UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES

Département d'informatique

Professeur Stéphane Marchand-Maillet
Docteur Alexandros Kalousis

Counterfactual Interactive Learning: designing proactive artificial agents that learn from the mistakes of other decision makers

THÈSE

Présentée à la Faculté des sciences de l'Université de Genève
Pour obtenir le grade de Docteur ès sciences, mention informatique

Par

Lionel BLONDÉ

de

Balschwiller (France)

Thèse No 5630

GENÈVE
2022

Abstract

The research endeavours laid out in this thesis set out to address weaknesses of reinforcement learning. As the branch of machine learning that orchestrates a reward-driven decision maker that must learn from its evolving world via interaction, reinforcement learning relies heavily on said reward's design. Modelling a reinforcement signal able to convey the right incentive to the agent for it to carry out the task at hand is far from obvious, and overall fairly tedious in terms of engineering and tweaking required. Besides, the interactive nature of the domain puts the agent and its world at satefy risk. The longer it takes to iterate over and empirically validate reward designs, the less sensible the suffered costs gradually become. Imitation learning bypasses this time-consuming hurdle by enticing the agent to mimic an expert instead of trying to maximize a potentially ill-designed reward signal. In effect, the shift from reinforcement to imitation learning trades the reward feedback the agent receives upon interaction off for the access to a dataset describing how an expert decision-making policy — assumed to perform optimally at the task — maps situations to decisions. Depite alleviating the reliance on potentially satefty-critical reward modelling, state-of-the-art imitation learning agents still suffered from severe sample-inefficiency in the number of interactions that need be carried out in the environment.

As such, we first set out to address the high sample complexity suffered by Generative Adversarial Imitation Learning, the state-of-the-art approach to performing imitation (albeit sample-inefficient). To that end, we build on the strengths of off-policy learning to design a novel adversarial imitation approach. Albeit coordinating infamously unstable routines — adversarial learning, and off-policy learning — our novel method remains remarkably stable, and dramatically shrinks the amount of interactions with the environment necessary to learn well-behaved imitation policies, by up to several orders of magnitude.

In an attempt to decipher and ultimately understand what made the previous method so stable in practice despite overwhelming odds of instability, we perform an in-depth review, qualitative and quantitative, of the method. We show that forcing the adversarially learned reward function to be local Lipschitz-continuous is a *sine qua non* condition for the method to perform well. We then study

the effects of this necessary condition and provide several theoretical results involving the local Lipschitzness of the state-value function. We complement these guarantees with empirical evidence attesting to the strong positive effect that the consistent satisfaction of the Lipschitzness constraint on the reward has on imitation performance. Finally, we tackle a generic pessimistic reward preconditioning add-on spawning a large class of reward shaping methods, which makes the base method it is plugged into provably more robust, as shown in several additional theoretical guarantees. We then discuss these through a fine-grained lens and share our insights. Crucially, the derived guarantees are valid for *any* reward satisfying the Lipschitzness condition, nothing is specific to imitation. As such, these may be of independent interest.

While imitation learning seems difficult to outperform once the agent has access to past experiences from an expert behaving optimally, the performance of the mimicking agent quickly deteriorates as the expert strays from optimality. Intuitively, it stands to reason that the agent should only imitate another if the latter solves the task (near-) optimally. If not, then imitation is not a well suited approach.

Datasets of optimal interactive experiences are difficult to come by, and deterringly tedious to design from scratch. Even then, these rarely perfectly align with the practitioner's desiderata for the set task. As a relaxation, interaction traces of sub-optimal behaviors for closely related (yet not exactly aligned) tasks are far more readily available (*e.g.* due to continually recording sensors present in robots, cars). As such, after studying how to *only learn from an external source* of *putatively optimal* data coming from an expert policy, we then study how to *only learn from an external source* of *putatively sub-optimal* data provided through an arbitrary dataset — this last one is the *offline* reinforcement learning setting.

While in the imitation learning setting we had no reward from the environment, we were still able to interact with the environment. In the offline reinforcement learning setting, the situation is reversed. The agent has access to logged interactive experiences from external sources, typically distinct agents. Said dataset of interactive traces is annotated with rewards perceived by the agents upon interaction. Yet, the agent learned via offline reinforcement learning is not allowed to interact with the world, and is therefore unable to witness its *own* impact on it when trying to make better decisions in it.

The performance of state-of-the-art baselines in the offline reinforcement learning regime varies widely over the spectrum of dataset qualities — ranging from *"far-from-optimal"* random data to *"close-to-optimal"* expert demonstrations. We re-implement these under a fair, unified, and highly factorized framework, and show that when a given baseline outperforms its competing counterparts on one end of the spectrum, it *never* does on the other end. This consistent trend prevents us from naming a victor that outperforms the rest across the board. We attribute the asymmetry in performance between the two ends of the quality spectrum to the amount of inductive bias injected into the agent to entice it to posit that what the behavior underlying the offline dataset is *optimal* for the task. The more

bias is injected, the higher the agent performs, *provided the dataset is close-to-optimal*. Otherwise, its effect is brutally detrimental. Adopting an advantage-weighted regression template as base, we conduct an investigation which corroborates that injections of such *optimality* inductive bias, when not done parsimoniously, makes the agent subpar in the datasets it was dominant as soon as the offline policy is sub-optimal. To design methods that perform well across the whole spectrum, we revisit the generalized policy iteration scheme for the offline regime, and study the impact of nine distinct newly-introduced proposal distributions over actions, involved in the proposed generalization of the policy evaluation and policy improvement update rules. We show that certain orchestrations strike the right balance and can improve the performance on one end of the spectrum *without* harming it on the other end.

In neither of the two studied settings (*a)* imitation learning and *b)* offline reinforcement learning) does the agent receive direct reward feedback from the world about the quality of the decisions it makes. It is never told *how well* it performed, and must infer how well it performed from traces of other policies, which *a priori* do not align with what the decision maker would have done. In that light, both studied settings can be cast as counterfactual learning, in which sequential decision-making agents must learn how to interact with their world, only from traces that *a priori* do not coincide with their own. *In fine*, we are concerned with decision makers that learn from the choices of others.

Résumé

Les travaux de recherche décris dans cette thèse visent les faiblesses majeures de l'apprentissage par renforcement; branche de l'apprentissage automatique qui tâche de concevoir des agents interactifs autonomes capables de choisir la sequence de decisions accompagnée des plus fortes récompenses. De tels agents dependent donc fortement de la façon dont laquelle ces récompenses sont formulées. Concevoir des récompenses parvenant à guider l'agent vers la résolution de la tâche à laquelle il fait face est laborieux et requiert de nombreuses interventions et réajustements. Ces derniers s'avèrent souvent onéreux et dangeureux en raison de la nature intéractive de l'apprentissage de l'agent qui apprend en intéragissant avec son environment. L'apprentissage par imitation n'est en revanche pas entravé par l'assemblage minutieux d'un signal de récompense, qui pourrait même ne pas être *in fine* aligné avec la tâche que l'agent doit résoudre. Au lieu de l'accumuler un maximum de récompenses lors de ses interractions avec son environment, l'agent doit imiter le comportement d'un expert, qui est supposé *a priori* optimal pour la tâche à accomplir, et qui lui est communiqué par l'intermediaire de démonstrations. Même si ces méthodes évitent l'étape préliminaire consistant à aligner un signal de récompense avec la tâche à résoudre, les meilleures methodes d'apprentissage par imitation demeuraient couteuse en terme du nombre d'interactions que l'agent devait effectuer avec l'environment avant d'espérer adopter un comportement proche de celui de l'expert.

Ainsi, nous cherchons dans un premier temps à reduire le nombre d'échantillons dont la méthode de pointe, l'apprentissage par imitation antagoniste generative, a besoin pour imiter l'expert. Pour ce faire, nous rendons la méthode "off-policy", ce qui permet à l'agent d'extraire davantage d'information de ses interactions passées en les préservant en mémoire et s'en remémorant occasionnelement. Malgré l'instabilité notoire des méthodes impliquées — apprentissage "off-policy" et methodes antagonistes generatives — notre nouvelle méthode est quant à elle remarquablement stable, et reduit le nombre d'interactions nécessaires d'au moins un ordre de grandeur.

Afin d'élucider pour quelles exactes raisons la méthode d'imitation dévelopée reste si stable en dépit des fortes chances de divergence, nous étudions la méthode proposée en profondeur. Ainsi, dans un

deuxième temps, nous démontrons que, afin d'assurer la stabilité de la méthode, la fonction de récompense apprise de façon antagoniste doit être une application lipschitzienne. Nous étudions les effets de cette condition nécessaire et dérivons plusieurs résultats théoriques caractérisant la fonction de valeur état-action, communément notée "Q". Ces garanties sont complémentées de preuves empiriques attestant de l'importance qu'ont les contraintes encourageant la récompense apprise à une application lipschitzienne sur la performance de l'agent. Enfin, nous focalisons notre attention sur une large classe de préconditioneurs visant à être appliqués à la récompense, et montrons via plusieurs garanties théoriques que les membres de la classe introduite rendent les récompenses auxquelles ils sont appliqués plus robuste. Les garanties dérivées s'étendent à toutes les récompenses lipschitziennes; elles ne se limitent pas à l'apprentissage par imitation.

Imiter le comportement d'un tier agent ne fait sens que lorsque ce dernier agis de façon optimale, mais l'obtention de démonstrations d'un tel expert est loin d'être une mince affaire, et devient vite laborieux lorsque les objectifs s'avèrent complexes. Des traces d'interaction décrites par un comportement sous-optimal sont bien plus facile à obtenir (*e.g.* capteurs d'une voiture autonome prenant continuellement des mesures lors de chaque navigation). Ainsi, après avoir étudié l'apprentissage de comportement interactif à partir de données provenant d'un expert supposé optimal, nous considérons, dans un troisième temps, la situation dans laquelle ces données de source(s) extérieure(s) sont potentiellement sous-optimales, et la traitons par apprentissage par renforcement hors-ligne (ou "offline"). Alors que l'agent qui apprenait par imitation ne recevait pas de récompense de son environement, il était tout du moins autorisé à interagir avec ce dernier. Pour l'agent qui apprend par renforcement hors-ligne, la situation est inversée. L'agent a accès a un historique d'experiences correspondant aux intéractions passées d'un ou de plusieurs autres agents, où chaque situation et décision prise par le tiers agent en question est annoté de la récompense qu'il a perçu lorsque la décision a été exécutée. En revanche, étant hors-ligne, l'agent ne peut lui-même pas interagir avec son environment, et ne peut de ce fait pas observer l'effet de ses propres décisions alors qu'il essaie de s'améliorer.

La performance d'agents apprenant par renforcement hors-ligne fluctue grandement en fonction de la qualité des données, qui peut varier sur une échelle allant de "loin d'être optimal" (*e.g.* aléatoires) à "près d'être optimal". En nous reposant sur notre ré-implémentation des méthodes de pointe d'apprentissage par renforcement hors-ligne au sein d'un système unifié, nous montrons que lorsqu'une de ces méthodes de pointe surpasse ses concurrentes sur une extremité de l'echelle, elle ne les surpasse *jamais* sur l'autre extremité. Ce motif récurrent nous empêche de nommer un vainqueur sur l'intégralité de l'échelle de qualité. Nous attribuons cette asymétrie au biais inductif qu'injecte ces méthodes dans leur agents réspectifs, où le biais en question pousse l'agent à partir du principe que le comportement décris dans les données hors-ligne est *optimal* pour la tâche à effectuer. Plus il y a de biais injecté, meilleur sera l'agent *à condition que* le dataset hors-ligne soit "près d'être optimal".

Sinon, l'injection a un effet néfaste. Nous les nommons biais inductif d'optimalité. En nous fondant sur une méthode de régression pondéré par la function valeur état-action, nous menons une investigation qui corrobore que l'injection de bias inductif d'optimalité reduit considérablement la performance de l'agent sur des données sont sous-optimales, si cette injection n'est pas effectuée avec parcimonie. Afin de concevoir une classe de méthodes qui permettrait à l'agent d'avoir de bonnes performance sur l'intégralité de l'echelle de qualité, nous revisitons le schéma de l'itération de politique généralisée. Nous étudions l'impact de nouvelles distributions sur l'espace de décisions dans le cadre d'une généralisation spécifique au régime hors-ligne de ce schéma que nous introduisons. Nos études montrent qu'il existent des orchestrations de distributions dans la généralisation proposée qui permettent à l'agent d'aboutir à un bon équilibre de performance où l'agent devient bien plus compétitif sur l'intégralité de l'échelle de qualité de données.

Contents

Contents

Chapter 1

Introduction

In this first chapter, we give a primer on *Reinforcement Learning*, abbreviated *"RL"* throughout the thesis. RL constitutes the common denominator of every research endeavors undertaken, reported, and discussed at length in this thesis. We lay out the foundations in this chapter, and will enrich this federating introduction, at the beginning of each subsequent chapter, with additional introductory elements in order for the reader to grasp how each of the treated settings branches out of RL.

The chapters of this thesis align with the contributions laid out in the following articles:

Chapter 2 Lionel Blondé, Alexandros Kalousis. *Sample-Efficient Imitation Learning via Generative Adversarial Nets.* Presented at AISTATS (poster); Apr 2019; Naha, JP

Chapter 3 Lionel Blondé, Strasser Pablo, Alexandros Kalousis. *Lipschitzness Is All You Need To Tame Off-policy Generative Adversarial Imitation Learning.* Post-rebuttal at the MLJ 2021

Chapter 4 Lionel Blondé, Alexandros Kalousis. *Where is the Grass Greener? Revisiting Generalized Policy Iteration for Offline Reinforcement Learning.* To be submitted at the JMLR 2021

1.1 RL in a nutshell

Reinforcement learning is the field of sequential decision making under uncertainty, where the decision maker must learn to maximize its long-term utility by interacting with a stateful world that rewards it with feedback upon interaction. The world or *environment* is unknown to the agent who can only learn from it through explicit interactive queries. Upon reacting to the agent's decision, the state of the world changes. From a practitioner standpoint, one must design a learning procedure that

enables the decision maker to figure out how to best answer the following question: ***what decision rule should I adopt to act* optimally *in any given situation?*** An agent is deemed *optimal* if its decision rule or *policy* enables it to accumulate more rewards in every episode of interaction than any other strategy, regardless of how the world reacted to each intermediary decision. Rewards are often delayed, delivered to the agent only after it has already committed to an interaction strategy. Receiving a comparatively high reward at a given state of the multi-stage decision process does not necessarily means that the action executed at said state was good, it might instead be due to having made a good decision a few steps prior. Identifying which decisions led to a satisfactory or disappointing outcome — equivalently, which actions *deserve credit* for the perceived result — is a tall order. This challenge is known as the *temporal credit assignment problem*, across naturalistic behavioral disciplines [1].

1.2 An artificial agent in a Markovian world

Facing the resolution of the reinforcement learning problem, the decision maker or agent is set in an interactive world, referred to as environment, and denoted by $\mathcal{E}$. The environment $\mathcal{E}$ is modeled as a memoryless, infinite-horizon, and stationary *Markov Decision Process* (*abbrv.* MDP), noted $\mathbb{M}$. Formally, the MDP is defined as the tuple $\mathbb{M} := (\mathcal{S}, \mathcal{A}, p, \rho_0, u, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ are respectively the state space and action space of the MDP. describing the states and actions that *can* occur in the decision process. Intuitively, states correspond to *situations*, and actions to *decisions*. Due to the discrete-time structure of a MDP, the experiences of the agent are divided across discrete timesteps t, or *stages*, where $t \geq 0$. One may refer to any number of sources to get a comprehensive account on these multi-stage decision processes, *e.g.* [Ros83] or [Put94]. Note, these books tackle *finite* MDPs (FMDPs) for the most part, defined as MDPs whose state and action spaces are both finite. Due to their conceptual simplicity compared to their continuous alternative, FMDPs have unsurprisingly been the object of far more studies. Besides, as stressed by Sutton in [SB98], one generally need not look further that FMDPs to deal with 90% of modern reinforcement learning problems.

The *dynamics* of the environment $\mathcal{E}$, *i.e.* how the Markovian world reacts to the agent's decisions, are determined by both p and ρ_0. The initial state probability distribution is built from the associated density ρ_0, while p is the stationary and stochastic transition function that decides how the environ-

[1] The temporal credit assignment problem has first been addressed by [Min61] in an artificial intelligence context. It was also studied, albeit later, in a branch of experimental psychology focused on animal behavior processes sometimes referred to as animal learning, *e.g.* in [Mac75]. Operant conditioning — class of associative learning, and subfield of animal learning; *cf.* [Ski48] for an early work on- and [Mac83] for an account on operant conditioning — have been shown to have a lot in common with RL, *e.g.* in [SZ97] and [TS97]. Note, operant conditioning exposes the animal to *reinforcers* (counterpart of RL's rewards) depending on its *decisions*, unlike Pavlovian (or classical) conditioning in which the animal's behavior is *not* reinforced *w.r.t.* what it does (*cf.* Dickinson's animal learning book [Dic80]). Connections have also been made in neuroscience between RL's rewards and dopamine neuron responses (*e.g.* [LAS92], [MDS96], [SDM97], [DKNU+20]).

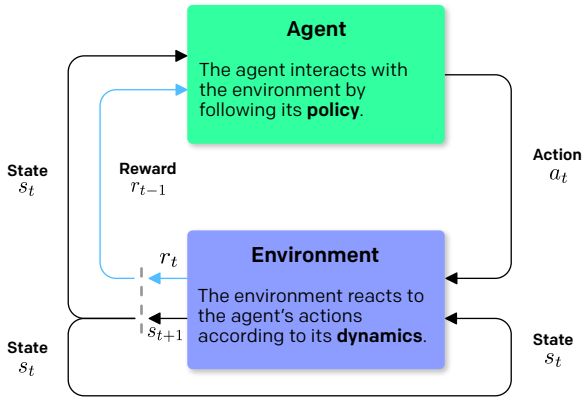


Figure 1.1: Reinforcement learning interaction diagram (similar to the one in [SB98].

ment's MDP transitions from one state to the next upon being affected by a decision from the agent. In concrete terms, $p(s'|s,a)$ is the conditional probability density concentrated at the (next) state s' when action a is executed in state s. Upon transitioning from s to s' via the action a in $\mathcal{E}$, the agent from which a originated is provided with feedback from $\mathcal{E}$, called reinforcement signal, or simply reward. The real-valued reward r received by the agent in response to this interaction is distributed as $r \sim u(\cdot|s,a)$, where u is modeled as a stationary reward process. Completing the $\mathbb{M}$ tuple, the discount factor γ is a coefficient that discounts the perceived payoff, to be picked in the $[0,1)$ real interval. It remains frozen at the preset value throughout the learning process; usually set at a value near 0.99.

The decision maker behaves in line with what its policy π dictates. π is followed consistently — at every stage of the decision process, and is modeled as a stochastic function that maps states to probability distributions over actions, *i.e.* as $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ (or as $\pi : \mathcal{S} \to \mathcal{A}$, if π is chosen to be deterministic). Concretely, at $s \in \mathcal{S}$, the agent concentrates a probability density equal to $\pi(a|s)$ at action $a \in \mathcal{A}$. If π puts a density $\pi(a|s)$ at a every time s is visited (whatever the stage), then π is said to be *stationary*.

Rather than considering a *finite*-horizon MDP, the *infinite*-horizon MDP setting is adopted. As such, the state t of the multi-stage process is *a priori* unbounded above in the formalism. Still, despite being within the infinite-horizon setting, one can posit that every trace contains at least one absorbing state, effectively making the MDP episodic. The discount factor γ is artificially set to zero when one such state is reached in order to emulate termination, hence formally constructing an *episode*. This will become more apparent when discussing the concept of *return*. When the analyses carried out call for the involvement of a finite time horizon T, how far the horizon is set will be adjusted accordingly.

The stateful interactive process between the agent and the environment is depicted in FIGURE 1.1.

For most of the theory developed in RL over the last decade to apply, one must assume that the Markovian world satisfied the *Markov property*. More accurately, one must assume that the conceptual MDP used to model how the worlds reactions to the agent's decision follows said property. The latter is likely not a property verified by the *real* world, but one that we can assume for the model set over the environment to best represent its inner workings and structure. The Markov property states that all that can be deemed as relevant in the history of interaction is present in the current state. Looking backwards at anything older that the current state would be redundant. By relying on this property, the decision maker can then base its current action solely on the current state, and need neither use nor even know about the MDP's past. As such, the transition and reward processes need (p and u, respectively) depend only upon the state and the action from the *current* stage, and upon *none* of the states and actions respectively occupied and executed by the agent at previous stages. This explains the design choices made to model π, p, and u in the paragraphs above. The Markov property is well illustrated by Silver: *"the future is independent of the past given the present"*, in lecture [Sil15].

Reinforcement learning relies on the *reward hypothesis*, stipulating that one can equivalently treat the satisfaction of a preset goal as the maximization of an accumulation of perceived scalar rewards. In line with this hypothesis, the completion of a task should coincide, from the agent's perspective, with having reached the utmost cumulative amount of rewards achievable during its learning lifetime. Adopting the practitioner's perspective, the reward function must be purposely designed such that it aligns with the satisfaction of this desideratum, both qualitatively (the agent must receive rewards when it does well) and quantitatively (it must receive increasingly higher rewards as it gets better at solving the task). Reward alignment is tedious to carry out, as rewards are easily prone to misspecification, and therefore usually require an inordinate amount of trial-and-error prior experimentations, going through handcrafted iterates of designs until it encodes the task properly, and instills the desired behavior primitives into the agent. This intensive iterative process is usually referred to as *reward shaping*, for which [NHR99] has been the reference. For example, [RA98] gives an account on how they managed to teach an artificial agent how to drive a bicycle, via reinforcement learning, yet through the extensive (albeit successful) use of reward shaping. On the flip side, there have various reported occurrences of reward misalignment leading to the unintended collapse of the agent onto another (usually trivial) task. From the practitioner's standpoint, it might look like the agent is malevolently bypassing the task it was asked to complete, while in effect, it merely was not able to understand the intent behind the reward signal crafted purposely by the practitioner to induce the desired (unachieved) behavior. Such a phenomenon has been aptly dubbed reward *hacking*, or reward *hijacking*, or even reward *corruption*, *cf.* OpenAI's blog post [AC16] for an animated account on faulty reward function and companion paper [AOS+16] for guidelines and pointers on reward modelling.

A particularly striking occurrence of reward misspecification can be witnessed in the biorobotics

works lead by Taylor and reported in [HT81] and [FT91]. Albeit separated by a ten-year gap, these works tackle the problem of identifying and understanding what is the primary physiological signal whose optimization would best explain the observed kinematics in horse gaits. First, based on their empirical findings, [HT81] claimed that animals (such as the horses they studied) regulate the speed at which they switch gaits such that it minimizes the energy costs provoked by the change of gait. It was later disproved in [FT91] who showed that natural occurrences of gait switches can be accompanied with higher energy consumption. Instead, it was reported that transitions occur once musculoskeletal forces put muscles, tendons, or bones under critical strain. Avoiding chances of injury then seemed to be what physiologically determined the speed at which horses switch gait, not the energetic cost of the switch. This line of work showcases that solving the inverse problem of identifying *what is the signal that, when optimized, would explain the observed behavior* (exactly what *inverse* reinforcement learning, or IRL, aims to solve) is a tall order. One must therefore be cautious when it comes to specifying rewards to induce a certain behavior, as one's intuitions (sometimes backed by published research) about the inverse problem — what is the objective (reward) that is being optimized by the actor displaying the observed behavior — might be off track, leading to fatally erroneous specifications. In the abstract [Rus98], Russell also draws a parallel between reinforcement learning and econometrics, noticing that the inverse problem of RL has been tackled by econometricians as the problem referred to as *structural estimation of Markov decision processes*, of which [Rus94] gives an account in 1994.

Reward design constitutes an active subfield of reinforcement learning research. The research endeavors reported in both Chapter 2 and Chapter 3 tackle the problem of *learning* reward functions in the context of imitation learning — drawing stronger ties with inverse reinforcement learning and apprenticeship learning specifically, thereby avoiding the burden of manual reward shaping.

Making the agent decide what decision to make at the current state *solely* so as to maximize the *immediate* reward is far from ideal, since rewards can be delayed (*cf.* Watkins' thesis, *"Learning from Delayed Rewards"*). The agent's decision or action for the current state might only reveal itself as good at a later stage, even if not immediately rewarding at execution. An action that appears non-rewarding now might lead to areas of the state space where the decisions made by the agent would *then* yield high rewards, even it very few rewards have been perceived along the way — hence, *"delay"*.

Instead, it is far better for the decision maker to maximize the rewards it gets and will get ***in every stage from the current one to the end of the episode***, *i.e.* over ***current and future*** stages. In order to formally account for these future rewards, albeit unoccurred at the current stage, the can maximize the entity dubbed *return*, defined here as the discounted sum of future rewards (including the immediate reward received upon execution of the current action). While several alternative definitions of the return have been proposed and studied in prior art (*e.g.* average reward, undiscounted

sum of future rewards), we stick to the discounted total sum of rewards mainly due to its widespread usage, but also due to its convenient analytical behavior in the infinite-horizon setting. The *discount* in question for the considered return variant is determined by the discount factor $0 \leq \gamma < 1$ introduced earlier. Consider the reward r_t as being the reward received by the agent upon acting at stage t. In the adopted discounted infinite-horizon setting, the return R_t^γ of the agent at stage t is, $\forall t \in \mathbb{N}$:

$$R_t^\gamma := \sum_{k=0}^{+\infty} \gamma^k r_{t+k} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots + \gamma^u r_{t+u} + \cdots \tag{1.1}$$

Note, if one were to use a *finite* horizon T, the return at T would coincide with the reward at T since there is no future to account for, no future rewards to assess, and nothing to discount: $R_T^\gamma = r_T$. To get a grasp on the extent to which future rewards are discounted as we step further into the future, consider a discount factor $\gamma = 0.99$. With such a γ value, r_{t+1} is discounted by a factor $\gamma = 0.99$, r_{t+2} by $\gamma^2 \approx 0.98$, r_{t+50} by $\gamma^{50} \approx 0.61$, r_{t+100} by $\gamma^{100} \approx 0.37$, r_{t+200} by $\gamma^{200} \approx 0.14$, r_{t+500} by $\gamma^{500} \approx 0.0066$. In other words, with $\gamma = 0.99$, the return R_t^γ effectively neglects the rewards the rewards received by the agent past 500 steps into the future starting from t. By contrast, when picking a higher discount factor, *i.e.* $\gamma = 0.995$, r_{t+500} is then only discounted by a factor of $\gamma^{500} \approx 0.082$. The closer γ is to 1, the further R_t^γ can *propagate* rewards backwards, along the chain of decisions, from stage $t + u$ to stage t, *i.e.* handle greater delays in rewards. γ determines from how far into the future the agent can propagate rewards to the current stage. Picking a greater γ nevertheless comes at the cost of being exposed to a harder credit assignment — *backward reward propagation* being the latest refinement of this problem — since the effective horizon is larger (less discount from one stage to the next). *In fine*, if $\gamma = 0$, the return trivially collapses onto the immediate reward, ($R_t^\gamma = r_t$), a case of little interest.

The aims for the RL agent is then to find an *optimal* policy to follow — noted π^*, where *optimality* aligns with the ability of π^* to always collect the maximum achievable return R_t^γ for every stage t. Wherever the decision maker starts, it will always pick the best action in terms of long-term gains. To instill such an optimal behavior into the agent, one must be able to identify what are the actions executed in the course of a given episode by the decision maker that most *deserve* to be given credit. The concept of return naturally plays an instrumental role in determining which are responsible for allowing the agent to be rewarded for its performance at progressing towards task completion, albeit with a delay. This problem of knowing who to assign credit to or who to blame in the context of a sequential decision process is known as the *(temporal) credit assignment problem*, and the notion of return is at the center of its resolution. By definition, optimal policies — non-unique *a priori*; unicity of optimality in policy space discussed momentarily — solve the temporal credit assignment problem, and as such, are the ones one must teach one's RL agent how to search for. *Where should the one look for to find an optimal policy?* As shown by Ross in [Ros83], in every FMDP, among all the optimal

policies, there exists at least one optimal policy that is stationary and deterministic — *stationarity* (or time-homogeneous; defined earlier) for deterministic policies signifies that the agent always picks the *same* action whenever it faces the same state. Since there exists a deterministic stationary policy that is optimal (proven by Ross in a *F*MDP), and since verifying these two properties is *a priori* more appealing than the alternative, it is generally-speaking a good idea for the agent to search for an optimal policy *solely* in the space of stationary deterministic policies. Concretely, determinism is naturally modeled via a functional mapping without probabilistic sampling unit of any kind, and the added stationarity can be urged by learning a *single* deterministic policy for all states and for all stages. Learning one distinct policy *per stage in the episode* would result in a *non*-stationary (or time-heterogeneous) policy, which are far more tedious to learn in terms of compute and implementation costs. As such, one usually leverages the previous theoretical results first articulated by Ross, and set out to look for an optimal policy in the space of stationary deterministic policies, whose existence has been proved in FMDPs. Puterman enriches the theoretical results of Ross by a considerable margin in [Put94].

1.3 How to solve the temporal credit assignment problem?

Now that we have established that temporal credit assignment in multi-stage decision processes is the problem that the interactive agent must solve to carry out its assigned task optimally, we need answer the following: ***How should one go about solving the (temporal) credit assignment problem?*** Historically, there have been two main methods, both spanning their own respective collection of algorithms, that have aimed at figuring out how to assign credit to decisions sequentially made by artificial agents over time. Whether there is actual interaction with the Markovian world as the optimal strategy is sought after is conceptually outside the scope of this investigation at least at the level of granularity we target here). These two wide-ranging methods are *1)* dynamic programming and *2)* reinforcement learning (the one we focus on in this thesis). There is a considerable overlap between the two approaches to assigning credit in time, as both are built upon the same principle of optimality to derive optimal behavior primitives. Said principle of optimality will be laid out momentarily.

Albeit constructed from the same foundational basis to craft agents that act optimally, the two approaches and the algorithms that follow their respective paradigms differ by what they assume is known about the world, and therefore what they are allowed to leverage from this knowledge. Not to be mistaken for its computer science counterpart (although its underlying, defining pattern of *overlapping sub-problems* fits the considered control theory interpretation too), dynamic programming *need* access to a perfect model of the environment. Besides, the latter need be modeled as a MDP, which concretely allows the artificial agent to evaluate the probability of a transition $(s, a \rightarrow s')$ to occur by having access to p, or the probability of receiving the reward r upon executing a in s by being

also granted access to u from the decision process, $\forall s, s' \in \mathcal{S}, \forall a \in \mathcal{A}, \forall r \in \mathbb{R}$. By having perfect knowledge about how the environment react to its decisions, the agent can simulate and evaluate any possible future, and plan on it over. It need not experience the world itself, as it can foresee and forecast *in silico* what it ought to do, and how its projected trajectories would roll out in the MDP. As such, due to this perfect knowledge about the Markovian world — allowing the agent to roll out any possible variation of the future exactly, thereby freeing the agent from having to guess how its actions might affect its surroundings — we say that dynamic programming algorithms *compute* optimal decision makers, rather than *learn* them. Dynamic programming deals with discrete-time problems, as our multi-stage decision problem. If we were to work in a continuous-time scenario, we would turn to its continuous-time counterpart, called optimal control (naturally, it has its own dedicated corpus or research, *cf.* [Kir04] for a comprehensive *introduction* to optimal control which starts with the simpler case of dynamic programming in the first chapter, or the reference book of Bertsekas on both dynamic programming and optimal control [Ber00]). First proposed in [Bel57], the diversity of applications dynamic programming allows for was then laid out in [BD62], and later in [DL77].

Conversely, in reinforcement learning, the agent need *not* access to a perfect model of the world. It *only* need access a set of samples — which the agent was provided with in dynamic programming too. So, while dynamic programming assumes both data and a perfect model of the MDP are available, reinforcement learning asks the agent to devise an optimal strategy from data alone. Several question might arise. *a) Were the data collected by the agent by following the most recent update of its policy? Or were the interaction data collected by the agent in the past, then stored, and re-sampled? Are the decisions present in the data even coming from the agent, or do they originate from interaction traces of another policy, in the same or closely related MDP? b) Is any piece of datum (state, decision, reward) present in the data even coming from the agent at all? Are the data being updated or augmented with the traces made by the agent in the environment as it learns via interaction? Or is the dataset fixed, frozen before training begins?* These questions are both viable and relevant in the current RL research landscape: question *a)* refers to whether one is learning *on-policy* or *off-policy*, and to what extent, while question *b)* refers to whether one is learning *online* or *offline*. The research endeavors reported in Chapter 2 and Chapter 3 perform online, off-policy learning. The ones laid out in Chapter 4 however conduct offline, off-policy learning. Still, we limit *this* exposition to simply touch on these questions and give pointer, as these are orthogonal to the ones we set out to lay out and answer in the scope of this introductory chapter. Besides, the generic way we introduce the concepts, approaches, and methods in this corpus remains nonetheless valid, whatever the answers to either *a)* and *b)* might be. In contrast with dynamic programming, we say that RL algorithms *learn* optimal decision makers, rather than *compute* them. RL decision-making strategies are *learned* since the decision maker has to figure out on its own how the world reacts to its actions. It can not foresee the MDP's reactions to its

projected decision rule and plan out future hypothetical routes *in silico* as it does not enjoy access to a perfect model of the world as a MDP. The term *learning* is also more fitting due to the agent having to internalize its own implicit mental image of the world in an attempt to grasp how the world works and what decision rule could *exploit* it for higher reward gains. It does so little by little as it sees samples (whatever their source might be) and in particular new samples that can bring in fresh knowledge for the decision maker to be internalized, making it less oblivious to the world surrounding it. As such, injecting behavior primitives urging to the agent to *explore* is of primary importance — provided the agent collects training data with its own policy in the training loop. We address this aspect of RL later in the chapter. While of great concern for the RL decision maker, balancing exploration and exploitation need not concern the dynamic programming agent, as it can fictitiously simulate any possible future outcome. Since it can rely on a queryable oracle giving capable of answering any question about the world, the dynamic programming agent need not understand the inner workings of the environment. Conversely, the RL agent must be able to *generalize* from experience, which qualifies as the ability to understand the world (to some extend) incrementally, coinciding with the definition of *learning*. Note, having the decision maker learn an approximate model of the of the environment from samples (concretely, of p and u, defining our multi-stage MDP), and use it to assist the policy is what is referred to as *model-based* RL. Here, we only consider *model-free* RL methods.

The earliest work (1961) isolating temporal credit assignment as the challenge to be addressed in a setting drawing strong similarities with the current status of reinforcement learning is [Min61]. More than two decades later, Sutton and Watkins explicitly tackle how to assign credit to decisions made by an agent in their Ph.D. theses ([Sut84] and [Wat89], respectively), and propose various solutions.

To go about solving the temporal credit assignment problem, dynamic programming and RL construct an *evaluation* function — called the *value function* — over the state space. By design, the values returned by the value function are tied to a policy, say π, and as such is denoted by V^π. We say that V^π *evaluates* π, and that $V^\pi(s)$ is the (evaluation) value of π at state $s \in \mathcal{S}$. For legibility purposes, let us consider consider stage-indexed transitions, or rather, states, actions, and rewards. At the stage t in the trajectory traced by the agent in the MDP, it carries out action a_t in state s_t, ends up in the next state s_{t+1} $\big($sampled from $p(\cdot|s_t, a_t)\big)$, and receives the reward r_t $\big($sampled from $u(\cdot|s_t, a_t)\big)$ upon transitioning. This schema is followed at every stage t in $\mathbb{N}$ of the multi-stage decision process. The return R_t^γ, defined earlier as the total discounted reward encountered by the agent in an episode (among the three traditional design choices for the concept of return), plays a central in the value function: the value of a policy at state s_t, $V^\pi(s_t)$, is defined as the *expected return* along all the possible traces made

by the policy in the MDP from stage t onward — again, the agent is in s_t at stage t. Formally, $\forall t \in \mathbb{N}$:

$$V^{\pi}(s_t) \coloneqq \mathbb{E}_{\pi}^{\geq t}\left[R_t^{\gamma}\right] \tag{1.2}$$

where the syntactic sugar $\mathbb{E}_{\pi}^{\geq t}[\cdot]$ signifies that the expectation of R_t^{γ} is taken *w.r.t.* every possible outcome ensuing from starting at s_t and following π thereafter in the MDP. Unpacking the notation,

$$\mathbb{E}_{\pi}^{\geq t}[\cdot] \coloneqq \mathbb{E}_{a_t \sim \pi(\cdot|s_t), r_t \sim u(\cdot|s_t,a_t), s_{t+1} \sim p(\cdot|s_t,a_t), a_{t+1} \sim \pi(\cdot|s_{t+1}), \ldots}[\cdot] \tag{1.3}$$

The definition of value given in EQ 1.2 remains unaltered regardless of how the return is defined (*e.g.* average reward, total reward), or whether one choose to instead consider the finite-horizon setting.

When we introduced the agent and its Markovian world, the *optimal* policy for the agent to follow was loosely defined as the behavior or strategy that, provided the agent acts in line with it at every single stage of the multi-stage decision process, yields the highest possible return, wherever the agent starts from. For the behavior to be optimal, it must maximize the expected return already from the very first stage in any trajectory that could result from following said behavior. With the concept of value function, we can equivalently say that, for the strategy to be optimal, it must have the *highest possible value at states sampled from the initial state distribution of the MDP*. Encoding the behavior of the RL decision maker, the policy π acts optimally at *every* stage $t \in \mathbb{N}$ if and only if, for any given start state $s_0 \sim \rho_0$, the policy π maximizes $V^{\pi}(s_0)$. If π maximizes the value at any *initial* state, then π maximizes the value at any ensuing state too, due to the value's *forward-view* structure. *In fine*, our objective is for the agent to learn an optimal policy π^*, as a non-unique solution to the problem:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s_0) \tag{1.4}$$

for any given start state $s_0 \sim \rho_0$, in the targeted policy search space.

1.4 Bellman's equation and principle of optimality

In his seminal work that has stood the test of time, [Bel57], Bellman showed that, in a *stationary* infinite-horizon MDP with discount factor $\gamma < 1$ (the MDP is said to be *"discounted"*), the value function is solution to a *recursive* functional equation, usually referred to as *Bellman's equation*. The recursive equation is directly derived via simple algebraic manipulations from the very definition of value (*cf.* EQ 1.2), and links the value of a given policy at state s with its value at the subsequent state s' in the interaction trace with the MDP. By formally packing the recursion over V^{π} under an operator $\mathcal{T}$ to be applied on V^{π}, and relying on the fact that the discount was purposely set to be inferior

to 1, Bellman showed that said operator $\mathcal{T}$ is a *contraction* ($\mathcal{T}$ is ℓ-Lipschitz-continuous with $\ell < 1$), under the infinite (*i.e.* supremum) norm. Since the operator $\mathcal{T}$ is a contraction mapping, Banach's fixed-point theorem tells us that $\mathcal{T}$ has a *unique* fixed point. In other words, not only is the value function a solution of Bellman's equation, it is its *unique* solution. As such, by iteratively and recursively applying the Bellman's operator $\mathcal{T}$ over a randomly initialized value V^π, we would ultimately converge, at a certain number of iterations, the value function V^π, since it is $\mathcal{T}$'s unique fixed point. The Bellman equation in V^π, for any transition (s, a, s', r) and $\gamma < 1$, writes as follows:

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\Big[\mathbb{E}_{r\sim u(\cdot|s,a)}[r] + \gamma\,\mathbb{E}_{s'\sim p(\cdot|s,a)}\big[V^\pi(s')\big]\Big] \tag{1.5}$$

for any policy π interacting with the MDP. Besides, if any function satisfies the equation EQ 1.5, and provided the MDP is discounted and stationary, then said function coincides with V^π, by unicity. In other words, we can retrieve V^π by finding a solution to EQ 1.5 under the right MDP conditions.

While this gives us a practical way to find the value function of an arbitrary policy π, this is not *per se* the end-goal we set out to attain. What we *really* want to have access to is V^*, the value of the optimal policy π^* — aligning with the discussion laid out earlier about the notion of *optimality* for policies. In line with this desideratum, Bellman articulated the *"principle of optimality"*, which any optimal decision rule π^* and associated value V^*, *w.r.t.* the tackled MDP, must satisfy. For any state s occupied by the agent at any stage in sequential multi-stage Markovian process, a policy π^* is optimal for s if and only if said policy π^* is also an optimal policy for the subsequent state $s' \sim p(s'|s, a)$ returned by the MDP upon executing $a\ \pi^*(s)$ in s. It only minor errata to EQ 1.5 to formalize the principle of optimality, which naturally gives rise to the *optimality* variant of Bellman's equation:

$$V^*(s) = \max_{a\in\mathcal{A}}\Big[\mathbb{E}_{r\sim u(\cdot|s,a)}[r] + \gamma\,\mathbb{E}_{s'\sim p(\cdot|s,a)}\big[V^*(s')\big]\Big] \tag{1.6}$$

that the optimal value *uniquely* satisfies. Indeed, the Bellman operator associated with the optimality specialization of Bellman's equation is also a contraction (again, using the supremum norm) in a discounted MDP scenario, which allows one to leverage Banach's fixed-point theorem. Ultimately, echoing the previous line of reasoning carried out for EQ 1.5, it ensues that the the optimality Bellman question has a unique solution V^*. Adopting a more practical lens, finding *a* solution to the optimality equation in EQ 1.6 signifies that one has found *the* optimal value V^* to the tackled MDP. Note, the presence of the max operator is contingent on how the very notion of optimality is defined, which is for the policy π^* to yield the *highest* possible expected return, whichever state it starts from.

Since π^* aligns with the strategy whose choices yield the highest values for any state, we write:

$$\pi^*(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left[\mathbb{E}_{r \sim u(\cdot|s,a)}[r] + \gamma \, \mathbb{E}_{s' \sim p(\cdot|s,a)} \left[V^*(s') \right] \right] \tag{1.7}$$

where the RHS of EQ 1.7 is obtained from EQ 1.6 by replacing the max operator with an argmax. EQ 1.7 is not a recursive functional equation, and as such, π^* can be obtained directly from the optimal value V^*, itself obtainable via a resolution of the optimality Bellman equation laid out in EQ 1.6.

We have seen that, by a unicity argument, any solution to the optimality Bellman equation (*cf.* EQ 1.6) must coincide with the optimal value V^*. When it comes to the unicity of the optimal policy, the optimal policy π^* defined in line with EQ 1.7 is unique *unless* there are states at which there are several choices of actions that yield maximal value, in which case, any such policy is *an* optimal policy.

1.5 On assigning credit: a case study

For didactic purposes, in the remainder of this section, we adopt a narrower scope by tackling the resolution of *finite* MDPs. We remind the reader that a finite MDP, or FMDP, is a sequential, multi-stage, Markov decision process in which both the state and action spaces ($\mathcal{S}$ and $\mathcal{A}$, respectively) are finite — hence *a fortiori* discrete. As reminded in Sutton's book, [SB98], FMDPs are by far the most common specialized form of MDPs, particularly common in RL theory. Besides, Sutton emphasizes that FMDPs *"are all you need"* to understand 90% of modern reinforcement learning. Consequently, we do *not* turn to more complex settings since these might come at the expense of conceptual clarity.

As one considers the interactive resolution of a *finite-horizon*, finite MDP, Bellman's principle of optimality allows one to compute the optimal value (*cf.* EQ 1.6) and optimal decision rule (*cf.* EQ 1.7) by *backward induction* — starting at the terminal stage (which coincides with the time horizon T if the episode did not terminate prematurely) and moving backwards to the start of the trajectory. As underlined in [Rus92] where the author discusses whether humans act according to Bellman's principle of optimality, the value function plays a central role in the backwards induction algorithm. The backward view adopted by the procedure is laid out in ALGORITHM 1. Note, although dynamic programming — the more general method that subsumes the technique of backward induction — originates from [Bel57], the constructive procedure of backward induction, aimed for solving sequential decision problems in stateful environments, could be attributed to research endeavors released prior to 1957 (*e.g.* in [Wal47]). The value function assigns, to any state, a measure of quality aligned with the expected future rewards the agent could accumulate by starting from this state *and* following the optimal policy thereafter. Specifically, in the FMDP context considered here, the value can be seen as the expected, projected future utility assigned to every node of the decision tree effectively described

by the FMDP that the agent will navigate through to one leaf, one decision at a time. The backward-view logic that is integral to the way backward induction approaches dynamic programming is not limited, in principle, to finite settings. Among others, [Bla62] and [Den67] show that the procedure consisting in computing the optimal value, starting from the leaves of the decision tree (described by the possible outcomes of interactions between the policy and the FMDP), and working its way backwards to the root of said tree, could be extended to *infinite*-horizon, *non*-finite MDPs.

As such, as long as the MDP is stationary, backward induction is *a priori* viable in virtually any MDP setting. Through the lens of an economist, [Rus94] describes the procedure of dynamic programming via backward induction as a constructive subroutine that computes the optimal decision rule by using the values returned by the estimated optimal value V^* as *shadow prices* to reduce the original stochastic, multi-stage, and therefore tedious resolution of the MDP into a sequence of single-step, deterministic, and static decision-making problems. Note, the shadow price is the *correct* valuation of the future consequences — typically in econometrics, monetary costs — of current decisions, whether these impending future costs are difficult to calculate or unknowable, *e.g.* undisclosed market prices, decisions of external financial actors, in the multi-agent environments that are financial markets. The concept of shadow price in econometrics is therefore the counterpart of the concept of *optimal* value in reinforcement learning, V^*. Since by design the value function fictitiously simulates every possible outcomes resulting from (by essence unobserved) future interactions, and *bottles* the associated expected future utility of each state in a value for that state, it stands to reason why Howard called the value function a *portable genius* in [How71]: wherever we place it, it tells us, how rewarding it would be to follow our current strategy from this state onward into the virtually unpredictable future.

Parsing Algorithm 1, one can note a few things about what it entails to use backward induction, and by extension, the overarching paradigm of dynamic programming, to solve finite-horizon FMDPs. First, since the time-horizon T, and both cardinalities of $\mathcal{S}$ and $\mathcal{A}$ are finite, *a)* the values (and decisions, indifferently) that the procedure yields over its entire lifetime can fit in table a *table* of $(T+1)\cdot|\mathcal{S}|$ entries — one cell per value ever calculated. Although it need accommodate $(T+1)\cdot|\mathcal{S}|$ values, the V-table set each cell only once. The backward induction algorithm computes and stores, once and without second pass, one optimal value $V_t^*(s)$ (and optionally the one associated optimal decision $\pi_t^*(s)$), $\forall t \in [0, T] \cap \mathbb{N}$ and $\forall s \in \mathcal{S}$. If one were to pack these stage-dependent values per state s, but across stages given said state, one would in effect define the ensuing compacted optimal value as $V^*(s) \coloneqq \big(V_0^*(s), V_1^*(s), \dots, V_{T-1}^*(s), V_T^*(s)\big)$ and policy as $\pi^*(s) \coloneqq \big(\pi_0^*(s), \pi_1^*(s), \dots, \pi_{T-1}^*(s), \pi_T^*(s)\big)$, for every state s in the finite state space of the tackled FMDP. As such, *b)* by being defined as a sequence of $T+1$ distinct per-stage optimal values and a sequence of $T+1$ distinct per-timestep non-stationary optimal deterministic policies respectively, both V^* and π^* are non-stationary, by definition (for formal

Algorithm 1: Dynamic programming via *backward induction* to solve Bellman's (optimality) equation [Bel57], for a finite MDP, and finite time horizon T.

Returns: the optimal value function V^*, and the optimal decision rule or policy π^*.

Init: we initialize (with zeros) a lookup table of size $(T+1) \cdot |\mathcal{S}|$ in which we will save the optimal values of every state at every timesteps, $V_t^*(s), \forall s \in \mathcal{S}$, and $\forall t \in [0, T] \cap \mathbb{N}$. We have access to the probability tables of the transitions and rewards, p and u respectively. We also have a dataset of trajectories of various lengths $\ell \leq T$. We define, $\forall t \in [0, T] \cap \mathbb{N}$, $\mathcal{S}_t$ as the set of states visited by the agent at the step t of any given trajectory in the dataset. Analogously, $\mathcal{A}_t$ the set of actions, and $\mathcal{R}_t$ the set of rewards at for the timestep t.

Note, we omit the complications due to premature episode termination for legibility.

As the name of the procedure hints at, start at $t = T$, and move *backwards* from there:

$$\left(\forall s \in \mathcal{S}_T\right) \qquad V_T^*(s) = \max_a r(s, a)$$

and store the $V_T^*(s)$ values, $\forall s \in \mathcal{S}_T$.

for $t \in \textsc{Reverse}(1, \ldots, T)$ **do**

  Retrieve $V_t^*(s)$ from the value lookup table.

  Optimal Value. $\forall s \in \mathcal{S}_{t-1}$, use the retrieved values to compute $V_{t-1}^*(s)$ as:

$$V_{t-1}^*(s) = \max_{a \in \mathcal{A}_{t-1}} \left[\sum_{r \in \mathcal{R}_t} u(r|s,a)\, r + \gamma \sum_{s' \in \mathcal{S}_t} p(s'|s,a) V_t^*(s') \right]$$

  Optimal Control. $\forall s \in \mathcal{S}_{t-1}$, use the retrieved values to compute $\pi_{t-1}^*(s)$ as:

$$\pi_{t-1}^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}_{t-1}} \left[\sum_{r \in \mathcal{R}_t} u(r|s,a)\, r + \gamma \sum_{s' \in \mathcal{S}_t} p(s'|s,a) V_t^*(s') \right]$$

  Write these values and decisions in the value lookup table.

end

expositions and discussions about stationarity in stochastic dynamic programming, whose principles and results naturally extend to reinforcement learning, *cf.* [Ros83]). Another noteworthy aspect to note from the procedure laid out in ALGORITHM 1 is that, *c)* being a dynamic programming method, backward induction need access to a perfect model of the environment the agent must face, *i.e.* of the (F)MDP. Concretely, and specifically for the considered FMDP, the agent must be allowed to retrieve elements from both the transition table p and the reward table u in order to calculate the non-stationary V^* and π^*, as the procedure explicitly reads. Knowledge of the Markovian world is how dynamic programming can assemble values, with the end-goal of solving the temporal credit assignment problem.

The expected return for executing action a selected by the policy π in state s and following the policy π thereafter is packaged as $Q(s, a)$, which we here define as Watkins in his Ph.D. thesis [Wat89]:

$$Q^\pi(s, a) = \mathbb{E}_{r\sim u(\cdot|s,a)}[r] + \gamma\, \mathbb{E}_{s'\sim p(\cdot|s,a)}\left[V^\pi(s')\right] \tag{1.8}$$

It was defined such that V^π would be the expectation of Q^π over the actions that *could* be chosen by the policy π at the current state. As such, while $V^\pi(s)$ requires a complete evaluation of the value of the policy for all of its potential (unobserved) actions, computing $Q^\pi(s, a)$ only requires the evaluation of one step a made by the policy. Computing $Q^\pi(s, a)$ is thus far less tedious than computing $V^\pi(s)$, hence the practical impact that made the introduction of the Q-function in the first place. This quantity, also appropriately referred to as *action-value*, is a measure of quality of an action performed by the policy it evaluates (*i.e.* the policy that we posit will be followed after executing a). Based on how we defined $V^\pi(s)$ in EQ 1.5, and how we just defined $Q^\pi(s, a)$ in EQ 1.8, $V^\pi(s')$ verifies:

$$V^\pi(s') = \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q^\pi(s', a')\right] \tag{1.9}$$

Hence, by plugging $V^\pi(s')$ as expressed in EQ 1.9 into the expression of $Q^\pi(s, a)$ reported in EQ 1.8, we can derive a recursive equation in $Q^\pi(s, a)$ satisfying the general form of Bellman's equation [Bel57]:

$$Q^\pi(s, a) = \mathbb{E}_{r\sim u(\cdot|s,a)}[r] + \gamma\, \mathbb{E}_{s'\sim p(\cdot|s,a)}\mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q^\pi(s', a')\right] \tag{1.10}$$

By symmetry, we repeat the process we laid out for Q^π, but for the optimal action-value Q^* associated with the optimal value V^*. In essence, Q^* is to π^* what Q^π is to π. Like with V^*, in the form, Q^* does not involve π^* as we model the pure greediness of π^*'s optimal behavior with max operators. As such:

$$Q^*(s, a) = \mathbb{E}_{r\sim u(\cdot|s,a)}[r] + \gamma\, \mathbb{E}_{s'\sim p(\cdot|s,a)}\left[V^*(s')\right] \tag{1.11}$$

Based on how we defined $V^*(s)$ in EQ 1.6, and how we just defined $Q^*(s, a)$ in EQ 1.11, $V^*(s')$ verifies:

$$V^*(s') = \max_{a' \in \mathcal{A}} Q^*(s', a') \tag{1.12}$$

Thus, by plugging $V^*(s')$ as expressed in EQ 1.12 into the expression of $Q^*(s, a)$ reported in EQ 1.11, we can derive a recursive equation in $Q^*(s, a)$ also satisfying the general form of Bellman's equation:

$$Q^*(s, a) = \mathbb{E}_{r \sim u(\cdot|s,a)}[r] + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)} \max_{a' \in \mathcal{A}} \left[Q^*(s', a') \right] \tag{1.13}$$

The concept of action-value, compared to the one of state-value, is also particularly appealing as it allows one to express the optimal (deterministic) policy in a more digestible form that in EQ 1.7. By substituting EQ 1.8 in EQ 1.7, we immediately see that one need only maintain an estimate of Q^* and act greedily on it to (implicitly) encode the behavior determined by the optimal policy π^*, as follows:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s, a) \tag{1.14}$$

Estimating the optimal action value Q^* by attempting to solve the recursive functional equation it is (uniquely) solution of (*cf.* EQ 1.13) is the crux of the seminal *Q-learning* algorithm [Wat89, WD92]. Concretely, the Q-learning agent *iteratively* updates its randomly initialized, stationary Q-value $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ towards the optimal action-value Q^* according to the following Q-learning update rule:

$$Q_{\text{new}}(s_t, a_t) \leftarrow Q_{\text{old}}(s_t, a_t) + \alpha \, \Delta^1_{\text{TD}} \tag{1.15}$$

where $\Delta^1_{\text{TD}} \coloneqq Q^{\text{targ}}_{\text{old}} - Q_{\text{old}}(s_t, a_t)$, and $Q^{\text{targ}}_{\text{old}} \coloneqq r_t + \max_{\tilde{a} \in \mathcal{A}} Q_{\text{old}}(s_{t+1}, \tilde{a})$. The shorthands Q_{old} and Q_{new} are the pre- and post-updates estimates of Q, respectively, and (s_t, a_t, r_t, s_{t+1}) is the *transition* (atomic unit of interaction, or trace, of the policy in the MDP) given to the agent at the depicted iteration. In addition, the scaling coefficient α is the learning rate, $Q^{\text{targ}}_{\text{old}}$ is the *Bellman target* of Q_{old} (RHS of Bellman's equation), and the *signed* gap Δ^1_{TD} is called the (1-step) *temporal difference* (signed gap between the LHS and RHS of Bellman's equation, *abbrv.* TD). The Bellman target $Q^{\text{targ}}_{\text{old}}$ does not coincide exactly with the one laid out in the RHS of EQ 1.13, but instead corresponds to a *point estimate* of the expectations over reward and next state displayed in EQ 1.13. As such, the Q-learning update rule in EQ 1.15 solely uses the reward r_t and the next state s_{t+1} resulting from a *single* execution of the policy in the MDP. One could also design the temporal difference to spread over $n > 1$ steps, in which case the gap Δ^1_{TD} in EQ 1.15 would be replaced by Δ^n_{TD}. This variant, called multi-step or n-step TD, was introduced in [PW96]. We unpack Δ^n_{TD} in both CHAPTER 2 and CHAPTER 3.

In essence, through this update rule, Watkins' Q-learning method [Wat89, WD92] is formulated to

urge the action-value Q being updated to be solution of the *optimality* Bellman equation (*cf.* EQ 1.13). Since Q^* is the *unique* solution of the latter, succeeding in making Q a solution of Bellman's optimality equation would fatally mean that the learned Q coincides with the optimal value Q^*. In order for Q to achieve this goal, EQ 1.15 updates the action-value as follows. *a)* Q is *decreased* ($Q_{\text{new}} < Q_{\text{old}}$) if $\Delta^1_{\text{TD}} < 0$, *i.e.* if $Q^{\text{targ}}_{\text{old}} < Q_{\text{old}}$, meaning that Q *overshoots* its Bellman target. Conversely, *b)* Q is *increased* ($Q_{\text{new}} > Q_{\text{old}}$) if $\Delta^1_{\text{TD}} > 0$, *i.e.* if $Q^{\text{targ}}_{\text{old}} > Q_{\text{old}}$, meaning that Q *undershoots* its Bellman target.

Note, even though we have presented the concept of action-value in its canonical form, we are, at this point in the exposition, in the *tabular* setting, where both value functions are represented via tables. Keeping track of such tables is viable since we are (for now) tackling the resolution of *finite* MDPs. If that were *not* the case (*e.g.* if either or both the state and action spaces $\mathcal{S}$ and $\mathcal{A}$ were continuous or discrete *infinite*), then we would not be able to organize $Q(s, a), \forall (s, a) \in \mathcal{S} \times \mathcal{A}$, in a tabular format.

Backward induction (a dynamic programming method, *cf.* ALGORITHM 1) and Q-learning (a reinforcement learning method, *cf.* EQ 1.15) are both built from Bellman's optimality equation. Yet, they adopt radically different approaches to solving the credit assignment problem the agent faces. Abbreviating backward induction with its acronym BI and Q-learning as QL, we carry out a side-by-side comparison of the key aspects and features of BE and QL. By laying out such a perspective, our aim is to shed light on *why* QL's *update scheme* has seen a wide-spread and almost hegemonic adoption in the years that followed its inception — in contrast with BI's *one-shot* resolution of finite MDPs.

The reported differences articulate around three main points, omitting the obvious distinction that BI maintains a *V-table*, while QL maintains a *Q-table* — with the ensuing advantage that computing Q has over computing V, as laid out earlier. First, *a)* BI walks backwards and *sets* the value function V^* iteratively from the horizon $t = T$ to the beginning of the episode $t = 0$. In particular, BI sets each value $V^*_t(s)$ of its $(T + 1) \cdot |\mathcal{S}|$ V-table *once* (without ever updating them), starting from the row of index T in the V-table, and gradually progressing towards the row of index 0. By design, *b)* this results in BI learning non-stationary values and policies (one per stage, from start to horizon), therefore hindering their ability to generalize across stages, which is a sought-after property in the tackled sequential multi-stage problem modeled via an MDP. Finally, *c)* to compensate for the fact that BI sets each value cell once and never update them once they are set, BI fictitiously simulates every possible outcome of a decision potentially made by the agent's policy at the current stage, using the transition table p and reward table u (tabular representations too, since the MDP is finite).

In contrast with BI, *a)* QL can update cells of the Q-table more than once, depending on the transitions presented to the agent. Besides, *b)* QL learns stationary values and policies by design (learned values are not state-specific), thereby allowing said learned value to generalize better across the stages spread along the episode. Experiences need not be organized in connex trajectories (which *is* needed

in BI) since QL merely ingests one atomic unit of interaction, *i.e.* transition, to carry out an update (provided we are using 1-step temporal difference gap Δ^1_{TD} in EQ 1.15). By being processed in such a non-connex fashion, correlations between transitions in a the various trajectories are removed, which further reinforced the agent's ability to generalize across stages. Note, the existence of a stationary optimal policy in the considered setting (stationary MDP) has been proven by Ross in [Ros83]. This result does not tell us whether our QL agent is able to learn an optimal policy, but merely that looking for an optimal policy in the set of stationary ones (like QL does) is a goal worth aiming for. Concretely, QL keeps its stationary Q-values $Q(s, a)$ is a Q-table of $|\mathcal{S}| \cdot |\mathcal{A}|$ entries, one for each of the values that the action-value $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ can take. Lastly, *c)* contrary to BI, QL requires access to neither p nor u tables — real ones or empirical estimates of them, and as such qualifies as a *model-free* method. Building on its greater genelizability capabilities brought forth by *i)* value stationarity (*w.r.t.* the stage) and *ii)* temporally decorrelated atomic experiences (also *w.r.t.* the stage), the QL agent gradually internalizes p and u as it gets exposed to more and more transitions. Since a *single* transition only conveys information about how p and u react to a *single* decision made by the QL agent, the latter will greatly benefit from witnessing a variety of outcomes to ultimately internalize and *understand* how the world (MDP) works. Such a variety desideratum is typically achieved by investing the agent (*any* agent) with a *dithering* mechanism. This enables the agent to discover new reactions of the MDP in response to the agent's decisions (*"exploration"*), rather that acting greedily *w.r.t.* its Q-value (*"exploitation"*). While this enables the model-free agent to gradually grasp how the world reacts to its decisions, potentially enabling it to refine its strategy to maximize its long-term return, devising an exploration strategy allowing the agent to branch out from its greedy behavior (*w.r.t.* Q) to uncover new, highly rewarding pathways in the MDP is a challenge on its own — aptly named the *"exploration-exploitation"* dilemma or trade-off. Devising ways to best address said trade-off is the crux of the archetypal multi-armed bandit problem, originated in [Rob52], the pioneering work of Robbins, later built upon in [LR85] by Lai and Robbins, also held as a major milestone in the field.

All in all, the laid out comparison between BI and QL, both aiming to solve the temporal credit assignment problem, allowed us to tackle a number of problems inherent to RL, and establishes QL as the baseline upon which every further mechanism with be built. Besides, the inspections of BI and QL carried out earlier in this case study made is clear that dynamic programming appears to be a far less appealing option than RL for non-trivial experimental scenarios, due to its reliance on the availability of a perfect model of the Markovian world the agent is facing to craft an optimal interactive strategy.

In what follows, we expand the reach of Q-learning by involving function approximation, freeing the method from the hindrances and practical limitation of the tabular setting, and as a byproduct enabling the learned Q-*function* to *generalize* uniformly over continuous state and action spaces.

1.6 Approximate RL via function approximation

Given that the concepts and tools purposely designed to enable the RL agent to deal with the burdensome temporal credit assignment problem have been laid out, the initial desideratum can be refined. As such, the resolution of the problem faced originally can be formally boiled down to searching and hopefully finding an *optimal* decision rule represented by a Q-function. By aligning the notion of optimality with Bellman's principle of optimality, converging to such optimal decision rule can equivalently be reduced as learning a Q-value that is solution to Bellman's optimality equation. Within the formal context laid out earlier, such an optimal decision rule pi^* is modeled implicitly in line with EQ 1.14 from the optimal Q-value Q^* learned as the solution of the optimality version of Bellman's equation, described in EQ 1.13. If the Q-function estimated by the decision maker satisfies the Bellman's optimality equation (in the case study delved into earlier, two distinct ways of urging the value to verify said optimality equation were investigated), then one is ensured of what follows: *a)* the explicitly learned Q^* is unique and therefore coincides with the real Q^*, and *b)* the implicitly learned π^* (defined from the Q^* estimate according to EQ 1.14) is *an* optimal policy, denoted by π^* despite its non-unicity — might be unique, but can not be guaranteed at this point, hence *a priori*) not unique.

It has also been established that exists a stationary deterministic policy that is optimal *for every F*MDP (*cf.* [Ros83, Put94]), naturally leading to the practical conclusion that, in light of this result, the most sensible initiative would be to look for the optimal policy in the set of deterministic stationary policies. In addition to this rigorously principled argument, such decision rules turn out to be far easier to implement, and cheaper to compute, than their stochastic or non-stationary counterparts. As reported in the case study carried out earlier in the chapter, the algorithm of backward induction (a dynamic programming approach) estimates the optimal policy via a deterministic non-stationary value, while the Q-learning algorithm (belonging to RL) estimates the optimal policy via a deterministic stationary value. Among the various reason laid out then (for one, the fact that dynamic programming requires a perfect model of the world as a MDP), Q-learning emerges as the undeniable victor and constitutes the obvious basis to build on — in large part due to it being an RL method, inherently superior to dynamic programming approaches when only samples are available. It then comes as no surprise that one can easily derive the template any modern RL algorithm from the skeleton of Q-learning. Nonetheless, everything tackled so far has been treated in a *finite* MDP, hence *a)* the Q-function is represented as a table whose cells are partially updated every iteration, and *b)* the Q-value update rule is tabular in nature, in can only be used on Q's represented as Q-tables as it is presented in EQ 1.15. To extend its range of application, Q-learning need be usable in more general MDPs.

As soon as one considers continuous state spaces (or even finite state spaces of high cardinality $|\mathcal{S}|$), modeling the Q-value *exactly* as a table becomes unfeasible. In his seminal paper responsible for the

inception of dynamic programming as a field of study [Bel57], Bellman addresses this phenomenon under the heading of *"curse of dimensionality"* — a phrase that has hitherto seen a widespread use across a wide range of domains whose models rely on the ingestion of data. The naming for such phenomenon stems from the inability of certain models (or more generally, frameworks) to scale and behave gracefully as the dimensionality of the tackled problem increases drastically. Usually, the phenomenon practically translates into an exponential growth in the computational intensity or footprint (non-exclusive) required to solve a discrete dynamic programming problem as its size increases — where said *size* is measured in terms of the total number of possible values the state and control variables can assume *in effect*. Due to the shared tabular format of their estimator, this *"curse"* burdening dynamic programming naturally hinders tabular RL algorithms in the same fashion. In response to this glaring need to alleviate the curse of dimensionality in these classes of value-based approaches, researchers have turned to substituting Q-tables and V-tables with *value function approximators*. The classes of methods that result from this transition are referred to as *approximate* dynamic programming and *approximate* reinforcement learning, respectively. Bertsekas and Tsitsiklis give an account of the fundamental theory for approximation methods in dynamic programming in [BT96].

In contrast with the *exact* learning update of tabular Q-learning laid out in EQ 1.15, one can now involve a function approximator Q_ω to estimate the optimal Q-value, and relax the update *rule* of the tabular *value iteration* method with an approximate minimization problem over the *temporal difference error* $\widehat{\Delta}^1_{\text{TD}}$ in EQ 1.16 (counterpart of Δ^1_{TD} in EQ 1.15). The method is referred to as the methods of temporal differences, or simply via the shorthand *"TD learning"*, *cf.* [Sut88, SMSM99].

The temporal-difference error or TD error $\widehat{\Delta}^1_{\text{TD}}$ writes as follows, for any transition (s_t, a_t, r_t, s_{t+1}):

$$\widehat{\Delta}^1_{\text{TD}} \coloneqq \left(Q_\omega(s_t, a_t) - Q^{\text{targ}}_\omega\right)^2 \coloneqq \left(Q_\omega(s_t, a_t) - \left(r_t + \max_{\tilde{a} \in \mathcal{A}} Q_\omega(s_{t+1}, \tilde{a})\right)\right)^2 \tag{1.16}$$

$\left(\text{where } Q^{\text{targ}}_\omega \coloneqq r_t + \max_{\tilde{a} \in \mathcal{A}} Q_\omega(s_{t+1}, \tilde{a})\right)$. By allowing the value function to be represented by an arbitrarily complex method able to implement the functional mapping, approximate RL increases the generalization capabilities of the base learning technique over the state space. Instead of updating individual, discrete cells in a table, the learning update now changes also the values in entire dense neighborhoods of the update location in the state space. This greater generalization capabilities are enables by the continuous and dense nature of the new representation, albeit yielding an *inexact* estimator in effect. Each update changes far more states than in the exact (tabular) setting, and will update states neighboring a given state s to similar of dissimilar values depending on the regularity (or smoothness, in terms of local Lipschitz-continuity) of the function approximator at s. Yet, as underlined by Sutton in [SB98], not every sophisticated mapping method is suited for the specific RL use case, despite the showcased empirical success of said models in other prolific subfields of machine

learning (*e.g.* supervised learning). For reinforcement learning method to leverage function approximation methods, these must be *plastic* enough to accommodate to samples collected gradually by the agent, and to be easily *remolded* as soon as fresh — and therefore more up-to-date — data becomes available to the decision maker. Also, the function approximator must be able to remain *stable* enough when dealing with non-stationary targets (changing over the course of the learning process), *e.g.* to weather the instabilities that might be caused by the presence of Q_ω in *both* the prediction *and* the target of the TD error $\widehat{\Delta}^1_{\text{TD}}$ (*cf.* EQ 1.16). This need for both stability and plasticity, despite being somewhat contradictory desiderata, has been addressed under the heading of *stability-plasticity dilemma* in [CG87]. We find another occurrence of this dilemma in CHAPTER 3. In fields backed by a considerable body of theoretical work (*e.g.* MDPs), gradient-based methods are usually preferred over their gradient-free alternatives, and RL is no exception. Neural networks updated via a stochastic gradient descent (SGD; or a more sophisticated counterpart) optimizer seem like the obvious candidate, due to their inherent plasticity (analogies involving brain activity are ubiquitous since their resurgence), their generalization capabilities, but also due to the advent of convenient *deep learning* frameworks (taking its name from the deep neural architectures they allow one to leverage as function approximators). The earliest work on neural networks can be traced back to 1970 with [Iva70], in the field of cybernetics, the study of self-regulating mechanisms in the face of feedback stimuli, originated in [Wie48]. Said deep learning frameworks *all* train the networks via the *back-propagation* algorithm, whose first occurrence can be attributed to Werbos who reports the developed procedure in his Ph.D. thesis [Wer74]. Its discovery is nevertheless largely attributed to [RHW86], arguably because application-oriented view they adopt enabled the authors to reach a far greater audience. The popular use of deep neural networks as function approximators in approximate reinforcement learning has given rise to the alternate name of *"deep RL"* for RL, although most endeavors in the field still report the use of fairly shallow architectures. How to best leverage deeper architectures (allowing for greater expressiveness as the number of parameters grow) in RL remains an open question. Note, when Q_ω is modeled via a parametric function approximator, *e.g.* a (deep) neural networks, the letter in index (here ω) usually corresponds to the model's parameter vector. From that point on, we follow this naming schema.

By *a)* modelling the value function via the neural function approximator, Q_ω can be updated iteratively to *b)* minimize the TD error $\widehat{\Delta}^1_{\text{TD}}$ (*cf.* EQ 1.16) over a (possibly evolving) set of transitions. The optimization *b)* of the value approximator *a)* can be done via any variant of SGD, to ultimately provide an appropriate RL counterpart of the Q-learning value iteration algorithm described in EQ 1.15. In effect, the approximation *a)* allows the value function to take in continuous states and actions as inputs (in stark contrast with being an set of values discretely updated and arranged in a table, where rows correspond to states and columns to actions, *w.l.o.g.*). Still, the approximation *b)* (TD error,

$\widehat{\Delta}^1_{\text{TD}}$) is *not* able to deal with continuous actions quite yet, due to the presence of a max operator over *all the possible actions* in the approximate Bellman target piece of $\widehat{\Delta}^1_{\text{TD}}$ (*cf.* CHAPTER 4 for an account of how said max operator could be mitigated in continuous action spaces, *e.g.* via search).

A priori, this approximate value iteration algorithm derived directly from Q-learning is best suited for continuous state spaces and (discrete) finite action spaces, due to *b)*, the optimality TD error $\widehat{\Delta}^1_{\text{TD}}$. Even then, the method might still suffer from the curse of dimensionality discussed earlier as the finite number of actions pickable per state gets larger. As such, in the approximate counterpart of Q-learning, the approximate value function Q_ω is best modeled as a mapping from a continuous state vector $s \in \mathcal{S}$ to a $|\mathcal{A}|$-dimensional value vector, whose $|\mathcal{A}|$ scalar components are the values of *every* actions $a \in \mathcal{A}$ at s. Such a value function approximation (*a)*) and learning objective (*b)*) were leveraged in *Deep Q-Networks* (*abbrv.* DQN) which marked a milestone of RL (and even more generally AI) by showing that RL can train artificial decision makers to playing Atari 2600 games at levels of performance exceeding human capabilities (*cf.* both companion papers [MKS+13, MKS+15]). Nonetheless, two extra techniques were required to allow for such a breakthrough to occur: *1)* a new *target network* trick (whose parameters are denoted by ω'), and *2)* an experience replay mechanism. The trick in *1)* consists in using $Q_{\omega'}$ instead of Q_ω in the Bellman target of $\widehat{\Delta}^1_{\text{TD}}$, where the parameters ω' are frozen copies of ω that are periodically replaced by up-to-date copies of ω throughout the course of the learning process. The use of a target network somewhat mitigate the non-stationarity of the Bellman target by making it move less than the predicted value in the TD loss $\widehat{\Delta}^1_{\text{TD}}$, effectively increasing the stability of the optimization (albeit making the TD loss a slightly worse approximation of what was first supposed to align exactly with Bellman's optimality equation). This trick was reported to yield considerable stability improvement. Experience replay (point *2)*) consists in first *storing* past experiences (collected in an online manner, by the agent, in the MDP) in a replay *buffer* of fixed size implemented as a FIFO data structure. As such, the replay buffer retains only relatively recent transitions; older ones are flushed out organically. The retention window depends on the data collection frequency *w.r.t.* model updates and on the buffer's size. Every iteration, the agent then picks transitions at random from the replay memory, and *replays* them, *i.e.* use the sampled transitions to carry out the learning update of Q_ω. Note, the (by construction *off-policy*) distribution that corresponds to sampling from the buffer at random is in effect a mixture of past policy iterates (the ones used to collect the samples effectively stored in memory). Experience replay is biologically grounded (artificial counterpart of hippocampal replay, studied in neuroscience, *e.g.* in [FW06, MD18]). Its first use *in silico* is attributed to [Lin92] in 1992, and has since been instrumental in various successful endeavors in RL. Trick *1)* and mechanism *2)* both appear in CHAPTER 2, CHAPTER 3, and CHAPTER 4.

Integrating the target network ω' and the replay buffer $\mathcal{R}$, DQN learns Q_ω by minimizing the loss:

$$\widehat{\Delta}^1_{\text{DQN}} \coloneqq \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{R}}\left[\left(Q_\omega(s_t,a_t) - \left(r_t + \max_{\tilde{a}\in c\mathcal{A}} Q_{\omega'}(s_{t+1},\tilde{a})\right)\right)^2\right] \tag{1.17}$$

where the expectation notation is exceptionally overloaded for legibility, and signifies that the loss $\widehat{\Delta}^1_{\text{DQN}}$ in EQ 1.17 is trained on transitions (s_t, a_t, r_t, s_{t+1}) sampled at random from replay memory $\mathcal{R}$.

Now that the case of value iteration has been treated, we move on to *policy iteration* next.

1.7 Generalized policy iteration

The archetypal approximate value iteration loss has been laid out in EQ 1.17, where the extra add-on techniques have proved necessary to stabilize the method in practice. Still, without the introduction of additional search or relaxation mechanisms, the value iteration archetype can only handle finite action spaces (and the set of viable actions in a given state must have low cardinality for the max operation to be efficiently solvable). To free the agent from such constraints on the action space, and enable it to deal with *any* action space specification, one must sidestep the agent's reliance on the max operation to learn an estimate of the optimal value Q^*, from which π^* is implicitly defined. We have hitherto aligned the notion of optimality with Bellman's principle of optimality, and as such, established the resolution of the optimality version of Bellman's equation (*cf.* EQ 1.13) as the natural yet sole means of learning the optimal action-value Q^*. Along this line of reasoning, one can not *a priori* hope to recover the optimal value without solving a max operation. An alternative solution that bypasses the need for such possibly arduous resolution consists in *1)* introducing a parametric policy π_θ, and learning it *explicitly* to be *greedy w.r.t.* Q_ω, and *2)* learning Q_ω to match Q^{π_θ} in line with EQ 1.10 instead of learning Q_ω to match Q^* directly in line with EQ 1.13 — which would then involve a max operator; yet, this is what is argued for and experimented with in [CB95], *cf.* ours extensive discussion laid out in CHAPTER 4 on that matter. Step *1)* has been referred to as the *greedification*, while step *2)* is usually called the *evaluation* step. Perhaps the most streamlined denominations for this steps are, equivalently, *policy improvement* for step *1)*, and *policy evaluation* for step *2)*. The alternation between these two step within the learning process is what, broadly speaking, defines what Sutton called *Generalized Policy Iteration*, or GPI, in [SB98]. The involvement of both an explicit policy and value approximators learned in an entangled way following the GPI scheme is the most broad way to frame what is called *policy iteration* (in contrast with *value* iteration). FIGURE 1.2 depicts GPI as a sequence diagram, illustrating how (potentially partial) alternating steps of greedification (π_θ update *w.r.t.* Q_ω) and evaluation (Q_ω update *w.r.t.* π_θ) converge to optimality (discussed shortly). Note, in this scenario, Q_ω takes continuous state and action vectors and returns scalar values in $\mathbb{R}$.

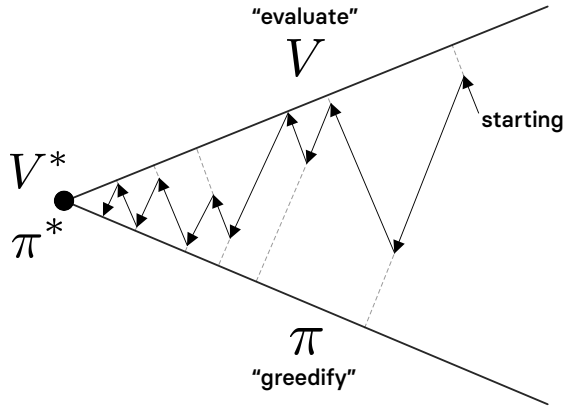


Figure 1.2: Sequence diagram representation of generalized policy iteration (GPI) — similar to the one in [SB98], except that the consecutive policy evaluation and improvement sub-goals are not performed to completion at each step. V and Q can be used interchangeably; the principle remains intact.

To get a better grasp of how entangled the objectives optimized by Q_ω and π_θ are in effect, we here denote their respective losses in the GPI procedure by ℓ_ω^{TD} and ℓ_θ. These losses write as follows:

$$\ell_\omega^{\text{TD}} \coloneqq \widehat{\mathbb{E}}_{s_t,a_t,r_t,s_{t+1}}\left[\left(Q_\omega(s_t,a_t) - \left(r_t + \mathbb{E}_{a'\sim\pi_\theta(\cdot|s_{t+1})}\left[Q_\omega(s_{t+1},a')\right]\right)\right)^2\right] \tag{1.18}$$

$$\ell_\theta \coloneqq -\widehat{\mathbb{E}}_{s_t}\left[\mathbb{E}_{a\sim\pi_\theta(\cdot|s_t)}\left[Q_\omega(s_t,a)\right]\right] \tag{1.19}$$

where $\widehat{\mathbb{E}}_{s_t,a_t,r_t,s_{t+1}}$ (and $\widehat{\mathbb{E}}_{s_t}$) signify that the expectation is taken over the effective distribution according to which the transitions (s_t, a_t, r_t, s_{t+1}) (and states s_t, respectively) are distributed. These samples may come from various sources, whose nature determines whether the method is online or offline, and whether it is on-policy or off-policy. Albeit out of scope for the currently followed line of reasoning, we revisit GPI in the *offline* RL setting through an in-depth and in-breadth study in Chapter 4.

Note, in GPI, the value function Q_ω need not be learned via TD learning (*cf.* eq 1.18), it can also be *fitted* directly in a standard supervised fashion to a Monte-Carlo estimation of the value. ℓ_ω^{MC} writes:

$$\ell_\omega^{\text{MC}} \coloneqq \widehat{\mathbb{E}}_{s_t,a_t,r_t,s_{t+1},\dots}\left[\left(Q_\omega(s_t,a_t) - \mathbb{E}^{>t}_{\pi_\theta}\left[R_t^\gamma\right]\right)^2\right] \tag{1.20}$$

where the syntactic sugar $\mathbb{E}^{>t}_{\pi_\theta}[\cdot]$ signifies that the expectation of R_t^γ (defined in line with eq 1.1) is taken *w.r.t.* every possible outcome ensuing from picking action a_t in state s_t and following π_θ there-

after in the MDP. Note, the notation used in EQ 1.2 was $\mathbb{E}_{\pi}^{\geq t}[\cdot]$, which did also take the expectation over the actions done at state s_t because it corresponded to the definition of V^{π}. By contrast, we are here handling *action*-values, which do not contain an expectation over the current action choice. Lastly for EQ 1.20, $\widehat{\mathbb{E}}_{s_t,a_t,r_t,s_{t+1},\ldots}$ signifies that, to carry out the learning update, the agent must have access to the episode segment starting from stage t and ending at episode termination. In practice, the agent must wait until the episode is over, compute the empirical discounted sum of reward from t to termination, and use this quantity as MC target for Q_{ω} at (s_t, a_t). One clear advantage of TD over MC is then that TD learning need not wait for the episode to terminate to be able to learn from the acquired samples. This is because TD learning *bootstraps* the value estimate at the current state with the value estimate at the subsequent state, in effect *"learning a guess from a guess"* (*cf.* [SB98]). As such, despite being being able to perform updates from incomplete trajectories (allowing for better sample-efficiency), the TD learning objective in EQ 1.18 is inherently less stable and tedious to tame than the MC-based objective in EQ 1.20. It is then not surprising that tricks like target networks — allowing the agent to mitigate the non-stationarity of the bootstrapped target — proved crucial empirically.

The crux of GPI and what FIGURE 1.2 illustrates is that if both processes (policy improvement and evaluation) stabilize, *i.e.* stagnate as the alternation keeps on being carried out, then the value function and the policy *must* be *jointly* optimal. Note, if the policy is almost perfectly greedy *w.r.t.* the value, the inner expectation $\mathbb{E}_{a'\sim\pi_{\vartheta}(\cdot|s_{t+1})}[Q_{\omega}(s_{t+1}, a')]$ that appears in EQ 1.18 is approximately equal to $\max_{\tilde{a}\in\mathcal{A}} Q_{\omega}(s_{t+1}, \tilde{a})$, itself appearing in EQ 1.16 (omitting here the target network use for symmetry). By training the value to be consistent *w.r.t* (equivalently, to evaluate) such a policy according to EQ 1.18, one is therefore essentially urging the value to be the solution of the optimality version of Bellman's equation (via TD error minimization). As such, as soon as the policy is close to optimal, the value is trained to align with the (unique; discussed earlier) optimal value. Conversely, as soom as the value is close to optimal, acting greedily *w.r.t* naturally defines *an* (*a priori* non-unique; discussed earlier) optimal policy. This discussion should not be held as rigorous proof of why GPI enables both policy and value to converge to their optimal counterparts — proof based on the *policy improvement theorem*, reported *e.g.* in [SB98]. Instead, it provides an intuitive depiction of how the interactive alternation of improvement and evaluation steps *cooperatively* moves the policy and value towards reaching a status of *shared* optimality they will ultimately reach together at the limit, while being functionally unable to achieve their common goal on their own (*cf.* their objectives in EQ 1.18 and EQ 1.19, displaying how they can *not* function in individually in isolation). Policy improvement and evaluation could also be seen as *competing* processes: *a)* when the policy changes, the value is not consistent *w.r.t* it, and *b)* when the value changes, the policy is not greedy *w.r.t* it anymore.

In order to optimize the decision rule followed by the agent encoded by the policy π_{ϑ}, the gradient of its objective, laid out in EQ 1.19, must be computer (gradient-based optimization dealt with

earlier). Computing and taking a step along the gradient of ℓ_θ *w.r.t* the parameter vector θ to optimize the policy via a gradient-based optimizer defines a spawns an entire class of algorithms called *policy-gradient* algorithms (*cf.* [SMSM99] for their inception, and [Sut01] where Sutton compares an array of policy-gradient algorithms). If TD learning is used to learn Q_ω in GPI, *i.e.* if Q_ω is updated to optimize EQ 1.18, then the resulting method is known as an *actor-critic*, where π_θ is the actor, and Q_ω is the critic (as it evaluates, or *critiques*, π_θ). These methods are described in [SB98] (CHAPTER 13.5). As such, actor-critic architectures are *a fortiori* a subclass of policy-gradient algorithm. Historically, the first notable occurrences of actor-critics in the policy-gradient research landscape were [KT00, PVS05, BSGL07, PS08, DPS12], tackling the on-policy setting. The first off-policy actor-critic algorithm and companion *Off-policy Policy-gradient Theorem* were developed by Degris in [DWS12], dealing with stochastic policies. Specific variants of the on-policy and off-policy policy-gradient algorithms were then developed by Silver in [SLH+14] for deterministic policies. Their main contribution is the *Deterministic Policy-gradient Theorem*, and is instrumental in both algorithms (on- and off-policy one).

All in all, a large majority of RL algorithms aiming to learn the optimal control in the sense of Bellman *can* be framed under the unifying GPI framework, with an identifiable learned policy and value.

Another type of approach, spawning the class of *likelihood-ratio* policy-gradient algorithms, does not learn a critic in its bare-bones form, just a policy π_θ. To get a gradient to update π_θ, it computes the *real* Monte-Carlo return and use this raw estimate of the value instead of Q_ω as it is currently laid out in EQ 1.19. Since the back-propagation of gradients can not occur through this raw estimate, these methods rely on the *score function gradient estimator* trick to estimate the gradient. This trick transforms the initially pathological gradient into the gradient of the logarithm of π_θ, weighted by the effective MC return, per action choice. In essence, the problem is reduced to a return-weighted log-likelihood maximization problem, which increases the probably for the agent of performing a certain action if the effective return of that action is high. The higher the return, the more this gradient estimate pushes up the log-probability of the action that led to this return (*e.g.* REINFORCE [Wil92], often used across ML disciplines to from gradients from non-differentiable losses and *black-boxes*).

Chapter 2

Sample-Efficient Imitation Learning via Generative Adversarial Nets

Abstract

GAIL is a recent successful imitation learning architecture that exploits the adversarial training procedure introduced in GANs. Albeit successful at generating behaviours similar to those demonstrated to the agent, GAIL suffers from a high sample complexity in the number of interactions it has to carry out in the environment in order to achieve satisfactory performance. We dramatically shrink the amount of interactions with the environment necessary to learn well-behaved imitation policies, by up to several orders of magnitude. Our framework, operating in the model-free regime, exhibits a significant increase in sample-efficiency over previous methods by simultaneously *a)* learning a self-tuned adversarially-trained surrogate reward and *b)* leveraging an off-policy actor-critic architecture. We show that our approach is simple to implement and that the learned agents remain remarkably stable, as shown in our experiments that span a variety of continuous control tasks. Video visualisations available at: `https://youtu.be/-nCsqUJnRKU`.

2.1 Introduction

Reinforcement learning (RL) is a powerful and extensive framework enabling a learner to tackle complex continuous control tasks [SB98]. Leveraging strong function approximators such as multi-layer neural networks, deep reinforcement learning alleviates the customary preliminary workload consisting in hand-crafting relevant features for the learning agent to work on. While being freed from this engineering burden opens up the framework to an even broader range of complex control and plan-

ning tasks, RL remains hindered by its reliance on reward design, referred to as *reward shaping*. Albeit appealing in theory, shaping often requires an intimidating amount of engineering via trial and error to yield natural-looking behaviours and makes the system prone to premature convergence to local minima [NHR99].

Imitation learning breaks free from the preliminary reward function hand-crafting step as it does not need access to a reinforcement signal. Instead, imitation learning learns to perform a task directly from expert demonstrations. The emerging policies mimic the behaviour displayed by the expert in those demonstrations. Learning from demonstrations (LfD) has enabled significant advances in robotics [BCDS08] and autonomous driving [Pom89, Pom90]. Such models were fit from the expert demonstrations alone in a supervised fashion, without gathering new data in simulation. Albeit efficient when data is abundant, they tend to be frail as the agent strays from the expert trajectories. The ensuing compounding of errors causes a covariate shift [RB10, RGB11]. This approach, referred to as *behavioral cloning*, is therefore poorly adapted for imitation. Those limitations stem from the sequential nature of the problem.

The caveats of behavioral cloning have recently been successfully addressed by Ho and Ermon [HE16] who introduced a model-free imitation learning method called *Generative Adversarial Imitation Learning* (GAIL). Leveraging Generative Adversarial Networks (GAN) [GPAM+14], GAIL alleviates the limitations of the supervised approach by *a)* learning a reward surrogate that explains the behaviour shown in the demonstrations and *b)* following an RL procedure in an inner loop, consisting in performing rollouts in a simulated environment with the learned surrogate as reinforcement signal. Several works have built on GAIL to overcome the weaknesses it inherits from GANs, with a particular emphasis on avoiding mode collapse [LSE17, HCS+17, KK17], causing policies to fail at displaying the diversity of demonstrated behaviours or skills [Goo17]. However, as the authors point out in the original paper ([HE16], Section 7), GAIL suffers from severe sample inefficiency. It is this limitation of GAIL that we address in this paper. "Sample-efficient" here means that we focus on limiting the number of agent-environment interactions, in contrast with reducing the number of demonstrations needed by the agent. Although learning from fewer demonstrations is not the primary focus of this work, our experiments span a spectrum of demonstration dataset sizes.

Failures of previous works to address the exceeding sample complexity stems from the on-policy nature of the RL procedure they employ. In such methods, every interaction in a given rollout typically is used to compute the Monte Carlo estimate of the state value by summing the rewards accumulated during the current trajectory. The experienced transitions are then discarded. Holding on to past trajectories to carry out more than a single optimization step might appear viable but often results in destructively large policy updates [SWD+17]. Gradients based on those estimates therefore

suffer from high variance, which can be reduced by sampling more intensively, hence the deterring sample complexity.

In this work, we introduce a novel method that successfully addresses the impeding sample inefficiency in the number of simulator queries suffered by previous methods. By designing an off-policy learning procedure relying on the use of retained past experiences, we considerably shrink the amount of interactions necessary to learn good imitation policies. Despite involving an adversarial training procedure and an actor-critic method, both notorious for being prone to instabilities and prohibitively difficult to train, our technique demonstrates consistent stability, as shown in the experimental section. Additionally, our reliance on the deterministic policy gradients allows us to exploit further information about the learned reward function, such as its gradient. Previous methods either ignore it by treating the reward signal as a scalar in a model-free fashion or train a forward model to exploit it. Our method achieves the best of both worlds as it can perform a backward pass from the discriminator to the generator (policy) while remaining model-free.

2.2 Related work

Imitation learning aims to learn how to perform tasks solely from expert demonstrations. Two approaches are typically adopted to tackle imitation learning problems: *a) behavioral cloning* (BC), originated in [Pom89, Pom90], which learns a policy via regression on the state-action pairs from the expert trajectories, and *b) apprenticeship learning* (AL) [AN04], which posits the existence of some unknown reward function under which the expert policy is optimal and learns a policy by *i)* recovering the reward that the expert is assumed to maximise (an approach called *inverse reinforcement learning* (IRL)) and *ii)* running an RL procedure with this recovered signal. As a supervised approach, BC is limited to the available demonstrations to learn a regression model, whose predictions worsen dramatically as the agent strays from the demonstrated trajectories. It then becomes increasingly difficult for the model to recover as the errors compound [RB10, RGB11, Bag15]. Only the presence of correcting behaviour in the demonstration dataset can allow BC to produce robust policies. AL alleviates this weakness by entangling learning the reward function and learning the mimicking policy, leveraging the return of the latter to adjust the parameters of the former. Models are trained on traces of interaction with the environment rather than on a fixed state pool, leading to greater generalization to states absent from the demonstrations. Albeit preventing errors from compounding, IRL comes with a high computational cost, as both modelling the reward function and solving the ensuing RL problem (per learning iteration) can be resource intensive [SBS08, SS08, HGE16, LPK11].

In an attempt to overcome the shortcomings of IRL, Ho and Ermon [HE16] managed to bypass the need for learning the reward function assumed to have been optimised by the expert when collect-

ing the demonstrations. The proposed approach to AL, *Generative Adversarial Imitation Learning* (GAIL), relies on an essential step consisting in learning a surrogate function measuring the similarity between the learned policy and the expert policy, using Generative Adversarial Networks (GAN) [GPAM+14]. The learned similarity metric is then employed as a reward proxy to carry out the RL step, inherent to the AL scheme. Recently, connections have been drawn between GANs, RL [PV16] and IRL [FCAL16]. In this work, we extend GAIL to further exploit the connections between those frameworks and overcome a limitation that was left unaddressed: the burdensome sample inefficiency of the method.

GANs involve a generator and a discriminator, each represented by a neural network, making the associated computational graph fully differentiable. The gradient of the discriminator with respect to the output of the generator is of primary importance as it indicates how the generator should change its output to have better chances at fooling the discriminator at the next iteration. In GAIL, the generator's role is carried out by a stochastic policy, causing the computational graph to no longer be differentiable end-to-end. Following a model-based approach, [BACM17] recovers the gradient of the discriminator with respect to actions (via reparametrization tricks) and with respect to states (via a forward model), making the computational graph fully differentiable. In contrast, the method introduced in this work can, by operating over deterministic policies and leveraging the deterministic policy gradient theorem [SLH+14], directly wield the gradient of the discriminator with respect to the actions, without requiring gradient estimation techniques (*e.g.* reparametrization trick [KW14], Gumbel-Softmax trick [JGP17, MMT17]). Since we stick to the model-free setting, states remain stochastic nodes and therefore block (backward) gradient flows.

An independent endeavour to overcome the data inefficiency of GAIL has very recently been reported in [KAD+19], in which the authors leverage a similar architecture, yet rely on an arguably ad-hoc preliminary preprocessing technique on the demonstrations before the imitation begins. In contrast, our method does not rely on any preprocessing to yield gains in sample efficiency by orders of magnitude.

2.3 Background

Setting

We address the problem of an agent learning to act in an environment in order to *reproduce* the behaviour of an expert demonstrator. No direct supervision is provided to the agent — she is never directly told what the optimal action is — nor does she receives a reinforcement signal from the environment upon interaction. Instead, the agent is provided with a pool of trajectories and must use them to guide its learning process.

World and agent

We model this sequential interactive problem over discrete timesteps as a *Markov decision process* (MDP) $\mathbb{M}$ [Put94], formalised as a tuple $(\mathcal{S}, \mathcal{A}, \rho_0, p, r, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ respectively denote the state and action spaces. The dynamics are defined by a transition distribution with conditional density $p(s_{t+1}|s_t, a_t)$, along with ρ_0, the density of the distribution from which the initial state is sampled. Finally, $\gamma \in (0, 1]$ denotes the discount factor and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function. We consider only the fully-observable case, in which the current state can be described with the current observation $o_t = s_t$, alleviating the need to involve the entire history of observations. Although our results are presented following the previous infinite-horizon MDP, the MDPs involved in our experiments are *episodic*, with $\gamma = 0$ at episode termination. In the theory, whenever we omit the discount factor, we implicitly assume the existence of an absorbing state along any agent-generated trajectory.

We formalise the sequential decision making process of the agent by defining a policy π_θ, modelled via a neural network with parameter θ. $\pi_\theta(a_t|s_t)$ designates the conditional probability density concentrated at action a_t when the agent is in state s_t. In line with our setting, the agent interacts with $\mathbb{M}^-$, an MDP comprising every element of $\mathbb{M}$ except its reward function r. Since our approach involves learning a surrogate reward function, we use $\mathbb{M}^+$ to denote the MDP resulting from the augmentation of $\mathbb{M}^-$ with the learned reward. We can therefore equivalently assume that the agent interacts with $\mathbb{M}^+$. *Trajectories* are traces of interaction between an agent and an MDP. Specifically, we model trajectories as sequences of *transitions* (s_t, a_t, r_t, s_{t+1}), atomic units of interaction. *Demonstrations* are provided to the agent through a set of expert trajectories τ_e, generated by an expert policy π_e in $\mathbb{M}$.

Objective

We now introduce additional concepts and notations that will be used in the remainder of this work. The *return* is the total discounted reward from timestep t onwards: $R_t^\gamma := \sum_{k=t}^{+\infty} \gamma^{k-t} r(s_k, a_k)$. The state-action value, or Q-value, is the expected return after picking action a_t in state s_t, and thereafter following policy π_θ: $Q^{\pi_\theta}(s_t, a_t) := \mathbb{E}^{>t}_{\pi_\theta}[R_t^\gamma]$, where $\mathbb{E}^{>t}_{\pi_\theta}[\cdot]$ denotes the expectation taken along trajectories generated by π_θ in $\mathbb{M}^+$ (respectively $\mathbb{E}^{>t}_{\pi_e}[\cdot]$ for π_e in $\mathbb{M}$) and looking onwards from state s_t and action a_t. We want our agent to find a policy π_θ that maximises the expected return from the start state, which constitutes our performance objective, $J(\pi) := \mathbb{E}_\pi[R_0^\gamma]$, i.e. $\pi_\theta = \operatorname{argmax}_\pi J(\pi)$, for any given start state $s_0 \sim \rho_0$.

To ease further notations, we finally introduce the *discounted state visitation frequency* of a policy π in the MDP $\mathbb{M}$, denoted by $\rho^\pi_\mathbb{M}$, and abbreviated ρ^π when non-ambiguous. Formally, it is defined by $\rho^\pi_\mathbb{M}(s) := \sum_{t=0}^{+\infty} \gamma^t \mathbb{P}^\pi_\mathbb{M}[S_t = s]$, where $\mathbb{P}^\pi_\mathbb{M}[S_t = s]$ is the probability of reaching state s at timestep t (S_t is a random variable) when sampling the initial state from ρ_0, and thereafter following policy π in $\mathbb{M}$.

Given that a simple derivation gives $\sum_{s\in\mathcal{S}}\rho^{\pi}_{\mathbb{M}}(s) = 1/(1-\gamma)$, the frequency $\rho^{\pi}_{\mathbb{M}}$ can be interpreted as a *probability distribution* over states up to a constant factor. In our experiments, we omit the discount factor for state visitation, in line with common practices.

Generative Adversarial Imitation Learning

Leveraging *Generative Adversarial Networks* [GPAM+14], *Generative Adversarial Imitation Learning* [HE16] introduces an extra neural network D_φ to play the role of *discriminator*, while the role of *generator* is carried out by the agent's policy π_θ. D_φ tries to assert whether a given state-action pair originates from trajectories of π_θ or π_e, while π_θ attempts to fool D_φ into believing her state-action pairs come from π_e. The situation can be described as a minimax problem $\min_\theta \max_\varphi \; V(\theta, \varphi)$, where the *value* of the two-player game is $V(\theta, \varphi) := \mathbb{E}_{\pi_\theta}[\log(1 - D_\varphi(s, a))] + \mathbb{E}_{\pi_e}[\log D_\varphi(s, a)]$. We omit the causal entropy term for brevity. The optimization is however hindered by the stochasticity of π_θ, causing $V(\theta, \varphi)$ to be non-differentiable with respect to θ. The solution proposed in [HE16] consists in alternating between a gradient step (Adam, [KB14]) on φ to increase $V(\theta, \varphi)$ with respect to D_φ, and a policy optimization step (TRPO, [SLM+15]) on θ to decrease $V(\theta, \varphi)$ with respect to π_θ. In other words, while D_φ is trained as a binary classifier to predict if a given state-action pair is real (from π_e) or generated (from π_θ), the policy π_θ is trained by being rewarded for successfully confusing D_φ into believing that generated samples are coming from π_e, and treating this reward as if it were an external analytically-unknown reward from the environment.

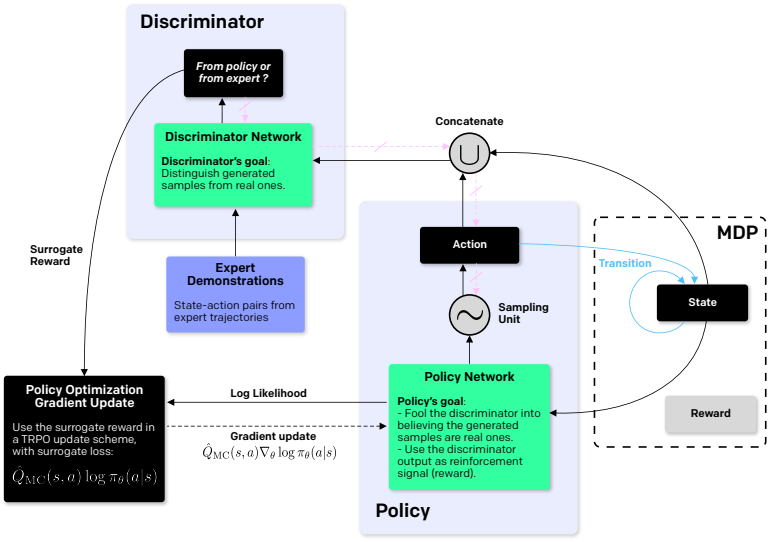


Figure 2.1: Inter-module relationships in Generative Adversarial Imitation Learning [HE16].

Actor-critic

Policy gradient methods with function approximation [SMSM99], referred to as *actor-critic* (AC) methods, interleave *policy evaluation* with *policy iteration*. Policy evaluation estimates the state-action value function with a function approximator called *critic* $Q_\omega \approx Q^{\pi_\theta}$, usually via either Monte-Carlo (MC) estimation or Temporal Difference (TD) learning. Policy iteration updates the policy π_θ by greedily optimising it against the estimated critic Q_ω.

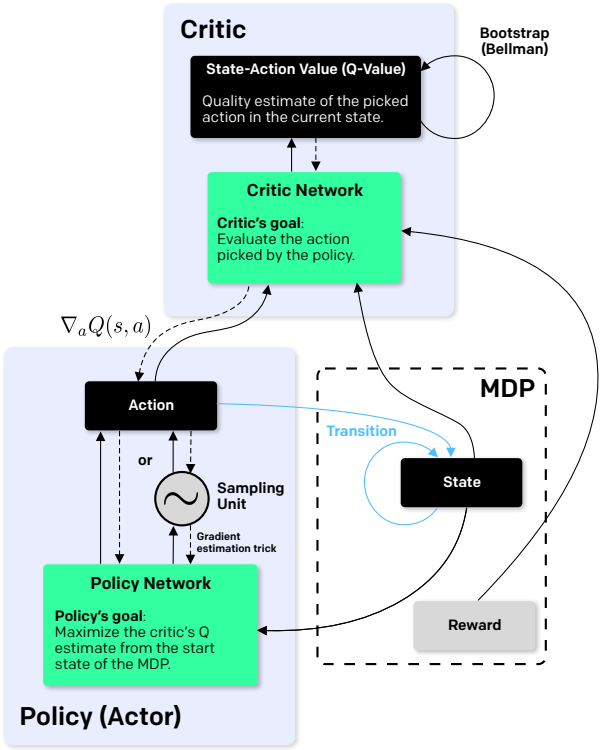


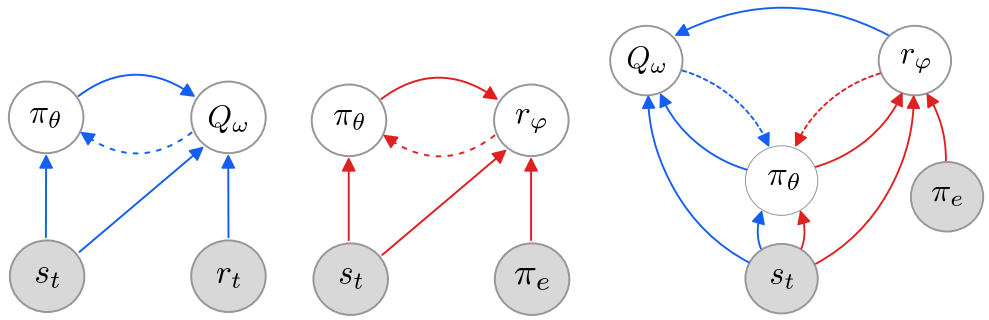
Figure 2.2: Inter-module relationships in an Actor-Critic architecture [SMSM99].

2.4 Algorithm

The approach in this paper, named *Sample-efficient Adversarial Mimic* (SAM), adopts an off-policy TD learning paradigm. By storing past experiences and replaying them in an uncorrelated fashion, SAM displays significant gains in sample-efficiency, in line with [WBH+16, GLG+16]. To solve the differentiability bottleneck of [HE16] caused by the stochasticity of its generator, we operate over deterministic policies. At a given state s_t, following its deterministic policy μ_θ, an agent selects the action $a_t = \mu_\theta(s_t)$. Alternatively, we can obtain a deterministic policy from any stochastic policy π_θ

by systematically picking the average action for a given state: $\mu_\theta(s_t) = \mathbb{E}_a[\pi_\theta(a|s_t)]$. By relying on an off-policy actor-critic architecture and wielding deterministic policies, SAM builds on the *Deep Deterministic Policy Gradients* (DDPG) algorithm [LHP+16], in the context of Imitation Learning.

SAM is composed of three interconnected learning modules: a *reward* module (parameter φ), a *policy* module (parameter θ), and a *critic* module (parameter ω) (*cf.* FIGURE 2.3 and 2.4). The reward and policy modules are both involved in a GAN's adversarial training procedure, while the policy and critic modules are trained as an actor-critic architecture. As reminded recently in [PV16], GANs and actor-critic architectures can be both framed as bilevel optimization problems, each involving two competing components, which we just listed out for both architectures. Interestingly, the policy module plays a role in both problems, tying the two bilevel optimization problems together. In one problem, the policy module is trained against the reward module, while in the other, the policy module is trained against the critic module. The reward and critic modules can therefore be seen as serving analogous roles in their respective bilevel optimization problems: forging and maintaining a signal which enables the reward-seeking policy to adopt the desired behaviour. How each of these component is optimised is described in the subsequent dedicated sections.



(a) Actor-Critic [SMSM99] (b) GAIL [HE16] (c) SAM [BK19]

Figure 2.3: Inter-module relationships in different neural architectures (the scope of this figure was inspired from [PV16]). Modules with distinct loss functions are depicted with empty circles, while filled circles designate environmental entities. Solid and dotted arrows respectively represent (forward) flow of information and (backward) flow of gradient. *a)* Generative Adversarial Imitation Learning [HE16] *b)* Actor-Critic architecture [SMSM99] *c)* SAM (this work). Note, in SAM, the critic takes in information from the reward module, while in the vanilla AC architecture, the critic receives the reward from the environment. The gradient flow from the critic to the reward module must however be sealed. Indeed, such a gradient flow would allow the policy to adjust its parameters to induce values of the reward which yield low TD residuals, hence preventing both critic and reward modules to be learned as intended.

As an off-policy method, SAM cycles through the following steps: *i)* the agent uses π_θ to interact with

$\mathbb{M}^+$, *ii)* stores the experienced transitions $\mathcal{C}$ in a replay buffer $\mathcal{R}$, *iii)* updates the reward module φ with an equal mixture of uniformly sampled state-action pairs from $\mathcal{C}$ and τ_e, *iv)* updates the reward module φ with an equal mixture of uniformly sampled state-action pairs from $\mathcal{R}$ and τ_e, and *v)* updates the policy module θ and critic module ω with transitions sampled from $\mathcal{R}$. Note that while sampling uniformly from $\mathcal{C}$ (iii) gives states and actions distributed as ρ^{π_θ} and π_θ respectively (on-policy), sampling uniformly from $\mathcal{R}$ (iv) gives states and actions distributed as ρ^β and β respectively, where β denotes the off-policy sampling mixture distribution corresponding to sampling transitions uniformly from the replay buffer. A more detailed description of the training procedure is laid out in the algorithm pseudo-code (*cf.* ALGORITHM 2).

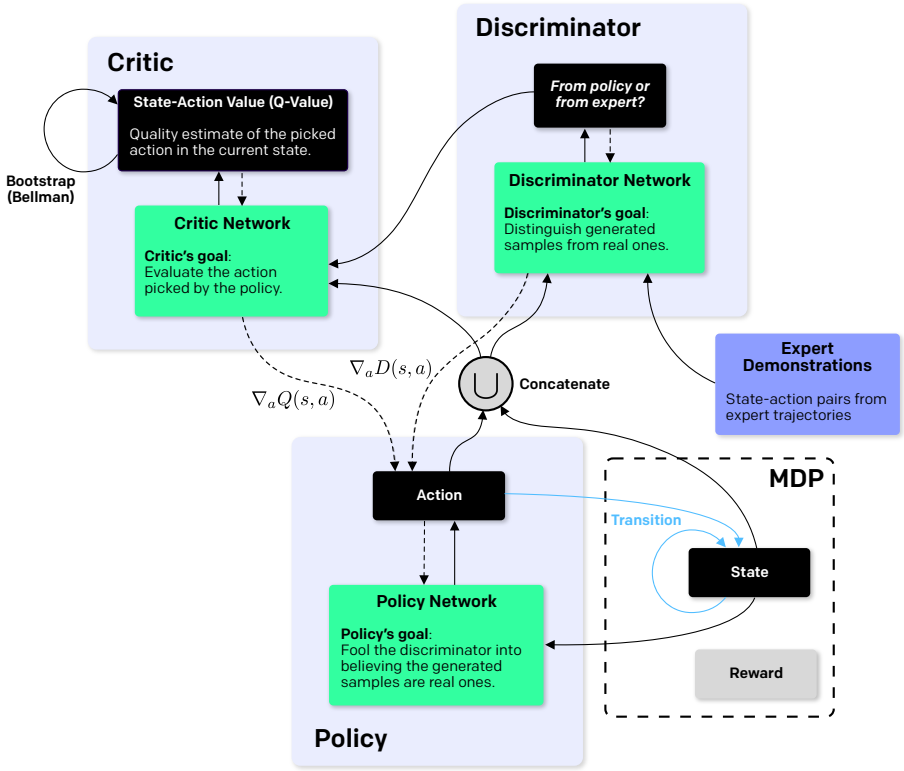


Figure 2.4: Inter-module relationships in SAM (this work).

2.4.1 Reward

We introduce a reward network with parameter vector φ, operating as the discriminator. The cross-entropy loss used to train the reward network is:

$$\mathbb{E}_{\pi_\theta}[-\log(1 - D_\varphi(s,a))] + \mathbb{E}_{\pi_e}[-\log D_\varphi(s,a)] + \lambda\, \mathfrak{R}_{\text{GP}}(\varphi) \tag{2.1}$$

where $\mathfrak{R}_{\text{GP}}(\varphi)$ is a penalty on the discriminator gradient, as introduced in [GAA+17], Section 4. [LKM+17] reports benefits from applying such regulariser to the non-saturated variant of the discriminator loss, although it was initially introduced for Wasserstein GANs [ACB17] in [GAA+17]. This penalty favours our method by further improving its stability.

The reward is defined as the negative of the generator loss. The later has been declined in many variants, which are thoroughly compared in [LKM+17]. We can therefore analogously define a synthetic reward for each of these forms. We go over and discuss major ones in supplementary material. Additionally, [FLL18] proposes an extra variant in the context of IRL. In the remainder, we use $r_\varphi(s_t, a_t) = -\log(1 - D_\varphi(s_t, a_t))$ as synthetic reward. The reward network is trained, each iteration, first on the mini-batch most recently collected by π_θ, then on mini-batches sampled from the replay buffer. Although [PV16] reports that using a replay buffer in GANs causes the generation to be poor, we do not seem to suffer the same detrimental effect in the continuous control tasks we tackle.

2.4.2 Critic

The loss optimised by the critic, noted $\ell(\omega)$, involves three components: *i)* a 1-step Bellman residual $\ell_1(\omega)$, *ii)* a n-step Bellman residual $\ell_n(\omega)$, and *iii)* a weight decay regulariser $\mathfrak{R}_{\text{WD}}(\omega)$. A similar loss is employed in [VHS+17] in the context of Reinforcement Learning from Demonstrations. While the authors use weight decay regularisers for both the policy and the critic, we restrain from decaying the policy's weights since, in our setting, the policy plays a role in two distinct optimization problems. We do not apply a weight decay regulariser for the discriminator either, as it was proven to cause the Wasserstein GAN *critic* (name given to the discriminator in Wasserstein GANs) to diverge [GAA+17].

We define the critic loss as follows:

$$\ell(\omega) = \ell_1(\omega) + \ell_n(\omega) + \nu\, \mathfrak{R}_{\text{WD}}(\omega) \tag{2.2}$$

where ν is a hyperparameter that determines how much decay is used. The losses i) and ii) are defined

respectively based on the 1-step and n-step lookahead versions of the Bellman equation,

$$\tilde{Q}^1_\omega(s_t, a_t) \coloneqq r_\varphi(s_t, a_t) + \gamma Q_\omega(s_{t+1}, \mu_\theta(s_{t+1})) \tag{2.3}$$

$$\tilde{Q}^n_\omega(s_t, a_t) \coloneqq \sum_{k=0}^{n-1} \gamma^k r_\varphi(s_{t+k}, a_{t+k}) + \gamma^n Q_\omega(s_{t+n}, \mu_\theta(s_{t+n})) \tag{2.4}$$

yielding the critic losses:

$$\ell_1(\omega) \coloneqq \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta}[(\tilde{Q}^1_\omega - Q_\omega)^2(s_t, a_t)] \tag{2.5}$$

$$\ell_n(\omega) \coloneqq \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta}[(\tilde{Q}^n_\omega - Q_\omega)^2(s_t, a_t)] \tag{2.6}$$

where $\mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta}[\cdot]$ signifies that transitions are sampled from the replay buffer $\mathcal{R}$, using in effect the off-policy distribution β. Both Q_ω and $\tilde{Q}^{\cdot}_\omega$ ((2.3), (2.4)) depend on ω, which might cause severe instability. In order to prevent the critic from diverging, we use separate *target* networks for both policy and critic (θ', ω') to calculate $\tilde{Q}^{\cdot}_\omega$, which slowly track the learned parameters (θ, ω). In line with results exhibited in the recent ablation study (Rainbow [HMvH+17]) assessing the influence of the various add-ons of DQN [MKS+13, MKS+15] on its performance, we studied the influence of two add-ons that were transposable to SAM: longer TD backups and replay prioritisation. n-step returns not only played a significant role in improving the sample complexity, but also had a positive influence on stability in the training regime. Prioritized Experience Replay [SQAS16] however prevented SAM from consistently learning well-behaved policies. Being already prone to overfitting in its original setting [SQAS16], we conjecture this phenomenon is amplified in our setting since the TD-errors, instrumental in the priority assignments, depend on rewards that are themselves learned. Uniform experience replay offers greater resilience against oversampling transitions that have wrongfully been assigned high synthetic rewards by the adversarially-trained reward module.

2.4.3 Policy

We update the policy μ_θ so as to maximise the *performance objective*, defined as the expected return from the start state. To that end, the policy is updated by taking a gradient ascent step along:

$$\nabla^{(1)}_\theta J(\mu_\theta) \approx \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_\theta Q_\omega(s_t, \mu_\theta(s_t))\right] \tag{2.7}$$

$$= \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_\theta \mu_\theta(s_t) \nabla_a Q_\omega(s_t, a)|_{a=\mu_\theta(s_t)}\right] \tag{2.8}$$

where the partial derivative with respect to the state is ignored since we consider the model-free setting. This gradient estimation stems from the policy gradient theorem proved by [SLH+14], and points towards regions of the parameter space in which the policy displays high similarity with the demon-

strator. We model the synthetic reward as a parametrised function that takes a state and an action as inputs. As such, we can take the derivative of the reward with respect to θ. By applying the chain rule, we obtain:

$$\nabla_\theta^{(2)} J(\mu_\theta) \approx \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_\theta r_\varphi(s_t, \mu_\theta(s_t)) \right] \tag{2.9}$$

$$= \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_\theta \mu_\theta(s_t) \nabla_a r_\varphi(s_t, a)|_{a=\mu_\theta(s_t)} \right] \tag{2.10}$$

which constitutes another estimate of how to update the policy parameters θ to increase the similarity between the policy and the expert ([SYK19] employs a similar estimate).

Each estimate of how well the agent is behaving, r_φ and Q_ω, is trained via a different policy evaluation method, each presenting its own advantages. The first is updated by adversarial training, providing an accurate estimate of the immediate similarity with expert trajectories. The second is trained via TD learning, enabling longer propagation of rewards along trajectories and effectively tackling the credit assignment problem. While our formulation enables us to use either of these gradient estimates, $\nabla_\theta^{(1)} J(\mu_\theta)$ is more suited to learn control policies in environments inducing delayed rewards. As the continuous control tasks we consider in this paper belong to this category, we use $\nabla_\theta^{(1)} J(\mu_\theta)$ to update the policy module. While we could use a mixture of $\nabla_\theta^{(1)} J(\mu_\theta)$ and $\nabla_\theta^{(2)} J(\mu_\theta)$, we found that the latter had a detrimental effect on the former, as it prevented the policy to reason across timesteps, resulting in poor reward propagation.

2.4.4 Exploration

Deterministic policies have zero variance in their predictions for a given state, translating to no exploratory behaviour. The exploration problem is therefore treated independently from how the policy is modelled, by defining a stochastic policy π_θ from the learned deterministic policy μ_θ. In this work, we construct π_θ via the combination of two fundamentally different techniques: *a)* by applying an adaptive perturbation to the learned weights θ (exploration by noise-injection in parameter space [PHD+18, FAP+17]) and *b)* by adding temporally-correlated noise sampled from a Ornstein-Uhlenbeck process $\mathfrak{N}_{OU}$ [LHP+16], well-suited for control tasks involving inertia (*e.g.* locomotion tasks, and simulated robotics in general). We denote the obtained policy by $\pi_\theta := \mu_{\tilde{\theta}} + \mathfrak{N}_{OU}$, where $\tilde{\theta}$ results from applying a) to θ. When interacting with the environment, SAM samples from the conditional distribution π_θ, and stores the collected transitions in the replay buffer $\mathcal{R}$. An interesting result is that the reward is adversarially trained on samples coming from the parameter-perturbed policy. Rather than causing severe divergence, it seems that it positively impacts the adversarial training procedure. This observation directly echoes noise-injection techniques from the GAN literature. The additive noise applied to the output of our policy (which plays the role of generator in our ar-

chitecture) aligns with [AB17] who add artificial noise to the inputs of the discriminator (although we do not perturb expert trajectories). Furthermore, perturbing μ_θ in parameter space draws strong similarities with [ZML17], in which the authors add Gaussian noise to the layers of the generator.

2.5 Results

Our agents were trained in physics-based control environments, built with the MuJoCo physics engine [TET12], and wrapped via the OpenAI Gym [BCP+16] API. Tasks simulated in the environments range from legacy balance-oriented tasks to simulated robotics and locomotion tasks of various complexities. In this work, we consider the 5 following environments, ordered by increasing complexity (aligned with their respective degrees of freedom in state and action spaces, *cf.* Table 2.A.1 in Section 2.A): `InvertedPendulum`, `InvertedDoublePendulum`, `Reacher`, `Hopper`, `Walker2d`. In the experiments presented in Figure 2.5, we explore how the performance of SAM and that of GAIL evolve as a function of the number of interactions they have with the environment.

For each environment, an expert was designed by training an agent for 10M timesteps using the Proximal Policy Optimization (PPO) algorithm [SWD+17] (with Generalized Advantaged Estimation, or GAE [SML+16]). The episode horizon (maximum episode length) was left to its default value per environment. We created a dataset of expert trajectories per environment. For every environment, we evaluated the performance of the agents when provided with various quantities of demonstrations, sampled for the demonstration dataset associated with the environment. We do so in order to explore how the two methods behave with respect to the number of demonstrations to which they are exposed. Both models are shown the same set of selected trajectories. We ensure that the two compared models are trained on exactly the same subset of extracted trajectories by training them with the same random seeds. We varied the cardinality of the set of selected trajectories as a function of the environment's complexity. We ran every experiment on the same range of 4 random seeds, namely $\{0, 1, 2, 3\}$. In Figure 2.5, we use scatter plots to visualise every episodic return, for every random seed. Solid blue and green lines represent the mean episodic return across the random seeds for the given number of interactions. The filled areas are confidence intervals around the solid lines, corresponding to a fixed fraction of the standard deviation around the mean for the given number of interactions. Every item coloured in red relates to the expert performance, for a given environment. The solid red line corresponds to the mean episodic return of the demonstrations present in the expert dataset associated with the given environment. The filled red region is a trust region whose width is equal to the standard deviation of returns in the expert dataset. The dotted line depicts the minimum return in the demonstration dataset while the dashed line represents the maximum. Having statistics about the demonstration datasets is particularly insightful when evaluating the results of experiments dealing

Algorithm 2: Sample-efficient Adversarial Mimic

Initialise replay buffer $\mathcal{R}$
Initialise network parameters $(\varphi, \theta, \omega)$
Initialise target network parameters (θ', ω') as respective copies of (θ, ω)
for $i \in 1, \ldots, i_{max}$ **do**
    `# Interact with environment`
    `# and store collected transitions`
    for $c \in 1, \ldots, c_{max}$ **do**
        Interact with environment following π_θ and collect the experienced transitions in C augmented with synthetic rewards
        Store C in the replay buffer $\mathcal{R}$
    end
    for $t \in 1, \ldots, t_{max}$ **do**
        `# Update reward module`
        for $d \in 1, \ldots, d_{max}$ **do**
            Sample uniformly a minibatch $\mathcal{B}^c$ of state-action pairs pairs from C
            Sample uniformly a minibatch $\mathcal{B}_e^c$ of state-action pairs from the expert dataset τ_e, with $|\mathcal{B}^c| = |\mathcal{B}_e^c|$
            Update synthetic reward parameter φ with the equal mixture $\mathcal{B}^c \cup \mathcal{B}_e^c$ by following the gradient:
            $\hat{\mathbb{E}}_{\mathcal{B}^c}[\nabla_\varphi \log(1 - D_\varphi(s,a))] + \hat{\mathbb{E}}_{\mathcal{B}_e^c}[\nabla_\varphi \log D_\varphi(s,a)] + \lambda \nabla_\varphi \mathfrak{R}_{\mathrm{GP}}(\varphi)$
            Sample uniformly a minibatch $\mathcal{B}^d$ of state-action pairs from $\mathcal{R}$
            Sample uniformly a minibatch $\mathcal{B}_e^d$ of state-action pairs from the expert dataset τ_e, with $|\mathcal{B}^d| = |\mathcal{B}_e^d|$
            Update synthetic reward parameter φ with the equal mixture $\mathcal{B}^d \cup \mathcal{B}_e^d$ by following the gradient:
            $\hat{\mathbb{E}}_{\mathcal{B}^d}[\nabla_\varphi \log(1 - D_\varphi(s,a))] + \hat{\mathbb{E}}_{\mathcal{B}_e^d}[\nabla_\varphi \log D_\varphi(s,a)] + \lambda \nabla_\varphi \mathfrak{R}_{\mathrm{GP}}(\varphi)$
        end
        `# Update policy and critic modules`
        for $g \in 1, \ldots, g_{max}$ **do**
            Sample uniformly a minibatch $\mathcal{B}^g$ of transitions from $\mathcal{R}$
            Update policy parameter θ by following the gradient: $\hat{\mathbb{E}}_{\mathcal{B}^g}[\nabla_\theta J(\mu_\theta)]$
            Update critic parameters ω by minimizing critic loss: $\hat{\mathbb{E}}_{\mathcal{B}^g}[\ell(\omega)]$
            Update target network parameters (θ', ω') to slowly track (θ, ω), respectively
        end
    end
end

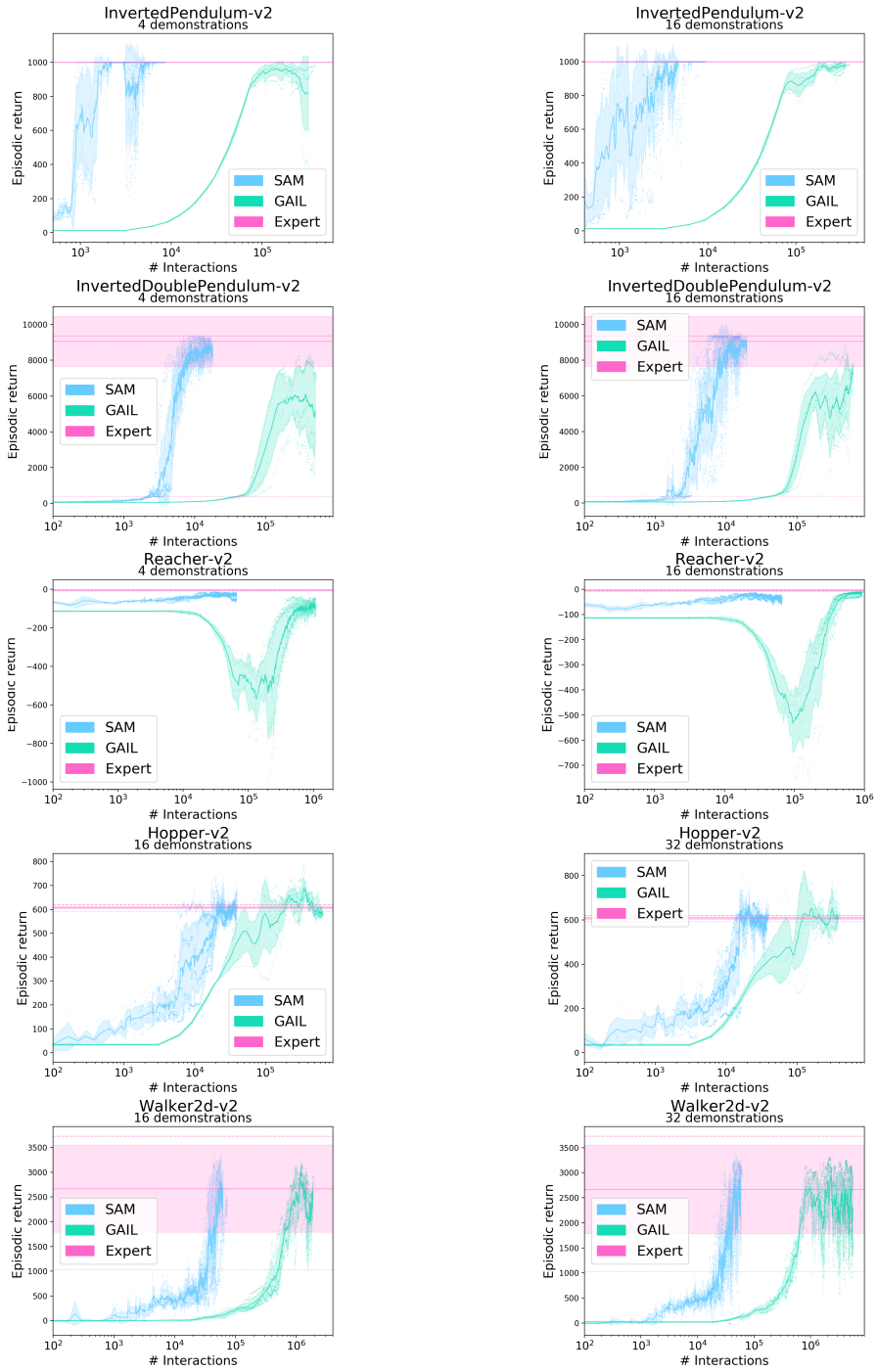


Figure 2.5: Performance comparison between SAM and GAIL in terms of episodic return. The horizontal axis depicts, in logarithmic scale, the number of interactions with the environment. While there is no ambiguity for GAIL, we used the unperturbed SAM policy μ_θ (without parameter noise and additive action noise) to collect those returns during a per-iteration evaluation phase. The figure shows that our method has a considerably better sample-efficiency than GAIL in various continuous control tasks, often by several orders of magnitude. Red-colored lines and filled areas indicate the performance range of the expert demonstrations present in the training set. The meaning of the different line styles and colors is given in-text.

with few demonstrations.

Every experiment runs with 4 parallel instantiations of the same model, initialised with different seeds. Each instantiation has its own interaction with the environment, its own replay buffer and its own optimisers. However, every iteration, the gradients are averaged per module across instantiations and the averaged gradients are distributed per module to every instantiation and immediately used to update the respective module parameters. Both SAM and GAIL experiments were run under this setting. This vertical scalability played a considerable role in speeding up training phases, equivalently for both models. Since every instantiation has its own random seed, the fairness of our performance comparison between SAM and GAIL is further strengthened [DCH+16, HIB+18]. We used layer normalisation [BKH16] in the policy module. Indeed, applying layer normalisation to every layer of the policy was instrumental in yielding better results, in line with the observations reported in [PHD+18]. To ensure symmetry within the actor-critic architecture, we also applied layer normalisation to the critic module. Pop-Art [vHGH+16] was also useful to our architecture as our learned reward would sometimes output scores of various magnitudes. Applying Pop-Art helped in overcoming the various scales.

Finally, note, *SAM and GAIL implementations use exactly the same discriminator implementation*.

We provide architecture, hyperparameter, implementation, and other details in the supplementary material. We also provide video visualisations at `https://youtu.be/-nCsqUJnRKU`, as well as the code associated with this work at `https://github.com/lionelblonde/sam-tf`.

The sample-efficiency we gain over GAIL is considerable: SAM needs one or two orders of magnitude less interactions with the environment to attain asymptotic expert performance. Note that the horizontal axis is scaled logarithmically. Additionally, we observe in Figure 2.5 that GAIL agents sometimes fall short of reaching the demonstrator's asymptotic performance (*e.g.* `Reacher` and `InverseDoublePendulum`). While GAIL requires full traces of agent–environment interaction per iteration as it relies on Monte-Carlo estimates, SAM only requires a couple of transitions per iteration since it performs policy evaluation via temporal-difference learning. Instead of sampling transitions from the environment, performing an update and discarding the transitions, SAM keeps experiential data in memory and can therefore leverage decorrelated transitions collected in previous iterations to perform an off-policy update. Our method therefore requires considerably fewer new samples (interactions) per iteration, as it can re-exploit the transitions previously experienced.

Since our approach trades interactions with the environment with replays with past experiences to extract more knowledge out of past interactions, echoing fictitious play in game theory, it generally takes a longer wall-clock time to train imitation policies. However, in real-world scenarios (*e.g.* robotic

manipulation, autonomous cars), reducing the required interaction with the world is significantly more desirable, for safety and cost reasons.

2.6 Conclusion

In this work, we introduced a method, called *Sample-efficient Adversarial Imitation Learning* (SAM), that meaningfully overcomes one considerable drawback of GAIL [HE16]: the number of interactions between agent and environment that it requires to learn expert-like policies. We demonstrate that our method shrinks the number of interactions by an order of magnitude, and sometimes more. Leveraging an off-policy procedure was key to that success.

Appendix

2.A Studied environments

The environments we dealt with were provided through the OpenAI Gym [BCP+16] API, building on the MuJoCo physics engine [TET12], to model physical interactive scenarios between an agent and the environment she is thrown into. The control tasks modelled by the environments involve locomotion tasks as well as tasks in which the agent must reach and remain in a state of dynamic balance.

Environment	s DoFs	a DoFs
InvertedPendulum-v2	4	1
InvertedDoublePendulum-v2	11	1
Reacher-v2	11	2
Hopper-v2	11	3
Walker2d-v2	17	6

Figure 2.A.1: Degrees of freedom (DoF) of the considered MuJoCo simulated environments. DoFs of both continuous action and state spaces are presented, for the studied physical control tasks. Actions spaces are bounded along every dimension, while the state spaces are unbounded.

2.B Reward function variants

The reward is defined as the negative of the generator loss. As for the latter, the former can be stated in two variants, the saturating version and the non-saturating version, respectively

$$r_{\phi}^{\frown}(s_t, a_t) = -\log\big(1 - D_{\phi}(s_t, a_t)\big) \tag{2.11}$$

$$r_{\phi}^{\smile}(s_t, a_t) = \log D_{\phi}(s_t, a_t) \tag{2.12}$$

The non-saturating alternative is recommended in the original GAN paper as well as in [FRL+18] more recently, as the generator loss suffers from vanishing gradients only in areas where the generated samples are already close to the real data. GAIL relies on policy optimization to update the generator, which makes this vanishing gradient argument vacuous. Besides, in the context of simulated locomotion environments, the saturated version proved to prevail in our experiments, as our agents were unable to overcome the extremely low rewards incurred early in training when using the non-saturating rewards. With the saturated version, signals obtained in early failure cases were close to zero, which was more numerically forgiving for our agents to kick off.

2.C Experimental setup

Both our algorithm and GAIL baseline model implement the MPI interface: each experiment has been launched concurrently with X parallel workers (each with its own random seed), each having its own interaction with the environment, its own replay buffer, its own optimisers and its own network updates. However, every iteration and for a given network, the gradients of the X Adam [KB14] optimisers are pulled together, averaged, and a unique average gradient is distributed to the worker for immediate usage. In the experiments reported in this paper, we used 4 parallel workers.

Our experiments have all been conducted on a single 16-core CPU workstation (reference: AMD Ryzen Threadripper® 1950X CPU).

2.D Hyperparameters settings

In our training procedure, we adopted an alternating scheme consisting in performing 3 training iterations of the actor-critic architecture for one training iteration of the synthetic reward, in line with common practices in the GAN literature (the actor-critic acts as generator, while the synthetic reward plays the role of discriminator). This training pattern applies for both the GAIL baseline and our algorithm, SAM.

As supported by the ending discussion of the GAIL paper, performing a behavioral cloning [Pom89, Pom90] pre-training step to warm-start GAIL can potentially yield expert-like policies in fewer number of ensuing GAIL training iterations. It is especially appealing in so far as the behavioral cloning agent does not interact with the environment at all while training. We therefore intended to precede the training of our experiments (for GAIL and SAM) with a behavioral cloning pre-training phase. However, although the previous training pipeline enables a reduction of training iterations for GAIL, we did not witness a consistent benefit for SAM in our preliminary experiments. Our proposed explanation of this phenomenon is that by pre-training both policy and critic individually as regression problems over the expert demonstrations dataset, we hinder the entanglement of the policy and critic training procedures exploited in SAM. We believe that by adopting a more elaborate pre-training procedure, we will be able to overcome this issue, and therefore leave further exploration for future work.

2.D.1 Generative Adversarial Imitation Learning

2.D.2 Sample-efficient Adversarial Mimic

Hyperparameter	Value
number of MPI workers	4
policy number of layers	2
policy layer widths	(100, 100)
policy hidden activations	tanh
discriminator number of layers	2
discriminator layer widths	(100, 100)
discriminator hidden activations	leaky ReLU
discount factor γ	0.995
generator training steps	3
discriminator training steps	1
non-saturating reward?	false
entropy regularization coefficient λ	0.
gradient penalty [GAA+17] coefficient	10.
one-sided label smoothing	true
number of interactions per iteration	1024
minibatch size	128
normalize observations?	true

Figure 2.D.1: Hyperparameters used to train GAIL agents.

Hyperparameter	Value
number of MPI workers	4
policy number of layers	2
policy layer widths	(64, 64)
policy hidden activations	leaky ReLU
policy layer normalisation [BKH16]	true
policy output activation	tanh
critic number of layers	2
critic layer widths	(64, 64)
critic hidden activations	leaky ReLU
critic layer normalisation	true
discriminator number of layers	2
discriminator layer widths	(64, 64)
discriminator hidden activations	leaky ReLU
discount factor γ	0.99
generator training steps	3
discriminator training steps	1
non-saturating reward?	false
entropy regularization coefficient	0.
gradient penalty [GAA$^+$17] coefficient	10.
one-sided label smoothing	true
number of interactions per iteration	4
minibatch size	32
number of training steps per iteration	20
replay buffer size	100K
normalise observations?	true
normalise returns?	true
Pop-Art [vHGH$^+$16]?	true
reward scaling factor	1.
critic weight decay coefficient ν	0.001
critic 1-step TD loss coefficient	1
critic n-step TD loss coefficient	1
TD lookahead length n	96
adaptive parameter noise for π_θ	0.2
Ornstein-Uhlenbeck additive noise	0.2

Figure 2.D.2: Hyperparameters used to train SAM agents.

Chapter 3

Lipschitzness Is All You Need To Tame Off-policy Generative Adversarial Imitation Learning

Abstract

Despite the recent success of reinforcement learning in various domains, these approaches remain, for the most part, deterringly sensitive to hyper-parameters and are often riddled with essential engineering feats allowing their success. We consider the case of off-policy generative adversarial imitation learning, and perform an in-depth review, qualitative and quantitative, of the method. We show that forcing the learned reward function to be local Lipschitz-continuous is a *sine qua non* condition for the method to perform well. We then study the effects of this necessary condition and provide several theoretical results involving the local Lipschitzness of the state-value function. We complement these guarantees with empirical evidence attesting to the strong positive effect that the consistent satisfaction of the Lipschitzness constraint on the reward has on imitation performance. Finally, we tackle a generic pessimistic reward preconditioning add-on spawning a large class of reward shaping methods, which makes the base method it is plugged into provably more robust, as shown in several additional theoretical guarantees. We then discuss these through a fine-grained lens and share our insights.

Crucially, the guarantees derived and reported in this work are valid for *any* reward satisfying the Lipschitzness condition, nothing is specific to imitation. As such, these may be of independent interest.

3.1 Introduction

Imitation learning (IL) [Bag15] sets out to design artificial agents able to adopt a behavior demonstrated via a set of expert-generated trajectories. Also referred to as *"teaching by showing"* [Sch97], IL can replace tedious tasks such as manual hard-coded agent programming, or hand-crafted reward design *"reward shaping"* [NHR99] for the agent to be trained via reinforcement learning (RL) [SB98]. Besides, in contrast with the latter, imitation learning does not necessarily involve agent-environment interactions. This feature is particularly appealing in real-world domains such as robotics [AS97, Sch97, RBS07, BCDS08], where the artificial agent is physically implemented with expensive hardware, and the environment contains enough external entities (*e.g.* humans, other artificial agents, other costly devices) to raise safety concerns [HXT+20, KZLA16, RAA19, HMZ+17]. When controls are provided in the demonstrations (or recovered via inverse dynamics from the available kinematics [HS17]), we can treat said controls as regression targets, and learn a mimicking policy with a simple, supervised approach. This interaction-free approach (simulated or physical, real-world interactions), called *behavioral cloning* (BC), has enabled the success of various endeavors in robotic manipulation and locomotion [RBS07, WMR+17], in autonomous driving — with the first self-driving vehicle [Pom89, Pom90] thirty years ago and more recently with [GLDS20] using Waymo's open dataset [SKD+19] — and also in grand challenges like AlphaGo [SHM+16] and AlphaStar [VBC+19]. Due to its conceptual simplicity, we expect BC to still be a part of the pipeline for the most ambitious enterprises going forward, especially as open datasets get slowly released.

Despite its practical advantages, BC is extremely data-hungry *w.r.t.* the amount of expert demonstrations it needs to yield robust, high-fidelity policies. Besides, unless corrective behavior is present in the dataset (*e.g.* in autonomous driving, how to drive back onto the road), the policy learned via BC will not be able to internalize this behavior. Once in a situation from which it can not recover, there will be a permanent *covariate shift* between its current observations and the demonstrated ones. The controls learned in a supervised manner on the expert dataset are therefore useless, due to the distributional shift. As a result, the agent's errors will compound, a phenomenon coined by [RB10] as *compounding errors*. In Section 3.6.2.3, we stress how the latter echoes the *compounding variations* phenomenon, exhibited as part of the theoretical contributions of this work. To address the shortcomings of BC, [AN04] proposes to harness the innate credit assignment [SB98] capabilities of RL, by first trying to learn the cost function underlying the demonstrated behavior (inverse RL [NR00]), before using this cost to optimize a policy via RL. The succession of inverse RL and RL is called apprenticeship learning (AL) [AN04], and can, by design, yield policies that can recover from out-of-distribution situations thanks to RL's built-in temporal abstraction mechanisms. Cost learning however is incredibly tedious, and successful approaches end up requiring coarse relaxations to avoid being deterringly

computationally-expensive [AN04, SS08, SBS08, HGE16]. Ultimately, as noted by [ZMBD08], setting out to recovering the cost signal under which the expert demonstrations are optimal (base assumption of inverse RL) is an ill-posed objective — echoing the reward shaping considerations from [NHR99]. In line with this statement, generative adversarial imitation learning (GAIL) [HE16] departs from the typical AL pipeline, and replaces learning the optimal cost ("optimal" in the inverse RL sense) by learning a *surrogate* cost function. GAIL does so by leveraging generative adversarial networks [GPAM$^+$14], as the name hints. The method is described in greater detail in Section 3.3. Due to the RL step it involves (like any AL method), GAIL suffers from poor sample-efficiency *w.r.t.* the amount of interactions it needs to perform with the environment. This caveat has since been addressed, notably by transposition to the off-policy setting, concurrently in SAM [BK19] and DAC [KAD$^+$19] (*cf.* Section 3.4). Both adversarial IL methods ([BK19, KAD$^+$19]) leverage actor-critic architectures, consequently suffering from a greater exposure to instabilities. These weaknesses are mitigated with various complementary techniques, and cautious hyper-parameter tuning.

In this work, we set out to first conduct a thorough investigation into off-policy generative adversarial imitation learning, to pinpoint which are the techniques that are instrumental in performing well, and shed light over which are ones that can be discarded or disregarded without decrease in performance. Ultimately, we would like to exhibit the techniques that are *sufficient* for the method to achieve peak performance. Virtually every algorithmic design choice made in this work is supported by an ablation study reported in the Appendix. We start by describing the base off-policy adversarial imitation learning method at the core of this work in Section 3.4. We then undertake diagnoses of the various issues that arise from the combination of bilevel optimization problems at the core of the investigated model in Section 3.5. A key contribution of our work consists in showing that enforcing a Lipschitzness constraint on the learned surrogate reward is a *necessary* condition for the method to even learn anything — in our consumer-grade, computationally affordable hardware setting. We study it closely, providing empirical evidence of the importance of this constraint through detailed ablation results in Section 3.5.5. We follow up on this empirical evidence with theoretical results in Section 3.6.1, characterizing the Lipschitzness of the state-action value function under said reward Lipschitzness condition, and discuss the obtained variation bounds subsequently. Crucially, we show that without variation bounds on the reward, a phenomenon we call *compounding variations* can cause the variations of the state-action value to explode. As such, the theoretical results reported in Section 3.6.1 — and discussed in Section 3.6.2 — corroborate the empirical evidence exhibited in Section 3.5.5. ***Note, the theoretical results reported in this work are valid for any reward satisfying the condition, nothing is specific to imitation.*** The theoretically-grounded Lipschitzness condition, implemented as a gradient penalty, is in practice a *local* Lipschitzness condition. We therefore investigate *where* (*i.e.* on which samples, on which input distribution) the local

Lipschitzness regularization should be enforced. We propose a new interpretation of the regularization scheme through an RL perspective, make an intuitively grounded claim on where to force the constraint to get the best results, and corroborate our claim empirically (*cf.* Section 3.6.3). Crucially, we show that the consistent satisfaction of the Lipschitzness constraint on the reward in a strong predictor of how well the mimicking agent performs empirically (*cf.* Section 3.6.4). Finally, we introduce a generic pessimistic reward preconditioning add-on technique which makes the base method it is plugged into provably more robust, as attested by its companion theoretical guarantees (*cf.* Section 3.6.5). ***Again, these guarantees are not not specific to imitation and may be of independent interest.*** Among the reported insights, we give an illustrative example of how the add-on technique can help further increasing the robustness of the method it is plugged into empirically.

3.2 Related work

Off-policy generative adversarial imitation learning, which is the object of this work, involves learning a parametric surrogate reward function, from expert demonstrations. By design [HE16, BK19, KAD+19], this signal is learned at the same time as the policy, and as such is prone to non-stationarities (*cf.* Section 3.5.2). This reward regime is reminiscent of *reward corruption* [EKO+17, RHP+18], which posits that the real-world rewards are imperfect (*e.g.* uncontrolled task specification change, sensor defects, reward hacking) and must therefore be treated as such, *i.e.* non-stationary at the very least. Despite being learned and therefore liable to non-stationary behavior, our reward is internal — as opposed to outside the agent's and practitioner's scope — and is therefore fully observable, as well as controllable via the practitioner-specified algorithmic design. The reward corruption can consequently be acted upon, and more easily mitigated than if it originated from a *black box* reward originating from the unknown environment.

The demonstrations on the other hand are available from the very beginning, and do not change as the policy learns. In that respect, our approach differs from *observational learning* [BPMP17], where the policy learns to imitate another by *observing* it itself learn in the environment — and therefore does not strictly qualify as an expert at the task. Observational learning draws clear parallels with the teacher-student scheme in policy distillation [RCG+15]. While our reward is changing since the policy changes and due to the inherent learning dynamics of function approximators, in observational learning, the reward is changing also due to the expert still learning, causing a distributional drift.

Multi-armed bandits [Rob52] have received a lot of attention in recent years to formalize and model problems of sequential decision making under uncertainty. In the context of this work, the most appropriate variants of bandits are *stateful* contextual multi-armed bandits. As the name hints, such models formalize decision making specific to given situations (*i.e.* contexts, states), in which the sit-

uations are *i.i.d.*-sampled. We consider the case of reinforcement learning, where the situations are entangled, along with the decisions themselves, in a Markov decision process (*cf.* Section 3.3). In particular, non-stationary reward channels in Markov decision processes have been studied extensively (*cf.* Section 3.5.2). Among these, adversarial bandits [ACBFS95] can be seen as the archetype or worst-case reward corruption scenario, in which an adversary — possibly driven by malevolent intents — decides on the reward given to the agent. In these models, the common way to deal with non-stationary reward processes is to assume the reward variations in time are upper-bounded, either per-decision or over longer time periods. We give a comprehensive account of sequential decision making under uncertainty in non-stationary Markov decision processes in Appendix 3.B. By contrast, our theoretical guarantees are built on the premise that the reward function's variations are bounded *over the input space* by assuming the reward function is locally Lipschitz-continuous over it. We make the same assumption on the dynamics of the multi-stage decision process, as well as on the control policy. In addition, while our theoretical results ultimately characterize the value function's robustness in terms of Lipschitz-continuity starting from these assumptions, [FMWE10, FMWE13] propose an estimator of the expected return and derive bounds on its bias and variance under similar Lipschitz-continuity assumptions on the policy, dynamics, and reward function. Derived in the offline RL setting, their bounds increase as the *"dispersion"* of the offline dataset increases, which echo our findings (*cf.* Section 3.6.2).

Several works have recently attempted to address the overfitting problem GAIL suffers from. This is due to the discriminator being able to trivially distinguish agent-generated samples from expert-generated ones, which occurs when the learning dynamics of the adversarial game are not properly balanced. As such, the gist of said techniques is to either weaken the discriminator directly or make its classification task harder, which unsurprisingly exactly coincides with the typical techniques used to cope with overfitting in (binary) classification. These techniques are, in no particular order: reducing the discriminator's capacity — by plugging the classifier on top of an independent perception stack (*e.g.* random features, state-action value convolutional layers) [RAW+18], smoothing the positive labels with uniform random noise [BK19], adopting a positive-unlabeled classification objective (instead of the traditional positive-negative one) [XD19], using a gradient penalty (originally from [GAA+17]) regularizer [BK19, KAD+19], leveraging an adaptive information bottleneck in the discriminator network [PKT+18], enriching the expert dataset via task-specific data augmentation [ZRN+19]. In this work, we do not propose a new regularization technique. Instead, we perform an in-depth analysis of the simplest techniques — in terms of conceptual simplicity, implementation time, number of parameters, and computational cost [HB20], and ultimately find that the gradient penalty regularizer achieves the best trade-off. Crucially, we show that *not using* gradient penalization performs poorly (*cf.* Section 3.5.5).

In [HR11], Hafner and Riedmiller advocate for the use of a *smooth* reward signal in RL. [LGR12] presents it as one key method to make learning values in offline RL less tedious. Sharp changes in reward value are hard to represent and internalize by the action-value neural function approximator. Using a smooth reward surrogate derived from the original *"jumpy"* reward signal such that the trends are preserved but the crispness is attenuated proved instrumental empirically. Our observation about reward Lipschitz-continuity being a crucial component of our off-policy imitation learning pipeline is in line with the suggestion of [HR11]. On top of providing empirical evidence of its benefits, we also provide a number of theoretical results characterizing what the reward smoothness does on the value function smoothness.

Finally, we point out that local Lipschitz-continuity conditions are also found in the adversarial robustness literature. Notably, [FCAO18] encourages Lipschitzness via gradient regularization, as is done in our work, From a strict optimization standpoint, [HRS15] derives bounds under a Lipschitz-continuity assumption on the loss.

3.3 Background

Setting

In this work, we address the problem of an agent whose goal is, in the absence of extrinsic reinforcement signal [SLB09], to *imitate* the behavior demonstrated by an expert [Bag15], expressed to the agent via a pool of trajectories. The agent is never told how well she performs or what the optimal actions are, and is not allowed to query the expert for feedback.

World and agent

The intrinsic behavior of the decision maker is represented by the *policy* π_θ, modeled by a neural network with parameter θ, mapping states to probability distributions over actions. Formally, the conditional probability density over actions that the agent concentrates at action a_t in state s_t is denoted by $\pi_\theta(a_t|s_t)$, for all discrete timestep $t \geq 0$. We model the environment the agent interacts with as an infinite-horizon, memoryless, and stationary *Markov Decision Process* (MDP) [Put94] formalized as the tuple $\mathbb{M} := (\mathcal{S}, \mathcal{A}, p, \rho_0, u, \gamma)$. $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ are respectively the state space and action space. p and ρ_0 define the *dynamics* of the world, where $p(s_{t+1}|s_t, a_t)$ denotes the stationary conditional probability density concentrated at the next state s_{t+1} when stochastically transitioning from state s_t upon executing action a_t, and ρ_0 denotes the initial state probability density. u denotes a stationary *reward process* that assigns, to any state-actions pairs, a real-valued reward r_t distributed as $r_t \sim u(\cdot|s_t, a_t)$.

Finally, $\gamma \in [0, 1)$ is the discount factor. We make the MDP *episodic* by positing the existence of an absorbing state in every trace of interaction and enforcing $\gamma = 0$ to formally trigger episode termination once the absorbing state is reached. Since our agent does not receive rewards from the environment, she is in effect interacting with an MDP lacking a reward process r. Our method however encompasses learning a surrogate reward parameterized by a deterministic function approximator such as a neural network with parameter φ, denoted by r_φ, and whose learning procedure will be reported subsequently. Consequently, our agent effectively interacts with the augmentation of the previous MDP defined as $\mathbb{M}^* := (\mathcal{S}, \mathcal{A}, p, \rho_0, r_\varphi, \gamma)$. A *trajectory* τ_θ is a trace of π_θ in $\mathbb{M}^*$, succession of consecutive *transitions* (s_t, a_t, r_t, s_{t+1}), where $r_t := r_\varphi(s_t, a_t)$. A *demonstration* is the set of state-actions pairs (s_t, a_t) extracted from a trajectory collected by the expert policy π_e in $\mathbb{M}$. The *demonstration dataset* $\mathcal{D}$ is a set of demonstrations.

Objective

Building on the reward hypothesis at the core of reinforcement learning (any task can be defined as the maximization of a reward), to act optimally, our agents must be able to deal with delayed signals and maximize the long-term cumulative reward. To address credit assignment, we use the concept of *return*, the discounted sum of rewards from timestep t onwards, defined as $R_t^\gamma := \sum_{k=0}^{+\infty} \gamma^k r_{t+k} := \sum_{k=0}^{+\infty} \gamma^k r_\varphi(s_{t+k}, a_{t+k})$ in the infinite-horizon regime. By taking the expectation of the return with respect to all the future states and actions in $\mathbb{M}^*$, after selecting a_t in s_t and following π_θ thereafter, we obtain the state-action value (Q-value) of the policy π_θ at (s_t, a_t):

$$Q^{\pi_\theta}(s_t, a_t) := \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t, a_t), a_{t+1} \sim \pi_\theta(\cdot|s_{t+1}), \dots}[R_t^\gamma] \tag{3.1}$$

(*abbrv.* $\mathbb{E}_{\pi_\theta}^{\geq t}[R_t^\gamma]$). At state s_t, a policy π_θ that picks a_t verifying:

$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}} Q^{\pi_\theta}(s_t, a)$$

therefore acts optimally looking onwards from s_t. Ultimately, an agent acting optimally at all times maximizes $V^{\pi_\theta}(s_0) := \mathbb{E}_{a_0 \sim \pi_\theta(\cdot|s_0)}[Q^{\pi_\theta}(s_0, a_0)]$ for any given start state $s_0 \sim \rho_0$. *In fine*, we can now define the *utility function* (also called *performance objective* [SLH+14]) to which our agent's policy π_θ must be solution of: $\pi_\theta = \operatorname{argmax}_{\pi \in \Pi} U_0(\pi)$ where $U_t(\pi) := V^\pi(s_t)$ and Π is the search space of parametric function approximators, *i.e.* deep neural networks.

Generative Adversarial Imitation Learning

GAIL [HE16] trains a binary classifier D_φ, called *discriminator*, where samples from π_e are positive-labeled, and those from π_θ are negative-labeled. It borrows its name from *Generative Adversarial Networks* [GPAM+14]: the policy π_θ plays the role of generator and is optimized to fool the discriminator D_φ into classifying its generated samples (negatives), as positives. As such, the prediction value indicates to what extent D_φ believes π_θ's generations are coming from the expert, and therefore constitutes a good measure of mimicking success. GAIL does not try to recover the reward function that underlies the expert's behavior. Rather, it learns a similarity measure between π_e and π_θ, and uses it as a *surrogate* reward function. We say that π_θ and D_φ are *"trained adversarially"* to denote the two-player game they are intricately tied in: D_φ is trained to assert with confidence whether a sample has been generated by π_θ, while π_θ receives increasingly greater rewards as D_φ's confidence in said assertion lowers. *In fine*, the surrogate reward measures the confusion of D_φ. In this work, the neural network function approximator modeling D_φ uses a sigmoid as output layer activation, *i.e.* $D_\varphi \in [0, 1]$. The exact zero case is bypassed numerically for $\log \circ D_\varphi$ to always exist, by adding an infinitesimal value $\varepsilon > 0$ to D_φ inside the logarithm. The same numerical stability trick is used for $\log \circ (1 - D_\varphi)$ to avoid the exact one case (*cf.* reward formulations in Section 3.4).

3.4 Comprehensive refresher on the sample-efficient adversarial mimic

Building on TRPO [SLM+15], GAIL [HE16] inherits its policy evaluation subroutine, consisting in learning a parametric estimate of the state-value function $V_\omega \approx V^{\pi_\theta}$ via Monte-Carlo estimation over samples collected by π_θ. While it uses function approximation to estimate V^{π_θ}, hoping it generalizes better that a straight-forward non-parametric Monte-Carlo estimate (discounted sum), we will reserve the term *actor-critic* for architectures in which the state-value $V^{\pi_\theta}(\cdot)$ or Q-value $Q^{\pi_\theta}(\cdot, \cdot)$ is learned via Temporal-Difference (TD) [Sut88]. This terminology choice is adopted from [SB98] (*cf.* Chapter 13.5). A *critic* is used for bootstrapping, as in the TD update rule (whatever the bootstrapping degree is). As such, TRPO is not an actor-critic, while algorithms learning their value via TD, such as DDPG [SLH+14, LHP+16], are actor-critic architectures. Albeit hindered from various weaknesses (*cf.* Section 3.5.1), and forgetting for a moment that it is combined with function approximation [SMSM99, SLH+14], the TD update is able to propagate information quicker as the backups are shorter and therefore do not need to reach episode termination to learn, in contrast with Monte-Carlo estimation. That is without even involving fictitious, memory, or experience replay mechanisms [Lin92]. By design, TD learning is less data-hungry (*w.r.t.* interactions in the environment), and involving replay mechanisms [Lin92, LHP+16, WBH+16] significantly adds on to its inherent

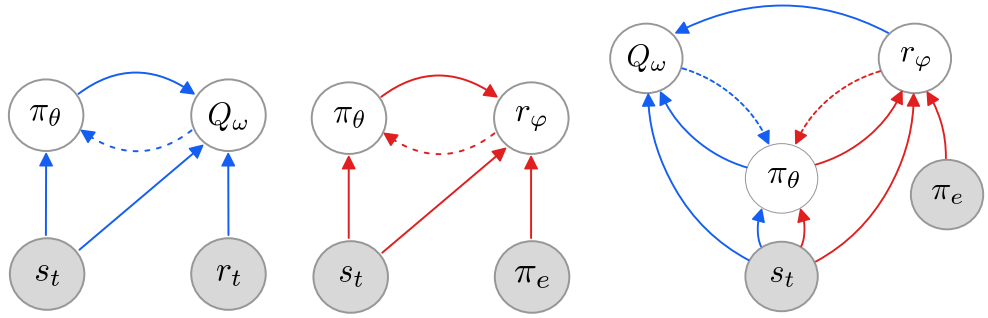
sample-efficiency. Based on this line of reasoning, SAM [BK19] and DAC [KAD+19] addressed the deterring sample-complexity of GAIL by, among other improvements (*cf.* [BK19, KAD+19]), using an actor-critic architecture to replace TRPO for policy evaluation and improvement. SAM [BK19] uses DDPG [LHP+16], whereas DAC [KAD+19] uses TD3 [FvHM18]. Both were released concurrently, and both report significant improvements in sample-efficiency (up to two orders of magnitude). Standing as the stripped-down model that brought sample-efficiency to GAIL, we take SAM as base. Albeit described momentarily in the body of this work, we urge the reader eager the understand every single aspect of the laid out algorithm to also refer to the section in which we describe the experimental setting, *cf.* 3.5.5.

We now lay out the constituents of SAM [BK19], and how their learning procedures are orchestrated. The agent's behavior is dictated by a *deterministic* policy μ_θ, the critic Q_ω assigns Q-values to actions picked by the agent, and the reward r_φ assesses to what degree the agent behaves like the expert. As usual, θ, ω, and φ denote the respective parameters of these neural function approximatiors. To explore when carrying out rollouts in the environment, μ_θ is perturbed both in parameter space by adaptive noise injection in θ [PHD+18, FAP+17], and action space by adding the temporally-correlated response of an Ornstein-Uhlenbeck noise process [UO30, LHP+16] to the action returned by μ_θ. Formally, in state s_t, action a_t is sampled from $\pi_\theta(\cdot|s_t) \coloneqq \mu_{\theta+\varepsilon}(s_t) + \eta_t$, where $\varepsilon \sim \mathcal{N}(0, \sigma_a^2)$ (σ_a adapts conservatively such that $|\mu_{\theta+\varepsilon}(s_t) - \mu_\theta(s_t)|$ remains below a certain threshold), and where η_t is the response of the Ornstein-Uhlenbeck process [UO30] $\mathfrak{N}_{OU}$ at timestep t in the episode, such that $\eta_t \coloneqq \mathfrak{N}_{OU}(t, \sigma_b)$. Note, $\mathfrak{N}_{OU}$ is reset upon episode termination. As a first minor contribution, we carried out an ablation study on exploration strategies, and report the results in Appendix 3.I. While the utility of temporally-correlated noise is somewhat limited to dynamical systems, both parameter noise and input noise injections have proved beneficial in generative modeling with GANs ([ZML17] and [AB17], respectively). As in GAIL [HE16] (described earlier in Section 3.3), the discriminator D_φ is trained via an adversarial training procedure [GPAM+14] against the policy π_θ. The surrogate reward r_φ used to augment MDP $\mathbb{M}$ into $\mathbb{M}^*$ is derived from D_φ to reflect the incentive that the agent needs to complete the task at hand. In the tasks we consider in this work (simulated robotics environments [BCP+16], based on the MuJoCo [TET12] physics engine, and described in Table 3.5.1) an episode terminates either *a)* when the agent *fails* to complete the task according to an task-specific criterion hard-coded in the environment, or *b)* when the agent has performed a number of steps in the environments that exceeds a predefined hard-coded *timeout*, which we left to its default value — with the exception of `HalfCheetah`, in which *a)* does not apply. Due to *a)*, the agent can decide to truncate its return by triggering its own failure, and decide to "cut its losses" when it is penalized too heavily for not succeeding according to the task criterion. Always-negative rewards (*e.g.* per-step "−1" reward to urge to agent to complete the task quickly [Kae93]) can therefore make the

agent give up and trigger termination the earliest possible, as this would maximize its return. On the other hand, always-positive rewards can make the agent content with its sub-optimal actions which would prevent it from pursuing higher rewards, as long as it remains alive. This phenomenon has been dubbed *survival bias* in [KAD+19]. Notably, this discussion highlights the tedious challenge that reward shaping [NHR99] usually represents to practitioners when designing a new task. Stemming from their generator loss counterparts in the GAN literature, the *minimax (saturating)* reward variant is $r_\varphi \coloneqq -\log\big(1 - D_\varphi\big)$, and the *non-saturating* reward variant is $\log\big(D_\varphi\big)$. The minimax reward is always positive, the non-saturating reward is always negative, and the sum of the two can take positive and negative values. We found empirically that using the minimax reward, despite being always positive, yielded by far the best results compared to the sum of the two variants. The performance gap is reduced in the `HalfCheetah` task which was expected since it is the only task in which the agent can not trigger an early termination. We report these comparative results in Appendix 3.F. Crucially, these results show that the base method considered in this work can already successfully mitigate survival bias, without requiring additional reward shaping. In summary, we use the formulation $r_\varphi \coloneqq -\log\big(1 - D_\varphi\big)$, unless stated otherwise explicitly.

We also adopt the mechanism introduced in [KAD+19] that wraps the absorbing transitions (agent-generated *and* expert-generated) to enable the discriminator to distinguish between terminations caused by failure and terminations triggered by the artificially hard-coded timeout. The method enables the discriminator to penalize the agent for terminating by failure when the expert would, with the same action and in the same state, terminate by reaching the episode timeout without failing. In such a scenario, without wrapping the absorbing transitions, the agent perfectly imitates the expert in the eyes of the discriminator, which is not the case. We use the wrapping mechanism in every experiment. Nonetheless, we omit it from the equations and algorithms for legibility. Giving the agent the ability to differentiate between terminations that are due to time limits and those caused by the environment had proved crucial for the decision maker to continue beyond the time limit. The significant role played by the explicit inclusion of the notion of time in RL has been established by Harada in [Har97], yet without much follow-up, until being revived in [PTLK18] where the authors demonstrate that a careful inclusion of the notion of time in RL can meaningfully impact performance.

By assuming the roles of opponents in a GAN, θ and φ are tied in a *bilevel* optimization problem (as highlighted in [PV16]). Similarly, by defining an actor-critic architecture, θ and ω are also tied in a bilevel optimization problem. We notice the dual role of θ, which is intricately tied in both bilevel problems. As such, what SAM [BK19] sets out to solve can be dubbed a *θ-coupled twin bilevel* optimization problem. Note, Q_ω uses the parametric reward r_φ as a scalar detached from the computational graph of the (θ, ω) bilevel problem, as having gradients flow back from Q_ω to φ would prevent D_φ from being learned as intended, *i.e.* adversarially in the (θ, φ) bilevel problem. The information



(a) Actor-Critic [SMSM99] (b) GAIL [HE16] (c) SAM [BK19]

Figure 3.4.1: Information flows (plain arrows) and gradient flows (dotted arrows) between modules. Best seen in color.

and gradient flows occurring between the components are illustrated in FIGURE 3.4.1. As we show via numerous ablation studies in this work, training this *θ-coupled twin bilevel* system to completion is severely prone to instabilities and highly sensitive to hyper-parameters. Ultimately, we show that r_φ's Lipschitzness is a *sine qua non* condition for the method to perform well, and study the effects of this necessary condition in several theoretical results in SECTION 3.6.1.

Sample-efficiency is achieved through the use of a replay mechanism [Lin92]: every component (every neural network, θ, ω, and φ) is trained using samples from the replay buffer $\mathcal{R}$, [MKS+13, MKS+15], a *"first in, first out"* queue of fixed retention window, to which new rollout samples (transitions) are sequentially added, and from which old rollout samples are sequentially removed. Note however that when a transition is sampled from $\mathcal{R}$, its reward component is re-computed using the most recent r_φ update. [BK19] and [KAD+19] were the first to train D_φ with experience replay, in a non-*i.i.d.* — Markovian — context, for increased learning stability. Borrowing the common terminology, the reward is therefore effectively *"learned off-policy"*. Let β be the off-policy distribution that corresponds to uniform sampling over $\mathcal{R}$. β is therefore effectively a mixture of past policy updates $[\theta_{i-\Delta+1}, \dots, \theta_{i-1}, \theta_i]$, where the mixing depends on $\mathcal{R}$'s retention window, and the number of collected samples (rollout size) per iteration.

We first introduce $\rho^\pi_{\mathbb{M}^*}$, which denotes the discounted state visitation frequency of an arbitrary policy π in $\mathbb{M}^*$. Formally, $\rho^\pi_{\mathbb{M}^*}(s) := \sum_{t=0}^{+\infty} \gamma^t \mathbb{P}^\pi_{\mathbb{M}^*}[S_t = s]$, where $\mathbb{P}^\pi_{\mathbb{M}^*}[S_t = s]$ is the probability of reaching state s at timestep t when interacting with the MDP $\mathbb{M}^*$ by acting according to π. Since $\sum_{s \in \mathcal{S}} \rho^\pi_{\mathbb{M}}(s) = 1/(1-\gamma)$, $\rho^\pi_{\mathbb{M}}$ can be seen as a probability distribution over states up to a constant factor. Due to the presence of the discount factor γ, $\rho^\pi_{\mathbb{M}^*}(s)$ has higher value if s is visited earlier than later in the infinite-horizon trajectory. In practice, we relax the definition to its non-discounted counterpart and to the

episodic regime case, as is usually done. Plus, since every interaction is done in MDP $\mathbb{M}^*$, we use the shorthand ρ^π. From this point forward, when states s_t are sampled uniformly from the replay buffer $\mathcal{R}$ — in effect, following policy β — the expectation over said samples will be denoted as $\mathbb{E}_{s_t\sim\rho^\beta}[\cdot]$.

We now go over how each module (θ, ω, and φ) is optimized in this work.

We optimize φ with the binary cross-entropy loss, where positive-labeled samples are from π_e, and negative-labeled samples are from β:

$$\ell_\varphi \coloneqq \mathbb{E}_{s_t\sim\rho^{\pi_e},a_t\sim\pi_e}[-\log(1-D_\varphi(s_t,a_t))] + \mathbb{E}_{s_t\sim\rho^\beta,a_t\sim\beta}[-\log(D_\varphi(s_t,a_t))] \tag{3.2}$$

Unless stated otherwise, φ is regularized with gradient penalization $\mathfrak{R}^\zeta_\varphi(k)$, subsuming the original formulation proposed in [GAA+17], which was used in SAM [BK19] and DAC [KAD+19]:

$$\ell^{\text{GP}}_\varphi \coloneqq \ell_\varphi + \lambda\,\mathfrak{R}^\zeta_\varphi(k) \coloneqq \ell_\varphi + \lambda\,\mathbb{E}_{s_t\sim\rho^\zeta,a_t\sim\zeta}[(\|\nabla_{s_t,a_t} D_\varphi(s_t,a_t)\| - k)^2] \tag{3.3}$$

The regularizer will be the object of several downstream analyses and discussions (*cf.* Sections 3.5.4 and 3.6.3). The meaning of λ, k and ζ will be given in Section 3.5.4.

The critic's parameters ω are updated by gradient decent on the TD loss [Sut88], using the multi-step version [PW96] (*"n-step"*) of the Bellman target (R.H.S. of the expected Bellman equation), which has proven beneficial for policy evaluation [HMvH+17, FHGS19]. The loss optimized by the critic is:

$$\ell_\omega \coloneqq \mathbb{E}_{s_t\sim\rho^\beta,a_t\sim\beta}[(Q_\omega(s_t,a_t) - Q^{\text{targ}})^2] \tag{3.4}$$

where the target Q^{targ} uses *softly*-updated [LHP+16] target networks [MKS+13, MKS+15], θ' and ω', and is defined as:

$$Q^{\text{targ}} \coloneqq \sum_{k=0}^{n-1} \gamma^k r_\varphi(s_{t+k},a_{t+k}) + \gamma^n Q_{\omega'}(s_{t+n},\mu_{\theta'}(s_{t+n})) \qquad \blacktriangleright \textit{ Bellman target} \tag{3.5}$$

$$(\theta',\omega') \leftarrow (1-\tau)(\theta',\omega') + \tau(\theta,\omega) \quad 0\le\tau\le 1 \qquad \blacktriangleright \textit{ target networks update} \tag{3.6}$$

Finally, since μ_θ is deterministic, its utility value at timestep t is $U_t(\mu_\theta) = V^{\mu_\theta}(s_t) = Q^{\mu_\theta}(s_t,\mu_\theta(s_t)) \approx \mathbb{E}_{s_t\sim\rho^\beta}[Q_\omega(s_t,\mu_\theta(s_t))] =: \mathcal{U}_\theta$, where the approximation is due to the actor-critic design involving the use of function approximators. To maximize its utility at t, θ must take a gradient step in the ascending

direction, derived according to the *deterministic policy gradient theorem* [SLH+14]:

$$\nabla_\theta U_t(\mu_\theta) \approx \nabla_\theta \mathcal{U}_\theta \tag{3.7}$$

$$= \nabla_\theta \mathbb{E}_{s_t \sim \rho^\beta}[Q_\omega(s_t, \mu_\theta(s_t))] \tag{3.8}$$

$$= \mathbb{E}_{s_t \sim \rho^\beta}[\nabla_\theta \mu_\theta(s_t) \nabla_a Q_\omega(s_t, a)|_{a=\mu_\theta(s_t)}] \tag{3.9}$$

This last step (EQ 3.9) emerges from the natural assumption that $\forall s \ \nabla_\theta s = 0$, since the analytical form of $\mathbb{M}$'s dynamics, p, is unknown. To overcome the inherent *overestimation bias* [TS93] hindering Q-Learning and actor-critic methods based on greedy action selection (*e.g.* DDPG [LHP+16]), and therefore suffered by our critic Q_ω, we apply the actor-critic counterpart of double-Q learning [vH10] — analogously, Double-DQN [vHGS15] for DQN — proposed in Twin-Delayed DDPG (*abbrv.* TD3) [FvHM18]. This add-on method, simply called *clipped double-Q learning* (*abbrv.* CD), consists in learning an additional (or *"twin"*) critic, and using the smaller of the two associated Q-values in the Bellman target, used in the temporal-difference error of both critics. For its reported benefits at minimal cost, we also use the other main add-on proposed in TD3 [FvHM18] called *target policy smoothing*. The latter adds noise to the target action in order for the deterministic policy not to pick actions with erroneously high Q-values, as such input noise injection effectively smooths out the Q landscape along changes in action. Target policy smoothing (or target smoothing, *abbrv.* TS) draws strong inspiration from the SARSA [SB98] learning update since it uses a perturbation of the greedy next-action in the learning update rule, which makes the method more robust against noisy inputs and therefore potentially safer in a safety-critical scenario. Note, while value overfitting primarily impedes policies that are deterministic by design, stochastic policies that prematurely collapse to their mode [SLM+15] are deterministic in effect and as such are impeded too. In particular, fitting the value estimate against an expectation of *similar* bootstrapped target value estimates forces similar actions to have similar values, which corresponds — by definition — to making the Q-function locally Lipschitz-continuous. As such, the induced smoothness over Q is to be understood in terms of *local Lipschitz-continuity* (or equivalently, *local Lipschitzness*), which we define in DEFINITION 3.4.1. More generally, the concept of smoothness that is at the core of the analyses laid out in this work is the concept of Lipschitz-continuity. Interestingly, we show later in SECTION 3.6.2.4, formally and from first principles, that target policy smoothing is equivalent to applying a regularizer on Q that induces Lipschitz-continuity *w.r.t.* the action input. In addition, we align the notion of *robustness* of a function approximator with the value of its *Lipschitz constant* (*cf.* DEFINITION 3.4.1): a k_1-Lipschitz-continuous function approximator will be characterized as *more robust* than another k_2-Lipschitz-continuous function approximator if and only if $k_1 \leq k_2$. As such, in this work, the notions of smoothness and robustness are both aligned with the notion of Lipschitz-continuity.

Definition 3.4.1 (local k-Lipschitz-continuity)**.** *Let f be a function $\mathcal{X} \subseteq \mathbb{R}^n \to \mathcal{Y} \subseteq \mathbb{R}^m$, $x \mapsto f(x)$, and C^0 (continuous) over $\mathcal{X}$. We denote the euclidean norms of $\mathcal{X}$ and $\mathcal{Y}$ by $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{Y}}$ respectively, and the Frobenius norm of the $\mathbb{R}^{m\times n}$ matrix space by $\|\cdot\|_F$. Lastly, let k be a non-negative real, $k \geq 0$.*

(a) f is k-Lipschitz-continuous over $\mathcal{X}$, iff, $\forall x, x' \in \mathcal{X}$,

$$\|f(x) - f(x')\|_{\mathcal{Y}} \leq k \|x - x'\|_{\mathcal{X}}$$

(b) If f is also differentiable, then f is k-Lipschitz-continuous over $\mathcal{X}$, iff, $\forall x, x' \in \mathcal{X}$,

$$\|\nabla f(x)\|_F \leq k$$

In either case, if the inequality is verified, k is called the Lipschitz constant of f. The symbol ∇, historically reserved to denote the gradient operator, is here used to denote the Jacobian operator of the vector function f, to maintain symmetry with the notations and appellations used in previous works.

(c) Let X be a subspace of $\mathcal{X}$, $X \subseteq \mathcal{X}$. f is said locally *k-Lipschitz-continuous over $X \subseteq \mathcal{X}$, iff, for all $x \in X$, there exists a neighborhood U_x of x such that f is k-Lipschitz-continuous over U_x.*

Based on Definition 3.4.1 *(b)* the gradient penalty in eq. 3.3, effectively enforces local Lipschitz-continuity over the support of the ζ distribution (described later in *cf.* Section 3.5.4), a subspace of the state-action joint space.

Unless specified otherwise, we use both the clipped double-Q learning and target policy smoothing add-on techniques in all the experiments reported in this work. We ran an ablation study on both techniques to illustrate their respective benefits, and support our algorithmic design choice to use them. We report said ablations in Appendix 3.D.

We describe the inner workings of SAM in Algorithm 3 [1]. Note, compared to SAM as it was introduced in Chapter 2, the SAM algorithm laid out in Algorithm 3 has gone through the three following simplifications. *1)* We do not use an extra, smaller replay buffer (retaining only recently collected transitions due to its size), in addition to the traditional replay buffer $\mathcal{R}$, to perform a learning update of D_φ on *"fresher"* samples. *2)* We only use a n-step TD loss as opposed to the sum of a n-step TD loss and a 1-step TD loss. *3)* We do not use any weight decay regularization on the critic Q_ω. Streamlining the algorithm in these three aspects proved not to be detrimental for the overall perfor-

[1] The symbols "◇" and "◇" appearing in front of line numbers in Algorithm 3 are related to the distributed learning scheme used in this work, which we describe in section 3.5.5.

mance. Trimming non-necessary add-ons from the method aligns with an important desideratum of this paper: finding the most minimal working configuration for SAM, or equivalently, finding what is necessary for the method's success.

Since our agent learns a parametric reward — differentiable by design — along with a deterministic policy, we *could*, in principle, use the gradient $\mathbb{E}_{s_t\sim\rho^\beta}[\nabla_\theta \mu_\theta(s_t)\nabla_a r_\varphi(s_t,a)|_{a=\mu_\theta(s_t)}]$ (constructed by analogy with EQ 3.9) to update the policy. In [BK19], we raised the question of whether one *should* use this gradient and answered in the negative: while the gradient in EQ 3.9 guides the policy towards behaviors that maximize the long-term return of the agent, effectively trying to address the credit assignment problem, the gradient involving r_φ in place of Q_ω is myopic, and does not encourage the policy to think more than one step ahead. It is obvious that back-propagating through Q_ω, literally designed to enable the policy to reason across longer time ranges, will be more helpful to the policy towards solving the task. We therefore discarded the gradient involving r_φ.

Nonetheless, we set out to investigate whether the latter can favorably assist the gradient in EQ 3.9 in solving the task, when both gradients are used *in conjunction*. Drawing a parallel with the line of work using unsupervised auxiliary tasks to improve representation learning in visual tasks [JMC+16, SMAD16, MPV+16, DGE15], we define the gradient $\mathbb{E}_{s_t\sim\rho^\beta}[\nabla_\theta \mu_\theta(s_t)\nabla_a Q_\omega(s_t,a)|_{a=\mu_\theta(s_t)}]$ as the *main* gradient, and $\mathbb{E}_{s_t\sim\rho^\beta}[\nabla_\theta \mu_\theta(s_t)\nabla_a r_\varphi(s_t,a)|_{a=\mu_\theta(s_t)}]$ as the *auxiliary* gradient, which we denote by g_m and g_a respectively. Based on our previous argumentation, allowing the myopic g_a to take the upper hand over g_m could have a disastrous impact on solving the task: combining the g_m and g_a must be done conservatively. As such, we use the auxiliary gradient only if it amplifies the main gradient. We measure the complementarity of the main and auxiliary tasks by the cosine similarity between their respective gradients, $\mathfrak{S}(g_m, g_a))$, as done in [DCJ+18], and assemble the new composite gradient $g_c \coloneqq g_m + \max(0, \mathfrak{S}(g_m, g_a))\, g_a$. By design, g_a is added to g_m only if the cosine similarity between them, $\mathfrak{S}(g_m, g_a))$, is positive, and will, in that case, be scaled by said cosine similarity. If the gradients are collinear, they are summed: $g_c = g_m + g_a$. If they are orthogonal or if the similarity is negative, g_a is discarded: $g_c = g_m$. Our experiments comparing the usage of g_c and g_m (*cf.* FIGURE 3.C.1) show that using the composite gradient g_c does not yield any improvement over using only g_m. By monitoring the values taken by $\mathfrak{S}(g_m, g_a))$, we noticed that the cosine similarity was almost always negative, yet close to 0, hence $g_c = g_m$, which trivially explains why the results are almost identical.

3.5 Lipschitzness is all you need

This section aims to put the emphasis on what makes off-policy generative adversarial imitation learning challenging. When applicable, we propose solutions to these challenges, supported by intuitive and empirical evidence. *In fine*, as the section name hints, we found that — in our experimental and

computational setting, described at the beginning of Section 3.5.5 — forcing the local Lipschitzness of the reward is a *sine qua non* condition for good performance, while also being *sufficient* to achieve peak performance.

3.5.1 A Deadlier Triad

In recent years, several works [FvHM18, FKSL19, AKA19] have carried out in-depth diagnoses of the inherent problems of Q-learning [Wat89, WD92] — and bootstrapping-based actor-critic architectures by extension — in the function approximation regime. Note, while the following issues directly apply to DQN [MKS+13, MKS+15], which even introduces additional difficulties (*e.g.* target networks, replay buffer), we limit the scope of this section to Q-learning, to eventually make our point. Q-learning under function approximation possesses properties that, when used in conjunction, make the algorithm brittle, prone to unstable behavior, as well as tedious to bring to convergence. Without caution, the algorithm is bound to diverge. These properties constitute the *deadly triad* [SB98, vHDS+18]: function approximation, bootstrapping, and off-policy learning.

Since the method we consider in this work *per se* follows an actor-critic architecture, it possesses all three properties, and is therefore inclined to diverge and suffer from instabilities. Additionally, since the learned reward r_φ is: *a)* defined from binary classifier predictions — discriminator's predicted probabilities of being expert-generated — estimated via function approximation, *b)* learned at the same time as the policy, and *c)* learned off-policy — with the negative samples coming from the replay distribution β, the method we study consequently introduces an extra layer of complication in the deadly triad. We now go over the three points and explain to what extent they each exacerbate the divergence-inducing properties that form the deadly triad.

To tackle point *a)*, we introduce explicit residuals to represent the various sources of error involved in temporal-difference learning, and illustrate how these residuals accumulate over the course of an episode. We will use the shorthand $\mathbb{E}[\cdot]$ for expectations for the sake of legibility. We take inspiration from eq (12) in [FvHM18], where a bias term is introduced in the TD error due to the function approximation of the Q-value, as the Bellman equation is never exactly satisfied in this regime. Borrowing the terminology from the statistical risk minimization literature, while the original bias suffered by the TD error was due to the *estimation error* caused by bootstrapping, function approximation is responsible for an extra *approximation error* contribution. The sum of these two errors is represented with the residual δ_ω. Let us now consider $D_\varphi(s, a)$, the estimated probability that a sample (s, a) is coming from expert demonstrations. Formally, $D_\varphi(s, a) = \mathbb{P}_\varphi[\textsc{Expert}(s, a)]$, where the event is defined as $\textsc{Expert}(s, a) :=$ "$s \sim \rho^{\pi_e} \wedge a \sim \pi_e$", and where $\mathbb{P}_\varphi$ denotes the probability estimated with the approximator φ. In the same vein, we distinguish the error contributions: the

Algorithm 3: SAM: Sample-efficient Adversarial Mimic

init: initialize the random seeds of each framework used for sampling, the random seed of the environment $\mathbb{M}$, the neural function approximators' parameters $(\theta, \varphi, \omega)$, their target networks as exact frozen copies, the rollout cache $\mathcal{C}$, the replay buffer $\mathcal{R}$.

while *no stopping criterion is met* **do**

/* Interact with the world to collect new samples */

repeat

Perform action $a_t \sim \pi_\theta(\cdot|s_t)$ in state s_t and receive the next state s_{t+1} and termination indicator d returned by the environment $\mathbb{M}^* - \{r_\varphi\}$;

Store the reward-less transition (s_t, a_t, s_{t+1}) in the rollout cache $\mathcal{C}$;

until *the rollout cache $\mathcal{C}$ is full*;

Dump the content of the rollout cache $\mathcal{C}$ into the replay buffer $\mathcal{R}$, then flush $\mathcal{C}$;

/* Train every modules */

foreach *training step per iteration* **do**

foreach *reward training step per iteration* **do**

Get a mini-batch of samples from the replay buffer $\mathcal{R}$;

Get a mini-batch of samples from the expert demonstration dataset $\mathcal{D}$;

Perform a gradient *descent* step along $\nabla_\varphi \ell_\varphi^{\text{GP}}$ (*cf.* EQ 3.2) using *both* mini-batches:

$$\ell_\varphi^{\text{GP}} := \mathbb{E}_{s_t \sim \rho^{\pi_e}, a_t \sim \pi_e}[-\log(1 - D_\varphi(s_t, a_t))] + \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta}[-\log(D_\varphi(s_t, a_t))] + \lambda \, \mathfrak{R}_\varphi^\zeta(k)$$

where $\mathfrak{R}_\varphi^\zeta(k) := \mathbb{E}_{s_t \sim \rho^\zeta, a_t \sim \zeta}[(\|\nabla_{s_t, a_t} D_\varphi(s_t, a_t)\| - k)^2]$ is a gradient penalty regularizer;

end

foreach *agent training step per iteration* **do**

Get a mini-batch of samples from the replay buffer $\mathcal{R}$;

Augment every reward-less transition sampled from $\mathcal{R}$ with the learned reward surrogate r_φ: $(s_t, a_t, s_{t+1}) \to (s_t, a_t, r_\varphi(s_t, a_t), s_{t+1})$ (omitting here the use of n-step returns for simplicity);

Perform a gradient *descent* step along $\nabla_\omega \ell_\omega$ (*cf.* EQ 3.4) using the mini-batch:

$$\ell_\omega := \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta}[(Q_\omega(s_t, a_t) - Q^{\text{targ}})^2]$$

where $Q^{\text{targ}} := \sum_{k=0}^{n-1} \gamma^k r_\varphi(s_{t+k}, a_{t+k}) + \gamma^n Q_{\omega'}(s_{t+n}, \mu_{\theta'}(s_{t+n}))$ is the n-step Bellman target;

Perform a gradient *ascent* step along $\nabla_\theta \mathcal{U}_\theta$ (*cf.* EQ 3.7) using the mini-batch:

$$\mathcal{U}_\theta := \mathbb{E}_{s_t \sim \rho^\beta}[Q_\omega(s_t, \mu_\theta(s_t))]$$

;

Update the target networks using the new ω and θ;

end

end

Adapt parameter noise standard deviation σ used to define π_θ from μ_θ (*cf.* SECTION 3.4);

/* Evaluate the trained policy */

foreach *evaluation step per iteration* **do**

Evaluate the empirical return of μ_θ in $\mathbb{M}$, using the task reward r (*cf.* SECTION 3.4);

end

end

approximation error is caused by the choice of function approximatior class (*e.g.* two-layer neural networks with hyperbolic tangent activations), and the estimation error is due to the gap between the estimations of our classifier and the predictions of the *Bayes classifier* — the classifier with the lowest misclassification rate in the chosen class. This gap can be written as $|D_\varphi(s_t, a_t) - \textsc{Bayes}(s_t, a_t)|$, where $\textsc{Bayes}(s, a) = \mathbb{P}_{\textsc{Bayes}}[\textsc{Expert}(s, a)]$, by analogy with the previous notations. *In fine*, we introduce the residual δ_φ that represents the contribution of both errors in the learned reward r_φ, hence:

$$
\begin{aligned}
Q_\omega(s_t, a_t) &= r_\varphi(s_t, a_t) - \delta_\varphi(s_t, a_t) + \gamma\mathbb{E}[Q_\omega(s_{t+1}, a_{t+1})] - \delta_\omega(s_t, a_t) && (3.10)\\
&= [r_\varphi(s_t, a_t) - \delta_\varphi(s_t, a_t) - \delta_\omega(s_t, a_t)] + \gamma\mathbb{E}[Q_\omega(s_{t+1}, a_{t+1})] && (3.11)\\
&= \Delta_{\varphi,\omega}(s_t, a_t) + \gamma\mathbb{E}[Q_\omega(s_{t+1}, a_{t+1})] && (3.12)\\
&= \Delta_{\varphi,\omega}(s_t, a_t) + \gamma\mathbb{E}[\Delta_{\varphi,\omega}(s_{t+1}, a_{t+1}) + \gamma\mathbb{E}[Q_\omega(s_{t+2}, a_{t+2})]] && (3.13)\\
&= \mathbb{E}\left[\sum_{k=0}^{+\infty} \gamma^k \Delta_{\varphi,\omega}(s_{t+k}, a_{t+k})\right] && (3.14)
\end{aligned}
$$

where $\Delta_{\varphi,\omega}(s_t, a_t) \coloneqq r_\varphi(s_t, a_t) - \delta_\varphi(s_t, a_t) - \delta_\omega(s_t, a_t)$.

As observed in [FvHM18] when estimating the accumulation of error due to function approximation in the standard RL setting, the variance of the state-action value is proportional to the variance of both the return and the Bellman residual δ_ω. Crucially, in our setting involving the learned imitation reward r_φ, it is *also* proportional to the variance of the residual δ_φ, containing contributions of both the approximation error *and* estimation error of r_φ. As a result, the variance of the estimate also suffers from a critically stronger dependence on γ (*cf.* ablation study in Appendix 3.G). Intuitively, as we propagate rewards further (higher γ^k value), their induced residual error triggers a greater increase in the variance of the Q-value estimate. In addition to its effect on the variance, the additional residual also clearly impacts the overestimation bias [TS93] it is afflicted by, which further advocates the use of dedicated techniques such as Double Q-learning [FvHM18, vH10], as we do in this work (*cf.* Section 3.4). All in all, by introducing an extra source of approximation and estimation error, we further burden TD-learning.

Moving on to points *b)* — the reward is learned at the same time as the policy — and *c)* — the reward is learned off-policy using samples from the replay policy β — we see that each statement allow us to qualify the reward r_φ as a *non-stationary* process. Conceptually, by considering a additive decomposition of the reward r_φ into a stationary r_φ^{STAT} and a non-stationary contribution $r_\varphi^{\text{NON-STAT}}$, we see that following an accumulation analysis similar to the previous one shows that the variance of the state-action value is proportional to the variances of each contribution. While the variance of r_φ^{STAT} can be important and therefore can have a considerable impact on the variance of the Q-value estimate,

it can usually be somewhat tamed with online normalization techniques and mitigated with techniques enabling the agent to cope with rewards of vastly different scales (*e.g.* POP-ART [vHGH+16]). We show later that such methods do not help when the underlying reward is non-stationary (*cf.* SECTION 3.5.2 for empirical results). The variance of the non-stationary contribution $r_\varphi^{\text{NON-STAT}}$, indeed is, due to its continually-changing nature, untameable with these regular techniques relying on the usual stationarity assumption — unless additional dedicated mechanisms are integrated (*e.g.* change point detection techniques). Naturally, the non-stationary contribution also has an effect on the bias of the estimation, and *a fortiori* on its overestimation bias (as with *a)*). We note that the argument made in the context of Q-learning by [FKSL19] naturally transfers to the TD-learning objective optimized in this work: the objective is non-stationary, due to *i)* the *moving target* problem — caused by using bootstrapping to learn an estimate that is updated every iteration and *ii)* the *distribution shift* problem — caused by learning the Q-value estimate off-policy using β, effectively being a mixture of past policies, which changes every iteration. Point *i)* is a source of non-stationarity since the target of the supervised objective is moving with the prediction as iterations go by, due to using bootstrapping. Fitting the current estimate against the target defined from this very estimate is an ordeal, and *b)* makes the task even harder by having the reward move too, given it is also learned, at the same time. The target of the TD objective therefore now has two moving pieces, one from bootstrapping (*i)*), one from reward learning (*b)*). The distribution shift problem *ii)*, stemming from the Q-value being learned off-policy, is naturally worsened by the reward being estimated off-policy *c)*. Note, although both the reward and Q-value are learned with samples from β, the actual mini-batches used to perform the gradient update of each estimate might be different in practice. As such, the TD error would be optimized using samples from a mixture of past policies that is different from the mixture under which the reward is learned, and then use this reward trained under a different effective distribution in the Bellman target. All in all, by introducing a extra sources of non-stationarity (*b)* and *c)*), we further burden the non-stationarity of TD-learning (*i)* and *ii)*).

3.5.2 CONTINUALLY CHANGING REWARDS

In a non-stationary MDP, the non-stationarities can manifest in the dynamics [NEG05, DSBBE06, XM07, LXM13, AK16a], in the reward process [EdKM05, DGS14], or in both conjointly [YM09a, YM09b, AYBS13, GOA18, PPB19, YS19, LR19] (*cf.* APPENDIX 3.B for a review of sequential decision making under uncertainty in non-stationary MDPs). In this work, we focus on the MDP $\mathbb{M}^*$ whose transition distribution p is stationary *i.e.* not changing over time. As discussed in SECTION 3.5.1, the reward process defined by r_φ is however non-stationary. In particular, r_φ is *drifting*, *i.e.* gradually changes at an unknown rate, due to the reward being learned at the same time as the policy, but also due to it being estimated off-policy. While the former reason is true in the on-policy setting as well,

the latter is specific to the off-policy setting, on which we focus in this work. Indeed, in *on-policy* generative adversarial imitation learning, the parameter sets φ and θ are involved in a bilevel optimization problem (*cf.* SECTION 3.3) and consequently are intricately tied. φ is trained via an adversarial procedure opposing it to θ in a zero-sum two-player game. At the same time, θ is trained by policy gradients to optimize π_θ's episodic accumulation of rewards generated by r_φ. The synthetically generated rewards perceived by the agent are, in effect, sampled from a stochastic process that incrementally changes over the course of the policy updates, effectively qualifying r_φ as a drifting non-stationary reward process.

By moving to the off-policy setting — for reasons laid out earlier in SECTION 3.4 — the zero-sum two-player game is not opposing r_φ and π_θ, but r_φ and β, where β is the off-policy distribution stemming from experience replay. As the parameter set θ go through gradient updates, the new policies π_θ are added to the mixture of past policies β. Crucially, to perform its parameter update at a given iteration, the policy π_θ uses transitions augmented with rewards generated by r_φ, whose latest update was trying to distinguish between samples from π_e and β (as opposed to π_e and π_θ in the on-policy setting). Since π_θ is drifting, β is also drifting based on how experience replay operates. Nevertheless, by being a mixture of previous policy updates, β potentially drifts less that π_θ, since, in effect, two consecutive β distributions are mixing over a wide overlap of the same past policies. In reality however, β corresponds to uniformly sampling a mini-batch from the replay buffer. Consecutive β can therefore be uncontrollably distant from each other in practice, making the distributional drift of the reward more tedious to deal with than in the on-policy setting. Using large mini-batches and distributed multi-core architectures somewhat levels the playing field though.

The adversarial bilevel optimization problem guiding the adaptive tuning of r_φ for every π_θ update is reminiscent of the stream of research pioneered by [ACBFS95] in which the reward is generated by an omniscient *adversary*, either arbitrarily or adaptively with potentially malevolent drive [YM09a, YM09b, LXM13, GOA18, YS19]. Non-stationary environments are almost exclusively tackled from a theoretical perspective in the literature (*cf.* previous references). Specifically, in the *drifting* case, the non-stationarities are traditionally dealt with via the use of sliding windows. The accompanying (dynamic) regret analyses all rely on strict assumptions. In the switching case, one needs to know the number of occurring switches beforehand, while in the drifting case, the change variation need be upper-bounded. Specifically, [BGZ14, CSLZ19a] assume the total change to be upper-bounded by some preset variation budget, while [CSLZ19b] assumes the variations are uniformly bounded in time. [OGA19] assumes that the *incremental* variation (as opposed to *total* in [BGZ14, CSLZ19a]) is upper-bounded by a *per-change* threshold. Finally, in the same vein, [LR19] posits *regular evolution*, by making the assumption that both the transition and reward functions are Lipschitz-continuous *w.r.t.* time. By contrast, our approach relies on imposing local Lipschitz-continuity of the reward over

the input space, which will be described later in SECTION 3.5.4.

Online return normalization methods — using statistics computed over the entire return history (reminiscent of sliding window methods) to whiten the current return estimate — are the usual go-to solution to deal with rewards (and *a fortiori* returns) whose scale can vary a lot, albeit still under stationarity assumption. We investigate whether online return normalization methods and POP-ART [vHGH+16] can have a positive impact on learning performance, when the process underlying the reward is learned at the same time as the policy, via experience replay. Given that the reward distribution can drift at an unknown rate (although influenced by the learning rate used to train φ), it is fair to assume that we might benefit from such methods, especially considering how unstable a twin bilevel optimization problem can be. On the other hand, as learning progresses, older rewards are – especially in early training — *stale*, which can potentially pollute the running statistics accumulated by these normalization techniques. The results obtained in this ablation study are reported in APPENDIX 3.H.

We observe that neither return normalization nor POP-ART provide an improvement over the baseline. On the contrary, in `Hopper` and `Walker2d`, we see that they even yield significantly poorer performance within the allowed runtime, compared to the base method using neither return normalization nor POP-ART (*cf.* FIGURE 3.H). We propose an explanation of this phenomenon based on the *stability-plasticity dilemma* [CG87]. In early training, the policy π_θ changes at a fast rate and with a high amplitude when going through gradient updates, due to being a randomly initialized neural function approximator. The reward r_φ is in a symmetric situation, but is also influenced by the rate of change of θ, being trained in an adversarial game. In order to keep up with this fast pace of change in early training, the critic Q_ω — using the reward r_φ in its own learning objective — needs to be sufficiently flexible to accommodate and adapt quickly to these frequent changes. In other words, the critic's *plasticity* must be high. Since reward estimates from r_φ become stale after a few φ updates, we also want our critic to avoid using stale reward to prevent the degradation of ω. This property is referred to as *stability* in [CG87]. *In fine*, the critic must be plastic and stable. Note, using the current reward update to augment the sample transitions with their reward, as done in this work, provides the critic with such stability. However, return normalization and POP-ART use stale running statistics estimates to whiten the state-action values returned by the critic, which prevents both plasticity (values need to change fast with the reward, normalization slows down this process) and harms stability due to the staleness of the obsolete reward that are *"baked in"* the running statistics. The obtained results corroborate the previous analysis (*cf.* APPENDIX 3.H).

We conclude this section by discussing the reward learning dynamics. While in the transient regime, the reward process is effectively non-stationary, it gradually becomes stationary as it reaches a steady-state regime. Nonetheless, the presence of such stabilization does not guarantee that the desired equi-

librium has been reached. Indeed, as we will discuss in the next section, adversarial imitation learning has proved to be prone to overfitting. We now address it.

3.5.3 Overfitting cascade

In recent years, neural networks have received an increasing amount of attention due to their sought-after ability to generalize over data they have not been trained on — albeit coming from a closely-related distribution. A model that generalizes well displays low errors on the train set (trained on) and particularly also on the test set (never seen before). When a model achieves low error on the train set and high error on the test set, it is said to *overfit*. That is, it has *memorized* the intended pattern on the data it has been shown, but what was learned critically *does not generalize* to new data. Avoiding overfitting is therefore paramount.

Being based on a binary classifier, the synthetic reward process r_φ is inherently susceptible to overfitting, and it has been shown (*cf.* subsequent references) that it indeed does. As exhibited in Section 3.2, several endeavors have proposed techniques to prevent the learned reward from overfitting, individually building on traditional regularization methods aimed to address overfitting in classification. These techniques either make the discriminator model weaker [RAW+18, BK19, KAD+19, PKT+18], or make the classification task harder [BK19, XD19, ZRN+19], to deter the discriminator from relying on non-salient features to trivially distinguish between samples from π_e and π_θ (π_e and β in our off-policy setting, *cf.* Section 3.5.2).

On a more fundamental level, the ability of deep neural networks to generalize (and *a fortiori* to circumvent overfitting) had been attributed to the flatness of the loss landscape in the neighborhoods of minima of the loss function [HS97, KMN+17] — provided the optimization method is a variant of stochastic gradient descent. While it has more recently been shown that sharp minima *can* generalize [DPBB17], we argue and show both empirically and analytically that, in the off-policy setting tackled in this work, flatness of the reward function around the maxima — corresponding to the positive samples, *i.e.* the expert data — is paramount for good empirical performance. In other words, we argue that the presence of peaks in the reward function caused by the discriminator overfitting on the expert data (non-salient features in the worst case) is the major source of optimization issues occuring in off-policy GAIL. As such, we focus on methods that address overfitting by inducing flatness in the learned reward function around expert samples, subject to being peaked on the reward landscape. An obvious candidate to enforce this desired flatness property is gradient penalty regularization, inducing Lipschitz-continuity on the reward function r_φ, over its input space $\mathcal{S}\times\mathcal{A}$, which has been described earlier in Section 3.4, and will be the object of Sections 3.5.4 and 3.6.3.

Simply put, reward overfitting translates to the presence of peaks on the reward landscape. Even in

the case where these peaks exactly coincide with the expert data (perfect classification, the discriminator coincides with the Bayes classifier of the function class), peaked reward landscapes (*i.e.* sparse reward setting) can be tedious to optimize over. Crucially, peaks in r_φ *can potentially* cause peaks in the state-action value landscape Q_ω. When policy evaluation is done via Monte-Carlo estimation, the length of the rollouts likely attenuates the contribution of individual peaked rewards aggregated during the rollout into a discounted sum. If the peaks were not predominant in the rollout, the associated empirical estimate of the value will not be peaked (relative to its neighboring values). By contrast, the TD's bootstrapping-based objective does not attenuate peaks in r_φ, which consequently causes peaks in Q_ω. Note, using multi-steps returns [PW96] can help mitigate the phenomenon and benefit from the attenuation effect witnessed in the Monte-Carlo estimation described above, hence our usage of multi-step returns in this work (*cf.* Section 3.4).

Narrow peaks in the state-action value estimate Q_ω can cause the deterministic policy μ_θ to itself overfit to these peaks on the Q_ω landscape. As such overfitting *cascades* from rewards to the policy, and hampers policy optimization (*cf.* eq. 3.9). Furthermore, peaks in Q-values can severely hinder temporal-difference optimization since, by design, these outlying values can appear in either the predicted Q-value or the target Q-value. As such, echoing the observations and analyses made in Sections 3.5.1 and 3.5.2, bootstrapping makes the optimization more tedious, when bringing sampled-efficiency to GAIL. These irregularities naturally transfer to the loss landscape, exacerbating the innate irregularity of loss landscapes when using neural networks as function approximators [LXT+18], making it harder to optimize over eq. 3.4. *In fine*, peaks on the reward landscape can cascade and impede both policy improvement and evaluation.

In the next section (Section 3.5.4), we discuss how to enforce Lipschitz-continuity in usual neural architectures, before going over empirical results corroborating our previous analyses (*cf.* Section 3.5.5). Ultimately, we show that *not* forcing Lipschitz-continuity on the learned surrogate reward yields poor results, making it a *sine qua non* condition for success.

3.5.4 Enforcing Lipschitz-continuity in deep neural networks

Designed to address the shortcomings of the original GAN [GPAM+14], whose training effectively minimizes a Jensen-Shannon divergence between generated and real distributions, the Wasserstein GAN (WGAN) [ACB17] leverages the Wasserstein metric. Specifically, the authors of [ACB17] use the dual representation of the Wasserstein-1 metric under a 1-Lipschitz-continuity assumption (see Definition 3.4.1) over the discriminator, which allow them to employ the Kantorovich-Rubinstein duality theorem, to eventually arrive at a tractable loss one can optimize over.

In the Wasserstein GAN [ACB17], the weights of the discriminator — called *critic* to emphasize that

it is no longer a classifier — are *clipped*. While not equivalent to enforcing the 1-Lipschitz constraint their model is theoretically built on, clipping the weights *does* loosely enforce Lipschitz-continuity, with a Lipschitz constant depending on the clipping boundaries. This simple technique however disrupts, by its design, the optimization dynamics. As emphasized in [GAA+17], clipping the weights of the Wasserstein critic can result in a pathological optimization landscape, echoing the analysis carried out in SECTION 3.5.3.

In an attempt to address this issue, the authors of [GAA+17] propose to impose the underlying 1-Lipschitz constraint via another method, fully integrated into the bilevel optimization problem as a gradient penalty regularization. When augmented with this gradient penalization technique, WGAN — dubbed WGAN-*GP* — is shown to yield consistently better results, enjoys more stable learning dynamics, and displays a smoother loss landscape [GAA+17]. Interestingly, the regularization technique has proved to yield better results even in the original GAN [LKM+17], despite it not being grounded on the Lipschitzness footing like WGAN [ACB17]. In addition, following in the footsteps of the comprehensive study proposed in [LKM+17], [KLZ+18] shows empirically that the WGAN loss does not outperform the original GAN consistently across various hyper-parameter settings, and advocates for the use of the original GAN loss, along with the use of spectral normalization [MKKY18], and gradient penalty regularization [GAA+17] to achieve the best results (albeit at an increased cost in computation in visual domains). In line with these works ([LKM+17, KLZ+18]), we therefore commit to the archetype GAN loss formulation [GPAM+14], as has been laid out earlier in SECTION 3.4 when describing the discriminator objective in EQ 3.2. We now remind the objective optimized by the discriminator (*cf.* EQ 3.3), where the generalized form of the gradient penalty, $\mathfrak{R}^{\zeta}_{\varphi}(k)$, subsumes the original penalty [GAA+17] as well as variants that will be studied later in SECTION 3.6.3:

$$\ell^{\text{GP}}_{\varphi} \coloneqq \ell_{\varphi} + \lambda\, \mathfrak{R}^{\zeta}_{\varphi}(k) \coloneqq \ell_{\varphi} + \lambda\, \mathbb{E}_{s_t \sim \rho^{\zeta}, a_t \sim \zeta}[(\|\nabla_{s_t, a_t} D_{\varphi}(s_t, a_t)\| - k)^2] \tag{3.15}$$

In EQ 3.15, λ corresponds to the weight attributed to the regularizer in the objective (*cf.* ablation in SECTION 3.6.3), and $\|\cdot\|$ depicts the euclidean norm in the appropriate vector space. ζ is the distribution defining *where* in the input space $\mathcal{S} \times \mathcal{A}$ the Lipschitzness constraint should be enforced. ζ is defined from π_e and β. In the original gradient penalty formulation [GAA+17], ζ corresponds to sampling points uniformly in segments [2] joining points from the generated data and real data, grounded on the derived theoretical results (*cf.* Proposition 1 in [GAA+17]) that the optimal discriminator is 1-Lipschitz along these segments. While it does not mean that enforcing such constraint will make the discriminator optimal, it yields good results in practice. We discuss several formulations of ζ in

[2] The segment joining the arbitrary points x and y in $\mathbb{R}^d$ is the set of points defined as $S \coloneqq \{(1-\alpha)x + \alpha y \mid \alpha \in [0, 1]\}$. Sampling a point $z \in \mathbb{R}^d$ uniformly from S corresponds to sampling $\alpha \sim \text{unif}(0, 1)$, before assembling $z \coloneqq (1-\alpha)x + \alpha y$.

Section 3.6.3, evaluate them empirically and propose intuitive arguments explaining the obtained results. In particular, we adopt an *RL viewpoint* and propose an alternate ground as to why the regularizer has enabled successes in control and search tasks, as reported in [BK19, KAD+19]. In particular, in [GAA+17], the 1-Lipschitz-continuity is encouraged by using $\mathfrak{R}^{\zeta}_{\varphi}(1)$ as regularizer.

Additionally, in line with the observations done in [GAA+17], we investigated with *a)* replacing $\mathfrak{R}^{\zeta}_{\varphi}(k)$ with a *one-sided* alternative defined as $\mathbb{E}_{s_t\sim\rho^{\zeta},a_t\sim\zeta}[\max(0, \|\nabla_{s_t,a_t} D_{\varphi}(s_t, a_t)\| - k)^2]$, and *b)* ablating online batch normalization of the state input from the discriminator. The alternative regularizer of *a)* encourages the norm to be *lower* than k (formally, $\|\nabla_{s_t,a_t} D_{\varphi}(s_t, a_t)\| \leq k$) in contrast to the original regularizer that enforces it to be *close* to k. While the one-sided version describes the notion of k-Lipschitzness more accurately (*cf.* Definition 3.4.1), it yields similar results overall, as shown in Appendix 3.E.1. Crucially, we conclude from these experiments that it is *sufficient* to have the norm remain upper-bounded by k, or equivalently, to have D_{φ} be Lipschitz-continuous. In other words, we do not need to impose a stronger constraint than k-Lipschitz-continuity on the discriminator to achieve peak performance, in the context of this ablation study. As for *b)*, online batch normalization of the state input is mostly hurting performance. as reported in Appendix 3.E.2. We therefore arrive at the same conclusions as [GAA+17]: *a)* we use the *two-sided* formulation of $\mathfrak{R}^{\zeta}_{\varphi}(k)$ described in eq. 3.15 since using the once-sided variant yields no improvement, and *b)* we omit the online batch normalization of the state input in the discriminator since it hurts performance, while still using this normalization scheme in the policy and critic (more details about the technique will be given when we describe our experimental setting in the next section, Section 3.5.5).

3.5.5 Diagnosing the importance of Lipschitzness in off-policy adversarial imitation learning

Before going over the empirical results reported in this section, we describe our experimental setting. Unless explicitly stated otherwise, every experiment — reported in both this section and Section 3.6.5 — is run in the same base setting. In addition, the used hyper-parameters are made available in Appendix 3.A.

3.5.5.1 Environments

In this work, we consider the simulated robotics, continuous control environments built with the MuJoCo [TET12] physics engine, and provided to the community through the OpenAI Gym API [BCP+16]. We use the following versions of the environments: `v3` for `Hopper`, `Walker2d`, `HalfCheetah`, `Ant`, `Humanoid`, and `v2` for `InvertedDoublePendulum`. For each of these, the dimension n of a given state $s \in \mathcal{S} \subseteq \mathbb{R}^n$ and the dimension m of a given action $a \in \mathcal{A} \subseteq \mathbb{R}^m$ scale as the degrees of

freedom (DoFs) associated with the environment's underlying MuJoCo model. As a rule of thumb, the more complex the articulated physics-bound model is (*i.e.* more limbs, joints with greater DoFs), the larger both n and m are. The intrinsic difficulty of the simulated robotics task scales super-linearly with n and m, albeit considerably faster with m (policy's output) than with n (policy's input).

Omitting their respective versions, Table 3.5.1 reports the state and action dimensions (n and m respectively) for all the environments tackled in this work, and are ordered, from left to right, by increasing state and action dimensions, `Humanoid-v3` being the most challenging. Since we consider, in our experiments, expert datasets composed of at most 10 demonstrations (10 is the default number; when we use 5, we specify it in the caption), we report return statistics (mean μ and standard deviation σ, formatted as $\mu(\sigma)$ in Table 3.5.1) aggregated over the set of 10 deterministically-selected demonstrations (the 10 first in our fixed pool) that every method requesting for 10 demonstrations will receive. To reiterate: in this work, every single method and variant will receive exactly the same demonstrations, due to an explicit seeding mechanism in every experiment. The reported statistics therefore identically apply to every method or variant using 10 demonstrations. By design, this reproducibility asset naturally extends to settings requesting fewer.

3.5.5.2 Demonstrations

As in [HE16], we subsampled every demonstration with a $1/u$ ratio — an operation called *temporal dropout* in [DAS+17]. For a given demonstration, we sample an index i_0 from the discrete uniform distribution unif$\{0, u-1\}$ to determine the first subsampled transition. We then take one transition every u transition from the initial index i_0. *In fine*, the subsampled demonstration is extracted from the original one of length l by only preserving the transitions of indices $\{i_0 + ku \mid 0 \leq k < \lfloor l/u \rfloor\}$. Since the experts achieve very high performance in the MuJoCo benchmark (*cf.* last column of Table 3.5.1) they never fail their task and live until the *"timeout"* episode termination triggered by OpenAI Gym API, triggered once the horizon of 1000 timesteps is reached, in every environments considered in this work. As such, most demonstrations have a length $l \approx 1000$ transitions (sometimes less but always above 950). Since we use the sub-sampling rate $u = 20$, as in [HE16], the subsampled demonstrations have a length of $|\{i_0 + ku \mid 0 \leq k < \lfloor l/u \rfloor\}| = \lfloor l/u \rfloor \approx 50$ transitions.

We wrap the absorbing states in both the expert trajectories beforehand and agent-generated trajectories at training time, as introduced in [KAD+19]. Note, this assumes knowledge about the nature — organic (*e.g.* falling down) and triggered (*e.g.* timeout flag set at a fixed episode horizon) — of the episode terminations (if any) occurring in the expert trajectories. Considering the benchmark, it is trivial to individually determine their natures in our work, which makes said assumption of knowledge weak. We trained the experts from which the demonstrations were then extracted using the on-

policy state-of-the-art PPO [SWD+17] algorithm (with Generalized Advantaged Estimation, or GAE [SML+16]). We used early stopping to halt the expert training processes when a phenomenon of diminishing returns is observed in its empirical return, typically attained by the 20 million interactions mark. We used our own parallel PPO implementation, written in PyTorch [PGM+19], and will share the code upon acceptance. The IL endeavors presented in this work have also been implemented with this framework.

3.5.5.3 Distributed training

The distributed training scheme employed to obtain every empirical imitation learning result exhibited in this work uses the MPI message-passing standard. Upon launch, an experiment spins n workers, each assigned with an identifying unique rank $0 \leq r < n$. They all have symmetric roles, except the rank 0 worker, which will be referred to as the *"zero-rank"* worker. The role of each worker is to follow the studied algorithm — SAM (*cf.* Algorithm 3) in the experiments reported in this section, and the proposed extension PURPLE (*cf.* Algorithm 4), in the experiments reported later in Section 3.6.5. The zero-rank worker exactly follows the algorithm, while the $n-1$ other workers omit the evaluation phase (denoted by the symbol "$\diamond$" appearing in front of the line number). The random seed of each worker is defined deterministically from its rank and the *base* random seed given as a hyper-parameter by the practitioner, and is used to *a)* determine the behavior of every stochastic entity involved in the worker's training process, and *b)* determine the stochasticity of the environment it interacts with.

Before every gradient-based parameter update step — denoted in Algorithm 3 and Algorithm 4 by the symbol "$\diamond$" appearing in front of the line number — the zero-rank worker gathers the gradients across the $n-1$ other workers, and aggregates them via an averaging operation, and sends the aggregate to every worker. Upon receipt, every worker of the pool then uses the aggregated gradient in its own learning update. Since the parameters are synced across workers before the learning process kicks off, this *synchronous* gradient-averaging scheme ensures that the workers all have the same parameters throughout the entire learning process (same initial parameters, then same updates). This distributed training scheme leverages learners seeded differently in their own environments, also seeded differently, to accelerate exploration, and above all provide the model with greater robustness.

Every imitation learning experiment whose results are reported in this work has been run for a fixed wall-clock duration — 12 or 48 hours, as indicated in their respective captions — due to hardware and computational infrastructure constraints. While the effective running time appears in the caption of every plot, the latter still depict the temporal progression of the methods in terms of *timesteps*, the number of interactions carried out with the environment. The reported performance corresponds to

Environment	State dim. n	Action dim. m	Expert Return $\mu(\sigma)$
`IDP`	11	1	9339.966(1.041)
`Hopper`	11	3	4111.823(56.81)
`Walker2d`	17	6	6046.116(13.76)
`HalfCheetah`	17	6	7613.154(36.25)
`Ant`	111	8	6688.696(48.83)
`Humanoid`	376	17	9175.152(98.94)

Table 3.5.1: State and action dimensions, n and m, of the studied environments from the MuJoCo [TET12] simulated robotics benchmark from OpenAI Gym [BCP+16]. (*abbrv.* `IDP` for `Inverted-DoublePendulum`, the continuous control counterpart of `Acrobot`.) In the last column, we report both the mean μ and standard deviation σ (formatted as $\mu(\sigma)$ in the table) of the expert's returns, aggregated across the set of 10 demonstrations used in this work.

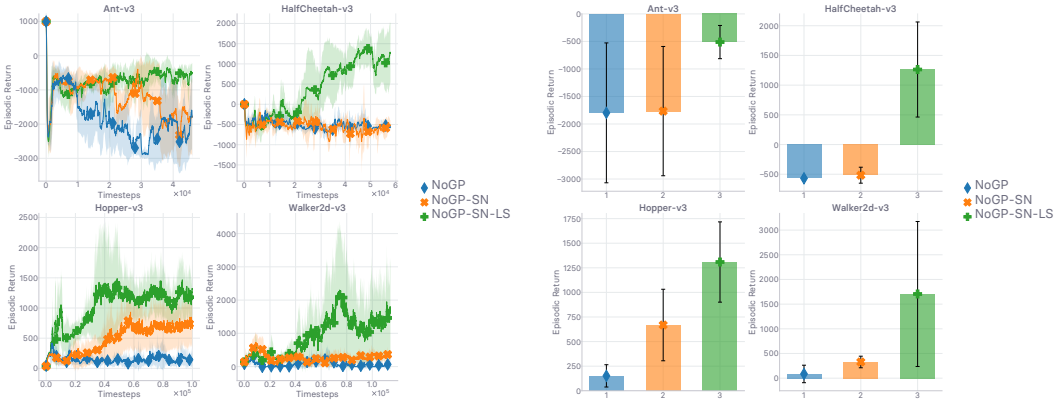
the undiscounted empirical return, computed using the reward returned by the environment (available at evaluation time), gathered by the non-perturbed policy μ_θ (deterministic) of the zero-rank worker. Every experiment uses 16 workers, and can therefore be executed on most desktop consumer-grade computers. Lastly, we monitored every experiment with the Weights & Biases [Bie20] tracking and visualization tool.

Additionally, we run each experiment with 5 different *base* random seeds (0 to 4), raising the effective seed count per experiment to 80. Each presented plot depicts the mean across them with a solid line, and the standard deviation envelope (half a standard deviation on either side of the mean) with a shaded area. Finally, we use an *online* observation normalization scheme, instrumental in performing well in continuous control tasks. The running mean and standard deviation used to standardize the observations are computed using an online method to represent the statistics of the entire history of observation. These statistics are updated with the mean and standard deviation computed over the concatenation of latest rollouts collected by each parallel worker, making is effectively an *online distributed* batch normalization [IS15] variant.

3.5.5.4 Empirical results

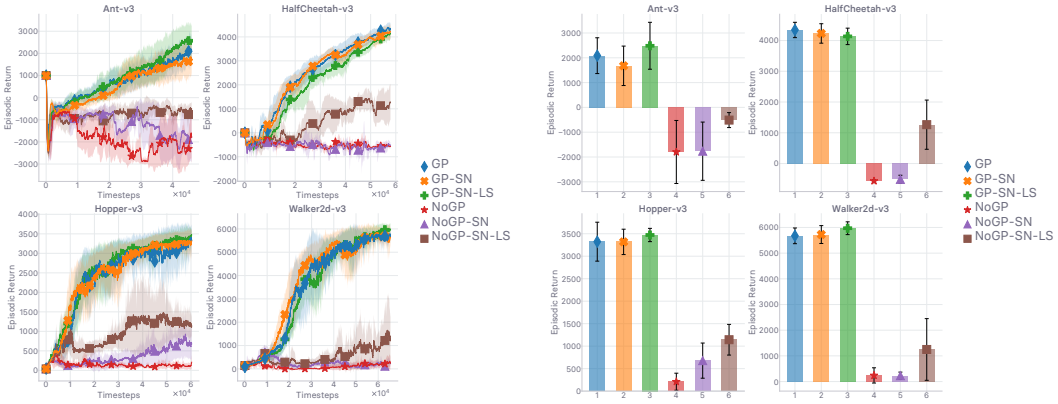
We now go over our first set of empirical results, whose goal is to show to what extent gradient penalty regularization is needed. The compared methods all use SAM (*cf.* Section 3.4) as base.

First, Figure 3.5.1 compares several modular configurations, which are described using the following handles in the legend. `GP` means that gradient penalization (GP) (*cf.* Section 3.5.4) is used. `NoGP` means that GP is not used (using ℓ_φ instead of $\ell_\varphi^{\mathrm{GP}}$). Note, `NoGP` is the only negative handle that we use, since it it central to our analyses. When any other technique is not in use, it is simply absent from



(a) Evolution of return values *(higher is better)* (b) Final return values at timeout *(higher is better)*

Figure 3.5.1: Evaluation of several methods while *not* using GP. Legend described in text. Runtime is 12 hours.



(a) Evolution of return values *(higher is better)* (b) Final return values at timeout *(higher is better)*

Figure 3.5.2: Evaluation of several methods showing the necessity of GP. Legend described in text. Runtime is ***12 hours***.

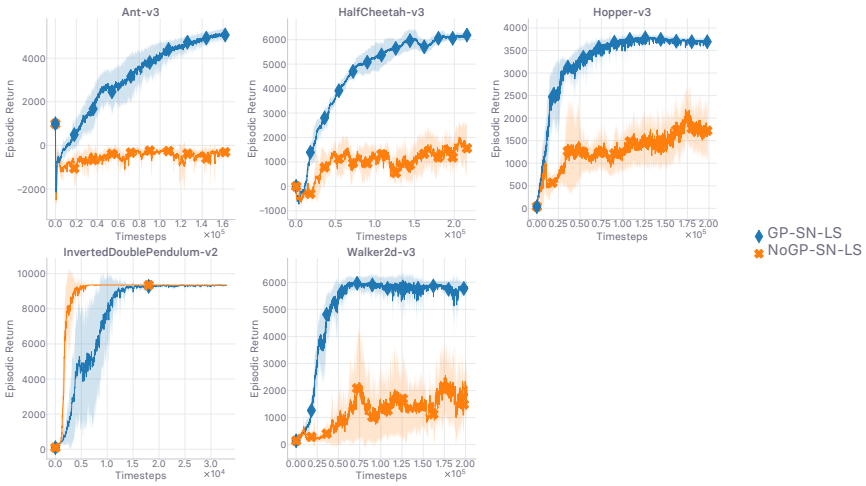


Figure 3.5.3: Evaluation of several methods showing the necessity of GP. Legend described in text. Runtime is ***48 hours***.

the handle in the legend. `SN` means that spectral normalization (SN) [MKKY18] is used. SN normalizes the discriminator's weights to have a norm close to 1, drawing a direct parallel with GP. In line with what the large-scale ablation studies on GAN add-ons advocate [LKM+17, KLZ+18], SN is used in most modern GAN architectures for its simplicity. We here investigate if SN is enough to keep the gradient in check, or if GP is necessary. `LS` denotes one-sided uniform label smoothing, consisting in replacing the positive labels only (hence *one-sided*), which are normally equal to 1 (expert, real), by a *soft label* u, distributed as $u \sim \text{unif}(0.7, 1.2)$. We do not consider Variational Discriminator Bottleneck (VDB) [PKT+18] in our comparisons since *a)* we prefer to focus on stripped-down canonical methods, and *b)* the information bottleneck forced on the discriminator's hidden representation boils down to smoothing the labels anyway, as shown recently in [MKH19].

In Figure 3.5.1, we see that *not using GP* (`NoGP`) prevents the agent from learning anything valuable: the agent barely collects *any reward at all*. While using SN can improve performance slightly (`NoGP-SN`), the addition of LS (`NoGP-SN-LS`) *considerably* improves performance over the two previous candidates. Nonetheless, despite the sizable runtime, all three perform poorly and are a far cry from achieving the same empirical return as the expert (*cf.* Table 3.5.1). In contrast with Figure 3.5.1, Figure 3.5.2 and Figure 3.5.3 show to what extent introducing GP in the off-policy imitation learning algorithm considered in this work impacts performance positively. The performance gap is *substantial* — in every environment except the easiest one considered, `InvertedDoublePendulum-v2`, as described in Table 3.5.1. As soon as GP is in use, the agent achieves near-expert performance (*cf.* Table 3.5.1). *In fine*, Figure 3.5.1 shows that *without* GP, neither SN nor LS are enough to enable the agent to mimic the expert with high fidelity, while Figure 3.5.2 and Figure 3.5.3 show that

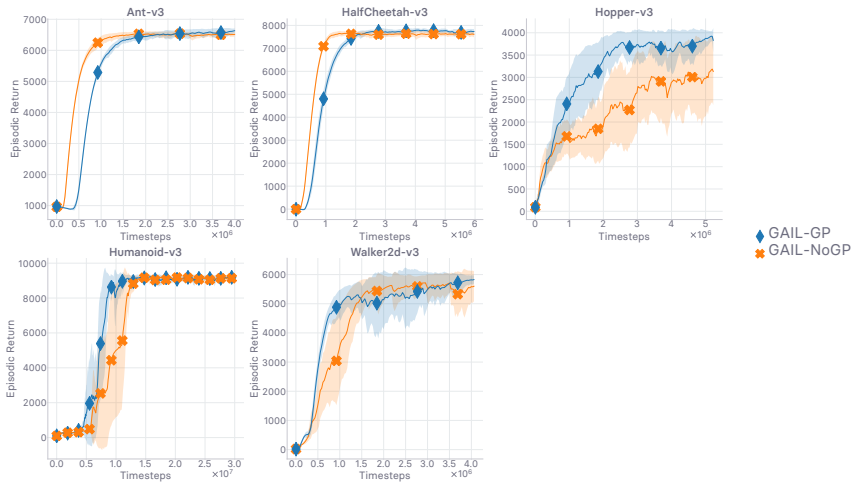
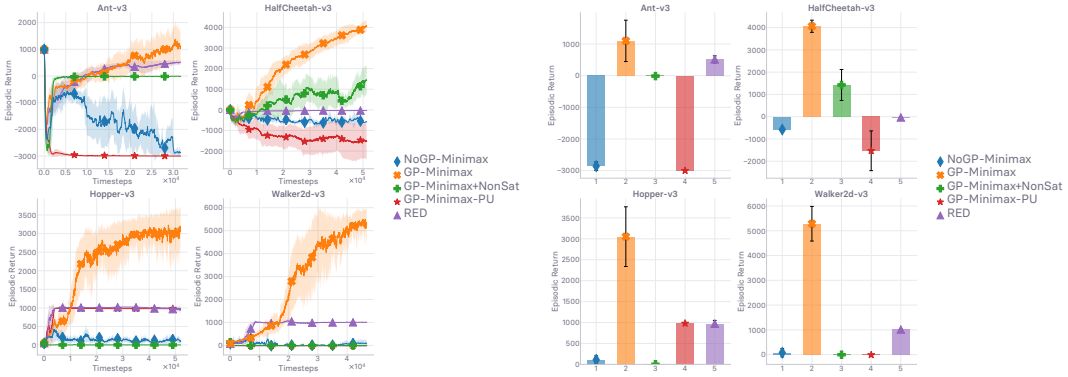


Figure 3.5.4: Ablation study on GP in *on-policy* GAIL. We see that the agent is still able to learn policies achieving peak performance even without GP, in contrast to the off-policy version of the algorithm. In the most difficult environment of the MuJoCo suite (*cf.* Table 3.5.1), `Humanoid`, GP achieves best performance. Runtime is 12 hours.

with GP, extra methods such as LS barely improve performance. These results support our claim: gradient penalty is, (*empirically*) *necessary* and *sufficient* to ensure near-expert performance in *off-policy* generative adversarial imitation learning, in our computational setting.

We also conducted an ablation of GP in the *on-policy* setting, reported in Figure 3.5.4. We see that across the range of environments, GP does not assume the same decisive role as in the off-policy setting. In fact, the agent reaches peak performance earlier *without* GP in two challenging environments, `Ant` and `HalfCheetah`, out of the five considered. Nevertheless, it still allows the agent to attain peak empirical return faster in `Hopper`, `Walker2d`, and perhaps most strikingly, in the extremely complex `Humanoid` environment. All in all, while GP can help in the on-policy setting, in is not *necessary* as in the off-policy setting studied in this work. In line with the analyses led in Sections 3.5.1, 3.5.2, and 3.5.3, the results of Figure 3.5.4 somewhat corroborate our claim that the presence of bootstrapping in the policy evaluation objective creates a *bottleneck*, that can be addressed by enforcing a Lipschitz-continuity constraint — GP — on the reward learned for imitation.

Figure 3.5.5 compares SAM, with and without GP, against several alternate versions of the objective used to train the surrogate reward for imitation. We introduce the following new handles to denote these methods. *"RED"* means that the random expert distillation (RED) [WCAD19] method is used to learn the imitation reward, replacing the adversarial one in SAM. RED is based on random network distillation (RND) [BESK18], an exploration method using the prediction error of a learned network



(a) Evolution of return values *(higher is better)* (b) Final return values at timeout *(higher is better)*

Figure 3.5.5: Evaluation of several alternate reward formulations. Legend described in text. Runtime is 12 hours.

against a random fixed target as a measure of novelty, and use it to craft a reward bonus. Instead of updating the network while training to keep the novelty estimate tuned to the current exploration level of the agent, RED trains the RND predictor network to predict the random fixed target on the expert dataset *before* training the policy. RED then uses the prediction error to assemble a reward signal for the imitation agent, who is rewarded *more* if the actions it picks are deemed *not novel*, as that means the agent's occupancy measure matches the occupancy of what has been seen before, *i.e.* the expert dataset. As such, RED is a technique that rewards the agent for matching the distribution support of the expert policy π_e. Note, as opposed to adversarial imitation, the RED reward is not updated during training, which technically protects it from overfitting. *"PU"* means that we learn the reward via adversarial imitation, but using the discriminator objective recently proposed in positive-unlabeled (PU) GAIL [XD19]. Briefly, the method considers that while the expert-generated samples are positive-labels, the agent-generated ones are unlabeled (as opposed to negative-labeled). Intuitively, it should prevent the discriminator overfitting on irrelevant features when it becomes difficult for the discriminator to tell agent and expert apart.

The wrapping mechanism — consisting in wrapping the *absorbing* transitions, which we described in Section 3.4 — is used in every experiment reported in Figure 3.5.5, *including RED*. In addition, note, we only use GP in the adversarial context we introduced it in. We do not use GP with RED. Each technique is re-implemented based on the associated paper, with the same hyper-parameters, with the exception of RED: instead of using the per-environment scale for the prediction loss on which the RED reward is built, we keep a running estimate of the standard deviation of this prediction loss and rescale said prediction loss with its running standard deviation. This modification is consistent with the rescaling done in the paper RED is based on RND. By contrast, the per-environment scales in

RED's official implementation span several orders of magnitude (four). We here opt for environment-agnostic methods.

The results in FIGURE 3.5.5 show that the wrapping techniques introduced in [KAD+19] and described in SECTION 3.4 increases performance overall. Like we have shown before in FIGURES 3.5.1, 3.5.2, and 3.5.3, not using GP causes a considerable drop in performance. PU prevents the agent to learn an expert-like policy, in every environment. Note, while the comparison is fair, PU was introduced in *visual* tasks. In particular, we see that, in `Hopper`, PU's empirical return hits a plateau at about 1000 reward units (*abbrv.* r.u.). We observe the exact same phenomenon with RED, for which it occurs in *every* environment. This is caused by the agent being stuck performing the same sub-optimal actions, accumulating sub-optimal outcomes until episode termination artificially triggered by timeout. The agent exploits the fact that it has a lifetime upper-bounded by said timeout and is therefore *biased* by its *survival* (survival bias, *cf.* SECTION 3.4). The RED agents are in effect staying alive until termination, and therefore avoid falling down (organic trigger) until the timeout (artificial trigger) is reached. While the reward used in RED is not negative, the agent quickly reaches a performance level at which all the rewards are almost identical — since the RED reward is trained *beforehand*, with no chance of adaptive tuning like training the reward *at the same time* allows in this work, and since RED's score is based on how the agent and expert distribution match. Once the agent is similar enough to the expert, it always gets the same rewards and has therefore no incentive to resemble the expert with higher fidelity. Instead, it is content and just tries to live through the episode. This propensity to survival bias explains why such care was taken to hand-tune its scale. Finally, even though wrapping absorbing transitions generally improves performance, FIGURE 3.5.5 shows that survival bias is avoided even *without* it (occurrence in `Hopper` has been overcome).

The results in FIGURE 3.5.2 provide empirical evidence that enforcing Lipschitz-continuity on D_φ over the input space via the gradient regularization (*cf.* EQ. 3.15) is *necessary* and *sufficient* for the agent to achieve expert performance in the considered off-policy setting. We therefore ask the question: is the positive impact that GP has on training imitation policies via bootstrapping explained *a)* by its *direct* effect on the reward smoothness, or *b)* by its *indirect* effect on the state-action value smoothness? We argue that *both* contribute to the stability and performance of the studied method. While point *a)* is intuitive from the analyses laid out in SECTION 3.5.1, 3.5.2, and 3.5.3, we believe that point *b)* deserves further analysis and discussion. As such, we derive theoretical results to qualify, both qualitatively and quantitatively, the Lipschitz-continuity that is potentially *implicitly enforced* on the state-action value when assuming the Lipschitz-continuity of the reward. These results are reported in SECTION 3.6.1, and will hopefully help us answer the previous question. A discussion of the *indirect* effect and how it compares to the direct effect implemented by target smoothing is carried out in SECTION 3.6.2.4.

3.6 Pushing the analysis further: robustness guarantees and provably more robust extension

3.6.1 Robustness guarantees: state-action value Lipschitzness

In this section, we ultimately show that enforcing a Lipschitzness constraint on the reward r_φ has the effect of enforcing a Lipschitzness constraint on the associated state-action value Q_φ. Note, Q_φ is the *real* Q-value derived from r_φ, while Q_ω is a function approximation of it. We discuss this point in more detail in Section 3.6.2. We characterize and discuss the conditions under which such result is satisfied, as well as how the exhibited Lipschitz constant for Q_φ relates to the one enforced on r_φ. We work in the *episodic* setting, *i.e.* with a finite-horizon T, which is achieved by assuming that $\gamma = 0$ once an absorbing state is reached. Note, since we optimize over mini-batches in practice, nothing guarantees that the Lipschitz constraint is satisfied by the learned function approximation *globally* across the whole joint space $\mathcal{S} \times \mathcal{A}$, at every training iteration. In such setting, we are therefore reduced to *local* Lipschitzness, defined as Lipschitzness in neighborhoods around samples at which the constraint is applied. The provenance of these samples is not the focus of this theoretical section and assume they are agent-generated. We study the effect of enforcing Lipschitzness constraints on other data distributions in Section 3.6.3.

Notations. Given a function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^d$, taking the pair of vectors (x, y) as inputs, we denote by $\nabla_{x,y} f$ the pair of Jacobians associated with x and y, $\nabla_x f$ and $\nabla_y f$ respectively, which are rectangular matrices in $\mathbb{R}^{d\times n}$ and $\mathbb{R}^{d\times n}$ respectively. Now that the stable concepts and notations have been laid out, we introduce the variables x_i and y_i, indexed by $i \in \mathcal{I} \subseteq \mathbb{N}$. Note, indices i's' do not depict different occurrences of the x variable: the x_i's and y_i's are distinct variables. These families of variables will enable us to formalize the Jacobian of f with respect to (x_i, y_i) evaluated at $(x_{i'}, x_{i'})$, defined as $\left(\mathrm{d}f(x_{i'}, y_{i'}) \,/\, \mathrm{d}x_i \,,\ \mathrm{d}f(x_{i'}, y_{i'}) \,/\, \mathrm{d}y_i \right)$, where $i' \in \mathcal{I}, i' \geq i$. To lighten the notations, we overload the symbol ∇ and introduce the shorthands $\nabla^i_x[f]_{i'} := \mathrm{d}f(x_{i'}, y_{i'}) \,/\, \mathrm{d}x_i$ and $\nabla^i_y[f]_{i'} := \mathrm{d}f(x_{i'}, y_{i'}) \,/\, \mathrm{d}y_i$. By analogy, the shorthand $\nabla^i_{x,y}[f]_{i'}$ denotes the pair $(\nabla^i_x[f]_{i'}, \nabla^i_y[f]_{i'})$. In this work, the difference between the index of derivation i and the index of evaluation i', $i - i' \leq 0$ will be referred to as *gap*. We use $\|\cdot\|_F$ to denote the Frobenius norm, which a) is naturally defined over rectangular matrices in $\mathbb{R}^{m\times n}$ and b) is *sub-multiplicative*: $\|UV\|_F \leq \|U\|_F \|V\|_F$, for U and V rectangular with compatible sizes (provable via Cauchy-Schwarz inequality). In proofs, we use "$\otimes$" for matrix multiplication, to avoid collisions with the scalar product.

Lemma 3.6.1 (recursive inequality — induction step)**.** *Let the MDP with which the agent interacts be deterministic, with the dynamics of the environment determined by the function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$.*

The agent follows a deterministic policy $\mu : \mathcal{S} \to \mathcal{A}$ to map states to actions, and receives rewards from $r_\varphi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ upon interaction. The functions f, μ and r_φ need be C^0 and differentiable over their respective input spaces. This property is satisfied by the usual neural network function approximators. The "almost-everywhere" case can be derived from this lemma without major changes (relevant when at least one activation function is only differentiable almost-everywhere, ReLU). **(a)** *Under the previous assumptions, for $k \in [0,\, T-k-1] \cap \mathbb{N}$ the following* ***recursive inequality*** *is verified:*

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \leq C_t \, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \tag{3.16}$$

where $C_t \coloneqq A_t^2 \max(1, B_{t+1}^2)$, A_t and B_t being defined as the supremum norms associated with the Jacobians of f and μ respectively, with values in $\mathbb{R} \cup \{+\infty\}$:

$$\forall t \in [0,\, T] \cap \mathbb{N}, \quad \begin{cases} A_t \coloneqq \|\nabla^t_{s,a}[f]_t\|_\infty = \sup\left\{\|\nabla^t_{s,a}[f]_t\|_F \,:\, (s_t, a_t) \in \mathcal{S} \times \mathcal{A}\right\} \\ B_t \coloneqq \|\nabla^t_{s}[\mu]_t\|_\infty = \sup\left\{\|\nabla^t_{s}[\mu]_t\|_F \,:\, s_t \in \mathcal{S}\right\} \end{cases} \tag{3.17}$$

(b) *Additionally, by introducing* ***time-independent*** *upper bounds $A, B \in \mathbb{R} \cup \{+\infty\}$ such that $\forall t \in [0,\, T] \cap \mathbb{N}$, $A_t \leq A$ and $B_t \leq B$, the recursive inequality becomes:*

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \leq C \, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \tag{3.18}$$

where $C \coloneqq A^2 \max(1, B^2)$ is the time-independent counterpart of C_t.

Proof of Lemma 3.6.1 (a). First, we take the derivative with respect to each variable separately:

$$\begin{aligned}
\nabla^t_s[r_\varphi]_{t+k+1} &= \mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1}) \,/\, \mathrm{d}s_t && (3.19)\\
&= \mathrm{d}r_\varphi\big(f(s_{t+k}, a_{t+k}), \mu(f(s_{t+k}, a_{t+k}))\big) \,/\, \mathrm{d}s_t && (3.20)\\
&= \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}s_{t+1}} \otimes \frac{\mathrm{d}f(s_t, a_t)}{\mathrm{d}s_t} && (3.21)\\
&\quad + \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}a_{t+1}} \otimes \frac{\mathrm{d}\mu(s_{t+1})}{\mathrm{d}s_{t+1}} \otimes \frac{\mathrm{d}f(s_t, a_t)}{\mathrm{d}s_t} \\
&= \nabla^{t+1}_s[r_\varphi]_{t+k+1} \otimes \nabla^t_s[f]_t + \nabla^{t+1}_a[r_\varphi]_{t+k+1} \otimes \nabla^{t+1}_s[\mu]_{t+1} \otimes \nabla^t_s[f]_t && (3.22)
\end{aligned}$$

$$\nabla_a^t [r_\varphi]_{t+k+1} = \mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1}) \,/\, \mathrm{d}a_t \tag{3.23}$$

$$= \mathrm{d}r_\varphi\big(f(s_{t+k}, a_{t+k}), \mu(f(s_{t+k}, a_{t+k}))\big) \,/\, \mathrm{d}a_t \tag{3.24}$$

$$= \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}s_{t+1}} \otimes \frac{\mathrm{d}f(s_t, a_t)}{\mathrm{d}a_t} + \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}a_{t+1}} \otimes \frac{\mathrm{d}\mu(s_{t+1})}{\mathrm{d}s_{t+1}} \otimes \frac{\mathrm{d}f(s_t, a_t)}{\mathrm{d}a_t} \tag{3.25}$$

$$= \nabla_s^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_a^t [f]_t + \nabla_a^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1} [\mu]_{t+1} \otimes \nabla_a^t [f]_t \tag{3.26}$$

By assembling the norm with respect to both input variables, we get:

$$\|\nabla_{s,a}^t [r_\varphi]_{t+k+1}\|_F^2$$

$$= \|\nabla_s^t [r_\varphi]_{t+k+1}\|_F^2 + \|\nabla_a^t [r_\varphi]_{t+k+1}\|_F^2 \tag{3.27}$$

$$= \|\nabla_s^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_s^t [f]_t + \nabla_a^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1} [\mu]_{t+1} \otimes \nabla_s^t [f]_t\|_F^2 \tag{3.28}$$
$$+ \|\nabla_s^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_a^t [f]_t + \nabla_a^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1} [\mu]_{t+1} \otimes \nabla_a^t [f]_t\|_F^2$$

$$\leq \|\nabla_s^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_s^t [f]_t\|_F^2 \qquad \blacktriangleright \textit{ triangular inequality} \tag{3.29}$$
$$+ \|\nabla_a^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1} [\mu]_{t+1} \otimes \nabla_s^t [f]_t\|_F^2$$
$$+ \|\nabla_s^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_a^t [f]_t\|_F^2$$
$$+ \|\nabla_a^{t+1} [r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1} [\mu]_{t+1} \otimes \nabla_a^t [f]_t\|_F^2$$

$$\leq \|\nabla_s^{t+1} [r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^t [f]_t\|_F^2 \qquad \blacktriangleright \textit{ sub-multiplicativity} \tag{3.30}$$
$$+ \|\nabla_a^{t+1} [r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^{t+1} [\mu]_{t+1}\|_F^2 \, \|\nabla_s^t [f]_t\|_F^2$$
$$+ \|\nabla_s^{t+1} [r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_a^t [f]_t\|_F^2$$
$$+ \|\nabla_a^{t+1} [r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^{t+1} [\mu]_{t+1}\|_F^2 \, \|\nabla_a^t [f]_t\|_F^2$$

$$= \|\nabla_s^{t+1} [r_\varphi]_{t+k+1}\|_F^2 \left(\|\nabla_s^t [f]_t\|_F^2 + \|\nabla_a^t [f]_t\|_F^2\right) \qquad \blacktriangleright \textit{ factorization} \tag{3.31}$$
$$+ \|\nabla_a^{t+1} [r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^{t+1} [\mu]_{t+1}\|_F^2 \left(\|\nabla_s^t [f]_t\|_F^2 + \|\nabla_a^t [f]_t\|_F^2\right)$$

$$= \|\nabla_s^{t+1} [r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_{s,a}^t [f]_t\|_F^2 \qquad \blacktriangleright \textit{ total norm} \tag{3.32}$$
$$+ \|\nabla_a^{t+1} [r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^{t+1} [\mu]_{t+1}\|_F^2 \, \|\nabla_{s,a}^t [f]_t\|_F^2$$

Let A_t, B_t and C_t be time-dependent quantities defined as:

$$\forall t \in [0, T] \cap \mathbb{N}, \quad \begin{cases} A_t \coloneqq \|\nabla_{s,a}^t [f]_t\|_\infty = \sup\left\{\|\nabla_{s,a}^t [f]_t\|_F \,:\, (s_t, a_t) \in \mathcal{S} \times \mathcal{A}\right\} \\ B_t \coloneqq \|\nabla_s^t [\mu]_t\|_\infty = \sup\left\{\|\nabla_s^t [\mu]_t\|_F \,:\, s_t \in \mathcal{S}\right\} \\ C_t \coloneqq A_t^2 \max(1, B_{t+1}^2) \end{cases} \tag{3.33}$$

Finally, by substitution, we obtain:

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \leq A_t^2 \|\nabla^{t+1}_s [r_\varphi]_{t+k+1}\|_F^2 + A_t^2 B_{t+1}^2 \|\nabla^{t+1}_a [r_\varphi]_{t+k+1}\|_F^2 \tag{3.34}$$

$$\leq A_t^2 \max(1, B_{t+1}^2) \left(\|\nabla^{t+1}_s [r_\varphi]_{t+k+1}\|_F^2 + \|\nabla^{t+1}_a [r_\varphi]_{t+k+1}\|_F^2\right) \tag{3.35}$$

$$= A_t^2 \max(1, B_{t+1}^2) \, \|\nabla^{t+1}_{s,a} [r_\varphi]_{t+k+1}\|_F^2 \qquad \blacktriangleright \textit{ total norm} \tag{3.36}$$

$$= C_t \, \|\nabla^{t+1}_{s,a} [r_\varphi]_{t+k+1}\|_F^2 \qquad \blacktriangleright \; C_t \textit{ definition} \tag{3.37}$$

which concludes the proof of Lemma 3.6.1 *(a)*. □

Proof of Lemma 3.6.1 (b). By introducing time-independent upper bounds A and B such that $A_t \leq A$ and $B_t \leq B \; \forall t \in [0, T] \cap \mathbb{N}$, as well as $C \coloneqq A^2 \max(1, B^2)$, we obtain, by substitution in eq 3.36:

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \leq A^2 \max(1, B^2) \, \|\nabla^{t+1}_{s,a} [r_\varphi]_{t+k+1}\|_F^2 \tag{3.38}$$

$$= C \, \|\nabla^{t+1}_{s,a} [r_\varphi]_{t+k+1}\|_F^2 \tag{3.39}$$

which concludes the proof of Lemma 3.6.1 *(b)*. □

Lemma 3.6.1 tells us how the norm of the Jacobian associated with a gap between derivation and evaluation indices equal to $t + 1$ relate to the norm of the Jacobian associated with a gap equal to t. We will use this recursive property to prove our first theorem, Theorem 3.6.2. Additionally, from this point forward, we will use the time-independent upper-bounds exclusively, *i.e.* Lemma 3.6.1 *(b)*.

Theorem 3.6.2 (gap-dependent reward Lipschitzness)**.** *In addition to the assumptions laid out in lemma 3.6.1, we assume that the function r_φ is δ-Lipschitz over $\mathcal{S} \times \mathcal{A}$. Since r_φ is C^0 and differentiable over $\mathcal{S} \times \mathcal{A}$, this assumption can be written as $\|\nabla^u_{s,a}[r_\varphi]_u\|_F \leq \delta$, where $u \in [0, T] \cap \mathbb{N}$.* **(a)** *Then, under these assumptions, the following is verified:*

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k}\|_F^2 \leq \delta^2 \prod_{u=0}^{k-1} C_{t+u} \tag{3.40}$$

where $k \in [0, T] \cap \mathbb{N}$ and C_v is defined as in Lemma 3.6.1 (a), $\forall v \in [0, T] \cap \mathbb{N}$. **(b)** *Additionally, by involving the time-independent upper bounds introduced in Lemma 3.6.1 (b), we have the following:*

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k}\|_F^2 \leq C^k \, \delta^2 \tag{3.41}$$

where $k \in [0, T] \cap \mathbb{N}$ and C is defined as in Lemma 3.6.1 (b).

Proof of Theorem 3.6.2 (a). We will prove Theorem 3.6.2 *(a)* by induction.

Let us introduce the dummy variable v, along with the induction hypothesis for v:

$$\|\nabla^t_{s,a}[r_\varphi]_{t+v}\|^2_F \leq \delta^2 \prod_{u=0}^{v-1} C_{t+u} \qquad \blacktriangleright \textit{ induction hypothesis} \tag{3.42}$$

where v represents the gap between the derivation timestep and the evaluation timestep.

Step 1: initialization. When the gap $v = 0$, EQ 3.42 becomes $\|\nabla^t_{s,a}[r_\varphi]_t\|^2_F \leq \delta^2, \forall t \in [0, T] \cap \mathbb{N}$, which is trivially verified since it exactly corresponds to THEOREM 3.6.2's main assumption.

Step 2: induction. Let us assume that EQ 3.42 is verified for v fixed, and show that EQ 3.42 is satisfied when the gap is equal to $v + 1$.

$$\|\nabla^t_{s,a}[r_\varphi]_{t+v+1}\|^2_F \leq C_t \, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+v+1}\|^2_F \qquad \blacktriangleright \textit{ LEMMA 3.6.1 (a)} \tag{3.43}$$

$$\leq C_t \, \delta^2 \prod_{u=0}^{v-1} C_{t+1+u} \qquad \blacktriangleright \textit{ EQ 3.42 since gap is v, at } t+1 \tag{3.44}$$

$$= C_t \, \delta^2 \prod_{u=1}^{v} C_{t+u} \qquad \blacktriangleright \textit{ index shift} \tag{3.45}$$

$$= \delta^2 \prod_{u=0}^{v} C_{t+u} \qquad \blacktriangleright \textit{ repack product} \tag{3.46}$$

EQ 3.42 is therefore satisfied for $v + 1$ when assumed at v, which proves the induction step.

Step 3: conclusion. Since EQ 3.42 has been verified for both the initialization and induction steps, the hypothesis is valid $\forall v \in [0, T] \cap \mathbb{N}$, which concludes the proof of THEOREM 3.6.2 *(a)*. □

Proof of THEOREM 3.6.2 (b). We will prove THEOREM 3.6.2 *(b)* by induction.

Let us introduce the dummy variable v, along with the induction hypothesis for v:

$$\|\nabla^t_{s,a}[r_\varphi]_{t+v}\|^2_F \leq C^v \, \delta^2 \qquad \blacktriangleright \textit{ induction hypothesis} \tag{3.47}$$

where v represents the gap between the derivation timestep and the evaluation timestep.

Step 1: initialization. When the gap $v = 0$, EQ 3.47 becomes $\|\nabla^t_{s,a}[r_\varphi]_t\|^2_F \leq \delta^2, \forall t \in [0, T] \cap \mathbb{N}$, which is trivially verified since it exactly corresponds to THEOREM 3.6.2's main assumption.

Step 2: induction. Let us assume that EQ 3.47 is verified for v fixed, and show that EQ 3.47 is satisfied

when the gap is equal to $v+1$.

$$\|\nabla^t_{s,a}[r_\varphi]_{t+v+1}\|_F^2 \leq C\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+v+1}\|_F^2 \qquad \blacktriangleright \textit{Lemma 3.6.1 (b)} \tag{3.48}$$

$$\leq C\,C^v\,\delta^2 \qquad \blacktriangleright \textit{eq 3.47 since gap is v} \tag{3.49}$$

$$= C^{v+1}\,\delta^2 \tag{3.50}$$

eq 3.47 is therefore satisfied for $v+1$ when assumed at v, which proves the induction step.

Step 3: conclusion. Since eq 3.47 has been verified for both the initialization and induction steps, the hypothesis is valid $\forall v \in [0,\,T] \cap \mathbb{N}$, which concludes the proof of Theorem 3.6.2 *(b)*. □

This result shows that when there is a gap k between the derivation and evaluation indices, the norm of the Jacobian of r_φ is upper-bounded by a *gap-dependent* quantity equal to $\sqrt{C^k}\delta$, over the entire input space. Crucially, this property applies if and only if the gap between the timestep of the derivation variable and the timestep of the evaluation variable is equal to 0, hence the use of the same letter u in the assumption formulation.

Theorem 3.6.3 (state-action value Lipschitzness)**.** *We work under the assumptions laid out in both Lemma 3.6.1 and Theorem 3.6.2, and repeat the main lines here for Theorem 3.6.3 to be self-contained: a) The functions f, μ and r_φ are C^0 and differentiable over their respective input spaces, and b) the function r_φ is δ-Lipschitz over $\mathcal{S}\times\mathcal{A}$, i.e. $\|\nabla^u_{s,a}[r_\varphi]_u\|_F \leq \delta$, where $u \in [0,\,T]\cap\mathbb{N}$. Then the quantity $\nabla^u_{s,a}[Q_\varphi]_u$ exists $\forall u \in [0,\,T]\cap\mathbb{N}$, and verifies:*

$$\|\nabla^t_{s,a}[Q_\varphi]_t\|_F \leq \begin{cases} \delta\sqrt{\dfrac{1-\left(\gamma^2 C\right)^{T-t}}{1-\gamma^2 C}}, & \text{if } \gamma^2 C \neq 1 \\ \delta\sqrt{T-t}, & \text{if } \gamma^2 C = 1 \end{cases} \tag{3.51}$$

$\forall t \in [0,\,T]\cap\mathbb{N}$, *where* $C \coloneqq A^2\max(1, B^2)$, *with A and B time-independent upper bounds of $\|\nabla^t_{s,a}[f]_t\|_\infty$ and $\|\nabla^t_s[\mu]_t\|_\infty$ respectively (see eq 3.33 for definitions of the supremum norms).*

Proof of Theorem 3.6.3. With finite horizon T, we have $Q_\varphi(s_t, a_t) \coloneqq \sum_{k=0}^{T-t-1}\gamma^k\, r_\varphi(s_{t+k}, a_{t+k})$, $\forall t \in [0,\,T]\cap\mathbb{N}$, since f, μ, and r_φ are all deterministic (no expectation). Additionally, since r_φ is assumes to be C^0 and differentiable over $\mathcal{S}\times\mathcal{A}$, Q_φ is by construction also C^0 and differentiable over $\mathcal{S}\times\mathcal{A}$. Consequently, $\nabla^u_{s,a}[Q_\varphi]_u$ exists, $\forall u \in [0,\,T]\cap\mathbb{N}$. Since both r_φ and Q_φ are scalar-valued (their output space is $\mathbb{R}$), their Jacobians are the same as their gradients. We can therefore use the linearity of the

gradient operator: $\nabla^t_{s,a}[Q_\varphi]_t = \sum_{k=0}^{T-t-1} \gamma^k \nabla^t_{s,a}[r_\varphi]_{t+k}, \forall t \in [0, T] \cap \mathbb{N}$.

$$\|\nabla^t_{s,a}[Q_\varphi]_t\|_F^2 = \left\| \sum_{k=0}^{T-t-1} \gamma^k \nabla^t_{s,a}[r_\varphi]_{t+k} \right\|_F^2 \qquad \blacktriangleright \textit{ operator's linearity} \tag{3.52}$$

$$\leq \sum_{k=0}^{T-t-1} \gamma^{2k} \|\nabla^t_{s,a}[r_\varphi]_{t+k}\|_F^2 \qquad \blacktriangleright \textit{ triangular inequality} \tag{3.53}$$

$$\leq \sum_{k=0}^{T-t-1} \gamma^{2k} C^k \delta^2 \qquad \blacktriangleright \textsc{Theorem 3.6.2} \tag{3.54}$$

$$= \delta^2 \sum_{k=0}^{T-t-1} \left(\gamma^2 C\right)^k \tag{3.55}$$

When $\gamma^2 C = 1$, we obtain $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F^2 = \delta^2 (T - t)$. On the other hand, when $\gamma^2 C \neq 1$:

$$\|\nabla^t_{s,a}[Q_\varphi]_t\|_F^2 \leq \delta^2 \frac{1 - \left(\gamma^2 C\right)^{T-t}}{1 - \gamma^2 C} \qquad \blacktriangleright \textit{ finite sum of geometric series} \tag{3.56}$$

$$\implies \quad \|\nabla^t_{s,a}[Q_\varphi]_t\|_F^2 \leq \begin{cases} \delta^2 \dfrac{1 - \left(\gamma^2 C\right)^{T-t}}{1 - \gamma^2 C}, & \text{if } \gamma^2 C \neq 1 \\ \delta^2 (T - t), & \text{if } \gamma^2 C = 1 \end{cases} \tag{3.57}$$

By applying $\sqrt{\cdot}$ (monotonically increasing) to the inequality, we obtain the claimed result. □

Finally, we derive a corollary from Theorem 3.6.3 corresponding to the infinite-horizon regime.

Corollary 3.6.3.1 (infinite-horizon regime)**.** *Under the assumptions of Theorem 3.6.3, including that r_φ is δ-Lipschitz over $\mathcal{S} \times \mathcal{A}$, and assuming that $\gamma^2 C < 1$, we have, in the infinite-horizon regime:*

$$\|\nabla^t_{s,a}[Q_\varphi]_t\|_F \leq \frac{\delta}{\sqrt{1 - \gamma^2 C}} \tag{3.58}$$

which translates into Q_φ being $\frac{\delta}{\sqrt{1-\gamma^2 C}}$-Lipschitz over $\mathcal{S} \times \mathcal{A}$.

Proof of Corollary 3.6.3.1. We now have $Q_\varphi(s_t, a_t) \coloneqq \sum_{k=0}^{+\infty} \gamma^k r_\varphi(s_{t+k}, a_{t+k}), \forall t \in [0, T] \cap \mathbb{N}$, since f, μ, and r_φ are all deterministic and are now working working under the infinite-horizon regime. Considering the changes in Q_φ's definition, the first part of the proof can be done by analogy with the proof of Theorem 3.6.3, until eq 3.55, which is our starting point. In this regime, $\gamma^2 C \geq 1$ yields an infinite sum in eq 3.55, which results in an uninformative (because infinite) upper-bound

on $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F$. On the other hand, when $\gamma^2 C < 1$ (note, we always have $\gamma^2 C \geq 0$ by definition), the infinite sum in EQ 3.55 is defined. Since we have shown that $\gamma^2 C < 1$ is the only setting in which the sum is defined, we continue from the infinite-horizon version of EQ 3.55 with $\gamma^2 C < 1$ onwards. Hence,

$$\|\nabla^t_{s,a}[Q_\varphi]_t\|_F^2 \leq \delta^2 \sum_{k=0}^{+\infty} \left(\gamma^2 C\right)^k = \frac{\delta^2}{1-\gamma^2 C} \qquad \blacktriangleright \textit{ infinite sum of geometric series} \tag{3.59}$$

Using $\sqrt{\cdot}$ (monotonically increasing) on both sides concludes the proof of COROLLARY 3.6.3.1. □

To conclude the section, we now give interpretations of the derived theoretical results, discuss the implications of our results, and also exhibit to what extent they transfer to the practical setting.

3.6.2 DISCUSSION I: IMPLICATIONS AND LIMITATIONS OF THE THEORETICAL GUARANTEES

3.6.2.1 FUNCTION APPROXIMATION BIAS

THEOREM 3.6.3 exhibits the Lipschitz constant of Q_φ when r_φ is δ-Lipschitz. In practice however, the state-action value (or value function) is usually modeled by a neural network, and learned via gradient descent either by using a Monte-Carlo estimate of the collected return as regression target, or by bootstrapping using a subsequent model estimate [Sut88]. We therefore have access to a learned estimate Q_ω, as opposed to the real state-action value Q_φ. As such, the results derived in THEOREM 3.6.3 will transfer favorably into the function approximation setting as Q_ω becomes a better parametric estimate of Q_φ. Note, the reward is denoted by r_φ for the reader to easily distinguish it from the *black-box* reward traditionally returned by the environment. Albeit arbitrary, the notation r_φ allows for the reward to be modeled by a neural network parameterized by the weights φ, and learned via gradient descent, as is indeed the case in this work. Crucially, having control over r_φ in practice allows for the enforcement of constraints, making the δ-Lipschitzness assumption in THEOREM 3.6.2, THEOREM 3.6.3 and COROLLARY 3.6.3.1 practically satisfiable via gradient penalization 3.5.4. It is crucial to note that, while function approximation creates a gap between theory and practice for the Q-value (*worse* when bootstrapping), there is a meaningfully lesser gap for the reward as the δ-Lipschitzness constraint is directly enforced on the parametric reward r_φ.

3.6.2.2 VALUE LIPSCHITZNESS

In COROLLARY 3.6.3.1 we showed that $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F \leq \delta/\sqrt{1-\gamma^2 C}$, in the infinite-horizon regime, when r_φ is assumed δ-Lipschitz over $\mathcal{S} \times \mathcal{A}$, and assuming $\gamma^2 C < 1$. In other words, in this setting, enforcing r_φ to be δ-Lipschitz causes Q_φ to be Δ_∞-Lipschitz, where $\Delta_\infty := \delta/\sqrt{1-\gamma^2 C}$, $C :=$

$A^2 \max(1, B^2)$, and A, B are upper-bounds of $\|\nabla^t_{s,a}[f]_t\|_\infty$, $\|\nabla^t_s[\mu]_t\|_\infty$. Starting from the assumption that $\gamma^2 C < 1$, we trivially arrive at $\sqrt{1-\gamma^2 C} < 1$, then $1/\sqrt{1-\gamma^2 C} > 1$, and since $\delta \geq 0$ by definition (*cf.* Section 3.5.4), we finally get $\Delta_\infty > \delta$. Without loss of generality, consider the case in which r_φ is *not* a contraction, *i.e.* r_φ is δ-Lipschitz C^0 over $\mathcal{S} \times \mathcal{A}$, with $\delta \geq 1$. As a result, $\Delta_\infty > \delta \geq 1$, *i.e.* $\Delta_\infty > 1$, which means that, under the considered conditions, Q_φ is *not* a contraction over $\mathcal{S} \times \mathcal{A}$ either. The latter naturally extends to any $u \in \mathbb{R}_+$ that lower-bounds δ: if $\delta > u$, then $\Delta_\infty > u, \forall u \in \mathbb{R}_+$. Lipschitz functions and especially contractions are at the core of many fundamental results in dynamics programming, hence also in reinforcement learning. Crucially, the Bellman operator being a contraction causes a fixed point iterative process, such as value iteration [SB98], to converge to a unique fixed point whatever the starting iterate of Q. Since we learn Q_φ with temporal-difference learning [Sut88] via a bootstrapped objective, the convergence of our method is a direct consequence of the contractant nature of the Bellman operator. As such the Lipschitzness-centric analysis laid out in this section is complementary to the latter. It provides a characterization of Q_φ's Lipschitzness over the input space $\mathcal{S} \times \mathcal{A}$ as opposed to over iterates, *i.e.* time. As such, our analysis therefore does not give convergence guarantees of an iterative process, which are already carried over from temporal-difference learning at the core of our algorithm. Rather, we provide *variation upper-bounds* for Q_φ when r_φ has upper-bounded variations: if r_φ is δ-Lipschitz, then Q_φ is Δ_∞-Lipschitz. *In fine*, this result has an immediate corollary, derived previously in this block: if the variations of r_φ are lower-bounded by δ, then the variations of Q_φ are lower-bounded by $\Delta_\infty > \delta$.

3.6.2.3 Compounding variations

The relative position of $\gamma^2 C$ with respect to 1 is instrumental in the behavior of the exhibited variation bounds, in both the finite- and infinite-horizon settings. In the latter, we see that the upper-bound gets to infinity when $\gamma^2 C$ (non-negative by definition, and lower than 1 as necessary condition for the infinite sum to exist) gets closer to 1 from below. In the former, we focus on the $\gamma^2 C \neq 1$ case, as in the other case, the bound does not even depend on $\gamma^2 C$. As such, we study the value of $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F$'s upper-bound in the finite-horizon setting when $\gamma^2 C \neq 1$, dubbed $\Delta_t \coloneqq \delta\sqrt{1-(\gamma^2 C)^{T-t}}/1-\gamma^2 C$. Beforehand, we would remind the reader how the bounded quantity should be behave throughout an episode. Since Q_φ is defined as the expected sum of *future* rewards r_φ, predicting such value should get increasingly tainted with uncertainty as it tries to predict across long time ranges. As such, predicting Q_φ at time $t = 0$ is the most challenging, as it corresponds to the value of an entire trajectory, whereas predicting Q_φ at time $t = T$ is the easiest (equal to last reward r_φ). Higher horizons T consequently make the prediction task more difficult, as do discount factors γ closer to 1. We now discuss Δ_t. As long as $\gamma^2 C \neq 1$, Δ_t gets to 0 as t gets to T. This is consistent with the previous reminder: as t gets to T, the Q_φ estimation task becomes easier, hence the variation bound (Δ_t) due to prediction uncertainty

should decrease to 0. As t gets to 0 however, the behavior of Δ_t depends on the value of $\gamma^2 C$: if $\gamma^2 C \gg 1$, Δ_t explodes to infinity, whereas for reasonable values of $\gamma^2 C$, Δ_t does not. Since $C \coloneqq A^2 \max(1, B^2)$, $\gamma^2 C \gg 1$ translates to $((\exists u > 1) : A \gg u) \vee ((\exists v > 1) : B \gg v)$. Let us assume that A (B) not only upper-bounds every A_t (B_t) but is also the tightest time-independent bound: $A \coloneqq A_{t'}$ ($B \coloneqq B_{t''}$) where $t' = \operatorname{argmax}_t A_t$ ($t'' = \operatorname{argmax}_t B_t$). We then have $((\exists u > 1)(\exists t') : A_{t'} \gg u) \vee ((\exists v > 1)(\exists t'') : B_{t''} \gg v)$, *i.e.* $((\exists u > 1)(\exists t') : \|\nabla^{t'}_{s,a}[f]_{t'}\|_\infty \gg u) \vee ((\exists v > 1)(\exists t'') : \|\nabla^{t''}_{s}[\mu]_{t''}\|_\infty \gg v)$ over $\mathcal{S} \times \mathcal{A}$. Note, the "or" is inclusive. In other words, if the variations (in space) of the policy or the dynamics are large in the early stage of an episode ($0 \leq t \ll T$), then Δ_t (variation bound on Q_φ) explodes. The exhibited phenomenon is somewhat reminiscent of the compounding of errors isolated in [RB10].

3.6.2.4 Is value Lipschitzness enough?

We showed that under mild conditions, and in finite- and infinite- horizon regimes, r_φ Lipschitzness implies Q_φ Lipschitzness, *i.e.* that if similar state-action are mapped to similar rewards by r_φ, then Q_φ also maps then to similar state-action values. This regularization desideratum is evocative of the *target policy smoothing* add-on introduced in [FvHM18], already presented earlier in Section 3.4. In short, target policy smoothing perturbs the target action slightly. In effect, the temporal-difference optimization now fits the value estimate against an expectation of *similar* bootstrapped target value estimates. Forcing similar action to have similar values naturally smooths out the value estimate, which by definition emulates the enforcement of a Lipschitzness constraint on the value, and as such mitigates value overfitting which deterministic policies are prone to. While its smoothing effect on the value function is somewhat intuitive, we set out to investigate formally how target policy smoothing affects the optimization dynamics, and particularly to what extent it smooths out the state-action value landscape. Since the function approximator Q_ω is optimized as a supervised learning problem using the traditional squared loss criterion, we first study how perturbing the inputs with additive random noise, denoted by ξ, impacts the optimized criterion, and what kind of behavior it encourages in the predictive function. As such, to lighten the expressions, we consider the supervised criterion $C(x) \coloneqq (y - f(x))^2$, where $f(x)$ is the predicted vector at the input vector x, and y is the supervised target vector. We also consider, in line with [FvHM18], that the noise is sampled from a spherical zero-centered Gaussian distribution, omitting here that the noise is truncated for legibility, hence $\xi \sim \mathcal{N}(0, \sigma^2 I)$. The criterion injected with input noise is $C_\xi(x) \coloneqq C(x + \xi) = (y - f(x + \xi))^2$. Assuming the noise has small amplitude (further supporting the original truncation), we can write

the second-order Taylor series expansion of the perturbed criterion near $\xi = 0$, as a polynomial of ξ:

$$C_\xi(x) = C(x) + \sum_i \left.\frac{\partial C}{\partial x_i}\right|_x \xi_i + \frac{1}{2}\sum_i\sum_j \left.\frac{\partial^2 C}{\partial x_i \partial x_j}\right|_x \xi_i\xi_j + O(\|\xi\|^3) \tag{3.60}$$

where $\|\cdot\|$ denotes the euclidean norm in the appropriate vector space. From this point forward, we assume the noise has a small enough norm to allow the third term, $O(\|\xi\|^3)$, to be neglected. By integrating over the noise distribution, we obtain:

$$\int C_\xi(x)p(\xi)d\xi = C(x) + \sum_i \left.\frac{\partial C}{\partial x_i}\right|_x \int \xi_i p(\xi)d\xi + \frac{1}{2}\sum_i\sum_j \left.\frac{\partial^2 C}{\partial x_i \partial x_j}\right|_x \int \xi_i\xi_j p(\xi)d\xi \tag{3.61}$$

Since the noise is sampled from the zero-centered and spherical distribution $\mathcal{N}(0, \sigma^2 I)$, we have respectively that $\int \xi_i p(\xi)d\xi = 0$ and

$$\int \xi_i\xi_j p(\xi)d\xi = \int \xi_i^2 \delta_{ij} p(\xi)d\xi = \delta_{ij}\int \xi_i^2 p(\xi)d\xi = \delta_{ij}\sigma^2$$

, where δ_{ij} is the Kronecker symbol. By injecting these expressions in EQ. 3.61, we get:

$$\int C_\xi(x)p(\xi)d\xi = C(x) + \frac{\sigma^2}{2}\sum_i \left.\frac{\partial^2 C}{\partial x_i^2}\right|_x = C(x) + \frac{\sigma^2}{2}\,\mathrm{Tr}(H_x\, C) \tag{3.62}$$

where $\mathrm{Tr}(H_x\, C)$ is the trace of the Hessian of the criterion C, *w.r.t.* the input variable x. We now want to express the exhibited regularizer $\mathrm{Tr}(H_x\, C))$ as a function of the derivatives of the prediction function f, and therefore calculate the consecutive derivative sums:

$$\sum_i \left.\frac{\partial C}{\partial x_i}\right|_x = -2\sum_i \big(y - f(x)\big)\left.\frac{\partial f}{\partial x_i}\right|_x \tag{3.63}$$

$$\sum_i \left.\frac{\partial^2 C}{\partial x_i^2}\right|_x = 2\sum_i \left[\left(\left.\frac{\partial f}{\partial x_i}\right|_x\right)^2 - \big(y - f(x)\big)\left.\frac{\partial^2 f}{\partial x_i^2}\right|_x\right] \tag{3.64}$$

hence,

$$\int C_\xi(x)p(\xi)d\xi = C(x) + \sigma^2\sum_i \left[\left(\left.\frac{\partial f}{\partial x_i}\right|_x\right)^2 - \big(y - f(x)\big)\left.\frac{\partial^2 f}{\partial x_i^2}\right|_x\right] \tag{3.65}$$

In fine, we can write, in a more condensed form:

$$\mathbb{E}_\xi[C(x+\xi)] = C(x) + \sigma^2\Big[\|\nabla_x f\|^2 - \mathrm{Tr}\left(C(x)H_x f\right)\Big] \tag{3.66}$$

The previous derivations — derived somewhat similarly in [Web94] and [Bis95] — show that minimizing the criterion with noise injected in the input is equivalent to minimizing the criterion without any noise *and* a regularizer containing norms of both the Jacobian and Hessian of the prediction function f. As raised in [Bis95], the second term of the regularizer is unsuitable for the design of a practically viable learning algorithm, since *a)* it involves prohibitively costly second-order derivatives, and *b)* it is not positive definite, and consequently not lower-bounded, which overall makes the regularizer a bad candidate for an optimization problem loss. Nevertheless, [Bis95] further shows that this regularization is equivalent to the use of a standard Tikhonov-like positive-definite regularization scheme involving *only* first-order derivatives, provided the noise has small amplitude — ensured here with a small σ and noise clipping. As such, the regularizer induced by the input noise ξ is equivalent to $\sigma^2\left[\|\nabla_x f\|^2\right]$, and by direct analogy, we can say that target policy smoothing induces an implicit regularizer on the TD objective, of the form $\sigma^2\left[\|\nabla_a Q_{\omega'}\|^2\right]$, Note, ω' are the target critic parameters, given that target policy smoothing adds noise to the target action, an input of target critic value $Q_{\omega'}$. By construction, the target parameters ω' slowly follow the online parameters ω (*cf.* Section 3.4). In addition, temporal-difference learning urges Q_ω to move closer to $Q_{\omega'}$ by design (*cf.* eq 3.4). Consequently, properties enforced on one set of parameters should *eventually* be transfered to the other, such that *in fine* both ω and ω' possess the given property only explicitly enforced on one (albeit delayed). Based on this line of reasoning, the temporal-difference learning dynamics and soft target updates should make the theoretically equivalent $\sigma^2\left[\|\nabla_a Q_{\omega'}\|^2\right]$ regularizer enforce smoothness on the online parameters ω too, even if it explicitly only constrains the target weights ω'. All in all, we have shown that target smoothing is equivalent to adding a regularizer to the temporal-difference error to minimize when learning Q_ω, where said regularizer is reminiscent of the gradient penalty regularizer, presented earlier in eq 3.15. As such, target smoothing *does* implement a gradient penalty regularization, but on Q_ω. Crucially, the gradient in the penalty is only taken *w.r.t.* the action dimension, but not *w.r.t.* the state dimension. In spite of the use of target policy smoothing in our method, it was not enough to yield stable learning behaviors, as shown in Section 3.5.5. Gradient penalization was an absolute necessity. Even though both methods encourage Q_ω to be smoother (directly in [FvHM18], and indirectly via reward Lipschitzness in this work), on its own, learning a smooth Q_ω estimate seems not to be *sufficient* for our method to work: learning a smooth r_φ estimate to serve as basis for Q_ω seems to be a *necessary* condition.

3.6.2.5 Indirect reward regularization

The theoretical guarantees we have derived (*cf.* Theorem 3.6.2, Theorem 3.6.3 and Corollary 3.6.3.1) all build on the premise that the reward r_φ is δ-Lipschitz over the joint input space $\mathcal{S} \times \mathcal{A}$, *i.e.* that $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$. Crucially, we do *not* enforce this regularity property *directly* is practice,

but instead urge the discriminator D_φ to be k-Lipschitz by restricting the norm of the Jacobian of the latter via regularization (*cf.* EQ 3.3). We here set out to figure out to what extent the k-Lipschitzness enforced onto D_φ propagates and transfers to r_φ; in particular, whether it results in the *indirectly*-urged δ-Lipschitzness of r_φ, with $\delta \neq k$ outside of edge cases. While k is fixed throughout the lifetime of the agent, δ need not be. As such, discussing the behavior of this evolving Lipschitz constant *w.r.t.* the learning dynamics is crucial to better understand *when* the guarantees we have just derived (whose main premise is $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$) apply in practice. As laid out ealier in SECTION 3.4, in this work, we consider two forms of reward, crafted purely from the scores returned by D_φ: the minimax (saturating) one $r^{\text{MM}}_\varphi \coloneqq -\log(1 - D_\varphi)$ and the non-saturating one $r^{\text{NS}}_\varphi \coloneqq \log(D_\varphi)$ (names purposely chosen to echo their counterpart GAN generator loss). Although we opted for the minimax form (based on the ablation study we carried out on the matter, *cf.* APPENDIX 3.F), we here tackle and discuss both forms, as we suspect there could be more to it than just zero-order numerics. Analyzing first-order behavior is the crux of most GAN design breakthroughs, which is far from surprising, considering how intertwined the inner networks are (generator G, and discriminator D). Yet, in adversarial IL, the policy (playing the role of G) does not receive gradients flowing back from D like in GANs. Instead, it gets a reward signal crafted from D's returned scalar value, detached from the computational graph, and try to maximize it over time via *policy*-gradient optimization. The discussion in adversarial IL has thus always limited to the numerics of the reward signal and how to shape it in a way that faciliates the resolution of the task at hand (similarly to how we discuss the impact of its shape when reporting our last empirical findings of SECTION 3.5.5).

By constrast, we here are interested in the gradients of these rewards ($r_{\varphi,\text{MM}}$ and $r_{\varphi,\text{NS}}$) in this studied adversarial IL context, with the end-goal of characterizing their Lipschitz-continuity (or absence thereof). Their respective Jacobians' norms, under the setting laid out earlier in SECTION 3.6.1, are $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F = \|\nabla^t_{s,a}[D_\varphi]_t\|_F \,/\, (1 - D_\varphi(s_t, a_t))$ and $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F = \|\nabla^t_{s,a}[D_\varphi]_t\|_F \,/\, D_\varphi(s_t, a_t)$, with $D_\varphi(s_t, a_t) \in (0, 1)$ (D_φ's score is wrapped with a sigmoid). As laid out above, we here posit that D_φ is k-Lipschitz-continuous as founding assumption — $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$. We can now upper-bound the Jacobians' norms unpacked above with the Lipschitz constant of D_φ: $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F \leq k \,/\, (1 - D_\varphi(s_t, a_t))$ and $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F \leq k \,/\, D_\varphi(s_t, a_t)$. Since $D_\varphi(s_t, a_t) \in (0, 1)$, both denominators (for either reward form) are in $(0, 1)$, which makes the Jacobian's norm of either reward form unbounded over its domain (due to $D_\varphi \to 0$ from above for r^{NS}_φ; due to $D_\varphi \to 1$ from below for r^{MM}_φ), despite the D_φ's k-Lipschitzness. Since treating the entire range of values that *can* be taken by $D_\varphi(s_t, a_t)$, $(0, 1)$, lead us to a dead end, and leaving us unable to upper-bound neither $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F$ nor $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F$, we now adopt a more granular approach and procede by dichotomy. As such, $\exists\, \ell \in (0, 1)$ verifying $0 < \ell \ll 1$ such that $1 \,/\, D_\varphi(s_t, a_t)$ (and as a result also $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F \leq k \,/\, D_\varphi(s_t, a_t)$) is unbounded when $D_\varphi(s_t, a_t) \in (0, \ell]$ and bounded when $D_\varphi(s_t, a_t) \in (\ell, 1)$. Sim-

ilarly, $\exists\, L \in (0,1)$ verifying $0 \ll L < 1$ such that $1 \,/\, (1 - D_\varphi(s_t, a_t))$ $\big(\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F \leq k \,/\, (1 - D_\varphi(s_t, a_t))$ too, as a result$\big)$ is bounded when $D_\varphi(s_t, a_t) \in (0, L]$ and unbounded when $D_\varphi(s_t, a_t) \in (L, 1)$. If we were to figure out the *effective* range covered by D_φ's values throughout the learning process, we woud maybe be able to exploit the dichotomy.

In practice, the untrained agent initially performs poorly at the imitation task, and is therefore assigned low scores by D_φ (near 0, as "0" is the label assigned to samples from the agent in the classification update D_φ goes through every iteration). As learning progresses, the agent's scores gradually shift towards 1 — the label used for expert samples in D_φ's update, and *optimally* converge to the central value of 0.5 in the $(0, 1)$ range that D_φ can describe. Indeed, the *perfect* discriminator consistently predicts scores equal to 0.5 for the agent's actions [Goo17]: the agent has managed to perfectly confuse D_φ as to where the data it is fed comes from (both sources, expert and agent, are perceived as equiprobable). What matters for $\|\nabla^t_{s,a}[r_\varphi]_t\|_F$ (either form) to be bounded *in practice* is for it to be bounded for values of D_φ in $(0, M]$, where $0.5 \leq M < 1$ (the values *realistically* taken by D_φ throughout the learning process). Since $M < L$ in effect (for L, *cf.* dichotomy above), we can conclude that $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F$ is effectively bounded: $\exists\, \partial,\ 0 \leq \partial < +\infty$, such that $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F \leq \partial$. We however can not conclude as such for $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F$, however close to zero ℓ might be (for ℓ, *cf.* dichotomy above). It is not rare for D_φ to take 0 as value early in training, which makes $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F$ unbounded in the interval described by the values taken by D in practice: $(0, M]$. Interestingly, when D_φ is near 0 early in training, $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F \leq k \,/\, (1 - D_\varphi(s_t, a_t)) \approx k$. The lowest upper-bound for $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F$ is $\partial \approx k$, and can only happen early in the training process, when D_φ correctly classifies the agent's actions as coming from the agent. In other words, the Lipschitz constant of r^{MM}_φ is at its lowest early in training. Besides, as the agent becomes more proficient at mimicking the expert and therefore collects higher scores from D_φ, ∂ increases monotonically and grows aways from its initial value k. Compared to the alternative (highest Lipschitz constant early in training and then monotonically decreasing as the scores increase when the agent gets better at the task, nearing the lowest value of k when $D_\varphi \to 1$), which as it turns out is exactly the behavior adopted by r^{NS}_φ, the behavior of r^{MM}_φ is far more desirable.

Crucially, to sum up, r^{NS}_φ is not Lipschitz early in training when the agent would benefit most from regularity in the reward landscape. r^{MM}_φ however *is* Lipschitz-continuous early in training, with the lowest Lipschitz constant of its lifetime, which aligns with the Lipschitz constant enforced on D_φ ($\partial \approx k$). As such, r^{MM}_φ is at its most regular when the agent needs it most (early, when it knows nothing), and then becomes less and less restrictive (the Lipschitz constant ∂ increases) as the agent collects higher similarity scores with the expert from D_φ. One could therefore see r^{MM}_φ as having built-in *"training wheels"*, which gradually phase out as the agent becomes better, providing less safety as the agent becomes more proficient at the imitation task. To conclude this discussion point, with the

minimax reward form $r_\varphi \coloneqq r_\varphi^{\text{MM}}$, we have $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k \implies \|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ in practice. This means that the premise of our theoretical guarantees consisting in positing that the reward is δ-Lipschitz-continuous *can* be satisfied in practice by enforcing k-Lipschitz-continuity on D_φ via gradient penalty regularization (*cf.* EQ 3.15). This is *not* the case when $r_\varphi \coloneqq r_\varphi^{\text{NS}}$. We propose this analytical observation as an explanation as to why using r_φ^{NS} yields such poor results in our reported ablation, *cf.* APPENDIX 3.F. Our discussion detaches itself from the one adopting a zero-order numerics scope, laid out in SECTION 3.5.5, by discussing first-order numerics instead, which blends into our Lipschitzness narrative.

3.6.2.6 Local smoothness

The local Lipschitzness assumption is reminiscent of many theoretical results in the study of robustness to adversarial examples. Notably, [YRZ+20] shows that local Lipschitzness is correlated with empirical robustness and accuracy in various benchmark datasets. As mentioned when we justified the *local* nature of the Lipschitz-continuity notion tackled in this work (*cf.* DEFINITION 3.4.1), we optimize the different modules over mini-batches of samples. While forcing the constraint to be satisfied globally might be feasible in some low-dimensional supervised or unsupervised learning problems, the notion of fixed dataset does not exist *a priori* in reinforcement learning. SECTION 3.6.3 describes, compares and discusses the effect of *where* the local Lipschitzness constraint is enforced (*e.g.* expert demonstration manifold, fictitious replay experiences). Wherever the regularizer is applied, the constraint is local nonetheless. One can therefore not guarantee that the δ-Lipschitz-continuity of r_φ, formalized as $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$, and urged by enforcing $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ via gradient penalization (*cf.* our previous discussion on indirect reward regularization in SECTION 3.6.2.5), will be satisfied *everywhere* in $\mathcal{S} \times \mathcal{A}$. Plus, considering that THEOREM 3.6.3 and COROLLARY 3.6.3.1 rely on the satisfaction of the constraint on r_φ along every trajectory, which is likely not to be verified in practice, we can say with high confidence that the constraint on Q_φ, $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F \leq \Delta_\infty$, will not be satisfied over the whole joint input space either. Still, we can hope to enhance the coverage of the subspace on which the constraint $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ is satisfied, dubbed $\mathfrak{C}$, by doing more r_φ learning updates with the regularizer — technically, D_φ learning updates encouraging D_φ to satisfy $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ via gradient penalization, *cf.* EQ 3.15. From this point onward, we will qualify a state-action pair (s_t, a_t) — equivalently, an action a_t in a given state s_t — as *"$\mathfrak{C}$-valid"* if it belongs to $\mathfrak{C} \ni (s_t, a_t)$, *i.e.* if r_φ is δ-Lipschitz, verifying $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$. Note, the notion of $\mathfrak{C}$-validity is inherently local, since we have defined the notion for a single given input pair (s_t, a_t). As such, future statements about $\mathfrak{C}$-validity will all be local ones by essence. In addition, despite having $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k \implies \|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ *in practice* for the minimax reward form (*cf.* our previous discussion on indirect reward regularization in SECTION 3.6.2.5), there is not an exact equivalence

between r_φ being δ-Lipschitz and D_φ being k-Lipschitz in theory. Therefore, we will qualify a state-action pair (s_t, a_t) — equivalently, an action a_t in a given state s_t — as *"approximately $\mathfrak{C}$-valid"* if D_φ is k-Lipschitz, verifying $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$. As it has been made clear by now, D_φ's k-Lipschitzness is encouraged by plugging a gradient penalty regularizer $\mathfrak{R}^\zeta_\varphi(k)$ into D_φ's loss (*cf.* EQ 3.15). Despite being encouraged, $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ can nonetheless not be guaranteed solely from the application of the regularizer at (s_t, a_t). As such, to cover all bases, we will qualify a state-action pair (s_t, a_t) — equivalently, an action a_t in a given state s_t — as *"probably approximately $\mathfrak{C}$-valid"* if (s_t, a_t) is in the support of the distribution ζ that determines where the gradient penalty regularizer $\mathfrak{R}^\zeta_\varphi(k)$ of ℓ^{GP}_φ is applied in $\mathcal{S} \times \mathcal{A}$, *i.e.* if $(\text{supp } \zeta) \ni (s_t, a_t)$. A probably approximately $\mathfrak{C}$-valid point is supported by the distribution that describes where $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ is enforced, and as such, $\mathfrak{R}^\zeta_\varphi(k)$ may be applied at this point.

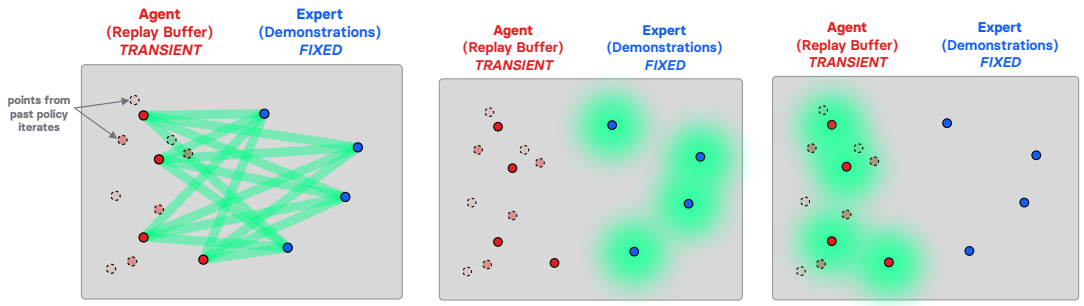
Importantly, the policy might, due to its exploratory motivations, pick an action a_t in state s_t that is not $\mathfrak{C}$-valid. Depending on where the constraint will then be enforced, the sample might then be $\mathfrak{C}$-valid after r_φ's update (technically, indirectly via D_φ's update; *cf.* SECTION 3.6.3). This observation motivates the investigation we carry out in SECTION 3.6.4, in which we define a soft $\mathfrak{C}$-validity pseudo-indicator of $\mathfrak{C}$ (*cf.* EQ 3.68) that enables us to assess whether the agent consistently performs approximately $\mathfrak{C}$-valid actions when it interacts with the MDP $\mathbb{M}^*$ following μ_θ.

3.6.3 A NEW REINFORCEMENT LEARNING PERSPECTIVE ON GRADIENT PENALTY

We begin by considering a few variants of the original gradient penalty regularizer [GAA+17] introduced in SECTION 3.5.4. Each variant corresponds to a particular case of the *generalized* version of the regularizer, described in EQ 3.15. Subsuming all versions, we remind EQ 3.15 here for didactic purposes:

$$\ell^{\text{GP}}_\varphi \coloneqq \ell_\varphi + \lambda\, \mathfrak{R}^\zeta_\varphi(k) \coloneqq \ell_\varphi + \lambda\, \mathbb{E}_{s_t \sim \rho^\zeta, a_t \sim \zeta}[(\|\nabla_{s_t,a_t} D_\varphi(s_t, a_t)\| - k)^2] \tag{3.67}$$

where ζ is the distribution that describes *where* the regularizer is applied — where the Lipschitz-continuity constraint is enforced in the input space $\mathcal{S} \times \mathcal{A}$. In [GAA+17], ζ corresponds to sampling point uniformly along segments joining samples generated by the agent following its policy and samples generated by the expert policy, *i.e.* samples from the expert demonstrations $\mathcal{D}$. Formally, focusing on the action only for legibility — the counterpart formalism for the state is derived easily by using the visitation distribution instead of the policy — $a \sim \zeta$ means $a = u\, a' + (1 - u)\, a''$, where $a' \sim \pi_\theta$, $a'' \sim \pi_e$, and $u \sim \text{unif}(0, 1)$. The distribution ζ we have just described corresponds to the transposition of the GAN formulation to the GAIL setting, which is an *on*-policy setting. Therefore, in this work, we amend the ζ previously described, and replace it with its *off*-policy counterpart, where



(a) WGAN-GP [GAA+17] (b) DRAGAN [KAHK17] (c) NAGARD (illustrative *only*)

Figure 3.6.1: Schematic representation (in green) of the support of the ζ distribution, depicting *where* the gradient penalty regularizer is enforced, at a given iteration, and for all iterations throughout the lifetime of the learning agent. It corresponds to the subspace of $\mathcal{S} \times \mathcal{A}$ on which the Lipschitz-continuity constraint is applied: where the state-action pairs are *likely* $\mathfrak{C}$-valid. The intensity of the green color indicates the probability assigned by the distribution ζ on the state-action pair. The more opaque the coloration, the higher the probability. Best seen in color.

$a' \sim \beta$ (*cf.* Section 3.4). As for the penalty target, [GAA+17] use $k = 1$, in line with the theoretical result derived by the authors. By contrast, DRAGAN [KAHK17] use a ζ such that $a \sim \zeta$ means $a = a'' + \varepsilon$, where $a'' \sim \pi_e$, and $\varepsilon \sim \mathcal{N}(0, 10)$. Like WGAN-GP [GAA+17], DRAGAN uses the penalty target $k = 1$. Finally, for the sake of symmetry, we introduce a reversed version of DRAGAN, dubbed NAGARD (name reversed). To the best of our knowledge, the method has not been explored in the literature. NAGARD also uses $k = 1$ as penalty target, but perturbs the policy-generated samples as opposed to the expert ones: $a \sim \zeta$ means $a = a' + \varepsilon$, where $a' \sim \beta$ (*off*-policy setting), and $\varepsilon \sim \mathcal{N}(0, 10)$. We use $\lambda = 10$ in all the variants, in line with the original hyper-parameter settings in [GAA+17] and [KAHK17].

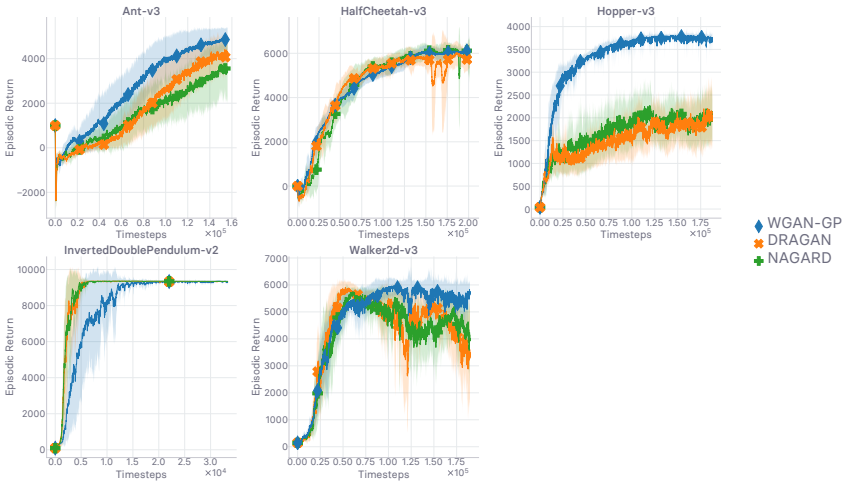
Figure 3.6.1 depicts in green the subspace of the input space $\mathcal{S} \times \mathcal{A}$ where the k-Lipschitz-continuity constraint, formalized as $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$, and enouraged in ℓ^{GP}_φ by $\mathfrak{R}^\zeta_\varphi(k)$, is applied. In other words, Figure 3.6.1 highlights the support of the distribution ζ for each variant, which have just been described above. As such, the green areas in Figures 3.6.1b, 3.6.1c, and 3.6.1a are schematic depictions of where the state-actions pairs are *probably approximately* $\mathfrak{C}$*-valid*.

One conceptual difference between the DRAGAN penalty and the two others is that the support of the distribution ζ does not change throughout the entire training process for the former, while is does for the latter. Borrowing the intuitive terminology used in [KAHK17], WGAN-GP proposes a *coupled penalty*, while DRAGAN (like NAGARD) propose a *local* penalty. In [KAHK17], the authors perform a comprehensive empirical study of mode collapse, and diagnose that the generator collapsing to single modes is often coupled with the discriminator displaying sharp gradients around

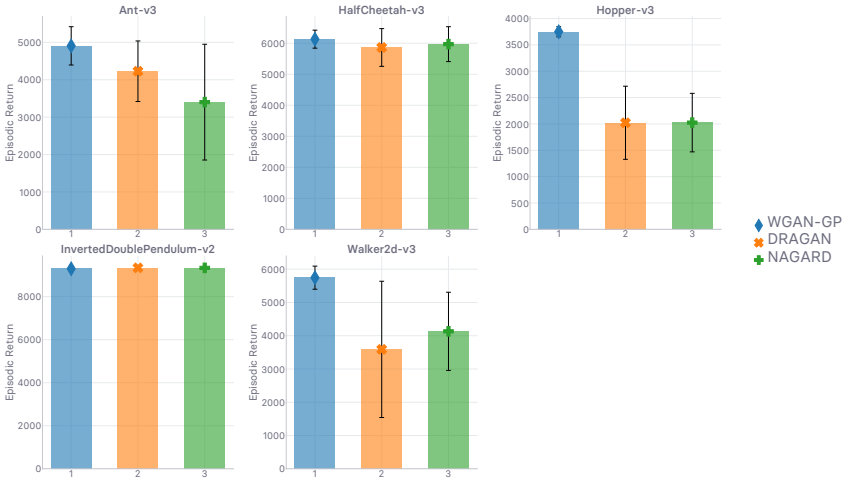
the samples from the real distribution. In model-free generative adversarial imitation learning, the generator does not have access to the gradient of the discriminator with respect to its actions in the backward pass, although it could be somewhat accessed using a model-based approach [BACM17]. In spite of not being accessible *per se*, the sharpness of the discriminator's gradients near real samples observed in [KAHK17] translates, in the setting considered in this work, to sharp rewards, which we referred to as reward overfitting and was discussed thoroughly in Section 3.5.3. As such, mode collapse mitigation in the GAN setting translates to a problem of credit assignment in our setting, caused by the peaked reward landscape (*cf.* Appendix 3.G to witness the sensitivity *w.r.t.* the discount factor γ, controlling how far ahead in the episode the agent looks). The stability issues the methods incur in either settings are on par. Both gradient penalty regularizers aim to address these stability weaknesses, and do so by enforcing a Lipschitz-continuity constraint, albeit on a different support $\operatorname{supp}\ \zeta$ (*cf.* Figure 3.6.1).

As mentioned earlier in Section 3.5.4, the distribution ζ used in WGAN-GP [GAA+17] is motivated by the fact that — as they show in their work — the *optimal* discriminator is 1-Lipschitz along lines joining real and fake samples. The authors of [KAHK17] deem the assumptions underlying this result to be unrealistic, which naturally weakens the ensuing method derived from this line of reasoning. They instead propose DRAGAN, whose justification is straightforward and unarguable: since they witness sharp discriminator gradients around real samples, they introduce a *local* penalty that aims to smooth out the gradients of the discriminator *around* the real data points. Formally, as described above when defining the distribution ζ associated with the approach, it tries to ensure Lipschitz-continuity of the discriminator in the neighborhoods (additive Gaussian noise perturbations) of the real samples. The generator or policy is more likely to escape the narrow peaks of the optimization landscape — corresponding to the real data points — with this extra stochasticity. *In fine*, in our setting, DRAGAN can dial down the sharpness of the reward landscape at expert samples the discriminator overfits on. This technique should therefore fully address the shortcomings raised and discussed in Section 3.5.4). While the method seem to yield better results than WGAN-GP in generative modeling with generative adversarial nets, the empirical results we report in Figure 3.6.2 show otherwise. All the considered penalties help close the significant performance gap reported in Figure 3.5.2, in almost every environment, but the penalty from WGAN-GP generally pulls ahead. Additionally, not only does is display higher empirical return, it also crucially exhibits more stable and less jittery behavior.

Despite the apparent disadvantage of *local* penalties (DRAGAN [KAHK17] and NAGARD) compared to WGAN-GP in terms of their schematically-depicted $\operatorname{supp}\ \zeta$ sizes (*cf.* Figure 3.6.1), it is important to remember that the additive Gaussian perturbation is distributed as $\mathcal{N}(0, 10)$. For these



(a) Evolution of return values *(higher is better)*



(b) Final return values at timeout *(higher is better)*

Figure 3.6.2: Evaluation of gradient penalty variants. Explanation in text. Runtime is 48 hours.

local methods, ζ is therefore covering a *large* [3] area around the central sample, including with high probability samples that are, according to the discriminator, from both categories — fake samples (predicted as from β), and real samples (predicted as from π_e). As such, the perceived diameter of the green disks in the schematic representations in FIGURES 3.6.1b and 3.6.1c maybe smaller than it would be in reality. It is crucial to consider the coverage of the different ζ distributions as they determine how strongly the Lipschitz-continuity property is potentially enforced at a given state-action pair, for a fixed number of discriminator updates. Consequently, for a given optimization step, while the *local* penalties are — somewhat ironically — applying the Lipschitz-continuity constraint on data points *scattered* around the agent- (NAGARD) or expert-generated (DRAGAN) samples, the supp ζ for WGAN-GP is less diffuse. Local penalties ensure the Lipschitzness is somewhat satisfied all around the selected samples, which for DRAGAN is motivated by the fact that there are narrow peaks on the reward landscape located at the expert samples, where it us prone to overfit (*cf.* SECTION 3.5.3). The distribution ζ used in WGAN-GP also supports data points near expert samples, but these are not scattered all around for the sole purpose of making the whole area smooth and escape bad basins of attraction like in DRAGAN. In other terms, the Lipschitz-continuity constraint is applied isotropically, from the original expert sample outwards. By contrast, WGAN-GP's ζ only supports a few discrete directions from a given expert sample, the lines joining said sample to all the agent-generated samples (of the mini-batch). Intuitively, while DRAGAN smooths out the reward landscape starting from expert data points and going in every direction from there, WGAN-GP smooths out the reward landscape starting from expert data points and going only in the directions that point toward agent-generated data points. As such, one could qualify DRAGAN as *isotropic* regularizer, and WGAN-GP as *directed* regularizer.

We believe that WGAN-GP outperforms DRAGAN in the setting and environments considered in this work (*cf.* FIGURE 3.6.2) due to the fact that the agent benefits from having smooth reward *pathways* in the reward landscape in-between agent samples and expert samples. Along these pathways, going from the agent sample end to the expert sample end, the reward *progressively* increases. For the agent trying to maximize its return, these series of gradually increasing rewards joining agent to the expert data points are akin to an *automatic curriculum* [KVDP12, Ope19] assisting the reward-driven agent and leading it towards the expert. FIGURE 3.6.2 shows that WGAN-GP indeed achieves consistently better results across every environment but the least challenging, as seen in the `IDP` environment (*cf.* TABLE 3.5.1). In the four considerably more challenging environments, the *directed* method allows the agent to attain overall significantly higher empirical return than its competitors. Besides, it displays greater stability when approaching the asymptotic regime, whereas the *local* regu-

[3] Considering the observations are clipped to be in $[-5.0, 5.0]$, as is customary in the MUJOCO [TET12] benchmark [BCP+16], an additive Gaussian perturbation with $\sigma^2 = 10$ can, in all fairness, be qualified as *large*.

larizers clearly suffer from instabilities, especially DRAGAN in the results obtained in environments `Walker2d` and `HalfCheetah`, depicted in FIGURE 3.6.2. While the proposed interpretation laid out previously corroborates the results obtained and reported in FIGURE 3.6.2, it does not explain the instability issues hindering the local penalties. We believe the jittery behavior observed in the results obtained in environments `Walker2d` and `HalfCheetah` (*cf.* FIGURE 3.6.2) — once the peak performance is attained — is caused by supp ζ (green areas in FIGURE 3.6.1) not changing *is size* as the agent learns to imitate and gets closer to the expert in $\mathcal{S} \times \mathcal{A}$.

Indeed, in DRAGAN, ζ is a stationary distribution: it applies the regularizer on perturbations of the expert samples, where the additive noise's underlying sufficient statistics are constant throughout the learning process, and where the expert data points are distributed according to the stationary policy π_e and its associated state visitation distribution. For NAGARD, the perturbations follow the same distribution, and remain constant across the updates. However, unlike DRAGAN, ζ is defined by adding the stationary noise to samples from the *current* agent, every update, distributed as β in our *off*-policy setting. Since β is by construction non-stationary across the updates, as a mixture of past π_θ updates, ζ is non-stationary in NAGARD. Despite ζ's having these different support and stationary traits, the results of either local penalties are surprisingly similar. This is due to the variance of the additive noise used in both methods being large relative to the distance between the expert and agent samples, at all times, in the considered environments. As such, their supp ζ are virtually overlapping, which makes the two local penalties virtually equivalent, and explains the observed similarities in-between them.

Coming back to the main point — *"why do local penalties suffer from instabilities at the end of training?"* — even though the agent samples are close to the expert ones, the local methods both apply the same large perturbation before applying the Lipschitz-continuity penalty. The probability mass assigned by ζ is therefore still spread similarly over the input space, and is therefore severely decreased in-between agent and expert samples since these are getting closer in the space. The local methods are therefore often applying the constraint on data points that the policy will never visit again (since it wants to move towards the expert) and equivalently, rarely enforces the constraint between the agent and the expert, which is where the agent should be encouraged to go. With this depiction, it is clearer why WGAN-GP pulls ahead. Compared to the fixed size of supp ζ in the local penalties, ζ *adapts* to the current needs of the agent (hence qualifying as non-stationary). As the agent gets closer to the expert, Lipschitz-continuity is always enforced on data points between them, which is where it potentially benefits the agent most. The support of ζ is therefore decreasing in size as the iterations go by, focusing the probability mass of ζ where enforcing a smooth reward landscape matters most: where the agent should go, *i.e.* in the direction of the expert data points.

Besides, considering the inherent sample selection bias [Hec79] the control agent is subjected to, where the latter end up in $\mathcal{S}\times\mathcal{A}$ depends on its actions, in every interaction with the dynamical system represented by its environment. This aspect dramatically differs from the traditional *non-Markovian* GAN setting — in which these penalties were introduced — where the generator's input noise is *i.i.d.*-sampled. Indeed, suffering from said sample selection bias, an imitation agent straying from the expert demonstrations is likely to keep on doing so until the episode is reset (*cf.* discussion in Section 3.5.4). Distributions ζ whose definition involve samples generated by the learning agent and adapt to the agent's current relative position *w.r.t.* the expert data points therefore provide valuable extra guidance in Markovian settings. Additionally, assuming the input also contained the *phase* — *"how far the agent/expert is in the current episode"*, $0 \leq t \leq T$ — (like in [PKT+18]) not only would the imitation task be easier, but the benefits of the WGAN-GP penalty would be further enhanced, as it would allow the models to exploit the temporal structure of to the considered Markovian setting.

In reaction to the recent interest towards *"zero-centered"* gradient penalties [RLNH17, MGN18], due to the theoretical convergence guarantees they allow for, we have conducted a grid search on the values of the Lipschitz constant k and the regularizer importance coefficient λ, as described in Section 3.6.3. The results are reported in Appendix 3.E.3. In short, the method performs poorly when $k = 0$, unless a very small value is used for λ. Enforcing 0-Lipschitzness is far too restraining for the agent to learning anything, unless this constraint is only loosely imposed. Conversely, a smaller λ value yields worse results when $k = 1$, revealing the interaction between the gradient penalty hyperparameters k and λ. In particular, we will momentarily provide comprehensive evidence along with a greater characterization of how the choice of scaling factor λ not only impacts the agent's performance (which is already depicted in Appendix 3.E.3), but how it correlates quantitatively with the approximate $\mathfrak{C}$-validity displayed by the agent (*cf.* Section 3.6.4). Unless explicitly stated otherwise, we use the WGAN-GP penalty variant, with Lipschitz constant target $k = 1$, and scaling coefficient $\lambda = 10$ throughout the empirical results exhibited in both the body and appendix.

3.6.4 Diagnosing $\mathfrak{C}$-validity: is the Lipschitzness premise of the theoretical guarantees satisfied in practice?

To put things in perspective, we first give a side-by-side rundown of how what we set out to tackle here compares to what we have just tackled in Section 3.6.3, thereby giving a glimpse of what we set out to investigate in what follows. In the previous section, we showed how *(a)* the choice of ζ (*where* do we want to encourage approximately $\mathfrak{C}$-valid behavior), and *(b)* the choice of λ (*to what degree* do we want to encourage approximately $\mathfrak{C}$-valid behavior) both independently impact the agent's performance in terms of empirical episodic return. In this section on the other hand, we will show how *(a)* the choice of ζ, and *(b)* the choice of λ both independently impact the agent's consistency at *effectively* selecting

approximately $\mathfrak{C}$-valid actions with its learned policy μ_θ. If we were to find a strong positive correlation between the agent's asymptotic return and its effectively measured approximate $\mathfrak{C}$-validity rate — high when high, low when low, for all tested ζ's and for all tested λ's — then we would have further quantitative evidence to support our work's main claim: reward Lipschitzness is necessary to achieve high return, and higher Lipschitzness uptime correlates strongly with higher return. Perhaps most crucially, we would be able to correlate high empirical episodic return with high chance of satisfying the premise of our theoretical guarantees (r_φ's Lipschitzness). As such, these would consequently apply in in practice too. This would attest to the practical relevance of Section 3.6.1.

We have shown that enforcing a Lipschitz-continuity constraint on the learned reward r_φ (albeit indirectly via D_φ) is instrumental in achieving expert-level performance in off-policy generative adversarial imitation learning (*cf.* Section 3.5.5). We have also shown that directed regularization techniques yield better results, seemingly due to the better guidance they provide to the mimicking agent, in the form of an automatic curriculum of rewards towards the expert data points (*cf.* Section 3.6.3). Such curriculum only exists where the Lipschitz-continuity constraint is satisfied. Said differently, it could not exist if the constraint were not satisfied along μ_θ's pathways which would then involve non-smooth hurdles. It is therefore crucially important for said constraint to be satisfied *in effect* for the state-actions pairs in the the support of the policy the agent uses in its learning update, μ_θ, *i.e.* supp $\mu_\theta \ni (s_t, a_t)$. Still, the deterministic policy μ_θ likely performs only approximately $\mathfrak{C}$-valid actions as it is trained with the sole objective to maximize cumulative rewards that represent its similarity *w.r.t.* the expert π_e. The imitation rewards corresponding to a greater degree of similarity are, by design of the generative adversarial imitation learning framework, situated between the agent's current position and the expert's position on the current reward landscape. Since this is where we apply the Lipschitzness constraint (with WGAN-GP, our baseline, as said above) — equivalently, since these regions are approximately $\mathfrak{C}$-valid — μ_θ is likely to never select $\mathfrak{C}$-invalid actions as it optimizes for its utility function (*cf.* Section 3.3). Conversely, in the considered setting, picking $\mathfrak{C}$-invalid actions could in theory hinder the optimization process the policy is subject to, as μ_θ would *a priori* venture in regions of the state-action space that do not increase its similarity with the expert policy π_e — or, at the very least, for which the *non*-satisfaction of the reward's Lipschitz-continuity premise $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ might lead to instabilities due to $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F > \Delta_\infty$ as a direct consequence of our theoretical guarantees (*cf.* Section 3.6.2). Since we do not have such a tight control over where and to what degree the Lipschitzness constraint over the reward r_φ is *satisfied* (hence our introduction of the notions of approximately $\mathfrak{C}$-valid samples and probably approximately $\mathfrak{C}$-valid samples), we instead turn to the closest surrogate over which we do have a tighter control: where and to what degree D_φ's constraint is *enforced*. The *"where"* is controlled by the choice of ζ (determined by the gradient penalty regularization method in use), and the *'to what degree'* by the choice of λ scale.

Still, even in the occurrence where D_φ's constraint is enforced by adding $\mathfrak{R}^\zeta_\varphi(k)$ as in ℓ^{GP}_φ (*cf.* EQ 3.15) at the point (s_t, a_t), the most we could say is that (s_t, a_t) is probably approximately $\mathfrak{C}$-valid, since $(s_t, a_t) \in \text{supp } \zeta$ — otherwise, the gradient penalty regularizer $\mathfrak{R}^\zeta_\varphi(k)$ could never have been applied at that point in the landscape $\mathcal{S} \times \mathcal{A}$. In effect, enforcing the constraint at the point was enough to guarantee that $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$, and we therefore do not know whether (s_t, a_t) is approximately $\mathfrak{C}$-valid, or not. As a direct consequence, we can *a fortiori* not guarantee that $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$; we do not know whether (s_t, a_t) is $\mathfrak{C}$-valid, or not — *cf.* SECTION 3.6.2.5 for our discussion on indirect reward regularization, in which we establish that D_φ's k-Lipschitzness causes r_φ to be δ-Lipschitz in practice. On the flip side, based on the latter result about indirect Lipschitz-continuity inducement, we can state that ensuring empirically that $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ is *enough* to ensure that $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ is verified in practice. In other words, showing that (s_t, a_t) is approximately $\mathfrak{C}$-valid can be used as a proxy for showing that (s_t, a_t) is $\mathfrak{C}$-valid, empirically. As such, in order to assess whether the premise of the theoretical guarantees we derived in SECTION 3.6.1 is satisfied in practice (r_φ's -δ-Lipschitz-continuity), it is sufficient to assess whether the agent's actions $a_t = \mu_\theta(s_t)$ are approximately $\mathfrak{C}$-valid. In particular, we want to know the relative impacts the choices of ζ and the λ in ℓ^{GP}_φ have on the propensity for an action from μ_θ to be approximately $\mathfrak{C}$-valid. So as to estimate how often the actions selected by the agent via μ_θ are approximately $\mathfrak{C}$-valid, we build an estimator that *softy* approximates $\mathbb{1}_{\mathfrak{C}} : \mathcal{S} \times \mathcal{A} \to \{0, 1\}$, the indicator of the $\mathfrak{C}$-validity subspace over $\mathcal{S} \times \mathcal{A}$, where $\mathbb{1}_{\mathfrak{C}}(s_t, a_t) = 1$ when $(s_t, a_t) \in \mathfrak{C}$, and $\mathbb{1}_{\mathfrak{C}}(s_t, a_t) = 0$ when $(s_t, a_t) \notin \mathfrak{C}$. Accordingly, we call our estimator *soft approximate $\mathfrak{C}$-validity pseudo-indicator*, implementing a soft, C^0 mapping $\widehat{\mathbb{1}}_{\mathfrak{C}} : \mathcal{S} \times \mathcal{A} \to (0, 1]$, and formally defined as, $\forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$:

$$\widehat{\mathbb{1}}_{\mathfrak{C}}(s_t, a_t) \coloneqq \exp\Big(- \max\big(0, \|\nabla_{s_t, a_t} D_\varphi(s_t, a_t)\| - k\big)^2 \Big) \tag{3.68}$$

► *soft approximate $\mathfrak{C}$-validity pseudo-indicator*

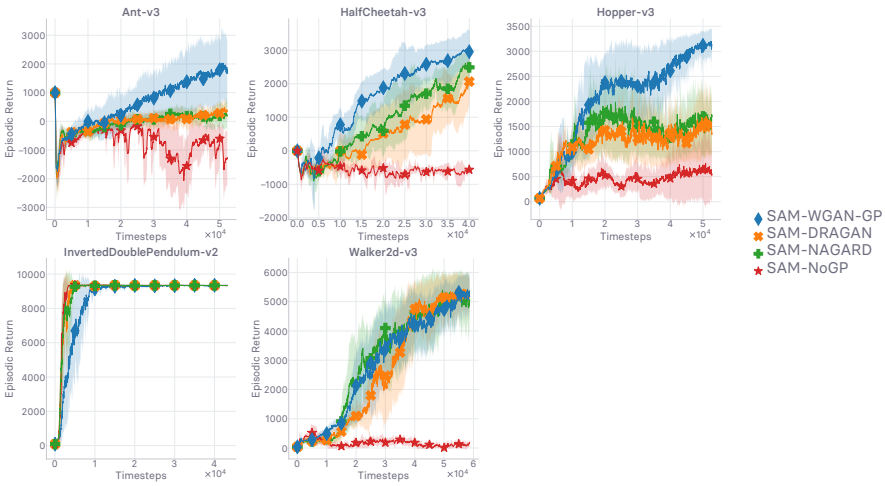
Thus, for a given pair (s_t, a_t), $\widehat{\mathbb{1}}_{\mathfrak{C}}(s_t, a_t) = 1$ when $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ and $\widehat{\mathbb{1}}_{\mathfrak{C}}(s_t, a_t) \to 0$ when $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \gg k$.

FIGURES 3.6.3 and 3.6.4 depict respectively the evolution of the values taken by the soft approximate $\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_{\mathfrak{C}}$ (*cf.* EQ 3.68) for different choices of ζ (different gradient penalty variants) and λ (sweep over $\mathfrak{R}^\zeta_\varphi(k)$'s scaling factor). In FIGURES 3.6.3 and 3.6.4, we also share the return accumulated by the agents throughout their respective training periods, (*cf.* 3.6.3a and 3.6.4a, respectively). In particular, what we report in FIGURES 3.6.3a and 3.6.4a echoes what we have already reported in FIGURES 3.6.2 and 3.E.3, but the settings in which the agents were trained differ (ever so) slightly. We indicate the specificities of the setting tackled in this section below, in this very paragraph. Still, since their settings do not match perfectly, we report their return along their soft approximate

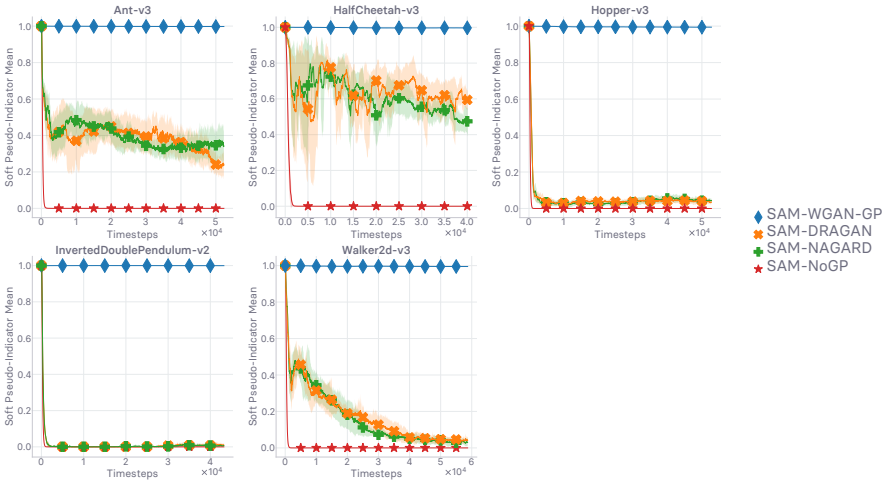
$\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values. We monitor and record these values during the evaluation trials the agent periodically goes through, in which the agent uses μ_θ to decide what to do in a given state. To best align with the definition of Lipschitz-continuity (*cf.* DEFINITION 3.4.1), which is also how we designed our soft approximate $\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_{\mathfrak{C}}$, we use one-sided gradient penalties $\mathfrak{R}^{\zeta}_{\varphi}(k)$ in the λ sweep — $\max(0, \|\nabla_{s_t,a_t} D_\varphi(s_t, a_t)\| - k)^2$, which *purely* encourages $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ to be satisfied (nothing more, nothing less) — although we have shown the variant presents very little empirical difference with the base two-sided one (*cf.* ablation in APPENDIX 3.E.1). It is worth noting that the experiments whose results are reported in FIGURES 3.6.3 and 3.6.4 carry out less iterations during the fixed allowed runtime, due to the substantial cost entailed by computing soft approximate $\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_{\mathfrak{C}}$ at every single evaluation step, in every evaluation trial. One could cut down that cost simply by evaluating $\widehat{\mathbb{1}}_{\mathfrak{C}}$ less frequently, but we decided otherwise, as we gave priority to having a finer tracking of $\widehat{\mathbb{1}}_{\mathfrak{C}}$. Besides, despite this slight apparent hindrance, the values of the proposed pseudo-indicator reported in either figure seem to have reached maturity, nearing their asymptotic regime, in the allowed runtime. We now go over the results reported in both figures.

In FIGURE 3.6.3, we observe that the monitored soft approximate $\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_{\mathfrak{C}}$ (*cf.* EQ 3.68) consistently takes values close to 1 when using the distribution ζ advocated in WGAN-GP to assemble the regularizer $\mathfrak{R}^{\zeta}_{\varphi}(k)$. Conversely, *not* using any gradient penalty regularizer causes the approximate $\mathfrak{C}$-validity rate to be in the vicinity of 0. Albeit *a priori* not surprising, it is still substantially valuable to notice that D_φ's k-Lipschitz-continuity (and thus r_φ's δ-Lipschitz-continuity; *cf.* SECTION 3.6.2) *never* happens by accident (or rather, by chance). As for DRAGAN and NAGARD (both being non-directed gradient penalty schemes, unlike WGAN-GP; *cf.* SECTION 3.6.3), both perform similarly across the board in terms of collected $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values. Their recorded soft pseudo-indicator values stay around a fixed value per environment, different for every one of them. These are within the $[0.1, 0.7]$ range, and as such, are definitely encouraging $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ in practice, yet are falling short of achieving the same *(a)* effective approximate $\mathfrak{C}$-validity value, and *(b)* effective approximate $\mathfrak{C}$-validity consistency as WGAN-GP. These phenomenona occur consistently across the spectrum of tackled environments.

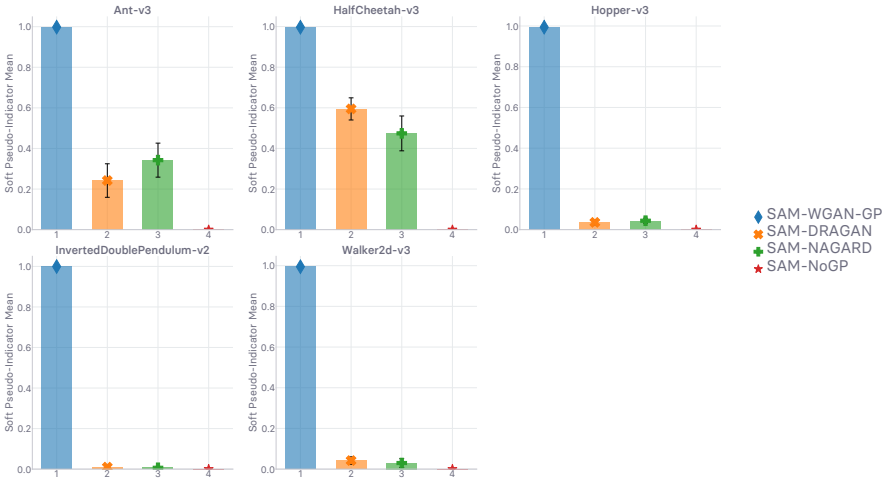
In FIGURE 3.6.4, we observe the unsurprising fact that the higher λ's value is — equivalently, the more we encourage the regularity property $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ to be satisfied — the more $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ is satisfied in effect. Besides confirming that gradient penalization indeed urges Lipschitzness (which we were not doubting), the figure helps us gauge to what degree the value of $\mathfrak{R}^{\zeta}_{\varphi}(k)$'s scaling coefficient in $\ell^{\text{GP}}_{\varphi}$ (*cf.* EQ 3.15) affects quantitatively the satisfaction of $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ monitored via the soft proxy $\widehat{\mathbb{1}}_{\mathfrak{C}}$. We considered powers of 10 for λ's sweep, tackling the values $\lambda_i := 10^i$, for $i \in \{-3, -2, -1, 0, 1\}$. The gap inbetween the $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values associated with each of these λ_i differ per environment, but their ranking remain the same (higher $\widehat{\mathbb{1}}_{\mathfrak{C}}$'s for higher i's). At its lowest (*i.e.* for



(a) Evolution of return values *(higher is better)*

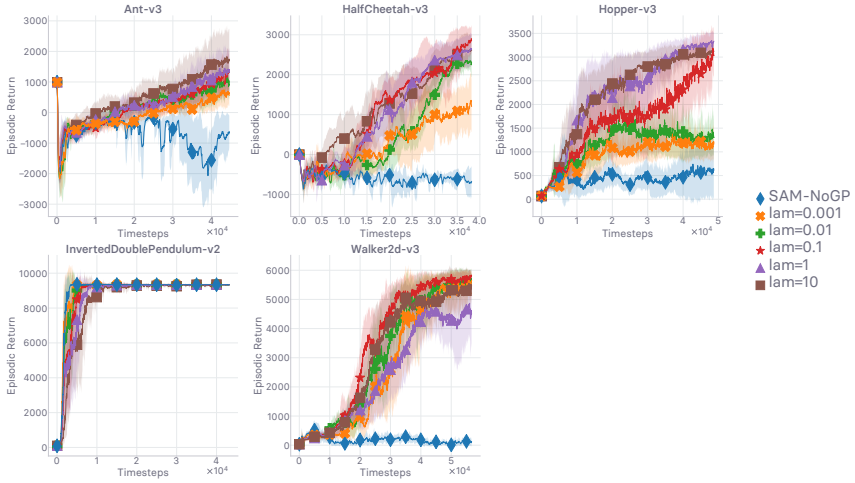


(b) Evolution of $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values *(higher means more approx. $\mathfrak{C}$-valid)*, *cf.* EQ 3.68.

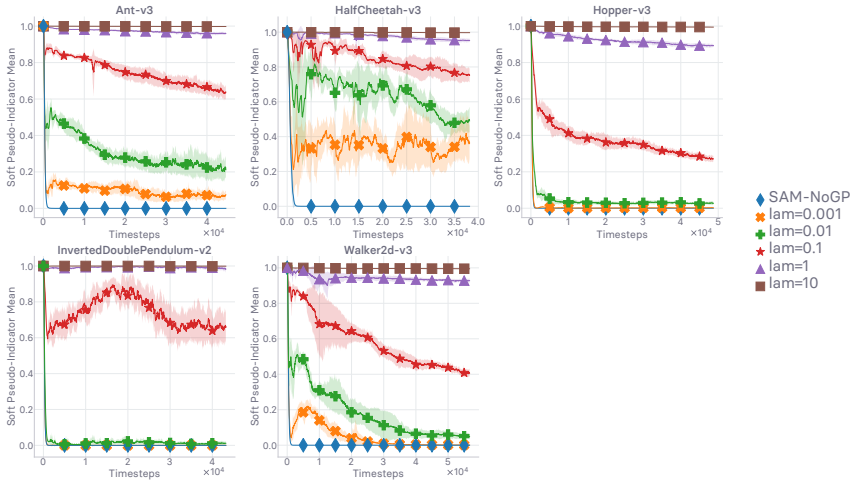


(c) Final $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values at timeout *(higher means more approx. $\mathfrak{C}$-valid)*, *cf.* EQ 3.68.

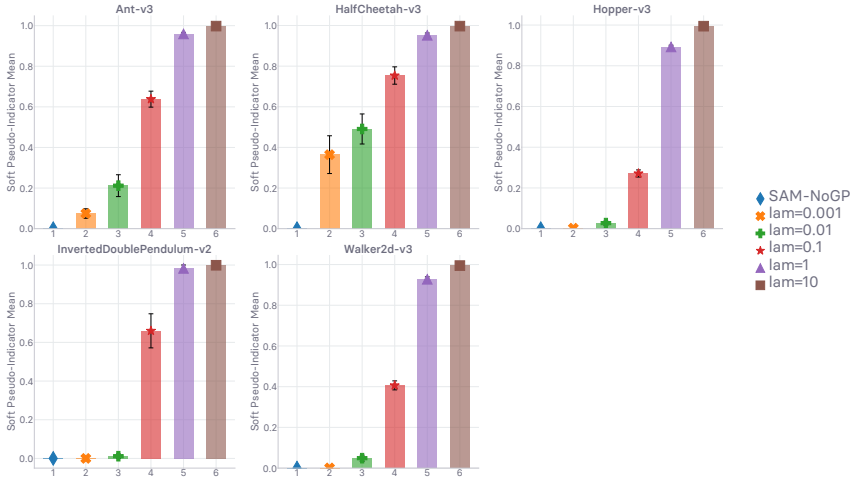
Figure 3.6.3: Evaluation of several GP methods differing by their ζ distribution In line with how we defined it in EQ 3.15, ζ controls ***"where"*** the GP constraint is enforced. Also, we report what happens without any GP regularization (`NoGP`). Explanation in text. Runtime is 48h.



(a) Evolution of return values *(higher is better)*



(b) Evolution of $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values *(higher means more approx. $\mathfrak{C}$-valid)*, *cf.* EQ 3.68.



(c) Final $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values at timeout *(higher means more approx. $\mathfrak{C}$-valid)*, *cf.* EQ 3.68.

Figure 3.6.4: Evaluation of several GP methods differing by their λ scaling factor In line with how we defined it in EQ 3.15, ζ controls ***"to what degree"*** the GP constraint is enforced. Also, we report what happens without any GP regularization (NoGP). Explanation in text. Runtime is 48h.

minimum i: $i = -3$) the soft pseudo-indicator values lie more often that not near 0. For $i = 1$, $\widehat{\mathbb{1}}_{\mathfrak{C}}$ perfectly aligns on the 1 value, meaning that the value we used so far ($\lambda = 10$, which corresponds to λ_i with $i = 1$) is enough for μ_θ to achieve a 100% satisfaction rate of $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$. The case $i = 0$ is right on the edge: in some environments, the approximate $\mathfrak{C}$-validity exactly equals 1, while for other environments, it nears it, yet does not quite reach it.

Since we use WGAN-GP's ζ in the experiments reported in FIGURE 3.6.4, we can first conclude that picking WGAN-GP's ζ variant and $\lambda = 10$ not only yields the best empirical return (as reported and discussed in SECTION 3.6.3), but also guarantees that the constraint $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ (and therefore $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$; *cf.* SECTION 3.6.2)) is satisfied for 100% of the actions performed by the agent's μ_θ in practice. As such, we can conclude that, in practice, the main premise of the theoretical guarantees we have derived in SECTION 3.6.1 — the reward δ-Lipschitz-continuity, $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ — is satisfied, hence making our theoretical guarantees *practically* relevant and insightful. In addition, since we showed that the learning agent's policy μ_θ (or rather, it's companion Q-value) is trained on a reward surrogate r_φ that verifies $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ almost 100% of the time, we have empirically proved that the agent effectively sees virtually uninterrupted sequences of smooth rewards. This new observation somewhat corroborates our RL-grounded interpretation of directed gradient penalization as as the automated and adaptive creation of reward curricula (*cf.* SECTION 3.6.3, and particularly our schematic depiction of WGAN-GP's supp ζ in FIGURE 3.6.1a).

Despite having answered the question we asked in the title of the section (in the block right above), interpreting the findings laid out both in this section and in the previous one side-by-side allows us to draw another critical conclusion, substantially more meaningful than if we were to interpret either in a vacuum. In SECTION 3.6.3, we studied the impact ζ and λ both have on the agent's performance, in terms of the empirical return in the MDP $\mathbb{M}$. We refer here to the latter via the shorthand RETURN. In *this* section, on the other hand, we have studied the impact ζ and λ both have on the effective approximate $\mathfrak{C}$-validity rate of the agent. We refer here to the latter via the shorthand VALIDITY. What emerges from comparing these two sets of results is that, for every given pair (ζ, λ) (*where* to apply the gradient penalty, and *to what degree*, respectively) in ℓ^{GP}_φ (*cf.* EQ 3.15): low RETURN co-occurs with low VALIDITY; intermediate RETURN co-occurs with intermediate VALIDITY; high RETURN co-occurs with high VALIDITY. Said differently, RETURN and VALIDITY behave similarly under the various pairings (ζ, λ) that we have considered. Through these observations, we therefore witness a strong correlation between RETURN and VALIDITY. Ultimately, by combining our two previous empirical analyses, we have shown that VALIDITY is a good predictor or RETURN, and *vice versa*.

In fine, compared to SECTION 3.5.5, SECTION 3.6.4 (this section) gives a far more fine-grained diagnostic of how reward Lipschitzness relates to empirical return, along with insights related to the

practicality of our theorerical guarantees.

3.6.5 Towards fulfilling the premise: a provably more robust way to further encourage Lipschitzness

We introduce two new entities, κ_t and $\tilde{r}_\varphi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, formally defined as:

$$\tilde{r}_\varphi(s_t, a_t) \coloneqq \kappa_t \, r_\varphi(s_t, a_t) \qquad \blacktriangleright \ \kappa_t\text{-preconditioned reward } \tilde{r}_\varphi \tag{3.69}$$

$\forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, where $0 < \kappa_t \leq 1, \forall t \in [0, T] \cap \mathbb{N}$ (in any episode).

We call κ_t a *reward preconditioner* since it functionally echoes the numerical transformation that conditions the tackled problem into a form that is more amenable to be solved via first-order optimization methods. Since our preconditioner is a scalar, we use the shorthand κ_t to constrast with the usual preconditioning matricies, denoted with capitalization. We have the following ranking of values, depending on the sign of the original learned synthetic reward r_φ: $\forall t \in [0, T] \cap \mathbb{N}$ and $\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, we have $\tilde{r}_\varphi(s_t, a_t) \leq r_\varphi(s_t, a_t)$ whenever $r_\varphi(s_t, a_t) > 0$, and conversely, we have $\tilde{r}_\varphi(s_t, a_t) > r_\varphi(s_t, a_t)$ whenever $r_\varphi(s_t, a_t) < 0$.

We posit that κ_t does not depend on (*i.e.*, is constant *w.r.t.*) the current state s_t and action a_t:

$$\mathrm{d}\kappa_t / \mathrm{d}s_t = 0 \qquad \text{and} \qquad \mathrm{d}\kappa_t / \mathrm{d}a_t = 0 \qquad \blacktriangleright \ \textit{property 1} \tag{3.70}$$

$\forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. Thus, we can write $\mathrm{d}\tilde{r}_\varphi(s_t, a_t) / \mathrm{d}s_t = \kappa_t \, \mathrm{d}r_\varphi(s_t, a_t) / \mathrm{d}s_t + \mathrm{d}\kappa_t(s_t, a_t) / \mathrm{d}s_t \; r_\varphi = \kappa_t \, \mathrm{d}r_\varphi(s_t, a_t) / \mathrm{d}s_t$, and similarly $\mathrm{d}\tilde{r}_\varphi(s_t, a_t) / \mathrm{d}a_t = \kappa_t \, \mathrm{d}r_\varphi(s_t, a_t) / \mathrm{d}a_t$. As such, we have $\|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|_F = \kappa_t \|\nabla^t_{s,a}[r_\varphi]_t\|_F$, hence $\|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|_F \leq \|\nabla^t_{s,a}[r_\varphi]_t\|_F$ since $0 < \kappa_t \leq 1$, $\forall t \in [0, T] \cap \mathbb{N}$. Applying such a preconditioner to r_φ therefore squashes the absolute value of r_φ and in effect *shrinks* r_φ's Lipschitz constant (assuming here that r_φ is δ-Lipschitz, with $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta < +\infty$) without regard to the sign of the signal. Formally, since κ_t is posited constant in s_t and a_t, we have, $\forall t \in [0, T] \cap \mathbb{N}$ and $\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$:

$$\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta \qquad \Longrightarrow \qquad \|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|_F = \kappa_t \|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \kappa_t \, \delta \quad (\leq \delta) \tag{3.71}$$

That is, if r_φ is δ-Lipschitz-continuous at t, then $\tilde{r}_\varphi$ is $\kappa_t \delta$-Lipschitz-continuous at t. Importantly, eq. 3.71 will be instrumental in proving the first stages of our next theoretical guarantees, in which we deal with the counterpart action-value of $\tilde{r}_\varphi$, denoted by $\widetilde{Q}_\varphi$.

Because of its *"reward-squashing"* effect, we name the method corresponding to the subtitution of r_φ with the preconditioned reward $\tilde{r}_\varphi$ *"Pessimistic" Reward Preconditioning Enforcing Lipschitzness*. We

dub the plug-in technique **"PURPLE"** (it is an acronym, with minor vowel filling and letter shuffle for legibility and easy of pronunciation).

We describe PURPLE's pseudo-code plugged into SAM ([BK19], *cf.* ALGORITHM 3) in ALGORITHM 4 [4]. Like with ALGORITHM 3, what we depict is a stripped down version of the full algorithm, omitting the various add-on techniques (*e.g.* label smoothing) for legibility. The hyper-parameters that complement ALGORITHM 4 are reported in APPENDIX 3.A.

We now study how the injection of PURPLE in SAM impacts the theoretical guarantees we have previously derived in SECTION 3.6.1. Concretely, we derive the PURPLE counterparts of LEMMA 3.6.1, THEOREM 3.6.2, THEOREM 3.6.3, and COROLLARY 3.6.3.1. In order for us to characterize the Lipschitzness of $\widetilde{Q}_\varphi$, we also posit that the introduced preconditioner does not depend on (*i.e.*, is constant *w.r.t.*) the *previously visited* (past) states and actions. Formally:

$$\mathrm{d}\kappa_{t+k+1}/\mathrm{d}s_t = 0 \qquad \text{and} \qquad \mathrm{d}\kappa_{t+k+1}/\mathrm{d}a_t = 0 \qquad \blacktriangleright \textit{ property 2} \tag{3.72}$$

$\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t-1] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. All in all, to develop the counterpart guarantees that will follow, the preconditioner κ_t must possess the following properties:

$\mathrm{d}\kappa_t/\mathrm{d}s_t = 0$ and $\mathrm{d}\kappa_t/\mathrm{d}a_t = 0$ ► *property 1, EQ 3.70*

► *gave us EQ 3.71, itself used in the proof (step 1) of THEOREM 3.6.5 (a)+(b)*

$\mathrm{d}\kappa_{t+k+1}/\mathrm{d}s_t = 0$ and $\mathrm{d}\kappa_{t+k+1}/\mathrm{d}a_t = 0$ ► *property 2, EQ 3.72*

► *used in the proof of LEMMA 3.6.4, itself then used to prove (step 2) THEOREM 3.6.5 (a)+(b)*

$\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t-1] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. Note, the last two properties, EQ 3.70 and EQ 3.72, can be condensed into, $\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$:

$$\mathrm{d}\kappa_{t+k}/\mathrm{d}s_t = 0 \qquad \text{and} \qquad \mathrm{d}\kappa_{t+k}/\mathrm{d}a_t = 0 \qquad \blacktriangleright \textit{ property 1+2 condensed into one} \tag{3.73}$$

PROPERTY THAT κ_t MUST HAVE. ***In plain English, to get our guarantees, we need the preconditioner to not depend on neither current nor past states visited and actions taken by the agent.*** Note, the property $\kappa_t \leq 1$ is only ever used in SECTION 3.6.6.1, and will not be leveraged anywhere else. The developed theory will still hold if $\exists t \in [0, T] \cap \mathbb{N}$ such that $\kappa_t > 1$.

Lemma 3.6.4. *Let the MDP with which the agent interacts be deterministic, with the dynamics of the*

[4] As indicated earlier, the symbols "◇" and "◇" appearing in front of line numbers in ALGORITHM 4 are related to the distributed learning scheme used in this work, which we describe in section 3.5.5.

Algorithm 4: SAM augmented with our provably more robust extension *Pessimistic Reward Preconditioning Enforcing Lipschitzness*, dubbed **"PURPLE"** (with minor vowel filling and letter shuffle for legibility).
Text in red color: differing from SAM (*cf.* ALGORITHM 3)

init: initialize the random seeds of each framework used for sampling, the random seed of the environment $\mathbb{M}$, the neural function approximators' parameters $(\theta, \varphi, \omega)$, their target networks as exact frozen copies, the rollout cache $\mathcal{C}$, the replay buffer $\mathcal{R}$.

while *no stopping criterion is met* **do**
  /* Interact with the world to collect new samples */
  repeat
    Perform action $a_t \sim \pi_\theta(\cdot|s_t)$ in state s_t and receive the next state s_{t+1} and termination indicator d returned by the environment $\mathbb{M}^* - \{r_\varphi\}$;
    Store the reward-less transition (s_t, a_t, s_{t+1}) in the rollout cache $\mathcal{C}$;
  until *the rollout cache* $\mathcal{C}$ *is full*;
  Dump the content of the rollout cache $\mathcal{C}$ into the replay buffer $\mathcal{R}$, then flush $\mathcal{C}$;
  /* Train every modules */
  foreach *training step per iteration* **do**
    foreach *reward training step per iteration* **do**
      Get a mini-batch of samples from the replay buffer $\mathcal{R}$;
      Get a mini-batch of samples from the expert demonstration dataset $\mathcal{D}$;
      Perform a gradient *descent* step along $\nabla_\varphi \ell_\varphi^{\mathrm{GP}}$ (*cf.* EQ 3.2) using *both* mini-batches:

$$\ell_\varphi^{\mathrm{GP}} := \mathbb{E}_{s_t \sim \rho^{\pi_e}, a_t \sim \pi_e}\left[-\log\left(1 - D_\varphi(s_t, a_t)\right)\right] + \mathbb{E}_{s_t \sim \rho^{\beta}, a_t \sim \beta}\left[-\log\left(D_\varphi(s_t, a_t)\right)\right] + \lambda\, \mathfrak{R}_\varphi^\zeta(k)$$

      where $\mathfrak{R}_\varphi^\zeta(k) := \mathbb{E}_{s_t \sim \rho^\zeta, a_t \sim \zeta}[(\|\nabla_{s_t, a_t} D_\varphi(s_t, a_t)\| - k)^2]$ is a gradient penalty regularizer;
    end
    foreach *agent training step per iteration* **do**
      Get a mini-batch of samples from the replay buffer $\mathcal{R}$;
      Compute the reward preconditioner $\kappa_t \in (0, 1]$;
      Assemble the new reward $\tilde{r}_\varphi(s_t, a_t)$ by preconditioning the learned surrogate r_φ with κ_t:

$$\tilde{r}_\varphi(s_t, a_t) := \kappa_t\, r_\varphi(s_t, a_t)$$

      as laid out in EQ 3.69;
      Augment every reward-less transition sampled from $\mathcal{R}$ with the reward surrogate $\tilde{r}_\varphi$ (*cf.* EQ 3.69): $(s_t, a_t, s_{t+1}) \to (s_t, a_t, \tilde{r}_\varphi(s_t, a_t), s_{t+1})$ (omitting here the use of n-step returns for simplicity);
      Perform a gradient *descent* step along $\nabla_\omega \ell_\omega$ (*cf.* EQ 3.4) using the mini-batch:

$$\ell_\omega := \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta}[(Q_\omega(s_t, a_t) - Q^{\mathrm{targ}})^2]$$

      where $Q^{\mathrm{targ}} := \sum_{k=0}^{n-1} \gamma^k r_\varphi(s_{t+k}, a_{t+k}) + \gamma^n Q_{\omega'}(s_{t+n}, \mu_{\theta'}(s_{t+n}))$ is the n-step Bellman target;
      Perform a gradient *ascent* step along $\nabla_\theta \mathcal{U}_\theta$ (*cf.* EQ 3.7) using the mini-batch:

$$\mathcal{U}_\theta := \mathbb{E}_{s_t \sim \rho^\beta}[Q_\omega(s_t, \mu_\theta(s_t))]$$

      ;
      Update the target networks using the new ω and θ;
    end
  end
  Adapt parameter noise standard deviation σ used to define π_θ from μ_θ (*cf.* SECTION 3.4);
  /* Evaluate the trained policy */
  foreach *evaluation step per iteration* **do**
    Evaluate the empirical return of μ_θ in $\mathbb{M}$, using the task reward r (*cf.* SECTION 3.4);
  end
end

environment determined by the function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. *The agent follows a deterministic policy* $\mu : \mathcal{S} \to \mathcal{A}$ *to map states to actions, and receives rewards from* $r_\varphi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *upon interaction. The functions* f, μ *and* r_φ *need be* C^0 *and differentiable over their respective input spaces. This property is satisfied by the usual neural network function approximators. The "almost-everywhere" case can be derived from this lemma without major changes (relevant when at least one activation function is only differentiable almost-everywhere, ReLU).* **(a)** *Under the previous assumptions, for* $k \in [0, T-k-1] \cap \mathbb{N}$ *the following* ***(non-recursive) inequality*** *is verified:*

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k+1}\|_F^2 \leq \kappa^2_{t+k+1} C_t \, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \tag{3.74}$$

where $0 < \kappa_u \leq 1 \; \forall u \in [0, T] \cap \mathbb{N}$, *and* $C_t \coloneqq A_t^2 \max(1, B_{t+1}^2)$, A_t *and* B_t *being defined as the supremum norms associated with the Jacobians of* f *and* μ *respectively, with values in* $\mathbb{R} \cup \{+\infty\}$*:*

$$\forall t \in [0, T] \cap \mathbb{N}, \quad \begin{cases} A_t \coloneqq \|\nabla^t_{s,a}[f]_t\|_\infty = \sup \left\{ \|\nabla^t_{s,a}[f]_t\|_F \, : \, (s_t, a_t) \in \mathcal{S} \times \mathcal{A} \right\} \\ B_t \coloneqq \|\nabla^t_{s}[\mu]_t\|_\infty = \sup \left\{ \|\nabla^t_{s}[\mu]_t\|_F \, : \, s_t \in \mathcal{S} \right\} \end{cases} \tag{3.75}$$

(b) *Additionally, by introducing* ***time-independent*** *upper bounds* $A, B \in \mathbb{R} \cup \{+\infty\}$ *such that* $\forall t \in [0, T] \cap \mathbb{N}$, $A_t \leq A$ *and* $B_t \leq B$, *and* κ *such that* $\kappa_u \leq \kappa \leq 1 \; \forall u \in [0, T] \cap \mathbb{N}$, *the non-recursive inequality becomes:*

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k+1}\|_F^2 \leq \kappa^2 C \, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \tag{3.76}$$

where $C \coloneqq A^2 \max(1, B^2)$ *is the time-independent counterpart of* C_t.

Proof of Lemma 3.6.4 (a). *(a)* First, we take the derivative with respect to each variable separately:

$$\begin{aligned} \nabla^t_s[\tilde{r}_\varphi]_{t+k+1} &= \mathrm{d}\tilde{r}_\varphi(s_{t+k+1}, a_{t+k+1}) \,/\, \mathrm{d}s_t && (3.77) \\ &= \kappa_{t+k+1} \, \mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1}) \,/\, \mathrm{d}s_t \qquad \blacktriangleright \textit{ we here assume that } \mathrm{d}\kappa_{t+k+1}/\mathrm{d}s_t = 0 && (3.78) \\ &= \kappa_{t+k+1} \, \nabla^t_s[r_\varphi]_{t+k+1} \qquad \blacktriangleright \textit{ repack} && (3.79) \end{aligned}$$

$$\begin{aligned} \nabla^t_a[\tilde{r}_\varphi]_{t+k+1} &= \mathrm{d}\tilde{r}_\varphi(s_{t+k+1}, a_{t+k+1}) \,/\, \mathrm{d}a_t && (3.80) \\ &= \kappa_{t+k+1} \, \mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1}) \,/\, \mathrm{d}a_t \qquad \blacktriangleright \textit{ we here assume that } \mathrm{d}\kappa_{t+k+1}/\mathrm{d}a_t = 0 && (3.81) \\ &= \kappa_{t+k+1} \, \nabla^t_a[r_\varphi]_{t+k+1} \qquad \blacktriangleright \textit{ repack} && (3.82) \end{aligned}$$

By assembling the norm with respect to both input variables, we get:

$$
\begin{aligned}
&\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k+1}\|^2_F \\
&\quad = \|\nabla^t_s[\tilde{r}_\varphi]_{t+k+1}\|^2_F + \|\nabla^t_a[\tilde{r}_\varphi]_{t+k+1}\|^2_F && (3.83)\\
&\quad = \kappa^2_{t+k+1}\,\|\nabla^t_s[r_\varphi]_{t+k+1}\|^2_F + \kappa^2_{t+k+1}\,\|\nabla^t_a[r_\varphi]_{t+k+1}\|^2_F && (3.84)\\
&\quad = \kappa^2_{t+k+1}\left(\|\nabla^t_s[r_\varphi]_{t+k+1}\|^2_F + \|\nabla^t_a[r_\varphi]_{t+k+1}\|^2_F\right) && (3.85)\\
&\quad = \kappa^2_{t+k+1}\,\|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|^2_F \qquad \blacktriangleright \textit{ total norm} && (3.86)
\end{aligned}
$$

As in Lemma 3.6.1, let A_t, B_t and C_t be time-dependent quantities defined as:

$$
\forall t \in [0, T] \cap \mathbb{N}, \quad
\begin{cases}
A_t \coloneqq \|\nabla^t_{s,a}[f]_t\|_\infty = \sup\left\{\|\nabla^t_{s,a}[f]_t\|_F \,:\, (s_t, a_t) \in \mathcal{S}\times\mathcal{A}\right\}\\
B_t \coloneqq \|\nabla^t_s[\mu]_t\|_\infty = \sup\left\{\|\nabla^t_s[\mu]_t\|_F \,:\, s_t \in \mathcal{S}\right\}\\
C_t \coloneqq A_t^2\max(1, B^2_{t+1})
\end{cases} \tag{3.87}
$$

Finally, by injecting eq. 3.36, we directly obtain:

$$
\begin{aligned}
\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k+1}\|^2_F &= \kappa^2_{t+k+1}\,\|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|^2_F && (3.88)\\
&\le \kappa^2_{t+k+1}\,A_t^2\max(1, B^2_{t+1})\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|^2_F \qquad \blacktriangleright \textit{ eq 3.36} && (3.89)\\
&= \kappa^2_{t+k+1}\,C_t\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|^2_F \qquad \blacktriangleright \textit{ } C_t \textit{ definition} && (3.90)
\end{aligned}
$$

which concludes the proof of Lemma 3.6.4 *(a)*. □

Proof of Lemma 3.6.4 (b). By introducing time-independent upper bounds A and B such that $A_t \le A$ and $B_t \le B\ \forall t \in [0, T] \cap \mathbb{N}$, $C \coloneqq A^2\max(1, B^2)$, and κ such that $\kappa_u \le \kappa \le 1\ \forall u \in [0, T] \cap \mathbb{N}$, we obtain, through eq. 3.89:

$$
\begin{aligned}
\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k+1}\|^2_F &\le \kappa^2\,A^2\max(1, B^2)\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|^2_F && (3.91)\\
&= \kappa^2\,C\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|^2_F && (3.92)
\end{aligned}
$$

which concludes the proof of Lemma 3.6.4 *(b)*. □

Theorem 3.6.5 (gap-dependent reward Lipschitzness)**.** *In addition to the assumptions laid out in lemma 3.6.4, we assume that the function r_φ is δ-Lipschitz over $\mathcal{S}\times\mathcal{A}$. Since r_φ is C^0 and differentiable over $\mathcal{S}\times\mathcal{A}$, this assumption can be written as $\|\nabla^u_{s,a}[r_\varphi]_u\|_F \le \delta$, where $u \in [0, T] \cap \mathbb{N}$.* **(a)** *Then,*

under these assumptions, the following is verified:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k}\|^2_F \leq \kappa^2_{t+k}\,\delta^2 \prod_{u=0}^{k-1} C_{t+u} \tag{3.93}$$

where $k \in [0, T] \cap \mathbb{N}$ *and* C_v *is defined as in* Lemma *3.6.4 (a),* $\forall v \in [0, T] \cap \mathbb{N}$. **(b)** *Additionally, by involving the time-independent upper bounds introduced in* Lemma *3.6.4 (b), we have the following:*

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k}\|^2_F \leq \kappa^2\, C^k\, \delta^2 \tag{3.94}$$

where $k \in [0, T] \cap \mathbb{N}$*; C and* κ *are defined as in* Lemma *3.6.4 (b).*

Proof of Theorem *3.6.5* (a). We will prove Theorem 3.6.5 *(a)* directly, as opposed to by induction (Lemma 3.6.4 proposes non-recursive inequalities, one side containing r_φ, the other $\tilde{r}_\varphi$). We want to prove the following eq. 3.95, $\forall v \in [0, T] \cap \mathbb{N}$:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+v}\|^2_F \leq \kappa^2_{t+v}\,\delta^2 \prod_{u=0}^{v-1} C_{t+u} \tag{3.95}$$

To do so, we will procede in two steps: *(1)* prove it for $v = 0$, and *(2)* prove it $\forall v \in [1, T] \cap \mathbb{N}$.

Step 1: case $v = 0$. When the gap $v = 0$, eq. 3.95 becomes $\|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|^2_F \leq \kappa^2_t\,\delta^2$, $\forall t \in [0, T] \cap \mathbb{N}$, which is verified by coupling Theorem 3.6.5's main assumption about the δ-Lipschitzness of r_φ and the observation laid out in eq. 3.71.

Step 2: case $v \in [1, T] \cap \mathbb{N}$. We start from the result we derived in Lemma 3.6.4 *(a)*, valid $\forall w \in [0, T-1] \cap \mathbb{N}$:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+w+1}\|^2_F \leq \kappa^2_{t+w+1}\, C_t\, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+w+1}\|^2_F \qquad \blacktriangleright \textit{ Lemma 3.6.4 (a)} \tag{3.96}$$

$$\leq \kappa^2_{t+w+1}\, C_t\, \delta^2 \prod_{u=0}^{w-1} C_{t+1+u} \qquad \blacktriangleright \textit{ Theorem 3.6.2 (a), at } t+1 \tag{3.97}$$

$$= \kappa^2_{t+w+1}\, C_t\, \delta^2 \prod_{u=1}^{w} C_{t+u} \qquad \blacktriangleright \textit{ index shift} \tag{3.98}$$

$$= \kappa^2_{t+w+1}\, \delta^2 \prod_{u=0}^{w} C_{t+u} \qquad \blacktriangleright \textit{ repack product} \tag{3.99}$$

This shows that eq. 3.95 is verified when $v = w + 1$, $\forall w \in [0, T-1] \cap \mathbb{N}$. eq. 3.95 is therefore valid $\forall v \in [1, T] \cap \mathbb{N}$.

Conclusion. We have shown that EQ 3.95 is valid $\forall v \in [0, T] \cap \mathbb{N}$, which concludes the proof of THEOREM 3.6.5 *(a)*. □

Proof of THEOREM 3.6.5 (b). We will prove THEOREM 3.6.5 *(b)* directly, as opposed to by induction (LEMMA 3.6.4 proposes non-recursive inequalities, one side containing r_φ, the other $\tilde{r}_\varphi$). We want to prove the following EQ 3.100, $\forall v \in [0, T] \cap \mathbb{N}$:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+v}\|^2_F \leq \kappa^2\, C^v\, \delta^2 \tag{3.100}$$

where κ satisfies $\kappa_u \leq \kappa \leq 1\, \forall u \in [0, T] \cap \mathbb{N}$.

To do so, we will procede in two steps: *(1)* prove it for $v = 0$, and *(2)* prove it $\forall v \in [1, T] \cap \mathbb{N}$.

Step 1: case $v = 0$. When the gap $v = 0$, EQ 3.100 becomes $\|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|^2_F \leq \kappa_t^2\, \delta^2 \leq \kappa^2\, \delta^2, \forall t \in [0, T] \cap \mathbb{N}$, which is verified by coupling THEOREM 3.6.5's main assumption about the δ-Lipschitzness of r_φ, the observation laid out in EQ 3.71, and finally the definition of κ (upper bound for all the κ_u's).

Step 2: case $v \in [1, T] \cap \mathbb{N}$. We start from the result we derived in LEMMA 3.6.4 *(b)*, valid $\forall w \in [0, T-1] \cap \mathbb{N}$:

$$\begin{aligned}
\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+w+1}\|^2_F &\leq \kappa^2\, C\, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+w+1}\|^2_F && \blacktriangleright \textit{ LEMMA 3.6.4 (b)} && (3.101)\\
&\leq \kappa^2\, C\, C^w\, \delta^2 && \blacktriangleright \textit{ THEOREM 3.6.2 (b), at } t+1 && (3.102)\\
&= \kappa^2\, C^{w+1}\, \delta^2 && \blacktriangleright \textit{ repack product} && (3.103)
\end{aligned}$$

This shows that EQ 3.100 is verified when $v = w+1, \forall w \in [0, T-1] \cap \mathbb{N}$. EQ 3.100 is therefore valid $\forall v \in [1, T] \cap \mathbb{N}$.

Conclusion. We have shown that EQ 3.100 is valid $\forall v \in [0, T] \cap \mathbb{N}$, which concludes the proof of THEOREM 3.6.5 *(b)*. □

Theorem 3.6.6 (state-action value Lipschitzness)**.** *We work under the assumptions laid out in both LEMMA 3.6.4 and THEOREM 3.6.5, and repeat the main lines here for THEOREM 3.6.6 to be self-contained: a) The functions f, μ and r_φ are C^0 and differentiable over their respective input spaces, and b) the function r_φ is δ-Lipschitz over $\mathcal{S} \times \mathcal{A}$, i.e. $\|\nabla^u_{s,a}[r_\varphi]_u\|_F \leq \delta$, where $u \in [0, T] \cap \mathbb{N}$. Then the quantity $\nabla^u_{s,a}[\widetilde{Q}_\varphi]_u$ exists $\forall u \in [0, T] \cap \mathbb{N}$, and verifies:*

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \begin{cases} \kappa\delta \sqrt{\dfrac{1 - \left(\gamma^2 C\right)^{T-t}}{1 - \gamma^2 C}}, & \text{if } \gamma^2 C \neq 1 \\ \kappa\delta\sqrt{T-t}, & \text{if } \gamma^2 C = 1 \end{cases} \tag{3.104}$$

$\forall t \in [0, T] \cap \mathbb{N}$, *where* $C \coloneqq A^2 \max(1, B^2)$, *with* A *and* B *time-independent upper bounds of* $\|\nabla^t_{s,a}[f]_t\|_\infty$ *and* $\|\nabla^t_s[\mu]_t\|_\infty$ *respectively (see* EQ *3.87 for definitions of the supremum norms), and where* κ *satisfies* $\kappa_u \leq \kappa \leq 1 \, \forall u \in [0, T] \cap \mathbb{N}$.

Proof of THEOREM *3.6.6.* With finite horizon T, we have $\widetilde{Q}_\varphi(s_t, a_t) \coloneqq \sum_{k=0}^{T-t-1} \gamma^k \, \tilde{r}_\varphi(s_{t+k}, a_{t+k}), \forall t \in [0, T] \cap \mathbb{N}$, since f, μ, r_φ, and $\tilde{r}_\varphi$ (*cf.* EQ 3.69) are all deterministic (no expectation). Additionally, since r_φ is assumes to be C^0 and differentiable over $\mathcal{S} \times \mathcal{A}$, $\widetilde{Q}_\varphi$ is by construction also C^0 and differentiable over $\mathcal{S} \times \mathcal{A}$. Consequently, $\nabla^u_{s,a}[\widetilde{Q}_\varphi]_u$ exists, $\forall u \in [0, T] \cap \mathbb{N}$. Since both r_φ and $\widetilde{Q}_\varphi$ are scalar-valued (their output space is $\mathbb{R}$), their Jacobians are the same as their gradients. We can therefore use the linearity of the gradient operator: $\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t = \sum_{k=0}^{T-t-1} \gamma^k \, \nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k}, \forall t \in [0, T] \cap \mathbb{N}$.

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 = \left\| \sum_{k=0}^{T-t-1} \gamma^k \, \nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k} \right\|_F^2 \qquad \blacktriangleright \textit{ operator's linearity} \tag{3.105}$$

$$\leq \sum_{k=0}^{T-t-1} \gamma^{2k} \, \|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k}\|_F^2 \qquad \blacktriangleright \textit{ triangular inequality} \tag{3.106}$$

$$\leq \sum_{k=0}^{T-t-1} \gamma^{2k} \, \kappa^2 \, C^k \, \delta^2 \qquad \blacktriangleright \textit{ THEOREM 3.6.5 (b)} \tag{3.107}$$

$$= (\kappa\delta)^2 \sum_{k=0}^{T-t-1} \left(\gamma^2 C\right)^k \tag{3.108}$$

When $\gamma^2 C = 1$, we obtain $\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 = \delta^2(T - t)$. On the other hand, when $\gamma^2 C \neq 1$:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 \leq (\kappa\delta)^2 \, \frac{1 - \left(\gamma^2 C\right)^{T-t}}{1 - \gamma^2 C} \qquad \blacktriangleright \textit{ finite sum of geometric series} \tag{3.109}$$

$$\implies \quad \|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 \leq \begin{cases} (\kappa\delta)^2 \, \dfrac{1 - \left(\gamma^2 C\right)^{T-t}}{1 - \gamma^2 C}, & \text{if } \gamma^2 C \neq 1 \\ (\kappa\delta)^2 (T - t), & \text{if } \gamma^2 C = 1 \end{cases} \tag{3.110}$$

By applying $\sqrt{\cdot}$ (monotonically increasing) to the inequality, we obtain the claimed result. □

Finally, we derive a corollary from THEOREM 3.6.6 corresponding to the infinite-horizon regime.

Corollary 3.6.6.1 (infinite-horizon regime)**.** *Under the assumptions of* THEOREM *3.6.6, including that* r_φ *is* δ*-Lipschitz and that* $\tilde{r}_\varphi$ *is defined as in* EQ *3.69 over* $\mathcal{S} \times \mathcal{A}$*, and assuming that* $\gamma^2 C < 1$*, we*

have, in the infinite-horizon regime:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \frac{\kappa\partial}{\sqrt{1-\gamma^2 C}} \tag{3.111}$$

which translates into $\widetilde{Q}_\varphi$ being $\frac{\kappa\partial}{\sqrt{1-\gamma^2 C}}$-Lipschitz over $\mathcal{S}\times\mathcal{A}$.

Proof of Corollary 3.6.6.1. By following the proof of Corollary 3.6.3.1, using Theorem 3.6.5 instead of Theorem 3.6.2, we arrive directly at the claimed result. □

Remark 1. *Say we were to write a proof analogous to the one laid out right above for* Theorem 3.6.6, *but using the time-*dependent *version of* Theorem 3.6.5 *instead of the time-*independent *version that we used in* eq 3.107 *(version 3.6.5 (a) instead of 3.6.5 (b)). Despite not being identifiable as a finite or infinite sum of geometric series, the expression we would get instead of* eq 3.107 *not only is a tighter bound by construction, but it also has an interesting form:*

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|^2_F \leq \sum_{k=0}^{T-t-1}\left[\gamma^{2k}\,\kappa^2_{t+k}\,\partial^2\prod_{u=0}^{k-1}C_{t+u}\right] \qquad \blacktriangleright \textit{ Theorem 3.6.5 (a)} \tag{3.112}$$

Going through the first operands of the sum, and looking solely at the "κ" and "C" factors, we have the following:

$$\kappa^2_t \to \kappa^2_{t+1}\,C_t \to \kappa^2_{t+2}\,C_tC_{t+1} \to \kappa^2_{t+3}\,C_tC_{t+1}C_{t+2} \to \ldots \to \kappa^2_T\,C_tC_{t+1}C_{t+2}\ldots C_{T-1} \tag{3.113}$$

This observation tells us that, in the derived Lipschitz constant of $\widetilde{Q}_\varphi$, the reward preconditioner κ_t at time t can compensate for all the past *values $\{C_v \mid v < t\}$. Intuitively, the more we wait to reduce κ_t, the more the* next *κ_t's will need to compensate for the "negligence" of their predecessors. Note, the product of $\{C_v \mid v < t\}$ compounds quickly.*

3.6.6 Discussion II: implications and limitations of the theoretical guarantees

3.6.6.1 Provably more robust

Given that, in this work, we aligned the notion of robustness of a function approximator with the value of its Lipschitz constant (*more* robust means *lower* Lipschitz constant, *cf.* Section 3.4), and given that κ_t's upper bound κ verifies $\kappa \leq 1$ (*cf.* Lemma 3.6.4), we can write, from the result of

Corollary 3.6.6.1:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \frac{\kappa\delta}{\sqrt{1-\gamma^2 C}} = \kappa\,\Delta_\infty \coloneqq \widetilde{\Delta}_\infty \leq \Delta_\infty \tag{3.114}$$

where $\Delta_\infty \coloneqq \delta/\sqrt{1-\gamma^2 C}$ is the upper bound of Q_φ's Lipschitz constant that we derived in Corollary 3.6.3.1. Note, all of what is written in this remark concerns the infinite-horizon regime, but one can derive the finite-horizon counterpart trivially — using Theorem 3.6.3 instead of Corollary 3.6.3.1, and Theorem 3.6.6 instead of Corollary 3.6.6.1 — to arrive at the same conclusion: $\widetilde{Q}_\varphi$ has a lower derived Lipschitz constant upper bound than Q_φ by a factor of $\kappa \leq 1$ and is therefore *provably more robust* than Q_φ. In other words, employing the simple PURPLE reward preconditioning to SAM has the effect of making the learned Q-value provably more robust.

3.6.6.2 Detached guide

Consider the following particular form for $\kappa_t, \forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$:

$$\kappa_t \coloneqq \exp(-\alpha\,\varepsilon_t) \quad \Longrightarrow \quad \tilde{r}_\varphi(s_t, a_t) \coloneqq \kappa_t\, r_\varphi(s_t, a_t) \coloneqq \exp(-\alpha\,\varepsilon_t)\, r_\varphi(s_t, a_t) \tag{3.115}$$

where α is an inverse temperature hyper-parameter involved in the definition of the kernel of the Boltzmann or Gibbs probability distribution $\kappa_t \coloneqq \exp(-\alpha\,\varepsilon_t)$, (hence $0 < \kappa_t \leq 1$), and where $\varepsilon_t \geq 0$ for now depicts an arbitrary non-negative energy function. κ_t is non-normalized, and as such, it is *not* a probability *per se*. Nonetheless, it still echoes the propensity or tendency of the state-action pair (s_t, a_t) to possess the property described by the non-negative energy ε_t, which we define momentarily. Low values of $\varepsilon_t \geq 0$ will push the preconditioner towards the upper limit $\kappa_t \to 1$, while high energy values will make it tend towards the lower limit $\kappa_t \to 0$ with $\kappa_t > 0$. Equivalently, the preconditioned reward $\tilde{r}_\varphi$ will verify the approximate identity $\tilde{r}_\varphi(s_t, a_t) \approx r_\varphi(s_t, a_t)$ whenever ε_t approaches zero (from above), and $\tilde{r}_\varphi(s_t, a_t) \approx 0$ whenever the energy ε_t grows towards higher levels. Under this orchestration, we need $\mathrm{d}\varepsilon_{t+k}/\mathrm{d}s_t = 0$ and $\mathrm{d}\varepsilon_{t+k}/\mathrm{d}a_t = 0$ to be satisfied $\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$ for the derived robustness guarantees to be readily applicable (we laid out the properties κ_t must possess in Section 3.6.5, right before exposing Lemma 3.6.4).

In particular, the soft approximate $\mathfrak{C}$-validity pseudo-indicator (*cf.* eq. 3.68) is an instantiation of the κ_t form laid out in eq. 3.115, where $\alpha = 1$ for the inverse temperature, and $\varepsilon_t = \max(0, \|\nabla_{s_t,a_t} D_\varphi(s_t, a_t)\| - k)^2$ for the energy. In such an instance, $\tilde{r}_\varphi(s_t, a_t) \approx r_\varphi(s_t, a_t)$ whenever the pair (s_t, a_t) is approximately $\mathfrak{C}$-valid, formally, $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$. Conversely, in the extreme scenario where $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \gg k$, ε_t grows large, κ_t is approximately equal to 0, and $\tilde{r}_\varphi(s_t, a_t) \approx 0$. As such, in effect, the agent's

policy μ_θ is punished for selecting actions that do not satisfy the approximate $\mathfrak{C}$-validity condition above. Besides, it is punished in accordance to how far outside the allowed range, $[0, k]$, the norm of the Jacobian of D_φ gets. Nonetheless, in this particular instance, the empirical observations we have made in SECTION 3.6.4 attest to the fact that, provided the right choice of λ scaling factor and ζ distribution (both characterizing the gradient penalization), the approximate $\mathfrak{C}$-validity constraint $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ can easily be satisfied 100% of the time by *only* regularizing D_φ. For D_φ's k-Lipschitzness to be ensured, there is therefore no need to further alter the rewards provided to the agent's policy μ_θ through PURPLE's pessimistic reward preconditioning. Note, however, that under such a ε_t formulation, we see that we clearly have $\mathrm{d}\varepsilon_{t+k}/\mathrm{d}s_t \neq 0$ and $\mathrm{d}\varepsilon_{t+k}/\mathrm{d}a_t \neq 0, \forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. While this does not mean that the studied entities are not robust, it prevents us from applying our derived results to guarantee such robustness. Still, without the full premise, they are not applicable.

Generally speaking, we will probably make the same observation whenever ε_t is defined from a constraint we want to enforce on a learned function approximation, for regularization purposes. Indeed, verifying said desideratum on the function approximator directly via the application of a regularizer seems to always be the easiest (since most direct) solution to encourage the satisfaction of a constraint on a *differentiable* function (*e.g.* D_φ, μ_θ). Constraints involving the Jacobian of a (*a fortioni* differentiable) function of the learned system (*e.g.* $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$) is a particular case of the general class of constraints for which *direct* regularization is *a priori* prefereable to an analogous reward shaping as dictated by EQ 3.115. On the flip side, due to the fact that the reward — albeit learned as a parametric function — is treated as an input in our computational graph, it is not differentiated through and *can* consequently be augmented with non-differentiable nodes through the design of ε_t. In other words, even if it is preferable to apply regularization directly the objective of the regularized function approximator for it to satisfy some constraint, it might not always be possible to do so directly. In that case, guiding the policy towards areas of the state-action landscape that satisfy said constraint could be a surrogate solution, albeit far less preferable than acting on the targeted approximator directly.

As such, by aligning ε_t with said constraint, EQ 3.115 offers a way for the policy to act in view of the satisfaction of said constraint *while* enjoying the considerable advantage of being able to treat ε_t as a *black box*. We will leverage this *universality* in the next discussion point.

3.6.6.3 PARTIAL COMPENSATION OF COMPOUNDING VARIATIONS

In reaction to the theoretical robustness guarantees derived in THEOREM 3.6.3 and COROLLARY 3.6.3.1, we have discussed earlier in SECTION 3.6.2.3 that, if the variations *in space* of the policy or the dynamics are large in the early stage of an episode (*i.e.* when $0 \leq t \ll T$), then Δ_t (the variation bound on

Q_φ) might explode. As results, $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F$ would then be unbounded, leaving us unable to guarantee the robustness of the learned Q-value Q_φ. The earlier large variations in either or both the policy and dynamics manifest, the more likely these variations are to compound to unreasonably high levels. Concretely, the degree of such compounding variations in space is entirely determined by the operand $\gamma^2 C$ that appears in the variation bounds derived in both Theorem 3.6.3 and Corollary 3.6.3.1. The exact same line of reasoning holds for the variation bounds laid out later in Section 3.6.5, in both Theorem 3.6.6 and Corollary 3.6.6.1 respectively. These guarantees unanimously agree on the critical role that C plays in the robustness bounds, which we here called variation bounds indifferently. Loosely, *high* values of C prevent Q_φ from enjoying the Lipschitzness guarantees laid out in Section 3.6.1 and Section 3.6.5. As such, it is paramount to devise a way to keep C in check by somewhat controling its magnitude, thereby preventing it from voiding our theoretical guarantees and from adopting a brittle behavior. We defined C in Lemma 3.6.1 *(b)* as $C \coloneqq A^2 \max(1, B^2)$, where A and B are time-independent upper bounds of the supremum Frobenius norms of the Jacobians of the dynamics f and the policy μ, $\|\nabla^t_{s,a}[f]_t\|_\infty$ and $\|\nabla^t_s[\mu]_t\|_\infty$, respectively (*cf.* eq. 3.33 for definitions of the supremum norms $\|\cdot\|_\infty$). Simply, $\forall t \in [0, T] \cap \mathbb{N}$, $\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, $\|\nabla^t_{s,a}[f]_t\|_\infty \leq A$ and $\|\nabla^t_s[\mu]_t\|_\infty \leq B$. As such, to devise a way to limit the magnitude of C, we seek ways to limit the respective magnitudes of the A and B majorants. Similarly to the learned surrogate reward core D_φ, the policy μ_θ followed by the agent (of which μ is a placeholder) is learned as a parametric function approximator, enabling us to tame B by applying a gradient penalty regularizer *directly* on the policy (exactly like we already do to ensure that D_φ remains k-Lipschitz-continuous).

By contrast, we can not tame A the same way (via direct regularization applied onto f), due to the transition function f of the world (whether real or simulated) being a black box that we can not even query at will. Not only is f non-differentiable (the real world never is; non-trivial simulated worlds virtually never are), but we also can *not* evaluate it at *any state-action pair whenever we want*. Our desideratum then ultimately boils down to finding a way to keep A in check, since the usual candidate to enforce Lipschitzness (applying a regularizer on the Jacobian directly) — which is the preferable option by far for D_φ and μ_θ — is out of the question for f, as we have established. Despite the fact that, by nature, we can not change f in the MDP $\mathbb{M}$, we *can* change the transition function f' that effectively takes the place of f in practice and underlies the effectively observed MDP $\mathbb{M}'$ by urging the agent's policy μ_θ to avoid areas of the state-action landscape $\mathcal{S} \times \mathcal{A}$ that display high $\|\nabla^t_{s,a}[f']_t\|_\infty$ values. In fact, f' changes continually (f' is non-stationary) throughout the learning process as the preferences of the agent evolve across learning episodes. It is therefore fair to posit that we can devise a way to skew the policy towards areas of $\mathcal{S} \times \mathcal{A}$ where $\|\nabla^t_{s,a}[f']_t\|_\infty$ is tightly upper-bounded. As such, we can keep A in check by keeping $\|\nabla^t_{s,a}[f']_t\|_\infty$ in check in practice, which can be approximately achieved by keeping $\|\nabla^t_{s,a}[f_\psi]_t\|_\infty$ in check, where $f_\psi : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a learned

functional approximation of the effective dynamics f'.

In fine, we urge the constraint $\|\nabla^t_{s,a}[f]_t\|_\infty \leq A$ to be satisfied by encouraging μ_θ to avoid areas where $\|\nabla^t_{s,a}[f_\psi]_t\|_\infty$ is high, which itself can be relaxed into $\|\nabla^t_{s,a}[f_\psi]_t\|_F$. Note, even if f_ψ is differentiable, regularizing it via gradient penalization does *not* have any effect on the value of $\|\nabla^t_{s,a}[f']_t\|_\infty$, since the agent does *not* interact with f_ψ, but with f'. For our line of reasoning to hold, we want $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ to be a high-fidelity depiction of $\|\nabla^t_{s,a}[f']_t\|_\infty$.

We maintain the parametric model f_ψ because it allows us to approximate the norm of the Jacobian of the dynamics wherever we want, whenever we want. In order for μ_θ to avoid areas where $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ is high, we leverage the universal preconditioner form exhibited in EQ 3.115. Concretely, we reward the agent *less* for *not* navigating areas of $\mathcal{S} \times \mathcal{A}$ that satisfy the constraint $\|\nabla^t_{s,a}[f_\psi]_t\|_F \leq \tau$. The Lipschitz constant τ we want to enforce onto f_ψ is a hyper-parameter that must be tuned, like k for D_φ. We push μ_θ towards areas where $\|\nabla^t_{s,a}[f_\psi]_t\|_F \leq \tau$ (where f_ψ is τ-Lipschitz-continuous, thereby also satisfying the premise of the guarantees) by defining the energy function ε^ψ_t in the *model-based* preconditioner κ^ψ_t as a one-sided gradient penalty, as follows:

$$\tilde{r}^\psi_\varphi(s_t, a_t) \coloneqq \kappa^\psi_t \, r_\varphi(s_t, a_t) \quad \text{where} \quad \begin{cases} \kappa^\psi_t & \coloneqq \max\left(\kappa_{\min}, \exp\left(-\alpha\, \varepsilon^\psi_t\right)\right) \quad \text{with} \\ \varepsilon^\psi_t & \coloneqq \max\left(0, \|\nabla^t_{s,a}[f_\psi]_t\|_F - \tau\right)^2 \Big/ \sigma^\psi_{\text{ON}} \end{cases} \tag{3.116}$$

$$\iff \qquad \tilde{r}^\psi_\varphi(s_t, a_t) \coloneqq \max\left(\kappa_{\min}, \exp\left(-\frac{\alpha}{\sigma^\psi_{\text{ON}}} \max\left(0, \|\nabla^t_{s,a}[f_\psi]_t\|_F - \tau\right)^2\right)\right) r_\varphi(s_t, a_t) \tag{3.117}$$

$\forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, where σ^ψ_{ON} denotes an online, running estimate of the standard deviation of $\max(0, \|\nabla_{s_t, a_t} f_\psi(s_t, a_t)\|_F - \tau)^2$. For completeness, we remind here that we used the same online normalization technique in our RED experiments (*cf.* SECTION 3.5.5), inspired by the discussion laid out in in [BESK18] on the importance of such normalization technique when the reward is grounded on a prediction loss. Considering the edge cases, and omitting here the clipping to $\kappa_{\min}$, when ε^ψ_t is close to zero, κ^ψ_t is approximately equal to 1, *i.e.* $\tilde{r}_\varphi(s_t, a_t) \approx r_\varphi(s_t, a_t)$ (*cf.* EQ 3.116, 3.117). Conversely, in the extreme scenario where ε^ψ_t is very large (*i.e.* $\|\nabla^t_{s,a}[f_\psi]_t\|_F \gg \tau$), κ^ψ_t is approximately equal to 0, and $\tilde{r}_\varphi(s_t, a_t) \approx 0$.

Looking at the model-based instantiation of PURPLE laid out in EQ 3.116, and specifically of the form exhibited in EQ 3.114, we see that the energy ε^ψ_t depends on the current state s_t and action a_t. Indeed, from the definitions of ε^ψ_t and κ^ψ_t, we immediately see that $\mathrm{d}\varepsilon^\psi_{t+k} \Big/ \mathrm{d}s_t \neq 0$ and $\mathrm{d}\varepsilon^\psi_{t+k} \Big/ \mathrm{d}a_t \neq 0$, which direclty leads to $\mathrm{d}\kappa^\psi_{t+k} \Big/ \mathrm{d}s_t \neq 0$ and $\mathrm{d}\kappa^\psi_{t+k} \Big/ \mathrm{d}a_t \neq 0$, $\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T - t] \cap$

$\mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. As such, the crafted preconditioner does not satisfy the eligibily conditions for the derived theoretical guarantees to be applicable, which were represented in condensed form in Section 3.6.5, right before exposing Lemma 3.6.4. If we had used the *supremum* Frobenius norm $\|\nabla^t_{s,a}[f_\psi]_t\|_\infty$ to formulate ε^ψ_t instead of relaxing it to $\|\nabla^t_{s,a}[f_\psi]_t\|_F$, its non-supremum counterpart, ε^ψ_t would *not* depend on s_t and a_t (or any visited state or picked action), and our robustness guarantees would be readily applicable. Still, such a supremum Frobenius norm is intractable in practice. In order for us to be able to evaluate the developed prototype empirically, we resorted to the obvious tractable relaxation consisting in simply dropping the supremum altogether for this diagnostics-oriented case.

Now that we have laid out how the pessimistic model-based preconditioner κ^ψ_t impacts the reward received by the agent artificially upon interaction, we consider how this preconditioning affects the Lipschitz constant of $\widetilde{Q}_\varphi$ in the infinite-horizon setting, denoted by $\widetilde{\Delta}_\infty$ (*cf.* eq 3.114). As $\|\nabla^t_{s,a}[f]_t\|_\infty$ grows larger, its upper-bound A grows larger. Assuming B (upper-bounding $\|\nabla^t_s[\mu]_t\|_\infty$) remains unaffected and remains constant, larger values of A cause larger values of $C \coloneqq A^2 \max(1, B^2)$, which in turn push the denominator of the Lipschitz constant $\widetilde{\Delta}^\psi_\infty \coloneqq \kappa^\psi_t \delta / \sqrt{1 - \gamma^2 C}$ towards 0 from above, exposing $\widetilde{\Delta}^\psi_\infty$ to diverge to $+\infty$. Without preconditioning ($\kappa^\psi_t = 1$), the task of compensating for such a low-valued denominator would be left to δ alone, and picking $\delta \approx 0$ would be the only way to maintain the robustness bound from diverging. With preconditioning however, we can also try to prevent it from diverging with the preconditioner κ^ψ_t, whose value can be set far more finely (*per* timestep). Specifically, with the κ^ψ_t formulation laid out in eq 3.116 and 3.117, and assuming $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ approximates $\|\nabla^t_{s,a}[f]_t\|_\infty$ well — *i.e.* $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ mirrors the behavior of $\|\nabla^t_{s,a}[f]_t\|_\infty$, we hold an analogous line of reasoning for the *numerator* of $\widetilde{\Delta}^\psi_\infty$. As $\|\nabla^t_{s,a}[f]_t\|_\infty$ grows larger, $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ grows larger (with we can translate into $\|\nabla^t_{s,a}[f_\psi]_t\|_F \gg \tau$), which consequently pushes the preconditioner κ^ψ_t towards 0 from above. As such, the premise *"$\|\nabla^t_{s,a}[f]_t\|_\infty$ grows larger"* pushes both the numerator and denominator of $\widetilde{\Delta}^\psi_\infty$ towards 0 from above, taming the quotient in effect. Nonetheless, note, we can not *eliminate* the influence of $\|\nabla^t_{s,a}[f]_t\|_\infty$ on the bound. Still, the *partial compensation* of the detrimental impact of $\|\nabla^t_{s,a}[f]_t\|_\infty$ on $\widetilde{\Delta}^\psi_\infty$ — that we were able to secure by proposing the model-based pessimistic reward preconditioning κ^ψ_t (*cf.* eq 3.116, 3.117) — can be tuned extensively in practice to achieve the desired level of compensation. We used $\kappa_{\min} = 0.7$, $\alpha = 1$, and $\tau \in \{6, 7\}$ in the experiments we conducted to showcase how the proposed model-based reward preconditioning laid out above can help us achieve our robustness desideratum.

Since we aim to *showcase* its potential benefits, as opposed to convince the reader to plug this preconditioning method in every future architecture, we conducted *illustrative* experiments only in the `Hopper` environment (neither the easiest, nor the hardest among the ones considered, *cf.* Table 3.5.1). Note, when it comes to D_φ's gradient penalty regularization, we use the default ζ and λ (*cf.* Section 3.6.3): the directed ζ distribution of WGAN-GP, with $\lambda = 10$ as scaling factor. Since

(a) Return values *(higher is better)* (b) G values *(lower is better)* (c) H values *(lower is better)*

Figure 3.6.5: Empirical evaluation of *(a)* the empirical return, *(b)* the norm of the Jacobian of the forward model f_ψ defined by $G := \|\nabla^t_{s,a}[f_\psi]_t\|_F$, and *(c)* the approximation of $\gamma^2 C$ defined by $H := \gamma^2\|\nabla^t_{s,a}[f_\psi]_t\|^2_F \max(1, \|\nabla^t_s[\mu_\theta]_t\|^2_F)$. *SAM-PURPLE-7* and *SAM-PURPLE-6* are two instantiations of SAM (*cf.* ALGORITHM 3), augmented with the *model-based* instantiation of PURPLE whose template is laid out in EQ 3.116 and 3.117, with $\tau = 7$ and $\tau = 6$ respectively. We indicate how to read the plots (whether *lower* or *higher* is better) in the caption of each column. Despite displaying overlapping return curves, note how *tighter* the standard deviation envelope is for PURPLE runs. Runtime is 96 hours.

the evaluated policy is penalized for navigating areas of $\mathcal{S} \times \mathcal{A}$ where $\|\nabla^t_{s,a}[f_\psi]_t\|_F > \tau$, we monitor $G := \|\nabla^t_{s,a}[f_\psi]_t\|_F$. We expect to observe *lower* values of G when using the studied preconditioning. In order to grasp the extent to which variations can compound in the system, and therefore highlight the need for mechanims allowing the main method to contain such compounding of variations (like the proposed one), we also monitor an approximation of $\gamma^2 C$, relaxed as $H := \gamma^2\|\nabla^t_{s,a}[f_\psi]_t\|^2_F \max(1, \|\nabla^t_s[\mu_\theta]_t\|^2_F)$. We expect to see the same ranking of methods in the plots depicting G and H respectively. These are all reported in FIGURE 3.6.5.

Note, the steep surge in overall computational cost caused by the evaluation of the monitored metrics (G and H) and expecially κ^ψ_t lowered the number of iterations our agent could do in the allowed runtime. As such, we increased said runtime from the usual 0.5-day or 2-day duration to a 4-day duration (or 96 hours) Such runs are more costly to orchestrate, hence the sparser array of experiments to offset the steeper cost in compute. In FIGURE 3.6.5, we observe that, at evaluation time, the model-based PURPLE instantiation in EQ 3.116 and 3.117 indeed enables the agent to achieve *lower* values of G and H, with the *same* episodic return. Said differently, it seems that the agent — with preconditioning, compared to the one without — achieves the same proficiency, with the same convergence speed, while making decisions that are *safer* in terms of incurred variations of the approximate dynamics f_ψ. ***So, even if the preconditioner is not needed to reach a higher return (or reach it faster) per se, we have showcased that the studied model-based reward preconditioning can increase the robustness of the main method by augmenting it with the means to tame a priori untamable entities in the system (here, the dynamics).*** Still, the studied model-based instantiation of PURPLE is set back by several drawbacks. *a)* We need to maintain a forward model

f_ψ that approximates the effective transition function f'. *b)* To be estimated, κ_t^ψ requires explicit calls to an automatic differentiation library, making its frequent computation (every time a mini-batch is sampled from the replay buffer) extremely expensive overall. *c)* The threshold τ (to be enforced as Lipschitz constant for f_ψ) must be set such that *not every* decision made by the agent is penalized, while making sure it is still strict enough in that respect. Besides, we observed in practice that the range of values taken by $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ varies greatly across environments. As such, τ must be tuned carefully per environment, making the overall process tedious and computationally expensive. In effect, this brings us back to the original issues of reward shaping [NHR99], that adversarial IL [HE16] circumvented.

3.6.6.4 Total compensation of compounding variations

Inspired by the insight laid out in Remark 1, we derive theoretical guarantees that characterize the robustness of $\widetilde{Q}_\varphi$ when using a preconditioner defined as follows:

$$\kappa_{t+k} \coloneqq \frac{1}{\sqrt{\prod_{u=0}^{k-1} C_{t+u}}} \qquad \text{where (\textit{cf.} eq 3.87) } \forall v \in [0, T-1],$$
$$C_v \coloneqq \|\nabla^v_{s,a}[f]_v\|^2_\infty \max\left(1, \|\nabla^{v+1}_s[\mu]_{v+1}\|^2_\infty\right) \tag{3.118}$$

$\forall t \in [0, T] \cap \mathbb{N}$, and $\forall k \in [0, T-t] \cap \mathbb{N}$. Since the norms involved in C_v are *supremum* ones, the preconditioner κ_t verifies $\mathrm{d}\kappa_{t+k}/\mathrm{d}s_t = 0$ and $\mathrm{d}\kappa_{t+k}/\mathrm{d}a_t = 0, \forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. The reward preconditioner therefore verifies the properties one must satisfy for the derived robustness guarantees to be applicable (*cf.* Section 3.6.5). Again, note, the property $\kappa_t \leq 1$ is only ever used in Section 3.6.6.1, and has not been leveraged anywhere else. Given that the developed theory still holds if $\exists t \in [0, T] \cap \mathbb{N}$ such that $\kappa_t > 1$, the fact that the preconditioner defined in eq 3.118 does not necessarily lie in the $(0, 1]$ interval is not an issue *a priori*. Still, in practice, it will virtually always be below 1.

We now derive the associated counterparts of Theorem 3.6.6 and Corollary 3.6.6.1.

Theorem 3.6.7 (state-action value Lipschitzness)**.** *We work under the assumptions laid out in both Lemma 3.6.4 and Theorem 3.6.5, and repeat the main lines here for Theorem 3.6.7 to be self-contained: a) The functions f, μ and r_φ are C^0 and differentiable over their respective input spaces, and b) the function r_φ is δ-Lipschitz over $\mathcal{S} \times \mathcal{A}$, i.e. $\|\nabla^u_{s,a}[r_\varphi]_u\|_F \leq \delta$, where $u \in [0, T] \cap \mathbb{N}$. Then the quantity $\nabla^u_{s,a}[\widetilde{Q}_\varphi]_u$ exists $\forall u \in [0, T] \cap \mathbb{N}$. Assuming in addition that the reward preconditioner*

used on r_φ to obtain $\tilde{r}_\varphi$ is defined according to EQ 3.118*, the action-value $\widetilde{Q}_\varphi$ verifies:*

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \delta \sqrt{\frac{1-\gamma^{2(T-t)}}{1-\gamma^2}} \tag{3.119}$$

$\forall t \in [0, T] \cap \mathbb{N}$. *Note, the bound now only depends on δ, γ, and $T - t$, the "remaining time in the episode".*

Proof of THEOREM 3.6.7. The reward preconditioner used to assemble $\tilde{r}_\varphi$ from r_φ is defined according to EQ 3.118. As carried out in REMARK 1, we start the proof of THEOREM 3.6.7 analogously to the one laid out for THEOREM 3.6.6, but using the time-*dependent* version of THEOREM 3.6.5 instead of the time-*independent* version that we used in EQ 3.107 (version 3.6.5 *(a)* instead of 3.6.5 *(b)*). Our starting point then aligns with the crux of REMARK 1. As such, we have:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|^2_F \leq \sum_{k=0}^{T-t-1}\left[\gamma^{2k}\,\kappa^2_{t+k}\,\delta^2\prod_{u=0}^{k-1}C_{t+u}\right] \qquad \blacktriangleright \textit{ THEOREM 3.6.5 (a)} \tag{3.120}$$

$$= \sum_{k=0}^{T-t-1}\left[\gamma^{2k}\,\frac{1}{\prod_{u=0}^{k-1}C_{t+u}}\,\delta^2\prod_{u=0}^{k-1}C_{t+u}\right] \qquad \blacktriangleright \textit{ EQ 3.118} \tag{3.121}$$

$$= \delta^2\sum_{k=0}^{T-t-1}(\gamma^2)^k \tag{3.122}$$

Since we defined γ to be within the interval $[0, 1)$ in SECTION 3.3, we trivially have $\gamma^2 < 1$, hence $\gamma^2 \neq 1$ and:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|^2_F \leq \delta^2\,\frac{1-\gamma^{2(T-t)}}{1-\gamma^2} \qquad \blacktriangleright \textit{ finite sum of geometric series} \tag{3.123}$$

By applying $\sqrt{\cdot}$ (monotonically increasing) to the inequality, we obtain the claimed result. □

Finally, we derive a corollary from THEOREM 3.6.7 corresponding to the infinite-horizon regime.

Corollary 3.6.7.1 (infinite-horizon regime)**.** *Under the assumptions of* THEOREM 3.6.7*, including that r_φ is δ-Lipschitz and that $\tilde{r}_\varphi$ is defined as in* EQ 3.69 *over $\mathcal{S} \times \mathcal{A}$, we have, in the infinite-horizon regime:*

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \frac{\delta}{\sqrt{1-\gamma^2}} \tag{3.124}$$

which translates into $\widetilde{Q}_\varphi$ being $\frac{\delta}{\sqrt{1-\gamma^2}}$-Lipschitz over $\mathcal{S} \times \mathcal{A}$.

Proof of Corollary 3.6.7.1. As we adapt the proof of Theorem 3.6.7 to the infinite-horizon regime, eq. 3.122 becomes

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|^2_F \leq \delta^2 \sum_{k=0}^{+\infty}(\gamma^2)^k = \frac{\delta^2}{1-\gamma^2} \qquad \blacktriangleright \textit{ infinite sum of geometric series} \tag{3.125}$$

since we defined γ to be within the interval $[0, 1)$ in Section 3.3, *i.e.* $\gamma^2 < 1$. We then apply $\sqrt{\cdot}$ to the inequality. □

In these theoretical guarantees, we have shown that by carefully crafting PURPLE's reward preconditioner according to eq. 3.118, we obtain upper-bounds $\widehat{\Delta}_\infty$ on the Lipschitz constant of the resulting action-value $\widetilde{Q}_\varphi$ that are *independent* of $C_v, \forall v \in [0, T-1]$ — where C_v is defined as in eq. 3.118, as $C_v \coloneqq \|\nabla^v_{s,a}[f]_v\|^2_\infty \max\left(1, \|\nabla^{v+1}_s[\mu]_{v+1}\|^2_\infty\right)$. In other words, we have shown that such preconditioner design allows us to *totally* compensate for the compounding variations *a)* first tackled in the discussion led in Section 3.6.2.3, and *b)* then addressed *only partially* by the model-based reward preconditioning discussed profusely in Section 3.6.6.3 (of which we showcase the applicability in practice). Echoing what motivated the emergence of Remark 1 in the first place, the form adopted by the reward preconditioning (*cf.* eq. 3.118) that allowed us to derive the robustness guarantees of Theorem 3.6.7 and Corollary 3.6.7.1 enjoys an insightful and intuitive *interpretation*. Going through the elements of the series described by the preconditioner of eq. 3.118, $(\kappa_{t+k})_k, \forall t \in [0, T] \cap \mathbb{N}$, and $\forall k \in [0, T-t] \cap \mathbb{N}$, we have the following sequence of consecutive preconditioning values:

$$\begin{aligned}\kappa_{t+k}\big|_{k=0} = \kappa_t \coloneqq 1 \;\rightarrow\; \kappa_{t+k}\big|_{k=1} = \kappa_{t+1} \coloneqq \frac{1}{\sqrt{C_t}} \;\rightarrow\; \kappa_{t+k}\big|_{k=2} = \kappa_{t+2} \coloneqq \frac{1}{\sqrt{C_t C_{t+1}}} \\ \rightarrow\; \kappa_{t+k}\big|_{k=3} = \kappa_{t+3} \coloneqq \frac{1}{\sqrt{C_t C_{t+1} C_{t+2}}} \;\rightarrow\; \dots \;\rightarrow\; \kappa_{t+k}\big|_{k=T-t} = \kappa_T \coloneqq \frac{1}{\sqrt{C_t C_{t+1} C_{t+2} \dots C_{T-1}}}\end{aligned} \tag{3.126}$$

We observe that, when purposely defined as such, the reward preconditioner κ_{t+k} at a given stage $t+k$ compensates for the C_v's of all the *previous* timesteps — backwards from $t+k-1$ to t, where $\widetilde{Q}_\varphi$'s Lipschitz constant is characterized. In order to prevent the upper-bound on $\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F$ to be burdened by incipient, potentially prone to compound, variation of $C_v \coloneqq \|\nabla^v_{s,a}[f]_v\|^2_\infty \max\left(1, \|\nabla^{v+1}_s[\mu]_{v+1}\|^2_\infty\right)$, the preconditioner can *actively anticipate* said incipient compounding variations to compound further within the time remaining in the episode by preemptively squashing the *current* surrogate reward at $t+k$ based on how much C_v's variations have accumulated since t until $t+k-1$. The proposed interpretation of the studied preconditioner aligns with our intuitive desideratum: *"if you want to fend off from compounding of variations that threaten the stability of your action-value, make the latter more robust as soon as you see, from past metrics — here, monitored C_v values — that said variations*

might actually compound soon".

Despite appealing in principle thanks to its salient interpretation, and justified by theoretical guarantees, we did not experiment with the proposed preconditioner in practice. Indeed, considering how we have shown in SECTION 3.6.6.3 that the values in effect taken by C_v (*cf.* EQ. 3.118) do not seem to affect the agent's return in practice, we do not expect the interpretable preconditioner tackled in this discussion to bring anything *practically* in the considered environments. Using a gradient penalty constraint to induce local Lipschitz-continuity of the function at the core of the reward function is, in a sense, *all you need* to achieve peak expert performance in the considered off-policy generative adversarial imitation learning setting. Still, we believe the design and study of methods able to actively tune their level of robustness — aligned in this work with the concept of spatial, local Lipschitz-continuity — depending on the choices (or more pessimistically, on the *mistakes*) made by the agent to be an interesting avenue of future work. Besides, by augmenting the reward-less MDP $\mathbb{M}$ (from which we first stripped the environmental reward) with our adversarially learned reward, preconditioned in line with EQ. 3.118, the resulting MDP has a *memory*, since the reward $\tilde{r}_\varphi$ depends on entities (C_v's) from previous timesteps in the episode. In effect, due to such a reward preconditioning formulation, the Markov property is not satisfied anymore as, given the present, the future now *does* depend on the past. We believe the observations made and results derived in this work could pave the way to further investigations aiming to decipher known methods and ultimately pinpoint the *most minimal* setup for it to still do well.

3.7 Conclusion

In this work, we conducted an in-depth study of the stability problems incurred by off-policy generative adversarial imitation learning. Our contributions closely follow the line of reasoning, and are as follows. ***(1)*** We characterized the various inherent hindrances the approach suffers from, in particular how learned parametric rewards affect the learned parametric state-action value. ***(2)*** We showed that enforcing a local Lipschitz-continuity constraint on the discriminator network used to formulate the imitation surrogate reward is a *sine qua non* condition for the approach to empirically achieve expert performance in challenging continuous control problems, within a number of timesteps that still enable us to call the method sample-efficient. ***(3)*** In line with the first and second steps, we derived theoretical guarantees that characterize the Lipschitzness of the Q-function when the reward is assumed δ-Lipschitz-continuous. ***Note, the reported theoretical results are valid for any reward satisfying the condition, nothing is specific to imitation.*** ***(4)*** We propose a new RL-grounded interpretation of the usual GAN gradient penalty regularizers — differing by *where* they induce Lipschitzness — along with an explanation as to *(a)* why they all have such a positive impact on stability,

but also *(b)* how to make sense of the empirical gap between them. ***(5)*** We show that, in effect, the consistent satisfaction of the Lipschitzness constraint on the reward in a strong predictor of how well the mimicking agent performs empirically. ***(6)*** Finally, we introduce a pessimistic reward preconditioning add-on technique which *(a)* makes the base method it is plugged into provably more robust, and *(b)* is accordingly backed by several theoretical guarantees. ***As in (3), these guarantees are not not specific to imitation and have a wide range of applicability.*** We give an illustrative example of how the add-on technique can help further increasing the robustness of the method it is plugged into empirically.

Appendix

3.A Hyper-parameters

The function approximators used in every learned module are two-layer multi-layer perceptrons, but the widths of their respective layers differ. We use layers of sizes 100-100 for the discriminator (from which the reward is formulated), 300-200 for the actor, and 400-300 for the critic, as they achieved the best overall result across the environments of the suite in our early experiments. Unless specifically stated otherwise, the discriminator network uses spectral normalization [MKKY18] at every layer, while the actor and critic networks both use layer normalization [BKH16] at every layer. Every neural network is initialized via orthogonal initialization [SMG13]. Each network has its own optimizer (*cf.* Section 3.4 for a complete description of the optimization problems the networks of parameter φ, ω, and θ are involved in, along with the loss they optimize). We use Adam [KB14] for each of them, with respective learning rates reported in Table 3.A.1, while the other parameters of the optimizer are left to the default PyTorch [PGM+19] values. In practice, we replace the squared error loss involved in the loss optimized by the critic (*cf.* eq. 3.4) by the Huber loss, as is commonly done in temporal-difference learning with function approximation and target networks [MKS+13, MKS+15]. As for the activations functions used in the neural networks, we used ReLU non-linearities in both the actor and critic, and used Leaky-ReLU [MHN13] non-linearities with a leak of 0.1 in the discriminator. We used an *online* version of batch normalization (described earlier in Section 3.5.5) to standardize the actor and critic observations before they are fed to them. We do not use any learning rate scheduler, for any module.

Hyper-parameter	Selected Value
Training steps per iteration	2
Evaluation steps per iteration	10
Evaluation frequency	10
Actor learning rate	2.5×10^{-4}
Critic learning rate	2.5×10^{-4}
Actor clip norm	40
Critic weight decay scale	0
Rollout length	2
Effective batch size	1024
Discount factor γ	0.99
Replay buffer $\mathcal{R}$ size	100000
Exploration (*cf.* Section 3.4)	$\sigma_a = 0.2, \sigma_b = 0.2$
Param. noise update frequency	50
Target update Polyak scale τ	0.005
Multi-step lookahead n	10
Target smoothing - noise σ [FvHM18]	0.2
Target smoothing - noise clip [FvHM18]	0.5
Actor update delay [FvHM18]	2
Reward training steps per iteration	1
Agent training steps per iteration	1
Discriminator learning rate	5.0×10^{-4}
Entropy regularization scale	0.001
Positive label-smoothing	Real labels $\sim$ unif$(0.7, 1.2)$
Positive-Unlabeled [XD19] - coeff. η	0.25

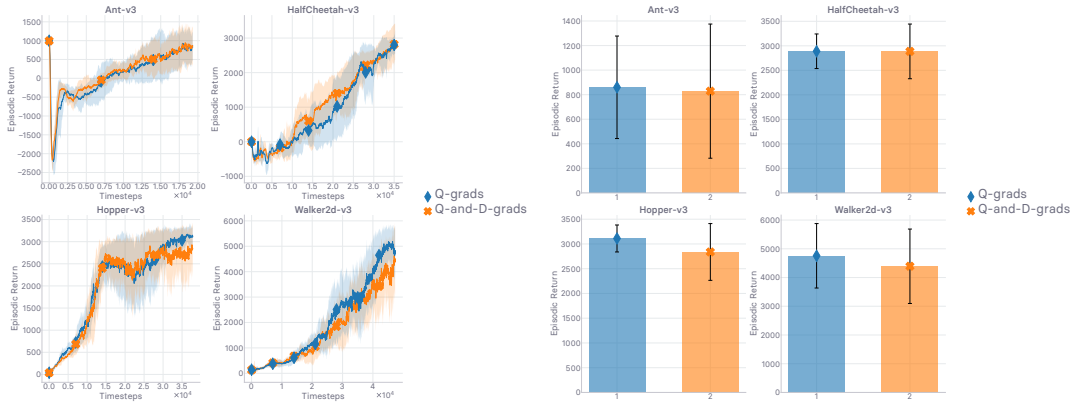
Table 3.A.1: Hyper-parameters used in this work. Unless explicitly stated otherwise, every method uses these. The *"effective"* batch size corresponds to the size of the mini-batch *aggregated* across parallel workers of the distributed architecture. In our case, every worker — of the grand total of $n = 16$ workers — samples a mini-batch of size 64 from its (individual) replay buffer, resulting in an effective batch size of $64 \times 16 = 1024$.

3.B Sequential Decision Making Under Uncertainty In Non-Stationary Markov Decision Processes

In Section 3.3, we have defined $\mathbb{M}$ as a stationary MDP, in line with a vast majority of works in RL. Note, a stochastic process or a distribution is commonly said *stationary* if it remains unchanged when shifted in time. While the stationarity assumption allows for the derivation of various theoretical guarantees and is overall easier to deal with analytically, it fails to explain the inner workings of complex realistic simulations, and *a fortiori* the real world. One critical challenge incurred when modeling the world as a non-stationarity MDP is the unavailability of convergence guarantees for standard practical RL methods. Crucially, assuming stationarity in the dynamics p is necessary for the Markov property to hold, which is required for the convergence of Q-learning [Wat89] algorithms [AK16a] like DQN [MKS+13, MKS+15]. As such, designing methods yielding agents that are robust against the non-stationarities naturally occurring in their realistic environments is a challenging yet timely milestone. Methods equipping models against unforeseen changes in the data distribution, a phenomenon qualified as *concept drift* [SG86], are surveyed in [GŽB+14] who dedicate the study to the supervised case. In RL, a analysis of non-stationarity issues inherent to the Q-learning loss optimization under function approximation [SMSM99] proposes qualitative and quantitative diagnostics along with a new replay sampling method to alleviate the isolated weaknesses [FKSL19]. Non-stationarities are characterized by how they manifest in time. A distribution is *switching* if abrupt changes, called *change points*, occur while remaining stationarity in-between, making it in effect piece-wise stationary [DSBBE06, JOA10, GM11, AK16a, GOA18, PPB19, AGO19]. The change points are either given by an oracle or discovered via change point detection techniques. Once exhibited, one can employ stationary methods individually on each segment. A distribution is *drifting* if it gradually changes at an unknown rate [BGZ14, AK16b, LWAL18, OGA19, CLLW19, CSLZ19a, CSLZ19b, RVC19]. The change can occur continually or as a slow transition between stationary plateaus, making it considerably more difficult to deal with, theoretically and empirically. In a non-stationary MDP, the non-stationarities can manifest in the dynamics p [NEG05, DSBBE06, XM07, LXM13, AK16a], in the reward process r [EdKM05, DGS14], or in both conjointly [YM09a, YM09b, AYBS13, GOA18, PPB19, YS19, LR19]. The adversarial bilevel optimization problem — guiding the adaptive tuning of the reward for every policy update — present in this work is reminiscent of the stream of research pioneered by [ACBFS95] in which the reward is generated by an omniscient *adversary*, either arbitrarily or adaptively with potentially malevolent drive [YM09a, YM09b, LXM13, GOA18, YS19]. Non-stationary environments are almost exclusively tackled from a theoretical perspective in the literature (*cf.* previous references in this section). Specifically, in the *drifting* case, the non-stationarities are traditionally dealt with via the use of sliding windows. The accompanying (dynamic) regret anal-

yses all rely on strict assumptions. In the switching case, one needs to know the number of occurring switches beforehand, while in the drifting case, the change variation need be upper-bounded. Specifically, [BGZ14, CSLZ19a] assume the total change to be upper-bounded by some preset variation budget, while [CSLZ19b] assumes the variations are uniformly bounded in time. [OGA19] assumes that the *incremental* variation (as opposed to *total* in [BGZ14, CSLZ19a]) is upper-bounded by a *per-change* threshold. Finally, in the same vein, [LR19] posits *regular evolution*, by making the assumption that both the transition and reward functions are Lipschitz-continuous *w.r.t.* time.

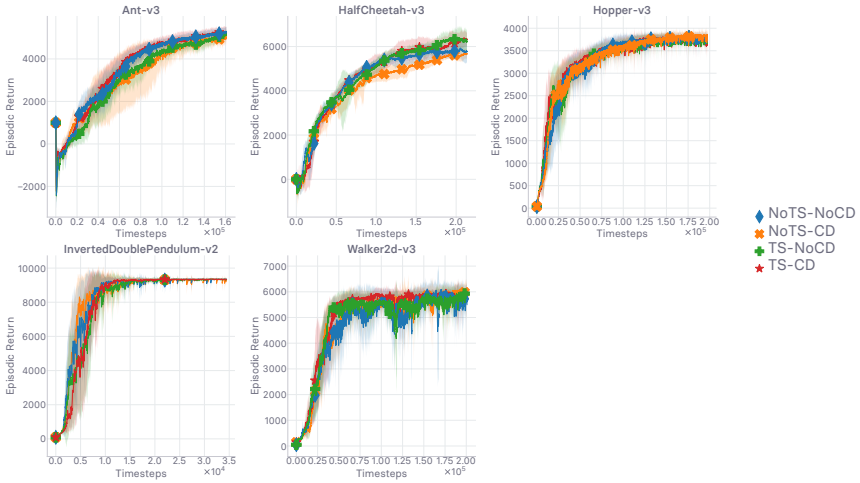
3.C Adaptive Policy Update based on Gradient Similarities



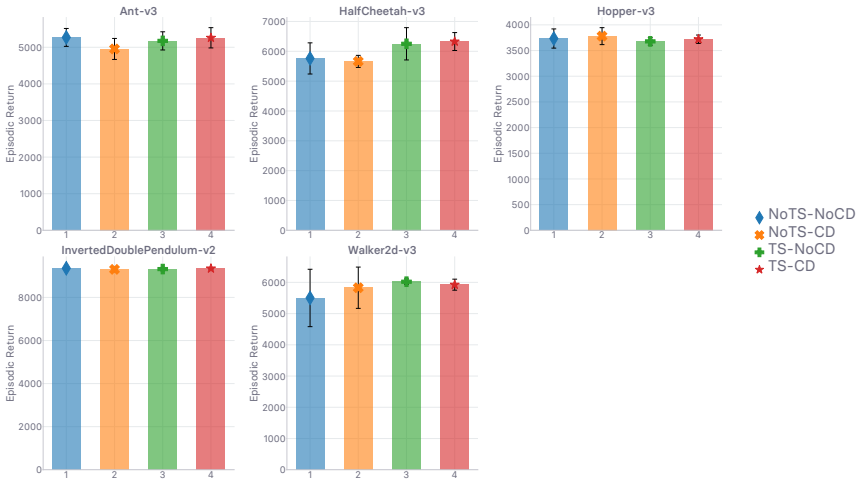
(a) Evolution of return values *(higher is better)* (b) Final return values at timeout *(higher is better)*

Figure 3.C.1: Comparison of the gradient used to update the policy in this work, involving the gradient of the state-action value, against an adaptive hybrid method involving *also* the gradient of the discriminator, and combining both gradients based on their cosine similarity. Runtime is 12 hours.

3.D Clipped Double-Q Learning and Target Policy Smoothing



(a) Evolution of return values *(higher is better)*

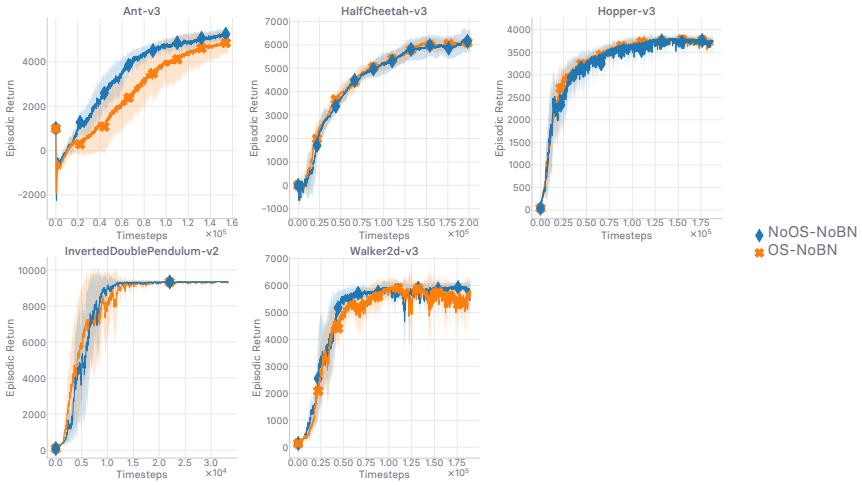


(b) Final return values at timeout *(higher is better)*

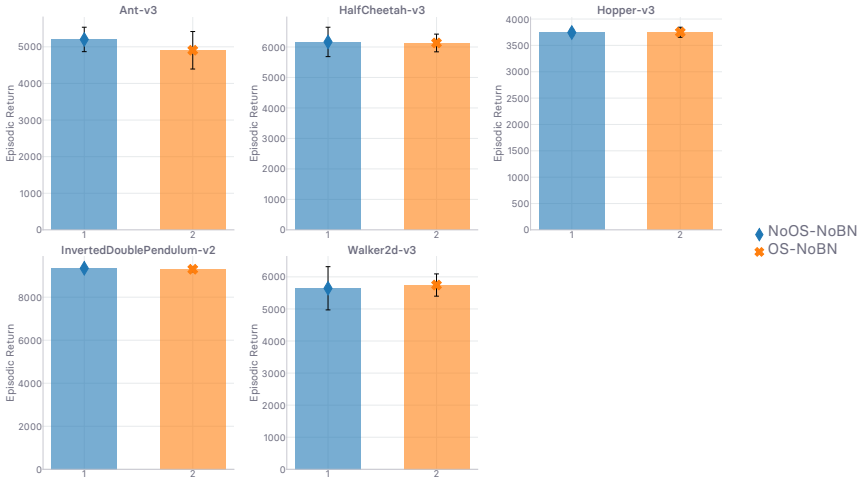
Figure 3.D.1: Ablation study on the use of the clipped double Q-Learning (CD) and target smoothing (TS) techniques, both from [FvHM18], *with* gradient penalty regularization [GAA+17]. Runtime is 48 hours

3.E Gradient Penalty

3.E.1 One-sided Gradient Penalty



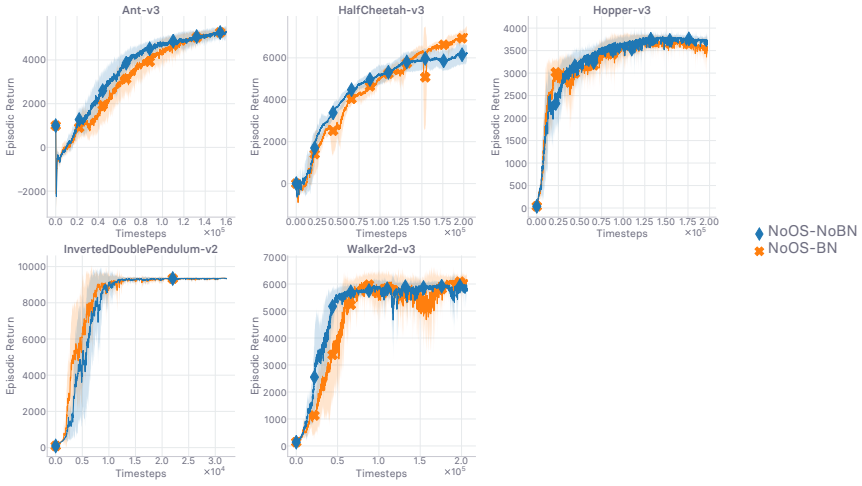
(a) Evolution of return values *(higher is better)*



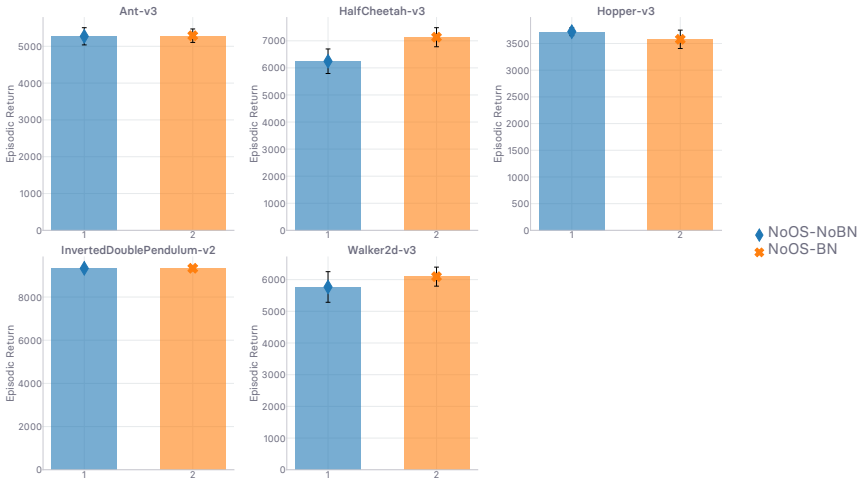
(b) Final return values at timeout *(higher is better)*

Figure 3.E.1: Ablation study on the use of the one-sided (OS) penalty variant [GAA+17]. Runtime is 48 hours

3.E.2 Online Batch Normalization in Discriminator



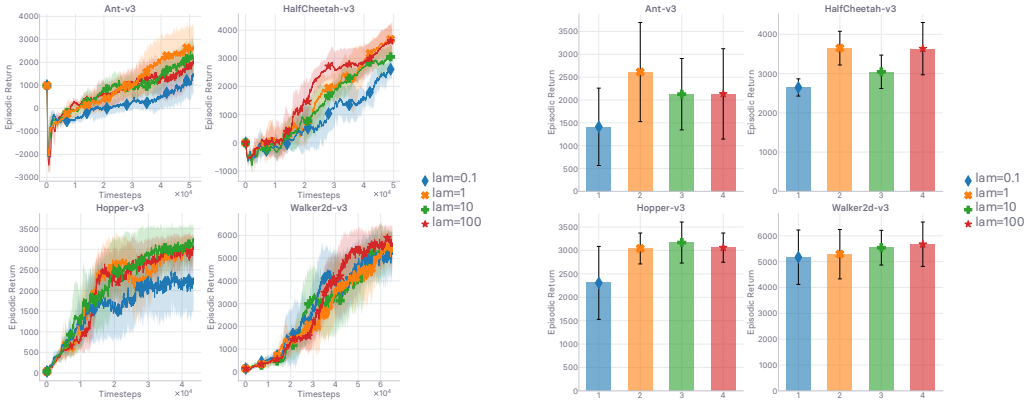
(a) Evolution of return values *(higher is better)*



(b) Final return values at timeout *(higher is better)*

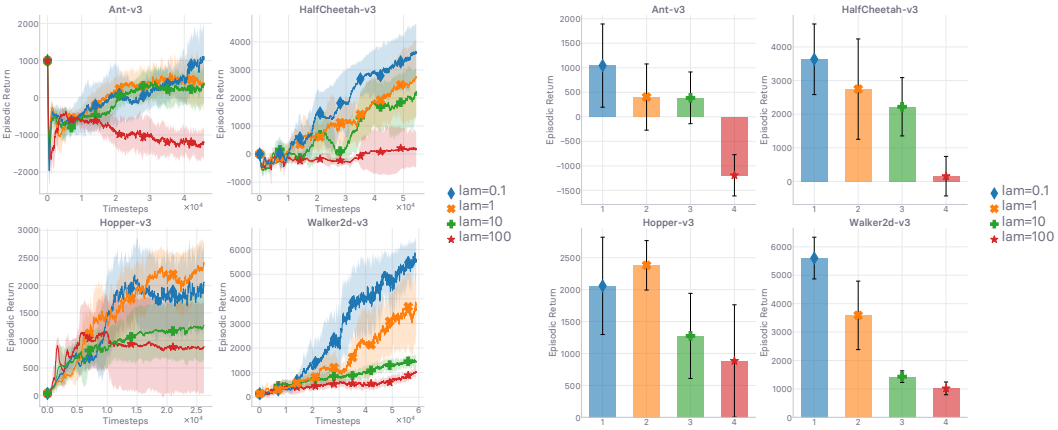
Figure 3.E.2: Ablation study on the use of online batch normalization (BN) in the discriminator for its impact on the gradient penalization [GAA+17]. Runtime is 48 hours

3.E.3 Target k and Coefficient λ Grid Search



(a) Evolution of return values *(higher is better)* (b) Final return values at timeout *(higher is better)*

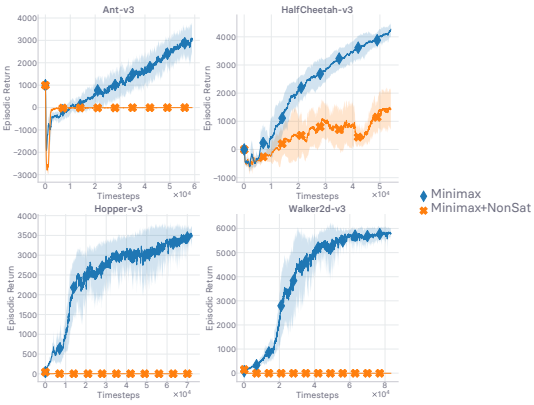
Figure 3.E.3: Grid search over the hyper-parameter λ when $k = 1$. Runtime is 12 hours.



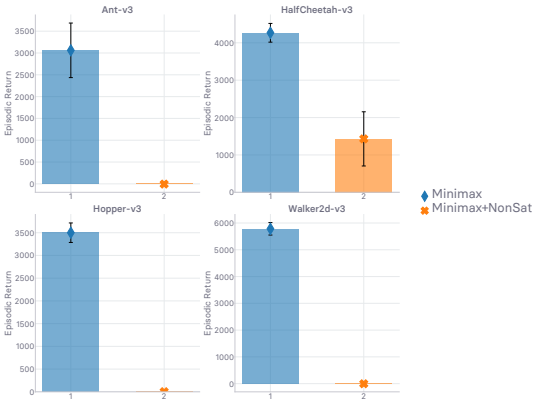
(a) Evolution of return values *(higher is better)* (b) Final return values at timeout *(higher is better)*

Figure 3.E.4: Grid search over the hyper-parameter λ when $k = 0$. Runtime is 12 hours.

3.F Reward Formulation



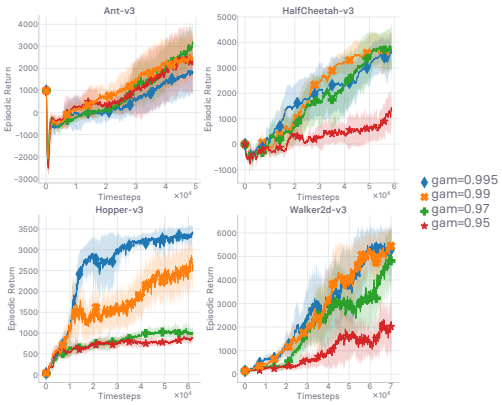
(a) Evolution of return values *(higher is better)*



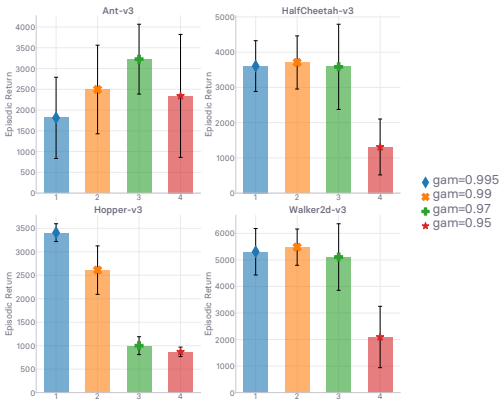
(b) Final return values at timeout *(higher is better)*

Figure 3.F.1: Comparison of two ways to define the surrogate imitation reward r_φ from the discriminator D_φ. *"Minimax"* corresponds to $r_\varphi^{\text{MM}} := -\log(1 - D_\varphi)$, while *"Minimax + Non-Saturating"* denotes the use of $r_\varphi^{\text{NS}} := -\log(1 - D_\varphi) + \log(D_\varphi)$, as described in Section 3.4. Runtime is 12 hours.

3.G Discount Factor



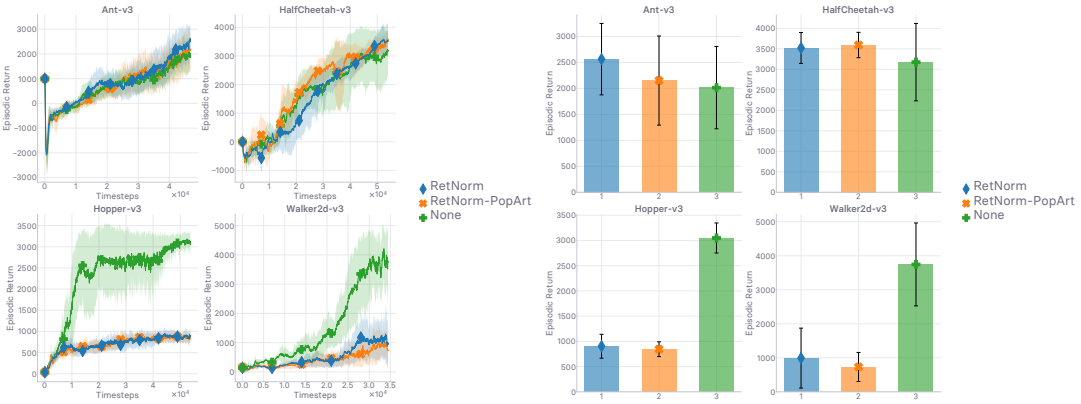
(a) Evolution of return values *(higher is better)*



(b) Final return values at timeout *(higher is better)*

Figure 3.G.1: Grid search over the discount factor γ. Runtime is 12 hours.

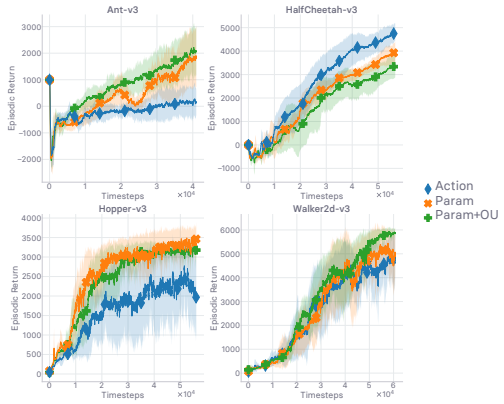
3.H Return Normalization



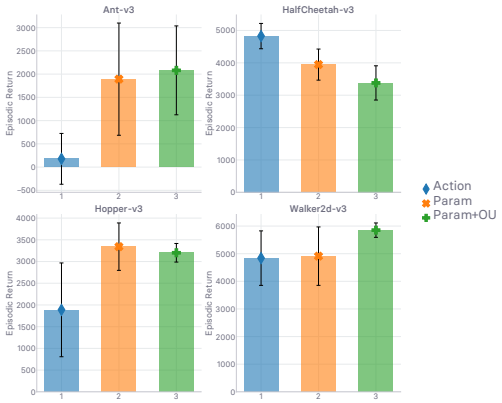
(a) Evolution of return values *(higher is better)* (b) Final return values at timeout *(higher is better)*

Figure 3.H.1: Ablation study on return normalization and Pop-Art [vHGH+16]. Runtime is 12 hours.

3.I Exploration



(a) Evolution of return values *(higher is better)*



(b) Final return values at timeout *(higher is better)*

Figure 3.I.1: Evaluation of the considered method under several exploration strategies. *"Action"* corresponds to defining π_θ by directly applying additive Gaussian noise to the *action* returned by μ_θ. As such, $\pi_\theta(\cdot, s_t) = \mu_\theta(s_t) + $, where $\sim \mathcal{N}(0, \sigma)$, with $\sigma = 0.2$. *"Param"* denotes the application of additive noise in the network *parameters* directly, and *"Param + OU"* corresponds to the additional application of temporally correlated noise, generated sequentially by a Ornstein-Uhlenbeck process, on the action (*cf.* Section 3.4 for a description of these two last approaches, and Table 3.A.1 for the associated hyper-parameters). Despite the absence of a clear winner, we use the combination of parameter noise and temporally correlated action noise in every experiment reported in this work, as it seems to yield the best results. Runtime is 12 hours.

Chapter 4

Where is the Grass Greener? Revisiting Generalized Policy Iteration for Offline Reinforcement Learning

Abstract

The performance of state-of-the-art baselines in the offline RL regime varies widely over the spectrum of dataset qualities — ranging from *"far-from-optimal"* random data to *"close-to-optimal"* expert demonstrations. We re-implement these under a fair, unified, and highly factorized framework, and show that when a given baseline outperforms its competing counterparts on one end of the spectrum, it *never* does on the other end. This consistent trend prevents us from naming a victor that outperforms the rest across the board. We attribute the asymmetry in performance between the two ends of the quality spectrum to the amount of inductive bias injected into the agent to entice it to posit that the behavior underlying the offline dataset is *optimal* for the task. The more bias is injected, the higher the agent performs, *provided the dataset is close-to-optimal*. Otherwise, its effect is brutally detrimental. Adopting an advantage-weighted regression template as base, we conduct an investigation which corroborates that injections of such *optimality* inductive bias, when not done parsimoniously, makes the agent subpar in the datasets it was dominant as soon as the offline policy is sub-optimal. In an effort to design methods that perform well across the whole spectrum, we revisit the generalized policy iteration scheme for the offline regime, and study the impact of nine distinct newly-introduced proposal distributions over actions, involved in proposed generalization of the policy evaluation and policy improvement update rules. We show that certain orchestrations strike the right balance and can improve the performance on one end of the spectrum *without* harming it on the other end.

4.1 Introduction

Reinforcement learning (RL) [SB98] is the branch of interactive machine learning that has received the most attention in recent years, due to its instrumental role in tackling a number of grand AI challenges (*e.g.* going beyond human performance in board and video games [MKS+13, MKS+15, SHS+17, SHM+16, VBC+19, OBB+19], hitting a new milestone in AI-operated hand dexterity leading to the resolution of a Rubik's cube [Ope19]). The *online* RL agents learn by acting in the world (either real or simulated), and update their intrinsic decision-making process by internalizing the feedback returned by the world upon interaction. The feedback takes the form of a reward signal, which scores the agent according to how appropriate its *own* executed actions were for the task at hand (how rewards are designed or come from is out of the scope of this paper, *cf.* [SLB09]). Nonetheless, while learning from our mistakes is undeniably valuable, it is of far greater value to be able to learn from the mistakes of others, a sub-branch of machine learning sometimes referred to as *counterfactual* learning [BPQC+13]. In the context of RL, *pure* counterfactual learning crystallizes as *offline* RL [LKTF20] (alternatively, *batch* RL [LGR12]). The agent is only allowed to learn from the feedback stored transitions from an offline dataset, collected by another policy, before the agent starts training. The agent is not allowed to interact with the world, and is consequently unable to learn from her *own* mistakes.

Offline RL shines *a)* when interaction data — albeit from multiple non-egocentric sources — are abundant and diverse (*e.g.* Waymo's self-driving open dataset [SKD+19]), and *b)* when simulator-in-the-loop approaches (*e.g.* [ABW+19, ZLV+20]) are impractical, due to deterringly high costs or safety concerns. In practice, it is incredibly tedious and challenging to design and implement a data collection strategy able to capture the diversity of the real world in a dataset. Besides, crafting a simulator able to model how the world reacts to the agent with high fidelity is an engineering feat too (*e.g.* the elaborate system of automatic domain randomization used in [Ope19] to assist in the resolution of of a Rubik's cube, which required the aid of the purposely-developed asset randomization engine reported in [CWW19]). Iteratively integrating hitherto-omitted edge cases is a long-term endeavor. Even when one manages to train agents yielding high return in simulation, they are not guaranteed to transfer to the real world, and often require additional model surgery (*cf. "sim-to-real"* research [HMC+21, HRX+20, SDZ+19, JWK+18, OAB+18, TZC+18, BIW+17, TFR+17, RVR+16], a subdomain of transfer RL [TS09, PBS16, BMSS16, GDL+17, BXCVdP18, WLBF18, NPH+18, GG19, PCZ+20]). As such, designing offline RL methods able to squeeze the most juice out of potentially imperfect offline real-world interaction data is both a crucial and timely problem to solve. The crying need for such methods in popular consumer-facing real-world applications (*e.g.* robotics [CCN+19, MRB+19], autonomous driving [CSKX15, GDL16, BDTD+16, SAPY17, MPLN17, CML+18, MSS18, RML18, SPZT18, KHJ+19, SKD+19, HXS+20, GSZ+20, YLCT20, GLDS20, GKM+20], healthcare

[JPS+16, GJM+18, GJK+19, FHDV20], conversational agents (*e.g.* dialog bots) [HLG05, PGCFB11, ZSRE17, GPL+20, JSG+20]) somewhat explains the resurgence of offline RL research in recent years [LKTF20].

Online off-policy actor-critic methods require the decision-maker to learn an estimate of the state-action value through bootstrapping with next actions generated by the agent, and as such implement the *SARSA* update rule [RN94, Thr95, Sut96, vSvHWW09]. These methods suffer from a *distributional shift* [FKSL19] as soon as the experience replay [Lin93] mechanism comes into play, since an action generated by the current policy is used to update the value at an action randomly picked from the replay memory — effectively distributed as a mixture of past policy updates. This covariate shift phenomenon is exacerbated in the *offline* setting, where the value is *only* subject to updates at the fixed set of actions present on the offline dataset. Since the distribution over actions underlying the offline dataset neither evolves nor need be tied to the agent's policy in any way, the distributional shift can grow arbitrarily large. By contrast, the mixture of past policy updates underlying a replay memory will by design always track (yet not necessarily closely) the current policy update, effectively limiting the gap between the action distributions. Since in the offline setting the value is solely updated over a *frozen* offline dataset detached from the policy followed by the agent, the value is all the more so exposed to being evaluated at *out-of-distribution* actions than its counterpart in the online (off-policy) setting. As such, the value can be arbitrarily erroneous at actions that are too far off the offline distribution underlying the dataset. This motivates the design of offline RL methods that deter the agent from straying onto out-of-distribution action by urging the agent to *"stay close to the offline dataset"* (*cf.* Figure 4.1.1), either explicitly or implicitly. We lay these out in Section 4.2.

We first propose fair re-implementations of state-of-the-art offline RL algorithms under a unified open-sourced framework, in Section 4.4.1. We then compare these competing methods empirically in Section 4.4.3. This study, along with all the other analyses reported in this work, are conducted under the experimental setting laid out in Section 4.4.2. In Section 4.4.4, we explicitly formalize the notion of *"optimality inductive bias"*. When injected into the decision-maker, it conceptually quantifies how much the latter posits that the policy underlying the offline dataset is optimally solving the task at hand. We investigate empirically to what extent the amount of inductive bias injected in the agent impacts its performance across the spectrum of considered datasets and environments. The performance comparison depicted in Section 4.4.3 shows how most baselines inject far too much optimality inductive bias to achieve reasonable levels of performance when the dataset is sub-optimal. Besides, these baselines have proven hard to tune [WTN19, LPPM+20, MKL+20], and the hyper-parameters displaying the largest swings in performance are the ones controlling the amount of bias injected. As such, from Section 4.4.4 (included) forward, we use the advantage-weighted regression template (*cf.* Algorithm 5) adopted identically in [WNZ+20] and [NDGL20], consid-

ering how it yields the most favorable empirical results in Section 4.4.3. In order to verify that its competitive superiority is due to the *implicit* nature of how such agents are injected with optimality inductive bias, we evaluate an extension of the base advantage-weighted regression method in Section 4.4.4. These reveal just how strikingly sensitive to the means of bias injection offline RL methods are. In Section 4.5 and Section 4.6, we propose generalizations of the value and policy objectives involved in the considered base actor-critic method (*cf.* Algorithm 5), respectively — Section 4.5 is dedicated to policy evaluation, while Section 4.6 is dedicated to policy improvement. These generalizations involve *proposal* policies, effectively acting as placeholders for a slew of different action distributions that we introduce (9 in total) and evaluate, in both contexts (*evaluation* in Section 4.5, *improvement* in Section 4.6). Each proposal strategy tackles, with its own distinct flavor, the *balancing act* that both the policy and value entangled in a GPI scheme must address in offline RL: ***how to come close to behaving optimally for the given task while avoiding being hindered by out-of-distribution actions?*** Under a notion of *safety* purposely derived from this desideratum, we can equivalently ask: how to teach offline agents to approach optimality *safely*? As illustrated in Figure 4.1.1, striking the right balance is far from obvious and highly dependent on the quality of the dataset, which we discuss in every single reported experiment. Nonetheless, we show that our new generalized *framework* introduced in Section 4.6.3 enables us to *safely* inject bias in the method in such a way that the results are improved when the policy underlying the dataset is optimal, while not being harmed when it is not. Note, crucially, our agents are *never* made aware of the quality of the offline dataset they are provided with.

4.2 Related Work

The subfield of offline RL has been outlined in two surveys, under slightly different scopes. In [LGR12], the authors give an overview of the *batch* setting, and a thorough picture of how it was tackled from its inception by Gordon in 1995 (*cf.* [Gor95]) to the release date of the survey. Eight years later, in [LKTF20], the offline RL landscape is painted by the authors through a more practice-oriented lens. While [LGR12] identified *stability* as the main culprit hindering the wide adoption of offline RL in real-world systems (defending that a genuine expertise was needed to pull it off), [LKTF20] can depict a more optimistic view of the domain since it has since received attention and been leveraged in various applications (*e.g.* such as dialog systems). Despite having been addressed numerous times in recent years, and approached via different angles, the practical stability issues discussed in [LGR12] have yet to be solved, and are barely mitigated in the current state-of-the-art methods. Going back to the inception of batch RL, the work of [Gor95] proposes a model-based method called Averagers that estimates the action-value exactly at *supports*. It does so by leveraging the exact dynamic programming operation, requiring the availability of a model of the transition function to be able

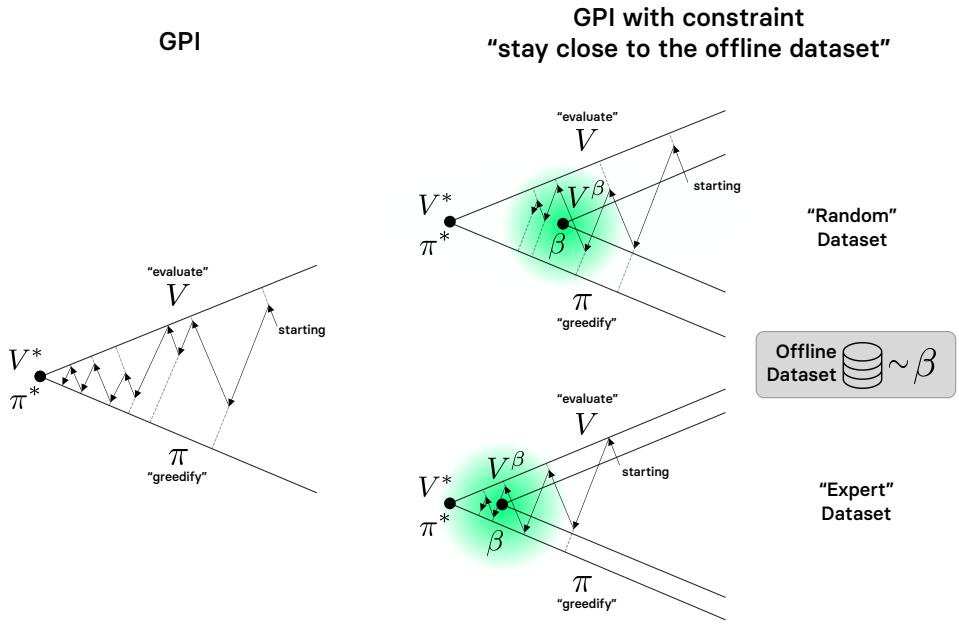


Figure 4.1.1: Sequence diagram representation of generalized policy iteration (GPI): as depicted in [SB98] for the online RL scenario on the left-hand side, and augmented for the offline RL case on the right-hand side. As reminded in [SB98], the real geometry is considerably more complex that this. In contrast with the diagram of [SB98] however, the consecutive policy *evaluation* and *improvement* sub-goals are not performed to completion at each step. The green area depicts the subspace of the policy-value joint parameter space where the constraint encoding the desideratum *"stay close to the offline dataset"* is satisfied (with the color density depicting the degree of satisfaction). The shape of the green area is determined by the metric used to enforce the constraint (although chosen here for legibility, we would have disks if we were employing an ℓ_2-loss over parameters). Crucially, this diagram illustrates *a)* that the gap between (π^*, V^*) and (β, V^β) is greater when the dataset is of poor quality, and *b)* how this gap would prevent an agent eager to remain close to the offline dataset (green area) from ever reaching optimal performance.

to solve the subsequent *planning* task with it. Besides, the approach of [Gor95] identifies as a *"fitted"* one, since it solves the action-value estimation problem before deriving a control policy from it. Note, the sequence diagrams depicted in FIGURE 4.1.1 do *not* temporally represent the learning process of agents learned under said *fitted* paradigm — otherwise, the dotted arrows would reach the solid lines every single time. Albeit raised and discussed, [Gor95] did not propose a model-free counterpart of the method, due to convergence issues for estimating the action-values at the supports from the action-values estimated at samples from the respective neighborhoods of the supports. Ormoneit and Sen [OS02] address this issue, though not in the AVERAGERS framework. Instead of attempting to estimate the values from neighboring samples at supports, they estimate the value directly via a

sample-based approximator (which is kernel-based). In other words, instead of relying on a transition model and carrying out the exact dynamic programming operation with it, their *kernel-based approximate dynamic programming* (KADP) method detaches itself from the need to possess a model of the world: it approximates the exact dynamic programming operator with the implicit sample-based transition model observable via the collected transitions in the batch (that we have access to, yet ignore the underlying dynamics and behavior models of). Our work is closer to the *least-squares policy iteration* (LSPI) method from [LP03] — whose first release was concurrent with KADP, since we set out to revisit the generalized policy iteration learning template in the offline regime (*cf.* SECTION 4.1 for what motivates our study, *cf.* FIGURE 4.1.1 for a conceptual depiction of the learning scheme). As such, we only consider methods that alternate between policy evaluation and policy improvement steps, in contrast with the distinct separation of these two problems that characterizes the *fitted* methods [Gor95, EGW05, Rie05, ASM07, NP08, LR10, HR11]. Note, LSPI [LP03] does not involve an explicit policy, the policy is defined implicitly from the learned value.

When it comes down to the modern state-of-the-art offline RL approaches which we critically evaluate in this work, one can divide them up into three distinct categories. These differ by how they encode the constraint enticing the agent to stay close to the offline dataset. They do so *1)* with *explicit* constraints on the *policy* (*e.g.* BRAC [WTN19], BEAR [KFTL19], WOP [JSG+20]), or *2)* with *explicit* constraints on the *value* (*e.g.* BRAC [WTN19], CQL [KZTL20]), or *3)* with *implicit* constraints on the *policy* (*e.g.* BC [Pom89, Pom90, RBS07, Bag15] BCQ [FMP18], MARWIL [WXH+18], AWR [PKZL19], ABM [SSB+20], CRR [WNZ+20], AWAC [NDGL20]). Note, we here only list out the model-free approaches — we will discuss the model-based ones momentarily. Optimizing an objective subjected to an implicit constraint corresponds to optimizing a *reduction* of an objective constrained explicitly. After reduction (*e.g.* via the Lagrangian method, with KKT conditions assumed to be satisfied in case of inequality constraint) the optimization problem is not constrained anymore, but it is still derived from an explicitly constrained problem, hence the *"implicit"* denomination. The derivation from a constrained objective to the unconstrained one optimized in MARWIL [WXH+18], AWR [PKZL19], ABM [SSB+20], CRR [WNZ+20], and AWAC [NDGL20] is laid out (albeit under a generalized form) in SECTION 4.6. Seeing BCQ [FMP18] as a perturbed, more sophisticated extension of BC, itself consisting in maximizing the likelihood of the learned policy over the data, we can equivalently interpret the objective of these algorithms as the minimization of the forward KL divergence between the true data distribution and the learned policy. As such, we *can* see these (BC and BCQ) as *implicitly* constrained.

Pretraining with offline datasets has been studied extensively in recent years. The pursued goal is simply to leverage offline datasets so as to increase the learning speed and final performance of downstream tasks in online RL, imitation learning [Bag15], and even offline RL (with a distinct task how-

ever). Such desideratum is pursued notably in [YN21], [AKA+20], [SLZ+20], and [NDGL20]. Nonetheless, we are *not* interested in this type of bootstrap and transfer capabilities in this work.

The endeavors we carry out in SECTION 4.5, including the formulation of the proposal action distributions (from which the next actions are to be sampled) relate to a slew of works that also attempted to skew the *SARSA* variant of Bellman's equation (used in usual actor-critic's) towards the *optimal* variant of Bellman's equation (used in Q-learning [Wat89, WD92]). Although plenty of modifications to the Bellman operator have been proposed throughout the years to entice the policy and value to adopt a certain desired behavior, we are here only interested in approaching the *optimal* action-value. The discussion that follows therefore focuses on how previous works have successfully been able to inject bias towards the optimal behavior in their respective agents. We align the notion of *optimality* with Bellman's principle of optimality. As such, a policy is optimal if and only if its value is solution to the optimal version of Bellman's equation. Whether an *expert* dataset (possibly human-generated) should be considered *optimal* is a valid question — discussed in [Rus92]. While one can just substitute the *SARSA* operator with the Q-learning one when dealing with discrete actions (*cf.* [CB95] for the *a priori* earliest adoption of such substitution in an actor-critic), how to proceed in continuous action spaces is less obvious (the max operation over actions creates an extra inner-loop optimization problem to solve *every iteration*). Different approaches have been adopted to circumvent this hindrance. As such, we discern four main strategies to tackle the max operation in continuous action spaces: *1)* discretizing the continuous action space into a discrete space of tractable dimension, then use Q-learning over the crafted discretized space (*e.g.* [MPD02, Kim07, MIJD17]), *2)* enforcing a strict structure over the Q-value in a way that facilitates the resolution of said extra inner-loop maximization, as done *e.g.* in NAF [GLSL16], *3)* getting rid of the hard max operation altogether by using a *soft* update instead [HTAL17], and finally *4)* tackling the resolution of inner-loop maximization problem every iteration, despite the leap in computational cost caused by the continuous nature of the action space. Approach *4)* has, for instance, been implemented via the use of the derivative-free, black-box, Cross-Entropy Method (CEM) [Rub99, MRG03] optimizer to estimate the maximizing action for the given state at each step, *e.g.* in QT-Opt [KIP+18, QJN+18], and in the Actor-Expert framework [LJL+18]. By contrast, the Amortized Q-learning (AQL) [VdWWFMM18] method follows approach *4)* by taking inspiration from the amortized inference literature and displays a conceptually simpler formulation. In effect, these methods replace the optimal policy that should in theory be used to generate the next action in the SARSA variant of Bellman's equation (to obtain the Q-learning variant of Bellman's equation), by a tractable approximation of it — which we refer to as *proposal* policy in SECTION 4.5. For example, QT-Opt [KIP+18, QJN+18] craft said proposal policy as the function that, for the given state, at the given iteration, returns the solution of the inner-loop, per-iteration maximization subproblem obtained via the CEM [Rub99, MRG03]. Tractable approximations of intractable policies

in continuous action spaces with proposal distributions over actions have also notably been carried out in [HBLH19] and [WADW18]. Finally, the most noteworthy method implementing approach *4)* is perhaps the operator introduced in EMaQ [GSG20], in the offline regime specifically. As we discuss in SECTION 4.5, this operator uses a proposal policy that interpolates between a tractable approximation of the optimal policy estimated via AQL, and an estimated clone of the policy underlying the offline dataset. The framework we propose for policy evaluation in SECTION 4.5 subsumes such operator. Finally, optimality *tightening* [HLSP17] also aligns with the optimality desideratum pursued here.

Based on how brittle state-action values estimated via temporal-difference learning are in the offline regime — due to the risk of using out-of-distribution actions for bootstrapping the value, one might want to ensure or even guarantee *safer* updates. This safety desideratum naturally links offline RL to the line of *Safe Policy Improvement* (SPI) methods (*e.g.* [TTG15, PCG16, LTdC19]), in the online regime. As reminded in SPIBB [LTdC19] (SPI by *Bootstrapping* the Q-value with a *Baseline*, not with the learned policy), the concept of *safety* in RL is overloaded to say the least (*cf.* [GF15]). Among all the possible hindrances one would desire to be shielded from — parametric (epistemic) uncertainty, internal (aleatoric) uncertainty, interruptibility, exploration in a hazardous environment — SPIBB sets out to design a method that guarantees safety against the potential damages that might be caused by an excessive epistemic uncertainty in the learned models. SPIBB builds on the work of Petrik (*cf.* [PCG16]), that designs a learning update rule for the actor that analytically guarantees a safe policy improvement, whatever the parameters of the model. Aligning its take on safety with Petrik's, SPIBB provides its agent access to a dataset as well as to a *baseline*. For most of their experimental endeavors, the dataset is assumed to be distributed as the baseline. Crucially, that is a condition that need be satisfied for the theory backing SPIBB to hold. Nevertheless, the authors also perform a set of experiments in which this restrictive assumption is relaxed. The relaxed scenario (the dataset was *not* collected with the baseline) is artificially implemented by replacing the dataset optained from the baseline by a dataset collected by a random policy, while the action bootstrap is still carried out using the baseline. This second setting shares a property we investigate in SECTION 4.5: how does detaching the proposal policy used to bootstrap from the policy that carried out the data collection, impact performance. We however do not have access to a baseline safeguard and our proposal policies do not require any additional privilied information; they are solely using the offline dataset (at most). The safe policy improvement rule proposed by SPIBB articulates as follows: when the agent is confident that it can improve upon the baseline, it will use its own learned policy, otherwise it will use the oracle baseline as a fallback. We take inspiration from these to formulate three of our proposal distributions, accordingly dubbed *safe*, and used in SECTIONS 4.5 and 4.6. As such, our work therefore also subsumes BRPO [SCO+20], which also proposes a learning update aligned with the one introduced in

the SPI line of works.

By consistently opting for the safer option when there is a potential risk (based on an estimated measure of uncertainty), SPI methods clearly follow a *pessimistic* heuristic, conceptually opposing the principle of *Optimism in the Face of Uncertainty Learning* (OFUL), ubiquitous in the Multi-Armed Bandits (MAB) and Online Learning (OL) literature. Note, any optimistic measure can trivially be made pessimistic by composing it with a monotonically decreasing function over reals — and optimistic measures of uncertainty or novelty are plentiful in the MAB, OL, and online RL literature to infuse the agent with exploratory incentives. Nonetheless, considering how the *offline* agent are unable to learn from their own mistakes (since their interactions are not recorded in the dataset used to train it), encourage exploration by advocating for optimism in the face of uncertainty should be avoided. Research endeavors in *offline* RL have instead turned to *pessimism*, as analyzed through a theoretical lens in [BGB20]. Pessimistic modifications to usual algorithms have recurrently been carried out via *model-based* approaches, consisting in replacing the rewards used by the learning agent with pessimistic reward surrogates. The first occurrence of such technique appears in the RaMDP technique propose by Petrik in [PCG16], where the author also introduced SPI, stressing how both SPI and RaMDP promote the adoption of pessimism when confidence is low. RaMDP transforms the rewards from the dataset via the application of a penalty. A reward from a given transition in the dataset will be penalized less if said transition appears often in the dataset. Conversely, it will be penalized more if the transition appears rarely in the dataset. The frequency of occurrence, used as a measure of confidence, is estimated via a *pseudo-count* [BSO+16, THF+16, OBvdOM17]. Years later, MOPO [YTY+20] and MoREL [KRNJ20] concurrently propose virtually identical model-based techniques penalizing the rewards in offline RL. In contrast with RaMDP, they formulate their reward penalties based on the uncertainty of a *forward* model (aiming to model the inner workings of the MDP). On the topic of reward re-shaping, [LGR12] suggests that using a smoother reward signal might help in stabilizing the learned Q-value, as first proposed and corroborated empirically in [HR11]. Yet, our preliminary investigation of reward smoothing did not yield improvement for the considered datasets. In this work, we consider only *model-free* approaches and, like mentioned earlier, dabble in pessimism only via SPI-based techniques.

Finally, the involvement of constraints consisting of KL divergences in the policy improvement objectives discussed and derived in Section 4.6 echoes the entire *"KL-control"* line of work, originating in [Kak01, KL02, KP08, PS08, VTKP09, TBS10, FB10, PMA10, KOP10, Neu11, KGO12]. Honorable mentions that distinguish themselves from these, yet are tightly related, could arguably be G-learning [FPT15], and ψ-learning [RTV13]. In the offline regime, WOP [JSG+20] notably implements a KL-control approach.

4.3 Background

Setting

In this work, we tackle the problem of *offline* RL — also sometimes referred to as *batch* RL [LGR12]: the autonomous agent must learn how to interact optimally in an environment without being allowed to interact with it during training. On the flip side, our learning agent has access to a collection of interactions (including the received feedback in the form of *reward*) from another distinct agent. This *counterfactual* interactive information storage is called the offline dataset, noted $\mathcal{D}$. The offline dataset is made available before the learning process starts, and is kept frozen throughout the entirety of the training procedure. Note, we do not consider the *"growing batch"* setting, in which the dataset *can* grow during training, either by the hands of the learning agent, or by an external source. Leveraging solely the offline dataset $\mathcal{D}$, the agent must learn an *online*, interactive policy that will enable the accumulation of the high rewards during *evaluation* phases. Note, we forbid the agent from accessing the offline dataset at evaluation time — allowing it would place our work in the *"observational learning"* setting. Albeit fairly realistic, we want our agents to be purely online when let loose for evaluation, with no means of tapping into pre-existing repositories of interaction data. As such, our agent uses only the offline dataset $\mathcal{D}$ without interacting online with its environment $\mathcal{E}$ at training time, but interacts online with $\mathcal{E}$ without ever using $\mathcal{D}$ at evaluation time — neither for *reading* $\mathcal{D}$, nor *writing* in $\mathcal{D}$.

World and agent

In this work, we model $\mathcal{E}$ as a memoryless, infinite-horizon, and stationary *Markov Decision Process* (MDP) [Put94], noted $\mathbb{M}$. Formally, $\mathbb{M} \coloneqq (\mathcal{S}, \mathcal{A}, p, \rho_0, u, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ are respectively the state space and action space. The *dynamics* of the world are determined by the stationary, stochastic transition function p, and the initial state probability density ρ_0. In effect, $p(s'|s, a)$ is the conditional probability density concentrated at the state s' when action a is executed in state s. The reward feedback that $\mathcal{E}$ returns upon executing a in s is modeled as the outcome of a stationary reward process u that assigns real-valued rewards distributed as $u(\cdot|s, a)$. The remaining piece of $\mathbb{M}$ is $\gamma \in [0, 1)$, the discount factor. The decision-making process of the learning agent is modeled by the parametric policy π_θ, under a neural representation with the parameter vector θ. The stochastic policy π_θ followed by the agent maps states to probability distributions over actions, which we denote by $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, or by the compact notation $\pi_\theta \in \mathcal{P}(\mathcal{A})^{\mathcal{S}}$, where $\mathcal{P}(\mathcal{A})^{\mathcal{S}} \coloneqq \{\pi \mid \pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})\}$. Concretely, the experiences of the agent are divided across discrete timesteps t, where $t \geq 0$. The analyses we carry out in this work do not require the involvement of a finite time horizon T, hence our decision to adopt the infinite-horizon MDP setting where t is *a priori* unbounded above in the

formalism. Despite being infinite horizon, we make the MDP episodic by assuming that every trace contains at least one absorbing state, and that when the first is reached by the agent, γ is artificially set to zero to emulate termination, hence formally constructing an *episode*. At each timestep $t \geq 0$, the agent is located at $s_t \in \mathcal{S}$, and concentrates a probability density over actions from $\mathcal{A}$ that is equal to $\pi_\theta(a|s_t)$ at action a. We denote the action selected by the agent's policy π_θ at timestep t by a_t, and the received reward by r_t.

Lastly, we introduce the discounted state visitation frequency for an agent following a policy π in the MDP $\mathbb{M}$, denoted as $\rho^\pi_\mathbb{M}$, and abbreviated ρ^π in the absence of ambiguity. Formally, $\rho^\pi_\mathbb{M}(s) \coloneqq \sum_{t=0}^{+\infty} \gamma^t \mathbb{P}^\pi_\mathbb{M}[S_t = s]$, where $\mathbb{P}^\pi_\mathbb{M}[S_t = s]$ is the probability of reaching state s at timestep t (S_t is a random variable) when following π in $\mathbb{M}$. Since an immediate derivation gives $\sum_{s\in\mathcal{S}} \rho^\pi_\mathbb{M}(s) = 1/(1-\gamma)$, $\rho^\pi_\mathbb{M}$ can be seen as a probability distribution over states up to a constant factor. Being in the episodic setting, we will use the *undiscounted* counterpart of $\rho^\pi_\mathbb{M}$, yet still artificially set $\gamma = 0$ when the absorbing state (posited earlier to always exist) is reached to emulate episode termination.

Dataset

We do not have access to the analytical form of the policy that generated the dataset, nor do we have the ability to sample from it. Besides, since none of the approaches mentioned in this work explicitly leverage the fact that the offline dataset might have been produced by multiple distinct sources, we posit *w.l.o.g.* that the offline dataset $\mathcal{D}$ contains interaction traces of a single conceptual policy β, dubbed the *offline* distribution. Despite being composed of traces of interaction, these are not necessarily available in connex trajectories, but rather as a shuffled collection of individual transitions. We do not know how the data was collected, in particular what the underlying strategy β was optimizing for, nor do we know the proficiency of β in satisfying the chased objective other than what we can infer and hopefully extrapolate from the offline dataset. $\mathcal{D}$ might have been collected with a single fixed snapshot of a policy, in which case β is this very snapshot; or $\mathcal{D}$ might be the training history of a policy, in which case β is a mixture of past iterates. For a given dataset $\mathcal{D}$, β might be focusing on covering the most ground while not paying too much attention to the collected rewards, or conversely covering the least amount of ground while accumulating rewards as greedily as possible. Note, in imitation learning and inverse RL [Bag15] where rewards are not known, one assumes the *expert* (β's counterpart) was acting optimally when collecting the demonstrations. This is *not* the case in this work. We use datasets whose quality range from *expert*-grade data (where β is close to optimality) to *random* data (where β wanders seemingly aimlessly) (*cf.* Section 4.4.2). Our agents are never made aware of $\mathcal{D}$'s quality or β's proficiency. The offline dataset $\mathcal{D}$ is formally defined as a collection of *SARS*-formatted transitions (s, a, r, s') collected by the underlying offline distribution β through interactions with $\mathbb{M}$. Being in the episodic setting, transitions also contain a termination indicator

in practice, taking value 1 when the transition is the last one in the episode, and 0 otherwise. When $\mathcal{D}$ has a richer structure — from transitions being *SARSA*-formatted to transitions being sequenced in full connex trajectories — we say so explicitly in the text. As noted in [FMP18] and [LTdC19], we have in effect two MDPs in the tackled offline setting: *1)* the real, non-observable, online MDP $\mathbb{M}$ underlying the inaccessible environment $\mathcal{E}$, and *2)* the fictitious, observable, offline MDP $\mathbb{M}^{\text{OFF}}$ effectively communicated through the dataset to the agent. As such, while β interacted with $\mathbb{M}$ to collect $\mathcal{D}$, π_θ interacts with $\mathbb{M}^{\text{OFF}}$ — in effect *detached* from the real world $\mathcal{E}$ — and collects nothing. Every state s in $\mathcal{D}$ is then distributed as $\rho^\beta_{\mathbb{M}}(\cdot)$. For legibility purposes, we will use ρ^β as a shorthand for $\rho^\beta_{\mathbb{M}}$. In practice, with a slight abuse of notation, we can note $(s, a, r, s') \sim \mathcal{D}$ to indicate that the transition (s, a, r, s') is in effect obtained by sampling from the offline dataset $\mathcal{D}$. Nevertheless, we will often opt for the explicit notation $\mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s),s'\sim\rho^\beta(\cdot)}[\cdot]$ — for *SARS*-formatted transitions, and $\mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s),s'\sim\rho^\beta(\cdot),a'\sim\beta(\cdot|s')}[\cdot]$ for *SARSA*-formatted transitions — as a drop-in replacement for $\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[\cdot]$ and $\mathbb{E}_{(s,a,r,s',a')\sim\mathcal{D}}[\cdot]$, with r being replaced by $r(s, a, s')$ in the operand of the expectation not to overload the $\mathbb{E}$ notation. We express the reward function as a function of the next state s' as well, as it is the most general setting and can be reduced to $r(s, a)$ by positing trivial assumptions. We do not indicate the states (and actions when applicable) at which the conditional densities are evaluated in the outer expectations throughout this work, to lighten the notation as much as possible.

Objective

We here describe the concepts that we need to add to our tool set so as to properly deal with the delayed nature of the reward feedback. In the infinite-horizon regime we placed our agent into, the *return* R^γ_t is the discounted sum of rewards from timestep $t \geq 0$ onwards, and is at the core of how RL methods attempt to solve the credit assignment problem created by said delayed feedback. We formalize the return of the state-action pair associated with timestep t as $R^\gamma_t \coloneqq \sum_{k=0}^{+\infty} \gamma^k r_{t+k}$. The expectation of the return along traces of the arbitrary policy π starting from the execution of action a_t in s_t defines the state-action value Q^π (also called action-value, or simply Q-value) such as $Q^\pi(s_t, a_t) \coloneqq \mathbb{E}_{s_{t+1}\sim p(\cdot|s_t,a_t),a_{t+1}\sim\pi(\cdot|s_{t+1}),\dots}[R^\gamma_t]$ (*abbrv.* $\mathbb{E}^{\geq t}_\pi[R^\gamma_t]$). We say that a policy π acts *optimally* at timestep $t \geq 0$ if the action π selects at state s_t, noted a_t verifies the following identity: $a_t = \text{argmax}_{a\in\mathcal{A}} Q^\pi(s_t, a)$. Such behavior coincides with π being purely greedy with respect to the *exact* action-value Q^π coupled with π (*cf.* Q-value definition right above). By extension, π acts optimally at *every* timestep $t \geq 0$ if and only if, for any given start state $s_0 \sim \rho_0$, the policy π maximizes $V^\pi(s_0) \coloneqq \mathbb{E}_{a_0\sim\pi(\cdot|s_0)}[Q^\pi(s_0, a_0)]$. Our objective is for the agent to learn a policy π_θ such that π_θ is *a* solution of the optimization problem $\pi_\theta = \text{argmax}_{\pi\in\Pi} U_0(\pi)$ for any given start state $s_0 \sim \rho_0$, where $U_t(\pi) \coloneqq V^\pi(s_t)$ is the *performance objective* [SLH+14], or *utility*, and Π is the neural search space for π_θ.

KL divergence notations

In addition, we adopt an alternative notation for the KL divergences by adding an arrow within the operator in order to disambiguate from a quick glance whether it is the *forward* (*inclusive* [Mac03]) or the *reverse* (*exclusive* [Mac03]) KL divergence, due to its asymmetric nature. The orientation of said arrow indicates the order in which one should write the predicted and target distributions respectively in the original KL divergence notation. As such, denoting the predicted and target distributions as q_θ and p respectively — the parameter "θ" being used here to highlight which distribution is parameterized and learned, and which is fixed and set as target — we write, for a given state s, $D^p_{\overrightarrow{\mathrm{KL}}}[q_\theta](s)$ to replace $D_{\mathrm{KL}}\big(p(\cdot|s)\,||\,q_\theta(\cdot|s)\big)$, and $D^p_{\overleftarrow{\mathrm{KL}}}[q_\theta](s)$ instead of $D_{\mathrm{KL}}\big(q_\theta(\cdot|s)\,||\,p(\cdot|s)\big)$. With this notation, the target distribution is always indicated in the exponent, of the divergence operator, while the learned distribution to fit to the target is alone in the operand of the operator. By following the arrow, we can then read, without ambiguity, for a given state s, *a)* $D^p_{\overrightarrow{\mathrm{KL}}}[q_\theta](s)$ as *"the forward KL between q_θ and p"*, and *b)* $D^p_{\overleftarrow{\mathrm{KL}}}[q_\theta](s)$ as *"the reverse KL between q_θ and p"*.

Stiffness

We propose the notion of hyper-parameter *"stiffness"*, defined by analogy with the notion of *stiff equation* in physics: we say that an algorithm A is *stiff* with respect to the hyper-parameter λ if slight changes in the value of λ cause large and non-monotonic variations in A's asymptotic performance, — or, to a more extreme extent, make A numerically unstable. In effect, from a practitioner's perspective, that translates into λ being tedious to tune. We introduce this notion due to various offline RL methods having been reported for their sensitivity and brittleness *w.r.t.* hyper-parameter choices (for instance in [LPPM+20] and [MKL+20]). The universal desideratum is to design algorithms A that are stiff *w.r.t.* none of their hyper-parameters λ, and this is naturally what we aspire to in this work.

4.4 The Offline Reinforcement Learning landscape

4.4.1 Competing baselines

As the first contribution of this work, we perform a thorough PyTorch [PGM+19] re-implementation of the *state-of-the-art* competing baselines in offline RL under a *common* computational infrastructure. This allows not only for meaningful comparisons in which we can vary in a very controlled manner different design choices, but also for the easy development of new offline RL algotithms. We describe the key elements of our empirical setting in Section 4.4.2, and urge the reader to refer to the provided codebase for further details.

The baselines investigate, analyze, and evaluate in this section are: SAC [HZAL18], D4PG [BMHB+18],

BCQ [FMP18], BEAR [KFTL19], CQL [KZTL20], BRAC [WTN19], BC [Pom89, Pom90, RBS07, Bag15], CRR [WNZ+20], and AWR [PKZL19] (*cf.* SECTION 4.2 for a categorization of these algorithms in semantic families by how they enforce *closeness* with respect to the dataset $\mathcal{D}$ to avoid the hindering distributional shift). These are all off-policy actor-critic [Sut84, KT00, KT03, DWS12] architectures — with the exception of BC, which is a pure imitation learning [Bag15] method (more on that momentarily). The actor or policy is modeled with π_θ (introduced in SECTION 4.3); the critic with the parametric model Q_ω. We study actor-critics and not *pure* value methods since we tackle *continuous* action spaces in this work (*cf.* SECTION 4.4.2 where we describe the continuous control tasks considered for this present work). In all of these implementations, we tried to strike the right balance between *a)* following the official implementation when made available by the authors, *b)* using the hyper-parameter values recommended in the associated paper (when not conflicting with the official implementation), and *c)* making the algorithms computational comparable in terms of flops, number of parameters, runtime, difficulty of implementation, accessibility to privileged information. Concretely, every single baseline *a)* starts learning *from scratch* (no warm-start), *b)* is provided with exactly the same information as input (*only* the offline dataset $\mathcal{D}$ at training time, *only* the online environment $\mathcal{E}$ modeled by the MDP $\mathbb{M}$ at evaluation time), *c)* is given access to the exact same computational resources (1 modern high-end GPU), and *d)* is allowed the same maximum runtime (12 hours). We now would like to draw the reader's attention to some of the aspects of the tackled algorithms.

The SAC [HZAL18] and D4PG [BMHB+18] algorithms were originally introduced as online RL algorithms, and we simply repurposed them as offline RL ones by *a)* removing the agent's ability to interact, collect, and store *new* data, and *b)* leaving the training loop unaltered, yet replacing the replay buffer $\mathcal{R}$, with the offline dataset $\mathcal{D}$. This candid transition from the online to the offline regime for SAC [HZAL18] and D4PG [BMHB+18] has proved disastrous numerous times, as reported in some of the candidate baselines above (SAC [HZAL18] has been reported the most; D4PG only rarely, despite the promising potential displayed by distributional values in [ASN20]). Behavioral Cloning (BC) [Pom89, Pom90, RBS07, Bag15] is the only imitation learning [Bag15] method of the above listing, and only uses the state-action pair (s, a) of transitions pulled from the offline dataset $\mathcal{D}$ to learn π_θ as a supervised learning regressor (states s are the inputs; actions a are the real-values outputs). Such method is not equiped to leverage the reward information communicated through the offline dataset $\mathcal{D}$, or equivalently via the observable, fictitious MDP $\mathbb{M}^{\text{OFF}}$. State-of-the-art imitation learning techniques that *are* able to do so (*e.g.* adversarial approaches like GAIL [HE16], SAM [BK19], and DAC [KAD+19]) require the agent to interact with $\mathbb{M}$ to collect more data to update their *surrogate* reward proxy. Since only $\mathbb{M}^{\text{OFF}}$ is accessible at training time, in the offline RL regime, such methods can not be used here. In addition, since AWAC [NDGL20] was released concurrently with

CRR [WNZ+20] and are essentially equivalent (*cf.* statement from the authors in [NDGL20]), we use the *"CRR"* notation to denote either indifferently. In line with the results reported in [WNZ+20] and [NDGL20], showing that ABM [SSB+20] is consistently outperformed by their respective approaches (CRR and AWAC respectively), we save valuable resources by not including ABM in the list of baselines. Besides, *a)* ABM's optimization shares its derivation with AWAC and CRR (*cf.* Section 4.6), *b)* the generalized framework we propose in Section 4.6 subsumes ABM. We encourage the reader to directly jump to that section for more details about how their (and our) objectives are derived. Finally, we omit approaches solely relying on ensemble learning (*e.g.* REM [ASN20], BEAR's UCB-like ensemble-based extension [KFTL19] based on Bootstrapped DQN [OBPVR16]), as we want to factor *out* ensembling techniques from the equation to figure out what are the *core* aspects of the studied approaches that are single-handedly responsible for the best performance. Then, one can trivially involve ensembling to reduce the epistemic (parametric) uncertainty.

4.4.2 Experimental setting

In this work, we carry out all our experiments in the D4RL suite of environment-dataset couples. We refer the reader to its companion paper [FKN+20] where the authors report in great detail *a)* the proficiencies an agent must possess to achieve high performance in the various physics-based simulated robotics tasks, and *b)* how the spectrum of datasets associated with each task were collected — ranging from data collected from an expert-grade agent (which we sometimes refer to as *high*-quality data) to purely random data (*low*-quality data). We focus on the subset of environment-dataset couples from D4RL that are based on the fast and scalable MuJoCo [TET12] physics engine and interfaced via OpenAI's Gym [BCP+16] API. That represents a total of 15 datasets per experiment: 3 distinct environments (corresponding to tasks involving a distinct MuJoCo-based simulated robot), and 5 distinct datasets for each environment (of different quality grades, yet the same 5 grades for all 3 environments). Such consistency enables us to draw more generalizable conclusions from our findings about how different algorithms perform when provided with dataset of various qualities from the available spectrum. Importantly, such reasoning consistency can only be ensured for the MuJoCo-based tasks of D4RL, since it is the only benchmark for which the spectrum of dataset qualities ranges (in 5 increments) from *low* to *high* for every single task.

Throughout this work, we consistently organize the results under this categorization by arranging the 15 plots in a grid where the *rows* correspond to the distinct five environments (or equivalently, tasks), while the *columns* describe the spectrum of the five dataset qualities — from *expert* for the left-most column, to *random* for the right-most column. Intermediate grades include *medium* (collected from a partially-trained agent), or even *replay* (contents of the replay buffer [Lin93] kept from the training procedure of an agent), *cf.* [FKN+20] for finer details about the spectrum. As such, in the

plots laying out the empirical results, looking at a *row* gives the respective performances in a given environments in five different datasets organized left-to-right from expert- to random-grade, while looking at a *column* informs the reader about how proficient the agent is at accumulating reward and achieving high return compared to its competitors for a given dataset quality, across 3 different MuJoCo-based locomotion tasks (*cf.* Figure 4.4.1 for an arbitrary example).

As mentioned above, we tried to use the hyper-parameter values suggested in the baselines' papers or codebases (provided the latter is provided, does not conflict with the companion report), unless the used values are clearly giving an unfair advantage to one method over the others, while not being part of the claimed reasons why said method outperforms its competitors. For instance, we aligned the number of layer, number of hidden units per layer, and output heads in the neural function approximator of CRR [WNZ+20] with the one used in SAC [HZAL18], BEAR [KFTL19], CQL [KZTL20], AWAC [NDGL20] (equivalent approach), among others. Consequently, we used a 2-layer MLP with 256 hidden units in each layer — for both the policy (actor) and the critic, with a single Gaussian head (the policy network returns a single mean μ and standard deviation σ pair). [WNZ+20] uses *mixtures* of Gaussian heads, by contrast. We make the option available via our available re-implementation codebase, yet do not report any result involving mixtures of Gaussian heads.

We only consider the variant of CRR [WNZ+20] where the function f used to wrap the advantage weighting the likelihood is the exponential function $x \mapsto \exp(x/\tau)$, along with the estimation of the advantage where the state-value V is estimated as an empirical average. The other variants (defining V in the advantage with a maximum operator instead of an expectation, Heaviside step function for f) were performing poorly across the board, for the suite of tasks and datasets we selected for this present study. As such, like AWAC [NDGL20], the objective used for policy improvement in CRR [WNZ+20] under the considered setting can be derived exactly in line with the analysis laid out in Section 4.6. Plus, by simply replacing the method used to estimate the advantage in AWR [PKZL19] from a Monte-Carlo one to a temporal-difference one, we end up collapsing onto the objective optimized by CRR [WNZ+20] and AWAC [NDGL20]. In line with what was first reported in the Appendix C of AWR [PKZL19], and in later CRR [WNZ+20], among others, we clamp the advantage-based exponential weights — re-weighting the actor's likelihood in the policy improvement objective — to remain below a maximum value of 20, for all the tackled methods sharing the same derivation (AWR [PKZL19], AWAC [NDGL20], and CRR [WNZ+20], *cf.* Section 4.6).

As for the temperature τ used in the advantage-based exponential weights objective of AWR [PKZL19] and CRR [WNZ+20], we use the values recommended in the respective reports. As such, we use $\tau = 1$ for CRR, and $\tau = 0.05$ for AWR. In an effort to uniformize the temperature across the two methods, we investigated how AWR would perform if the temperature τ was raised to 1, and report the associ-

ated auxiliary experiment in Appendix 4.E, Figure 4.E.1. Judging by these auxiliary results, which show that neither temperature value (neither $\tau = 0.05$, nor $\tau = 1$) clearly outperforms the other, we do not have reason enough to stray from the original suggested τ value. We thus use $\tau = 0.05$ in AWR.

A fair portion of the baselines listed out above involve a *warm-up* period consisting in using a behavioral cloning loss in the objective optimized by the actor's policy π_θ. This is done either by adding it as an extra piece of the main policy improvement loss, or by replacing the main loss with the cloning one until a certain iteration is reached. The thresholds used vary vastly from one baseline to another (values reported in the companion codebase or report). We use these original values per baseline, and otherwise do not use any warm-up at all (*e.g.* CRR [WNZ+20] does *not*).

Among the tackled baselines, AWR [PKZL19] is the only one that approximates the action-value Q^{π_θ} or the actor's policy π_θ with Q_ω learned via Monte-Carlo (MC) estimation, while *all* the others estimate it via temporal-difference (TD) learning [Sut84, Sut88, SMSM99]. As such, AWR needs the offline dataset $\mathcal{D}$ to be sequentially organized in connex trajectories, in order to be able to compute the Monte-Carlo returns of each state-action pairs in $\mathcal{D}$, and use them as targets for Q_ω. If such information about the $\mathcal{D}$'s sequentiality is *not* available, then AWR is *not* usable. Yet, if it is, one could then say that AWR has access to privileged information over the other baselines, and therefore benefits from a putative, unfair advantage. Either way, this reliance on full trajectories to estimate Q_ω is a clear drawback. On the flip side, by not involving bootstrapping over potentially out-of-distribution actions burdening Q_ω's stability in the offline regime, AWR is naturally shielded from such instabilities, or at the very least hindered by them to a lesser extent. This is a clear advantage for AWR. Nonetheless, we will see momentarily (*cf.* Section 4.4.3) that, despite possessing and leveraging privileged information about the sequentiality of the offline dataset $\mathcal{D}$, AWR is consistently outperformed by another baseline in every dataset of the tackled suite. Since it uses a critic Q_ω learned via MC-based policy evaluation, and is the only one to do so here, the methods that outperform it are TD-based.

It is rather surprising that this should happen in the offline RL regime, since the approximation of the action-value via temporal-difference learning is so much more sensitive to distributional shift (more so than in the online setting where the shift already takes a toll). Despite being more prone to Q_ω-related instabilities, it seems that temporal-difference learning is still the strongest contender relative to Monte-Carlo estimation in policy evaluation (*cf.* Section 4.4.3). To prevent Q_ω misestimation caused by out-of-distribution actions injected in Bellman's equation (TD learning), we use the same mechanism that is usually used in the online regime to counteract the *overestimation* bias [TS93] suffered by Q_ω. Said mechanism is Double Q-learning [vH10], or in particular the extension of Double Q-learning to modern deep neural models, Double DQN [vHGS15]. Considering our continuous

control setting, we use the direct counterpart of Double DQN for actor-critic architectures presented in [FvHM18]. As such, by default, every approach evaluated empirically involves a second Q-value estimate (called *twin* critic in actor-critics, *cf.* [FvHM18]). In the same vein as in SECTION 4.4.1, we do not study the extent to which more prolific ensembles (more than two action values) impact the agent's ability to fend off the overestimation bias, something that has shown limited success so far (*e.g.* in [KFTL19]).

When estimating Q_ω via temporal-difference learning, we use target networks, in line with all the value-based and actor-critic deep RL works that came after DQN [MKS+13, MKS+15] that originated the stabilization trick. When porting the various techniques and tricks introduced in DQN to the DPG algorithm [SLH+14], DDPG [LHP+16] opted for a slight variation in how the target network trick was executed. Instead of replacing the frozen parameters of the target networks *periodically* with a snapshot of the latest iterate (for both actor and critic, respectively), [LHP+16] makes the target networks slowly track the main networks by applying *every iteration* an update rule akin to Polyak's averaging technique [PJ92].

In this work, we evaluate a given offline RL agent by setting it loose in an *online* instance of the environment in which the *offline* dataset $\mathcal{D}$ it was trained with was collected. In other words, our agents are trained in the fictitious MDP $\mathbb{M}^{\text{OFF}}$, and evaluated the real MDP $\mathbb{M}$ (*cf.* SECTION 4.3). Concretely, we evaluate the agent every 5000 training iterations across all experiments. Evaluating agents offline, and consequently *a fortiori* off-policy, is a tedious and challenging feat to carry out properly, as attested by the myriad of works on off-policy evaluation (OPE) in recent years. We refer the reader to [VLJY19] for a comprehensive overview of the current OPE landscape. Questions related to opting for off-policy evaluation in offline RL have been raised very recently in [LPPM+20], and tackled there to an extent. Overall, how to best evaluate the agent *purely* offline, *i.e.* without any iteration with $\mathbb{M}$ at any point in the lifetime of the agent, remains an open question. In terms of metrics used to gauge the quality and proficiency of the learned agent, we assess to what extent the agent satisfies its performance objective by observing how high of a return it can accumulate over the course of an episode. We determine which agent is the best by comparing these average episodic returns — the higher, the better. Reaching the top performance *faster* also constitutes a valuable asset for an agent. During evaluation, we sample from the actor's policy π_θ directly. Note, since the actor's objective relies entirely on the critic Q_ω, by the very design of actor-critic architectures, the performance achieved by following π_θ in $\mathbb{M}$ is a direct image of how good of an estimate Q_ω is. For π_θ to perform well at evaluation time, designing a sensible update rule for Q_ω is paramount. Sampling from π_θ directly to evaluate the agent is therefore an excellent indicator of the *global* performance of the system.

We run each experiment for 0.5M iterations (the value most often used in the different baselines), or

for a threshold total duration of 12 hours, whichever occurs first. The stopping condition based on the runtime duration has the obvious advantage of penalizing the methods that take longer per iteration than their competitors, hence favoring the ones that are computationally cheaper to run in terms of flops. In every reported experiment, each agent uses its own modern high-end GPU. Only our D4PG implementation is distributed across more than one worker, but these parallel workers share the single GPU allocated to the agent. Yet, each worker is allocated its own CPU. Note, the codebase allows for more than one GPU to be used by an agent, and load-balances the distributed workers across the available ones automatically. The distributed paradigm used for D4PG is the following: *a)* at the start, each worker is assigned a distinct rank $k \geq 0$; *b)* at the start, each worker sets its random seed as the output of the same deterministic function that only depends on the rank k, making them in effect sample different minibatches from the offline dataset $\mathcal{D}$; *c)* at each iteration, each worker with rank $k > 0$ computes the gradient of its actor's loss, then sends it to the worker with rank $k = 0$, who aggregates all the k received gradients (including its own) by computing their empirical average, and finally sends the mean gradients to the $k - 1$ workers with $k > 0$ to replace their own gradients with. We decided not to distribute the baselines (except for D4PG whose distributed aspect is the very core of the method) primarily to save on computational budget, but also to prevent the gradient averaging scheme to conceal numerical instabilities some baselines might suffer from more than others. Additionally, every experiment is repeated over a fixed set of 4 random seeds, given to the agent beforehand. Every single plot reported in this work averages the statistics across these random seeds. Solid lines correspond to the *mean* μ over the seeds. Shaded areas correspond to trust regions around μ whose width are equal to $0.95\,\sigma$, where σ is the standard deviation of the studies recorded statistic (*e.g.* the return) over the fixed set of random seeds. We monitored every experiment with the Weights & Biases [Bie20] tracking and visualization tool.

4.4.3 Empirical assessment

In Figure 4.4.1, we depict the empirical comparison of the offline RL baselines laid out in Section 4.4.1, following the evaluation protocol exhibited near the end of Section 4.4.2. As an agent goes through parameter updates and gets better at interacting with $\mathbb{M}$, it will survive longer, leading to evaluation trials that also last longer. These extended survival periods due to the agent's own proficiency at tackling the task at hand have a direct effect on its total learning process, entangling alternatively training and evaluation phases. This increase in duration per online evaluation trial will in effect cause the agent (*e.g.* the BCQ [FMP18] agent in the top-left sub-plot) to hit the *timeout* before reaching the 0.5M iterations mark. Such preliminary termination in terms of iterations thus effectively does not impact how we rank the method, since there seems to be nothing left for the agent to learn then. By contrast with prolonged *evaluation* trials, the performance traces might ap-

pear truncated in FIGURE 4.4.1 (depicting that the agent has hit the runtime timeout before satisfying the *"number-of-iterations"* stopping criterion) due to considerably longer *training* durations (*e.g.* the CQL [KZTL20] agent in the top-left sub-plot). In our experimental setting, a longer training duration per iteration can only be caused by a higher computational complexity (allocated computational resources are identical across agents). While the cause underlying an increase in evaluation time is nonissue since the agent must already be proficient at the task for such an inflation to even occur, an extended training duration is more often than not an issue, since it does *not* depend on how well the agent performs. By displaying significantly longer training times, an agent might reach the timeout while still performing poorly and having much left to learn from $\mathcal{D}$. Limiting the allowed time for an agent to solve the task (like we do here) is therefore *penalizing* agents whose complexity (and by extension, computational cost) exceed the complexity of its competitors by too large of a margin. Note, we see in FIGURE 4.4.1 that the used runtime timeout is virtually always long enough to enable agents to reach the 0.5M iterations mark. Based on this observation *and* our compute budget, we did not deem necessary to increase said timeout period. Besides, it seems fair to punish methods that fail to achieve their final performance within the allowed runtime while so many manage to do so.

FIGURE 4.4.1 does not allow us to name a definitive *best* approach, as none of the baselines seem to perform well across the board. Yet, overall, it looks like the three main contenders for the title would be BCQ [FMP18], CRR [WNZ+20], and AWR [PKZL19].

As expected (and documented in past offline RL literature), the port of SAC [HZAL18] to the offline regime (described in SECTION 4.4.1) performs extremely poorly in every single dataset and environment. Motivated by the promise of distributional RL [BDM17, DRBM17, DOSM18, DKNU+20] in the offline setting underlined by REM [ASN20], D4PG [BMHB+18] displays higher returns than SAC [HZAL18], but is still mediocre compared to the baselines designed specifically for the offline regime. Nonetheless, the contrast in performance between SAC and D4PG suggests we might prompt noteworthy return gains simply by replacing the traditional critic used in natively offline RL methods with a distributional one. AWR [PKZL19], as discussed, leverages privileged information about the sequentiality of the offline dataset; this fact has to be taken into account when evaluating its performance. At first glance, it might appear as surprising that behavioral cloning (BC) gathers such high returns in a number of cases given that it is a *pure* imitation learning [Bag15] approach. Indeed, a BC agent need only use the state-action pairs (s, a) extracted from the transitions provided through the offline dataset $\mathcal{D}$, which is considerably less information than what the usual offline RL approaches use. BC discards the reward signal present in every transition, which essentially acts as a score on the decision made by β to execute action a at state s in the MDP $\mathbb{M}$. Instead, the BC agent only knows what β did, without knowing how proficient β was at accumulating rewards. As such, it is to be expected that BC *only* performs well when β does well, and becomes increasingly worse as the β underlying

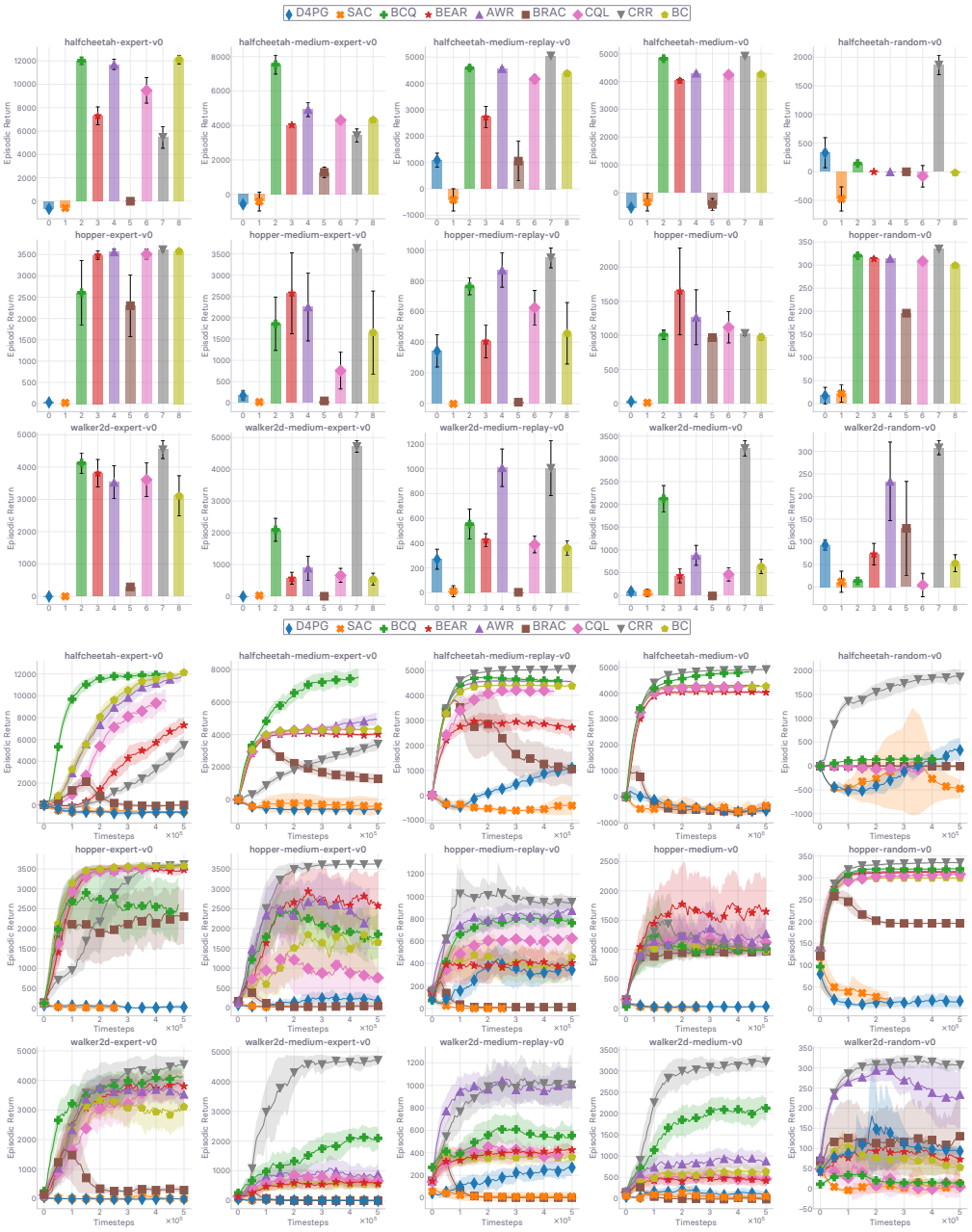


Figure 4.4.1: Empirical evaluation of our unified re-implementations of the offline RL baselines: SAC [HZAL18], D4PG [BMHB+18], BCQ [FMP18], BEAR [KFTL19], CQL [KZTL20], BRAC [WTN19], BC [Pom89, Pom90, RBS07, Bag15], CRR [WNZ+20], and AWR [PKZL19]. The first three rows give the return mean and standard deviation on training completion. The last three rows five the evolution of the return. Runtime is 12 hours. Best seen in color.

the offline dataset $\mathcal{D}$ strays from optimality. This is indeed what we observe in FIGURE 4.4.1: BC's return is highest for the *expert* datasets, and decreases relatively to its counterparts as we look from left (expert) to right (random). Since BC does not use the rewards in $\mathcal{D}$, it does not possess counterfactual learning abilities [BPQC+13]. Instead, it can only replicate the behavior demonstrated in the offline dataset with extremely limited extrapolation capabilities, and therefore can only achieve optimality if β is optimal for the task. Besides, BC is incredibly easier to implement that most of its baseline counterparts, which heavily plays in the imitation learning method's favor, from a purely practical perspective. CRR comes in a close second if we were to rank the methods by that criterion.

In FIGURE 4.4.1, we observe that CRR outperforms its competitors except in a handful of datasets and environments. In contrast to BC which imitates the policy β underlying the offline dataset $\mathcal{D}$, CRR only enforces the actor's policy π_θ to remain *somewhat* close to β via an *implicit* constraint (*cf.* EQ 4.34 for the generalized version of said constraint, and (*cf.* SECTION 4.6 for a derivation that could trivially be boiled down to obtain CRR's actor objective *exactly*). ***In an effort to ground the discussion with more conceptual formalism, we introduce the notion of*** **optimality inductive bias** $\mathcal{B}$***, grounded on the offline dataset*** $\mathcal{D}$***. Concretely, the amount of bias*** $\mathcal{B}$***, noted*** B***, injected into an agent represents to what degree the agent builds upon the belief that*** β ***underlying*** $\mathcal{D}$ ***is*** **optimal** ***to update its policy*** π_θ**.** Crucially, note, we do not built $\mathcal{B}$ to be aligned with the *genuine* intrinsic quality of the data present in $\mathcal{D}$, but on the agent's belief that $\mathcal{D}$ contains *expert*-grade data — that β is optimal. For example, BC injects the maximum possible amount of bias $\mathcal{B}$, B_{MAX}, into the agent since it *posits* by design that the provided demonstrations are originating from an optimal expert policy. In fact, this statement applies to every method following the imitation learning paradigm. Natively-offline baselines that enforce an explicit closeness constraint with β to avoid any distributional shift caused by out-of-distribution actions inject an inductive bias B_{EXPL} — and by symmetry, B_{IMPL} for the ones that involve an implicit constraint, like CRR. Intuitively, we loosely have: $B_{\text{MAX}} \geq B_{\text{EXPL}} \geq B_{\text{IMPL}} \geq 0$. One can move B_{EXPL} and B_{IMPL} within the interval $[0, B_{\text{MAX}}]$ by increasing the scaling coefficient associated with the constraint enforcing said *closeness* of π_θ with respect to β (equivalently, $\mathcal{D}$). Notably, we found that, in the case of BRAC [WTN19], — and to a lesser extent in the case of CQL [KZTL20] — finding the right level of bias B_{EXPL} to inject, proved to be tedious, and remarkably difficult to tune. The scaling coefficients controlling the injection of the optimality inductive bias in these methods thus qualify as stiff (in line with the notion of *stiffness* we have defined in SECTION 4.3). A similar observation has been made by [MKL+20] for CQL (BRAC was not tackled there). Interestingly, BCQ [FMP18] involves neither an explicit nor an implicit constraint between π_θ and β. Rather, we would categorize BCQ as a perturbed imitation learning method. As such, it injects an inductive bias $B \approx B_{\text{MAX}}$ into the agent. This is clearly illustrated in FIGURE 4.4.1, where BCQ performs well on expert dataset, yet poorly on random datasets.

As a rule of thumb, the closer the injected bias B is to B_{MAX} (pure imitation learning), *a)* the better the method performs in *expert* datasets, and *b)* the worse it performs in *random* ones, on the other side of the quality spectrum. Intuitively, treating everything as equally valuable in $\mathcal{D}$ is a bad idea if it is not the case, but is optimal if it is indeed the case. From a practitioner's perspective, it then all comes down to how much is known about the contents of the offline dataset. Consider a condition, dubbed $\mathcal{C}$, that is verified whenever we *know* that β is *optimal* for the task. When $\mathcal{C}$ is satisfied (we know that β is optimal), one should inject an inductive bias $B \approx B_{\text{MAX}}$ into the agent (*e.g.* via BCQ or via an imitation learning method like BC). Conversely, when $\mathcal{C}$ is *not* satisfied (either we do not know at all what is in $\mathcal{D}$ quality-wise, or we know that β is sub-optimal), one should inject an inductive bias $B < B_{\text{MAX}}$ into the agent (*e.g.* via CRR). Rephrasing what precedes, based on our results in FIGURE 4.4.1 it seems that the *best* course of action is for the offline RL practitioner to: *a)* use BC or BCQ (or any other method with high dataset-grounded bias) when $\mathcal{C}$ is satisfied, and *b)* use CRR (which is in effect with an *advantage re-weighted* BC) when $\mathcal{C}$ is not satisfied.

In the next section, we further corroborate these statements by showing that increasing the optimality bias of CRR in a minimalist and parsimonious fashion quickly makes the resulting method better in expert datasets and worse in random ones. Crucially, the same method can achieve state-of-the-art performance across the considered spectrum of datasets qualities via the adjustment of a single hyper-parameter, provided one knows whether β is optimal or not.

What's next. In the remainder of this work, we will use CRR [WNZ+20] (or equivalently, AWAC [NDGL20]) as base, since it is the method that seems to perform consistently well across the board. Given the central role it plays in what follows, we lay out the algorithm in ALGORITHM 5 under the name BASE, which denotes either CRR or AWAC indifferently.

4.4.4 DATASET-GROUNDED OPTIMALITY INDUCTIVE BIASES

We now investigate an extension of the BASE approach (denoting CRR or AWAC indifferently) approach, whose pseudo-code is described in ALGORITHM 5. We carry out a thorough analysis of the behavior of the method resulting from the addition of the CQL [KZTL20] constraints in BASE. Adding these constraints in effect provides us with a finely controllable handle on the further injection of inductive bias $\mathcal{B}$ (*cf.* SECTION 4.4.3) in BASE. These constrains introduced by CQL will be explicitly reported momentarily in this section. We call the composite method *"Reinforce The Gap"* (*abbrv.* RTG). The notion of *gap* (noted Δ_{GAP}) we use here aligns with the one introduced in CQL [KZTL20]:

$$\Delta_{\text{GAP}} := \mathbb{E}_{s\sim\rho^{\beta}(\cdot),a\sim\beta(\cdot|s)}\left[\max\left\{Q_{\omega}(s,a^{i}) \mid a^{i} \sim \text{unif}(\mathcal{A}[s])\right\}_{i\in[1,m]\cap\mathbb{N}} - Q_{\omega}(s,a)\right] \tag{4.1}$$

Algorithm 5: BASE (denotes either CRR [WNZ+20] or AWAC [NDGL20] indifferently)

init: initialize the random seeds of each framework used for sampling, the random seed of the environment $\mathbb{M}$, the neural function approximators' parameters (θ for the actor's policy π_θ, and ω for the critic's action-value Q_ω), the critic's target network ω' as an exact frozen copy, the offline dataset $\mathcal{D}$.

while *no stopping criterion is met* **do**

/* Train the agent in $\mathbb{M}^{\text{off}}$ */

Get a mini-batch of samples from the offline dataset $\mathcal{D}$;

Perform a gradient *descent* step along $\nabla_\omega \ell_\omega$ (*cf.* below) using the mini-batch;

$$\ell_\omega \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s),s'\sim\rho^\beta(\cdot)}\left[\left(Q_\omega(s,a) - \left(r(s,a,s') + \gamma\,\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\left[Q_{\omega'}(s',a')\right]\right)\right)^2\right]$$

where $r(s,a,s')$ was introduced as syntactic sugar in SECTION 4.3;

Perform a gradient *ascent* step along $\nabla_\theta \mathcal{U}_\theta$ (*cf.* below) using the mini-batch;

$$\mathcal{U}_\theta \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s)}\left[\exp\left(\frac{1}{\tau}A^{\pi_\theta}_\omega(s,a)\right)\log\pi_\theta(a|s)\right]$$

where $A^{\pi_\theta}_\omega(s,a) \coloneqq Q_\omega(s,a) - \mathbb{E}_{\bar{a}\sim\pi_\theta}[Q_\omega(s,\bar{a})]$, and τ is a temperature hyper-parameter;

Update the target network ω' using the new ω;

/* Evaluate the agent in $\mathbb{M}$ */

if *evaluation criterion is met* **then**

foreach *evaluation step per iteration* **do**

Evaluate the empirical return of π_θ in $\mathbb{M}$ (*cf.* evaluation protocol in SECTION 4.4.2);

end

end

end

where $\mathcal{A}[s]$ is the set of actions from $\mathcal{A}$ that are feasible in state s. The observation made in CQL is that the introduced constraints have the expected effect of increasing the maximum gap Δ_{GAP} (*cf.* definition in EQ 4.1) in action-value between random, uniformly-sampled actions, and actions from the offline dataset $\mathcal{D}$ at a given state from $\mathcal{D}$. We will report these gaps for both methods (BASE with and without CQL constraints) momentarily. Despite only being — in the context of our work — a toy extension of BASE that allows us to study the bias $\mathcal{B}$ more closely in a controlled environment, RTG also appears (very recently) in [MKL+20] as the combination of two state-of-the-art offline RL methods. As the direct combination of CQL and CRR, [MKL+20] names the method *conservative* CRR (*abbrv.* CCRR). CCRR was empirically evaluated in a handful of datasets of different qualities. We propose a far more fine-grained dataset design technique that enables us to finely control the percentage of random (or expert) data in the dataset.

We build RTG by adding both CQL's constraints in BASE, as add-on pieces to the loss optimized by Q_ω (*cf.* [KZTL20]). These are constraining Q_ω directly. Formally, the loss optimized by CQL and RTG to learn Q_ω articulates as follows (omitting numerical tricks; *cf.* [KZTL20] for the various versions of CQL):

$$\ell_\omega := \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s),s'\sim\rho^\beta(\cdot)}\left[\left(Q_\omega(s,a) - \left(r(s,a,s') + \gamma\,\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\left[Q_{\omega'}(s',a')\right]\right)\right)^2\right] \tag{4.2}$$

$$+\,\alpha\left(\mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\text{unif}(\mathcal{A}[s])}\left[Q_\omega(s,a)\right] - \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s)}\left[Q_\omega(s,a)\right]\right) \tag{4.3}$$

The loss laid out in EQ 4.2 is the standard temporal-difference (or TD) error minimized by CRR's critic Q_ω (as it appears in ALGORITHM 5). Brought over from CQL, the first piece of EQ 4.3 constitutes CQL's first constraint; it tries to minimize the action value everywhere — using uniformly sampled actions in $\mathcal{A}$ to apply the constraint onto. The second piece of EQ 4.3 constitutes CQL's second constraint; it tries to maximize the action-value over $\mathcal{D}$. In effect, the loss laid out above encourage the enlargement of the gap Δ_{GAP} in EQ 4.1. Note, the gap is signed. RTG's name stems from this desideratum, by plugging CQL's constraints into CRR, we urge the agent to deepen the gap in action-value between arbitrary actions and the ones from the offline dataset. In effect, the aggregation of these two constraints *increases* the learned Q_ω over $\mathcal{D}$, and *decreases* it everywhere else. As such, the higher the scaling coefficient for these constraints, the more quantity optimality inductive bias $\mathcal{B}$ we inject artificially into the CRR agent, while having a tight handle of how much we inject. We see it as a minimal and parsimonious way to study the impact of such injection on the performance of CRR. In FIGURE 4.4.2, we show how RTG compares to CRR in terms of return. Additionally, in FIGURE 4.4.3, we display the associated gaps Δ_{GAP} (*cf.* definition in EQ 4.1). FIGURE 4.4.4 puts things into perspective by depicting RTG's performance against the baselines that we laid out in SECTION 4.4.1. We observe in FIGURE 4.4.2 that RTG improved upon CRR in the 3 top-left corner

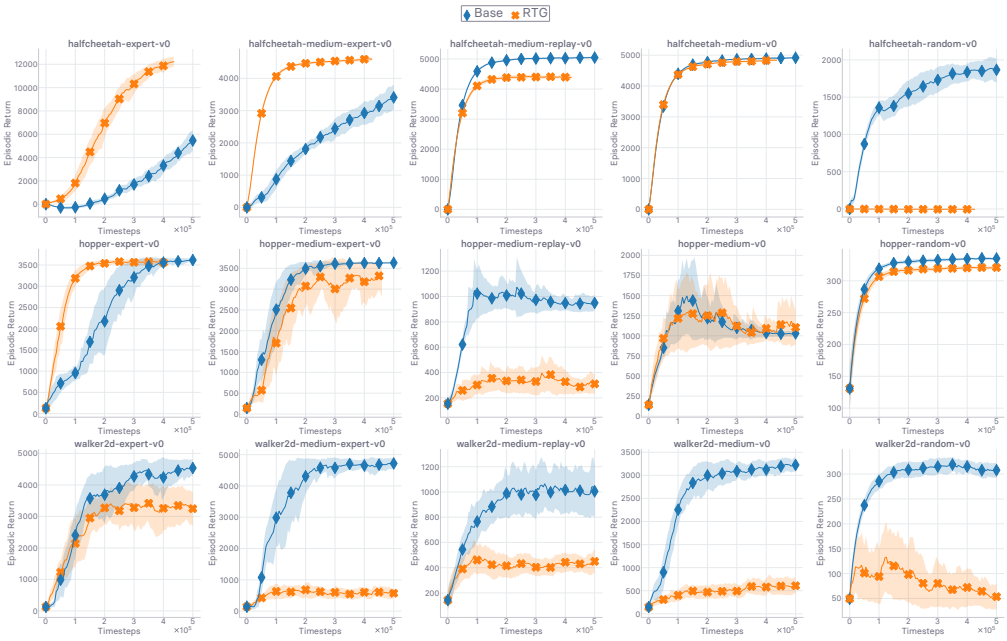


Figure 4.4.2: Empirical evaluation of the *return* of BASE and RTG. Runtime is 12 hours. Best seen in color.

subplots, and displays significantly worse results in the 12 other subplots of the grid. We arrive at the same expected conclusion: increasing a method's bias towards the optimality of β is a good idea if and only if β is at least close to being optimal. Naively forcing a method to imitate β by injecting more of $\mathcal{B}$ is therefore not a decision to be taken lightly and should be heavily grounded with respect to what we *know* (and perhaps more importantly, what we do *not* know) about the contents of $\mathcal{D}$. The greater bias is depicted clearly in FIGURE 4.4.3 where we see that the gaps displayed by RTG are consistently further away from zero than the ones in CRR. Interestingly, there are two datasets (random, top-right and bottom-right) in which the RTG gaps have high *positive* values, instead of low negative values like in the other subplots of the grid in FIGURE 4.4.3. Based on how the gap Δ_{GAP} is defined (*cf.* EQ 4.1), this means that, according to the RTG agent, actions inside the offline dataset have lower value than ones uniformly picked in $\mathcal{A}$. This goes against the desideratum that motivated the introduction of the bias-inducing constraints, and attests to the *brittleness* of such biasing mechanism. As expected, the RTG agent performs particularly horridly in these two datasets (random, top-right and bottom-right in FIGURE 4.4.2), in line with the nonsensical gaps displayed by RTG in these (*cf.* FIGURE 4.4.3). Interestingly, FIGURE 4.4.4 shows that RTG beats CQL in almost every tackled dataset. Nevertheless, one should only use RTG when $\mathcal{C}$ is satisfied — otherwise, CRR.

We have established through a series of experiments that RTG performs well on expert datasets and

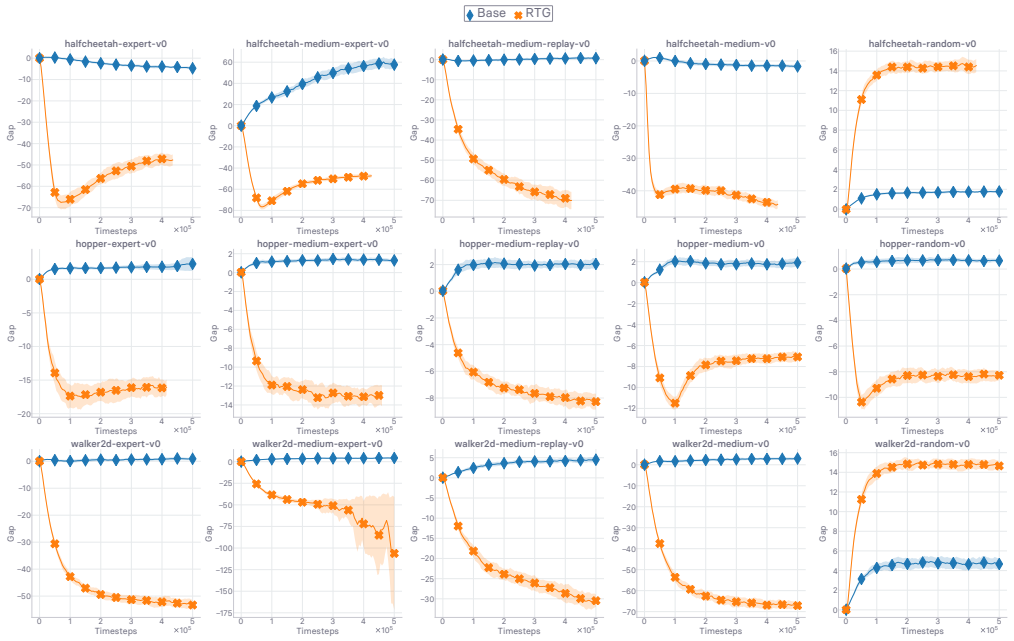


Figure 4.4.3: Empirical evaluation of the *gap* (*cf.* EQ 4.1) of BASE and RTG. Runtime is 12 hours. Best seen in color.

poorly on random datasets, due to how much optimality bias $\mathcal{B}$ is injected into the agent. BASE does not inject as much bias, and thus *a)* behaves far better when β is sub-optimal, but *b)* considerably lags behind RTG in expert datasets. We would like to know how both methods perform in between these dataset qualities, *i.e.* what happens when the grade of data in the offline dataset $\mathcal{D}$ gradually decreases from the maximum level (expert) to the minimum level (random). To answer this, we investigate how both BASE and RTG perform in a series of *mixed* datasets. These are crafted by aggregating a portion $p \in [0, 1]$ of the *expert* dataset for a given environment with a portion $1 - p$ of the *random* dataset for the same environment. In our experiments, p covers the set of values: $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. Note, we *shuffle* the datasets A and B using the agent's random seed, before merging the portion p extracted from dataset A with the portion $1 - p$ extracted from dataset B. Since we average every reported run across a set of random seeds fixed beforehand (*cf.* SECTION 4.4.2), the results reported in FIGURE 4.4.5 for this set of runs with mixed datasets are all the more robust and reproducible. As expected, RTG drops far quicker in performance than BASE as we increase the proportion of random data. Yet, RTG still manages to accumulate a *"fair"* return when the portion of random data in $\mathcal{D}$ is as high as $1 - p = 0.4$ across the range of environments. As such, the results of FIGURE 4.4.5 show us once more that *knowing* about $\mathcal{D}$'s quality (*i.e.* whether condition $\mathcal{C}$ is verified or not) to then chose a method with the *right* level of optimality inductive bias $\mathcal{B}$ is preferable over designing an algorithm than can *"do it all"*. Without this knowledge (*i.e.* condition $\mathcal{C}$ is not verified), then BASE

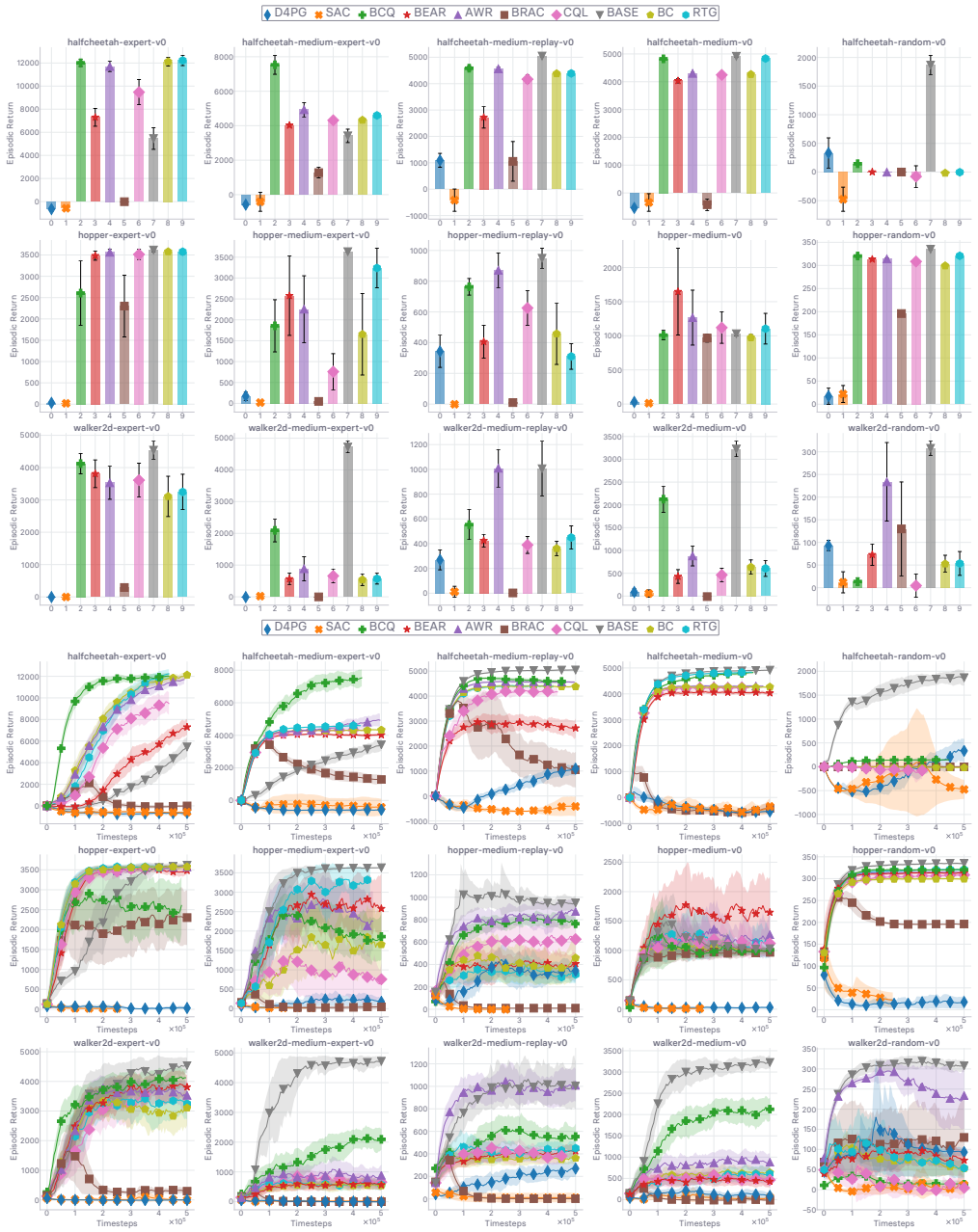


Figure 4.4.4: Empirical evaluation of the *return* of RTG among the baselines treated in SECTION 4.4.1. The first three rows give the return mean and standard deviation on training completion. The last three rows five the evolution of the return. Runtime is 12 hours. Best seen in color.

is the practitioner's best bet. Besides, in practical scenarios where the data source can oftentimes be compromised and polluted with random data, it is far easier for us to recommend the use of BASE over RTG (*cf.* FIGURE 4.4.5) — or any other method with high bias.

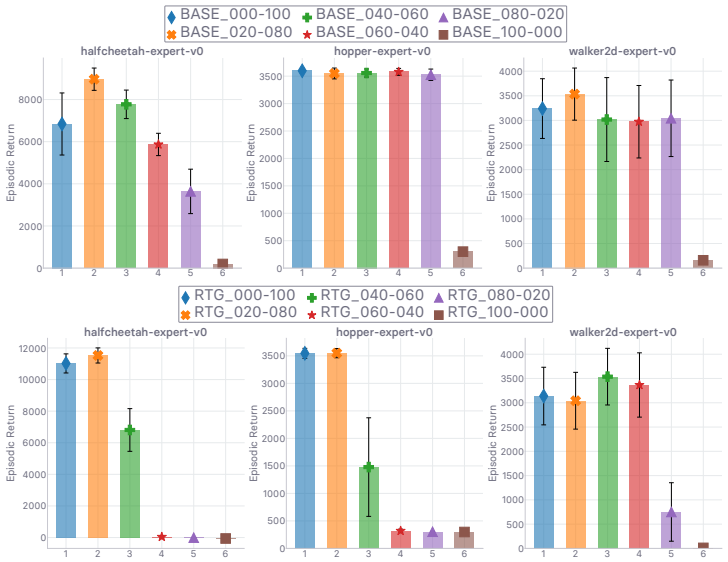


Figure 4.4.5: Empirical evaluation of the *return* of BASE (top row) and RTG (bottom row) in *mixed* datasets (*cf.* text for a complete description of the experimental design). In the legend, `020-080` means that the mixed dataset contains 20% of data from the random dataset, and 80% from the expert one (for a given environment), which corresponds to having $p = 0.8$. We cover the range $p \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. Runtime is 12 hours. Best seen in color.

What's next. Since in practice the optimality condition C is never satisfied, we set out to investigate how to improve the BASE approach by revisiting the *Generalized Policy Iteration* (GPI) learning procedure (subsuming BASE, as well as *any* actor-critic touched on or investigated in this work) in the *offline* regime (*cf.* FIGURE 4.1.1). In essence, methods that implement GPI alternate between a policy *evaluation* step (during which the value Q_ω is updated to be consistent *w.r.t.*, or evaluate, the policy π_θ) and policy *improvement* step (during which the policy π_θ is updated to be greedy *w.r.t.* its coupled value Q_ω). The loss optimized by BASE's critic Q_ω is laid out in ALGORITHM 5 and in EQ 4.2. Learning the critic Q_ω offline, characterized by the inability to acquire more data via interactions with the MDP, exposes said critic to a distributional shift due to out-of-distribution (or OOD) actions that can be involved in the Bellman target part of EQ 4.2. This phenomenon can manifest simply because any model likely evaluates arbitrarily poorly on data located outside the distribution said model was trained on. As such, since the Bellman target part of EQ 4.2 involves an evaluation of the critic Q_ω on an action from the learned policy π_θ (in line with GPI) then these evaluations might yield nonsen-

sical values as soon as π_θ's are too far off β's predictions (*i.e.* too far off the distribution underlying the dataset, which *is* the training distribution in the considered offline setting). Most of the methods touched on when we laid out the related works in SECTION 4.2, and studied in our first investigation in SECTION 4.4.1, stave off OOD actions by forcing the learned policy π_θ be be close to β, the distribution underlying the offline dataset $\mathcal{D}$. Since, in GPI, π_θ is used to generate the action employed in the Bellman target, updating π_θ in the vicinity of β allows Q_ω not to suffer the instabilities that would be caused by a distributional shift in target actions. In line with the goal of GPI, the alternation of policy evaluation and improvement must lead the estimated value and policy to coincide with their optimal counterparts in the sense of Bellman's optimality, *while* being tied to β for stability concerns (as illustrated in FIGURE 4.1.1). Unless the offline distribution β is optimal (*i.e.* the optimal policy coincides with β), the agent must face the following trade-off: ***to what extent should one aim for optimality at the expense of stability?***

As such, we investigate unifying generalizations of the value objective and policy objectives that consider how close to optimality the agent can get without being exposed to the dreaded distributional shift that hinders the offline agent. These policy evaluation and improvement studies are carried out in SECTIONS 4.5 and 4.6 respectively. These generalized evaluation and improvement objectives can be aligned with the traditional actor-critic ones implementing GPI in particular cases. Our investigations involve the introduction of a wide spectrum of ***proposal policies***. These proposal policies or distributions act as placeholders or substitutes for a slew of different action distributions, some ***safer*** than others in terms of exposure to distributional shift due to OOD actions. These investigations all take place over BASE. We chose CRR *a)* for the same reason we have done so in the investigation carried out in this section (it is the method that seems to perform consistently well across the board, as shown and concluded in SECTION 4.4.1), but also *b)* because we have just shown in this section that simply injecting dataset-grounded optimality bias in CRR (crystallized as RTG) enables the method to compete with top-performing baselines in the three datasets CRR was falling behind.

4.5 WHICH PROPOSAL DISTRIBUTION SHOULD Q EVALUATE?

4.5.1 UNIFYING OPERATORS

Before listing out the proposal distributions ζ's considered in this work, we first define homomorphic functional operators over the space of functions mapping states from $\mathcal{S}$ to (state-conditioned) probability densities over actions from $\mathcal{A}$. These operators — denoted by $\mathcal{T}_{\text{EVAL}}$ and $\mathcal{T}^{\omega,m}_{\text{MAX}}$ — transform stochastic policies from $\mathcal{P}(\mathcal{A})^{\mathcal{S}}$ into stochastic policies from $\mathcal{P}(\mathcal{A})^{\mathcal{S}}$, and differ by how the sampling unit of the former is used to build the samples of the latter: $(\forall \pi \in \mathcal{P}(\mathcal{A})^{\mathcal{S}})$, sampling from the s-conditioned policy $\mathcal{T}^{\omega,m}_{\text{MAX}}[\pi](\cdot|s)$ corresponds to sampling m actions from the policy π at

s and picking the action a among the m sampled ones that has the highest estimated action-value at s, $Q_\omega(s,a)$. Formally, $\mathcal{T}^{\omega,m}_{\textsc{Max}}[\pi] \in \mathcal{P}(\mathcal{A})^{\mathcal{S}}$ is defined to satisfy the following equivalence:

$$a \sim \mathcal{T}^{\omega,m}_{\textsc{Max}}[\pi](\cdot|s) \iff a = \underset{a^i}{\operatorname{argmax}} \left\{ Q_\omega(s,a^i) \mid a^i \sim \pi(\cdot|s) \right\}_{i \in [1,m] \cap \mathbb{N}} \tag{4.4}$$

$\big(\forall \pi \in \mathcal{P}(\mathcal{A})^{\mathcal{S}}\big)(\forall s \in \mathcal{S})$. For conceptual symmetry with $\mathcal{T}^{\omega,m}_{\textsc{Max}}$, we similarly introduce $\mathcal{T}_{\textsc{Eval}}$, defined as the identity homomorphic operator from and to the space of stochastic policies from $\mathcal{S}$ to $\mathcal{A}$. Trivially, $(\forall \pi \in \mathcal{P}(\mathcal{A})^{\mathcal{S}})$, sampling from the s-conditioned policy $\mathcal{T}_{\textsc{Eval}}[\pi](\cdot|s)$ corresponds to sampling a single action a from the policy π at s and picking this action. Maintaining the symmetry in notations, $\mathcal{T}_{\textsc{Eval}}[\pi] \in \mathcal{P}(\mathcal{A})^{\mathcal{S}}$ is formally defined to satisfy the following equivalence:

$$\big(\forall \pi \in \mathcal{P}(\mathcal{A})^{\mathcal{S}}\big)(\forall s \in \mathcal{S}) \qquad a \sim \mathcal{T}_{\textsc{Eval}}[\pi](\cdot|s) \iff a \sim \pi(\cdot|s) \tag{4.5}$$

We will leverage these operators as building blocks to assemble proposal policies with the maximum amount of notational overlap to keep our notation's verbosity to a bare minimum. Lastly, we introduce the operators $\mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\delta}_{\textsc{Cond-Eval}}$ and $\mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\delta}_{\textsc{Cond-Max}}$, both from $\mathcal{P}(\mathcal{A})^{\mathcal{S}}$ to $\mathcal{P}(\mathcal{A})^{\mathcal{S}}$ like $\mathcal{T}_{\textsc{Eval}}$ and $\mathcal{T}^{\omega,m}_{\textsc{Max}}$, which we define as follows, $\big(\forall \pi \in \mathcal{P}(\mathcal{A})^{\mathcal{S}}\big)(\forall s \in \mathcal{S})$,

$$a \sim \mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\delta}_{\textsc{Cond-Eval}}[\pi](\cdot|s) \iff a = \Gamma^\theta_\delta(s)\, \tilde{a}_{\textsc{Eval}} + \big(1 - \Gamma^\theta_\delta(s)\big)\, \tilde{a}^\theta_{\textsc{Max}} \tag{4.6}$$

$$\text{with} \quad \tilde{a}_{\textsc{Eval}} \sim \mathcal{T}_{\textsc{Eval}}[\pi](\cdot|s) \text{ and } \tilde{a}^\theta_{\textsc{Max}} \sim \mathcal{T}^{\omega,m}_{\textsc{Max}}[\pi_\theta](\cdot|s) \tag{4.7}$$

and

$$a \sim \mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\delta}_{\textsc{Cond-Max}}[\pi](\cdot|s) \iff a = \Gamma^\theta_\delta(s)\, \tilde{a}_{\textsc{Max}} + \big(1 - \Gamma^\theta_\delta(s)\big)\, \tilde{a}^\theta_{\textsc{Max}} \tag{4.8}$$

$$\text{with} \quad \tilde{a}_{\textsc{Max}} \sim \mathcal{T}^{\omega,m}_{\textsc{Max}}[\pi](\cdot|s) \text{ and } \tilde{a}^\theta_{\textsc{Max}} \sim \mathcal{T}^{\omega,m}_{\textsc{Max}}[\pi_\theta](\cdot|s) \tag{4.9}$$

where $\Gamma^\theta_\delta(s) \coloneqq \mathbb{1}\big[\rho\big(s, \tilde{a}^\theta_{\textsc{Max}}\big) \geq \delta\big]$, and where ρ is a potential function taking non-negative real values over the product space $\mathcal{S} \times \mathcal{A}$. Since Γ^θ_δ takes values in the binary set $\{0,1\}$, $a = \tilde{a}_{\textsc{Eval}}$ or $\tilde{a}_{\textsc{Max}}$ if $\rho\big(s, \tilde{a}^\theta_{\textsc{Max}}\big) \geq \delta$ — for $\mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\delta}_{\textsc{Cond-Eval}}$ and $\mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\delta}_{\textsc{Cond-Max}}$ respectively, and $a = \tilde{a}^\theta_{\textsc{Max}}$ if $\rho\big(s, \tilde{a}^\theta_{\textsc{Max}}\big) < \delta$. In practice, typical good candidates for ρ are density, novelty, or uncertainty estimates over $\mathcal{S} \times \mathcal{A}$, which can be obtained, among other techniques, via random network distillation (RND) [BESK18], or by estimating the epistemic uncertainty via an ensemble [OBPVR16]. Note, any signal over $\mathcal{S} \times \mathcal{A}$ that has shown promises when distilled into a reward function or even inspire the design of one is usually a suitable candidate for ρ. — *e.g.* signals derived from psychology and animal learning, typically categorized under the *intrinsic motivation* [Bar13, NBS10, Sch10] class of incentives to guide the artificial agent's exploration. We define the $\mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\delta}_{\textsc{Cond-Eval}}$ and $\mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\delta}_{\textsc{Cond-Max}}$ operators to later introduce proposal

policies inspired from *safe policy improvement* (SPI) [PCG16]. Specifically SPIBB [LTdC19] relies on an estimate of pseudo-counts $\tilde{N}_{\mathcal{D}}(s, a)$ [BSO+16, THF+16, OBvdOM17], themselves inspired from the counts involved in the design of upper confidence bounds in the multi-armed bandit literature building on the principle of OFUL (*cf.* Section 4.2). The framework we introduce in this work allows us to replicate SPIBB [LTdC19] by getting the actions used to bootstrap Q_ω from a proposal action selection method built with the operators $\mathcal{T}^{\omega,\theta,m,\Gamma^\rho_\delta}_{\text{Cond-Eval}}$ and $\mathcal{T}^{\omega,\theta,m,\Gamma^\rho_\delta}_{\text{Cond-Max}}$, where ρ would align with the pseudo-count estimator $\tilde{N}_{\mathcal{D}}$. In this work, we define $\rho(s, a) > 0$ to be a score aligned with the propensity of the pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ to be generated by the policy β underlying the offline dataset $\mathcal{D}$. We achieve this by learning a novelty score over the offline dataset $\mathcal{D}$ (at the same time as the other networks) via RND [BESK18]. This score, which we denote by $\eta(s, a)$, is defined as the prediction error between the outputs of a neural function approximator frozen after initialization and a non-frozen copy that is updated to predict the arbitrary frozen outputs of the first network. While aligned with a novelty signal in terms of variations, prediction errors (especially from quadratic losses) do not have appropriate scales to behave well as score surrogates. As such, we maintain an online rolling estimate of the standard deviation $\sigma^\eta_{\text{online}}$ of these prediction errors — as suggested originally in [BESK18] — and use a normalized novelty score instead, $\bar{\eta}(s, a) \coloneqq \eta(s, a)/\sigma^\eta_{\text{online}}$. *In fine*, we can now define the potential function ρ that we will use in all the reported empirical results thereafter:

$$(\forall s \in \mathcal{S})(\forall a \in \mathcal{A}) \qquad \rho(s, a) \coloneqq 1 - e^{-\bar{\eta}(s, a)/\tau} \tag{4.10}$$

where a sweep performed in preliminary searches lead us to choose the temperature $\tau = 0.06$ in every subsequent empirical studies reported in this work. Note, RND's temperature is akin to the bandwidth in kernel density estimation. The higher the bandwidth (or equivalently, the temperature), the smoother the density estimator. These sweeps also helped us pick a suitable value for the threshold variable δ, for which we assign the value $\delta = 0.6$. In terms of range, ρ takes values in $[0, 1)$ since $e^{-\bar{\eta}(s, a)/\tau}$ takes values in $(0, 1]$. If the pair (s, a) is deemed novel by η (*i.e.* $\bar{\eta}(s, a)$ has high value), then $\rho(s, a)$ is close to 1. Conversely, if (s, a) is not considered as novel, then $\rho(s, a)$ is close to 0. As we will show shortly, the assembled score ρ can therefore be instrumental in the design of *safe* action selection methods, which ultimately motivated the introduction of the operators $\mathcal{T}^{\omega,\theta,m,\Gamma^\rho_\delta}_{\text{Cond-Eval}}$ and $\mathcal{T}^{\omega,\theta,m,\Gamma^\rho_\delta}_{\text{Cond-Max}}$. These will be used to design *safe* proposal policies, which in the context of offline RL corresponds to presenting a *low risk* of injecting out-of-distribution action into function approximators — most critically, into the learned action-value Q_ω approximator at training time.

4.5.2 Offline dataset distribution clones

Finally, we introduce the policies β_{C} and β_{C}^{ξ}, the last prerequisites before laying out the proposal policies we considered for policy evaluation and improvement. β_{C} is a clone of β, the policy underlying the offline dataset $\mathcal{D}$. Concretely, the β_{C} policy is modeled via a state-conditional variational auto-encoder (VAE) [KW14, RMW14] trained to reconstruct the state-action pairing displayed in $\mathcal{D}$, effectively cloning β via behavioral cloning (BC), making it a policy one can sample from — given a state s — at training and evaluation time. While β_{C} stochastically generates actions from given states, ξ maps state-action pairs to actions, and should therefore be interpreted as a *state-conditional action perturbation* rather than as a policy. Leveraging the perturbation model ξ, we introduce β_{C}^{ξ} to satisfy the following equivalence, $(\forall s \in \mathcal{S})$:

$$a \sim \beta_{\mathrm{C}}^{\xi}(\cdot|s) \iff a = a_{\beta_{\mathrm{C}}} + \phi\, a_{\xi} \tag{4.11}$$

$$\text{with} \quad a_{\beta_{\mathrm{C}}} \sim \beta_{\mathrm{C}}(\cdot|s) \text{ and } a_{\xi} = \xi(s, a_{\beta_{\mathrm{C}}}) \tag{4.12}$$

Such a policy (perturbed clone β_{C}^{ξ}) was first introduced in BCQ [FMP18], where the authors suggest the relative action scaling value of $\phi = 0.05$, which we adopt in this work. As in [FMP18], we update the state-conditional action perturbation of the action predicted by the probabilistic clone to maximize $Q_{\omega}(s, \tilde{a})$, with $\tilde{a} \sim \beta_{\mathrm{C}}^{\xi}$ by leveraging the deterministic policy gradient theorem [SLH+14]. Note, since the action sampled from the β-clone β_{C} is an *input* to the perturbation model ξ, and that this is the only source of stochasticity in the β_{C}^{ξ} policy, the optimization of ξ does not involve any reparametrization trick — in contrast with the optimization of β_{C} which does (*cf.* [KW14, RMW14]).

4.5.3 Proposal policies and value simplex

We now lay out the proposal policies ζ considered in the empirical study that follows, defined as the state-conditioned distribution from which the *next* action a' is sampled to bootstrap Q_{ω} with at the *next* state s'. Formally, the proposal policies ζ's satisfy the following schema:

$$a' \sim \zeta(\cdot|s') \tag{4.13}$$

and the form of Bellman's equation considered in this work is the one where the target policy is not the optimal policy like in Q-learning [Wat89, WD92], but the proposal policy ζ. In other words, we involve the variant of Bellman's equation that urges Q_{ω} to *evaluate* the proposal policy ζ, *i.e.* that makes Q_{ω} consistent with ζ. Consequently, employing such a recursive equation to design the temporal difference update rule — with which Q_{ω} is updated via stochastic gradient descent — will in effect make Q_{ω} approximate Q^{ζ}, hence $Q_{\omega} \approx Q^{\zeta}$. For completeness, the loss used to update the action-

value's parameter vector ω is the following:

$$\ell_\omega \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s),s'\sim\rho^\beta(\cdot)}\left[\left(Q_\omega(s,a) - \left(r(s,a,s') + \gamma\,\mathbb{E}_{a'\sim\zeta(\cdot|s')}\left[Q_{\omega'}(s',a')\right]\right)\right)^2\right] \tag{4.14}$$

Nevertheless, since the quadruple (s, a, r, s') (*abbrv.* "*SARS* transition") is always coming from the offline dataset $\mathcal{D}$ assumed to have been generated by the interactions of a behavior policy or baseline β, and is therefore distributed as such, we can only have $Q_\omega \approx Q^\zeta$ if (and only if) the offline dataset $\mathcal{D}$ was generated by an artificial agent following the policy ζ when interacting with the world $\mathcal{E}$. In other words, we can write the *intuitive* equivalence:

$$Q_\omega \approx Q^\zeta \iff \zeta \approx \beta \tag{4.15}$$

As such, the closer to β we model and train ζ to be, the more we can expect the learned action-value function approximator Q_ω to accurately *evaluate* the proposal distribution ζ, in which case Q_ω is also a good surrogate for Q^β. This scenario corresponds to the virtual absence of *distributional shift*, since there is little discrepancy between the distribution underlying the dataset, β, and the proposal policy ζ used to generate the actions Q_ω must evaluate. In the specific case where ζ coincides exactly with the offline behavior policy β generating the offline data, *i.e.* $\zeta = \beta$, the critic loss ℓ_ω laid out in EQ 4.14 is in effect equivalent to a SARSA update [RN94, Thr95, Sut96, vSvHWW09], which is an on-policy (and therefore effectively *online*) update for the learned action-value Q_ω — which, as we have previously established in EQ 4.15, then approximates Q^ζ. Indeed, in that scenario, we would have the states s and s' distributed as ρ^ζ, and the actions a and a' distributed as ζ, making the behavior and target policies coincide in an on-policy fashion, as follows:

$$\ell_\omega^{\text{SARSA}} \coloneqq \mathbb{E}_{(s,s')\sim\rho^\zeta,(a,a')\sim\zeta}\left[\left(Q_\omega(s,a) - \left(r(s,a,s') + \gamma\,Q_{\omega'}(s',a')\right)\right)^2\right] \tag{4.16}$$

In this scenario, since $\zeta = \beta$, we can equivalently write the exact same expression for $\ell_\omega^{\text{SARSA}}$ with β instead of ζ. A natural first candidate for our proposal policy ζ is therefore β (exactly, not an approximation), which can be achieved by leveraging the availability of the *next* action for each *SARS* transition in the offline dataset $\mathcal{D}$. In the context of this specific proposal policy strategy, which we name *"beta sarsa"*, we therefore in effect use *SARSA* transitions from $\mathcal{D}$. Despite the setting laid out in SECTION 4.3, we here make an exception and benefit from the extra sequential information about β provided by these next actions attached to each transition. Importantly, none of the other proposal policy strategies use any privileged information of this kind, and stick to using *SARS* transitions to learn Q_ω. Thus, in the *"beta sarsa"* strategy, the proposal policy is β, and the next action a' is coming directly from the transition sampled from the dataset $\mathcal{D}$. Using the operators we have introduced at

the beginning of SECTION 4.5, we can write:

$$\zeta := \beta = \mathcal{T}_{\text{EVAL}}[\beta] \quad \Longrightarrow \quad a' \sim \mathcal{T}_{\text{EVAL}}[\beta](\cdot|s') \tag{4.17}$$

"BETA SARSA"

When the offline dataset $\mathcal{D}$ only contains *SARS* transitions, we can still, albeit to a lesser extent, leverage β's by-design protection against distributional shift caused by out-of-distribution *next* actions in Q_ω by using a learned clone β_C of β, which we introduced in SECTION 4.5. We name the strategy employing β_C as proposal policy *"beta clone"*. Note, as a side-effect, we can expect this new approach to reduce the exposure of Q_ω to overfitting, compared to adopting the parameter-free approach of simply using the available *SARSA* transitions (especially if the offline dataset coverage is poor). We can therefore expect Q_ω to generalize better when using the proposal β_C than β, making it less likely to inject out-of-distribution actions in Q_ω — unless the dataset covers $\mathcal{S}\times\mathcal{A}$ well, in which case both strategies are equally capable. Avoiding action-value overfitting is especially critical in actor-critic methods since the actor π_θ, trained to be greedy with respect to Q_ω, tends to overfit itself on spurious maxima of the action-value. Overcoming this compounding effect from critic to actor is as crucial during training — provided π_θ is used in the proposal policy design — as it is crucial at evaluation time, in the case of *on-policy* evaluation (*cf.* SECTION 4.4.2 for a description of our experimental setting and evaluation methods adopted in this work).

$$\zeta := \mathcal{T}_{\text{EVAL}}[\beta_C] \quad \Longrightarrow \quad a' \sim \mathcal{T}_{\text{EVAL}}[\beta_C](\cdot|s') \tag{4.18}$$

"BETA CLONE"

By using either $\zeta = \beta$ or $\zeta = \beta_C \approx \beta$ to produce a' in EQ 4.14, the equivalence of EQ 4.15 yields $Q_\omega \approx Q^\beta$ in both cases. We illustrate this, albeit through an abstract lens, in the diagrams of FIGURE 4.5.1, where the values learned by the strategies "BETA SARSA" and "BETA CLONE" are depicted by concentric disks centered at Q^β, signifying that both are approximating this value in functional space — the greater diameter for "BETA CLONE" echoes the wider trust region of the Q_ω approximation, due to β_C being itself an estimate of β. As such, using proposal policies that cause Q_ω to be near Q^β on the value simplex depicted in FIGURE 4.5.1 ($\zeta = \beta$ or $\zeta = \beta_C$) ensures Q_ω will not be evaluated at out-of-distribution actions. These proposal strategies are therefore *safe* with regards to distributional shift in Q_ω.

The offline RL algorithm we set out to use as baseline from SECTION 4.4.4 onwards uses the actor's policy π_θ as proposal policy. We name this strategy *"theta"*, add Q^{π_θ} as a corner of the action-value

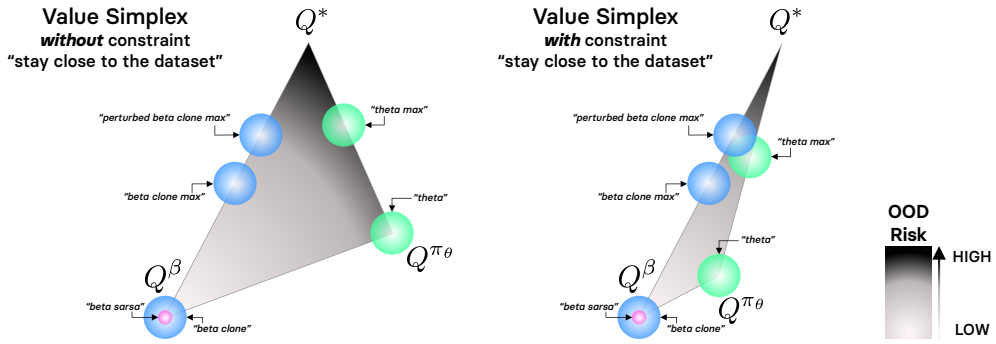


Figure 4.5.1: Abstract representation of the relative positioning of the action-values learned using the various proposal distributions laid out in SECTION 4.5 to generate the actions used to bootstrap Bellman's equation. These Q-values are depicted by disks over the simplex spanned by the optimal value Q^*, the value function perfectly evaluating the learned actor π_θ, Q^{π_θ}, and the exact value function Q^β associated with the policy underlying the offline dataset, β. The disk diameter roughly depicts how confident one can be about the placement of the various values associated with the tackled proposal policies (*cf.* SECTION 4.5) on the abstract simplex. Albeit only crudely estimating the actual geometry of the action-value simplex, this diagram can nevertheless help us categorize the different proposal distributions with respect to how they expose to agent and its value to *out-of-distribution* (OOD) actions *at training time*. We omit here the methods reminiscent of safe policy improvement (SPI) approaches as the associated proposal policies change their predictions based on an extra data-dependent condition being fulfilled, making them tedious to place on the simplex. Best seen in color: *pink* signifies that the proposal distribution is β — requiring access to *SARSA*-formatted transitions from the dataset, *blue* that the proposal distribution relies on an estimate of the β distribution, and *green* that only the actor π_θ is used to bootstrap.

simplex in FIGURE 4.5.1, and can similarly write:

$$\zeta := \mathcal{T}_{\text{EVAL}}\big[\pi_\theta\big] \quad \Longrightarrow \quad a' \sim \mathcal{T}_{\text{EVAL}}\big[\pi_\theta\big](\cdot|s') \tag{4.19}$$

"THETA"

Learning Q_ω with the loss $\ell_\omega^{\text{SARSA}}$ and $\zeta = \pi_\theta$ would yield $Q_\omega \approx Q^{\pi_\theta}$, as is commonplace in the online RL setting, where the *SARS* transitions originate either from π_θ (online, and on-policy), or an evolving mixture of previous iterates of π_θ (online, and off-policy with experience replay). By contrast, in *offline* RL, the *SARS* transitions are generated by the offline behavior policy β that has no ties with π_θ, and the dataset $\mathcal{D}$ produced by β remains frozen throughout the entirety of the learning process. Maintaining π_θ in the vicinity of β in some metric — as enforced in every single successful offline RL method reported in SECTION 4.4.1 — makes $\ell_\omega^{\text{SARSA}}$ using $\zeta = \beta$ coincide in analytical form with ℓ_ω using $\zeta = \pi_\theta$. In other words, optimizing the offline loss ℓ_ω with π_θ as proposal policy ζ while

constraining π_θ (and therefore by construction ζ) to be close to β (*i.e.* $\zeta \approx \beta$), we obtain, via the intuitive equivalence in EQ 4.15, $Q_\omega \approx Q^{\pi_\theta}$. Moreover, we have $Q_\omega \approx Q^\beta$ by transitivity, since π_θ is kept close to β according to some metric which has an effect on the *"closeness"* encoded here by the symbol "$\approx$" used as operator between action-value functions. We illustrate the effect of encouraging π_θ to be close to β in FIGURE 4.5.1 by representing the values Q^{π_θ} and Q^β (corresponding to the values learned using $\ell_\omega^{\text{SARSA}}$ with $\zeta = \pi_\theta$ and $\zeta = \beta$ or β_{C} respectively) closer to each other. As they get closer, the action-value simplex shrinks along the edge linking Q^{π_θ} to Q^β. Before shifting our attention to the third corner of the simplex depicted in FIGURE 4.5.1, the optimal action-value Q^*, note how forcing π_θ to be somewhat close to β to shield Q_ω from being evaluated at out-of-distribution actions (*black* gradient on the simplex of FIGURE 4.5.1) can have the *averse* effect of preventing Q_ω from ever reaching said optimal value Q^*. This undesirable consequence is depicted both in FIGURE 4.1.1 and FIGURE 4.5.1.

The diagrams of FIGURE 4.1.1 reminds us that the ultimate objective of generalized policy iteration (GPI) [SB98] is for Q_ω to converge to the *optimal* action-value Q^*, represented as a corner of the simplex in FIGURE 4.5.1. The canonical loss ℓ^* one uses to learn Q^* is derived from the optimal version of Bellman's equation — the one used in Q-learning [Wat89, WD92], and is defined as follows:

$$\ell_\omega^* \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s),s'\sim\rho^\beta(\cdot)}\left[\left(Q_\omega(s,a) - \left(r(s,a,s') + \gamma \max_{a'} Q_{\omega'}(s',a')\right)\right)^2\right] \tag{4.20}$$

By opting for an action-value Q_ω learned via Q-learning with ℓ_ω^*, instead of via SARSA updates with $\ell_\omega^{\text{SARSA}}$ — an adoption studied first in [CB95], then in [LJL+18] where the result of such an adoption was named an "actor-*expert*" algorithm — we align the signal returned by Q_ω with the identification of whether a given action a is the *best* action $a^* = \pi^*(s)$, rather than evaluating the proposal policy ζ used in EQ 4.16. As a result, due to the intertwined roles of the actor and critic (even in proposal policy strategies where π_θ is not involved in Q_ω's update, Q_ω is *always* used in π_θ's update), learning Q_ω via Q-learning will have a direct impact on π_θ, whose parameters will be updated to assign higher densities to actions that Q_ω, now estimating Q^*, believes are optimal. Provided the estimation $Q_\omega \approx Q^*$ is viable, this method has the clear advantage of *compartmentalizing* (containing and detaching from each other) Q_ω and π_θ, therefore preventing the compounding of errors (due to distributional shift and out-of-distribution actions, inherent to offline RL) in the alternating learning scheme between policy and value that characterizes GPI [SB98]. We could also write the last operand of EQ 4.20 as $\gamma\, Q_{\omega'}(s', \text{argmax}_{a'} Q_{\omega'}(s', a'))$, which has the added benefit of reminding us that the optimal policy π^* greedy with respect to Q^* is deterministic, since ℓ_ω^* coincides with ℓ_ω where $\zeta = \pi^*$. While the argmax operation is tractable in a reasonable compute time when the actions are discrete with a low number of dimensions, it is not a viable option *as is* when there is a plethora of discrete actions, or for

continuous action spaces. As such, the loss ℓ^*_ω is not always the best candidate to learn an estimate of the optimal value Q^*, and we reported the slew of works that designed alternatives to the raw argmax operation in these unviable scenarios in SECTION 4.2. We here opt for a simple stochastic sample-based *relaxation*, leveraging the operator $\mathcal{T}^{\omega,m}_{\text{MAX}}$ introduced earlier in EQ 4.4:

$$\underset{a'}{\text{argmax}}\, Q_{\omega'}(s', a') \approx \underset{a^i}{\text{argmax}} \left\{ Q_{\omega'}(s', a^i) \mid a^i \sim \pi(\cdot|s') \right\}_{i \in [1,m] \cap \mathbb{N}} = \mathcal{T}^{\omega',m}_{\text{MAX}}[\pi](\cdot|s') \tag{4.21}$$

where ω' is the parameter vector of the critic's target network introduced in SECTION 4.4.1, and π is a placeholder for a proposal distribution that the relaxation calls for, and for which we consider the following candidates: β_{C}, β^ξ_{C}, and π_θ. The relaxation proposed in EQ 4.21 therefore approximates the intractable loss ℓ^*_ω (which corresponds to ℓ_ω where $\zeta = \pi^*$) with the tractable loss ℓ_ω where $\zeta = \mathcal{T}^{\omega',m}_{\text{MAX}}[\pi]$, with $\pi \in \{\beta_{\text{C}}, \beta^\xi_{\text{C}}, \pi_\theta\}$ (*cf.* SECTION 4.5.2 for the definitions of β_{C} and β^ξ_{C}, the proposal policies derived from the offline dataset policy β). As such, in addition to the proposal policies ζ already introduced above, we also have the following ones, derived from β_{C}, β^ξ_{C}, and π_θ respectively:

$$\zeta := \mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta_{\text{C}}\right] \quad \Longrightarrow \quad a' \sim \mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta_{\text{C}}\right](\cdot|s') \tag{4.22}$$

"BETA CLONE MAX"

$$\zeta := \mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta^\xi_{\text{C}}\right] \quad \Longrightarrow \quad a' \sim \mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta^\xi_{\text{C}}\right](\cdot|s') \tag{4.23}$$

"PERTURBED BETA CLONE MAX"

$$\zeta := \mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta] \quad \Longrightarrow \quad a' \sim \mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta](\cdot|s') \tag{4.24}$$

"THETA MAX"

As for "BETA SARSA", "BETA CLONE", and "THETA", there is one colored disk depicted in FIGURE 4.5.1 for "BETA CLONE MAX", "PERTURBED BETA CLONE MAX", and "THETA MAX". The three latter are drawn closer to Q^* than the three former. In particular, "PERTURBED BETA CLONE MAX" is depicted closer to Q^* than "BETA CLONE MAX" since β^ξ_{C} is a clone β_{C} perturbed slightly to maximize Q_ω, and is therefore pushing Q_ω further towards the optimal action-value Q^*. Moreover, the value chosen for the hyper-parameter m can be used to modulate where these disks are located *a)* on the segment joining Q^β to Q^* for $\mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta_{\text{C}}\right]$ and $\mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta^\xi_{\text{C}}\right]$, and *b)* on the segment joining Q^{π_θ} to Q^* for $\mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta]$. Indeed, the proposal policies $\mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta_{\text{C}}\right]$, $\mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta^\xi_{\text{C}}\right]$, and $\mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta]$ are *a priori* expected to become better approximations of π^* when m is set to larger values. The quality of such approximation nevertheless strongly depends on the proposal distribution π introduced in EQ 4.21 for the relaxation of ℓ^*_ω, where $\pi \in \{\beta_{\text{C}}, \beta^\xi_{\text{C}}, \pi_\theta\}$. Hence, as m increases, we could draw the colored disks in FIGURE 4.5.1 — associated with the proposal distributions $\mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta_{\text{C}}\right]$, $\mathcal{T}^{\omega',m}_{\text{MAX}}\left[\beta^\xi_{\text{C}}\right]$, and $\mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta]$ — increasingly closer to Q^* (*cf.* APPENDIX 4.B for a sweep over several values of the hyper-parameter m,

showing a trade-off between performance and computational cost). All in all, tackling an offline RL task is a *balancing act*: we want to move Q^{π_θ} (the action-value consistent with the actor's policy π_θ) closer to Q^*, while not creating too much distance between Q^{π_θ} and Q^β to avoid unforgiving distributional shifts during training. What FIGURE 4.5.1 does not depict, in contrast with FIGURE 4.1.1, is what the diagram would look like for various qualities of dataset $\mathcal{D}$. In particular, if the dataset contains near-optimal data (*i.e.* $\beta \approx \pi^*$) the edge linking Q^β to Q^* would be considerably shorter, such that Q^β would almost overlap with Q^*. In other words, the objective of offline RL is to *shrink* this simplex until a *"sweet spot"* is reached. When the dataset contains optimal data, we want the simplex to shrink and collapse onto a single point ($Q^{\pi_\theta} = Q^\beta = Q^*$). When the dataset contains sub-optimal data, we want the simplex to reach a sweet spot that ought to manifest *before* all 3 corners collapse onto a single point, since $\beta \neq \pi^*$.

In an attempt to strike such balance, we lastly introduce proposal policies inspired from *Safe Policy Improvement* (SPI) [PCG16] (*cf.* SECTION 4.2). Instead of focusing only on a single edge among the two top edges of the simplex in FIGURE 4.5.1 (the edge linking Q^β to Q^*, and the one linking Q^β to Q^{π_θ}) these proposal policies would be located somewhere in between if they were depicted on the simplex of FIGURE 4.5.1 (not done for legibility reasons). We define these proposal policies, leveraging the operators $\mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\partial}_{\text{COND-EVAL}}$ and $\mathcal{T}^{\omega,\theta,m,\Gamma^\theta_\partial}_{\text{COND-MAX}}$ we introduced earlier in EQ 4.6 and EQ 4.8 respectively, as follows:

$$\zeta \coloneqq \mathcal{T}^{\omega',\theta,m,\Gamma^\theta_\partial}_{\text{COND-EVAL}}[\beta_{\text{C}}] \quad \Longrightarrow \quad a' \sim \mathcal{T}^{\omega',\theta,m,\Gamma^\theta_\partial}_{\text{COND-EVAL}}[\beta_{\text{C}}](\cdot|s') \tag{4.25}$$

"SPI BETA CLONE"

$$\zeta \coloneqq \mathcal{T}^{\omega',\theta,m,\Gamma^\theta_\partial}_{\text{COND-MAX}}[\beta_{\text{C}}] \quad \Longrightarrow \quad a' \sim \mathcal{T}^{\omega',\theta,m,\Gamma^\theta_\partial}_{\text{COND-MAX}}[\beta_{\text{C}}](\cdot|s') \tag{4.26}$$

"SPI BETA CLONE MAX"

$$\zeta \coloneqq \mathcal{T}^{\omega',\theta,m,\Gamma^\theta_\partial}_{\text{COND-MAX}}[\beta^\xi_{\text{C}}] \quad \Longrightarrow \quad a' \sim \mathcal{T}^{\omega',\theta,m,\Gamma^\theta_\partial}_{\text{COND-MAX}}[\beta^\xi_{\text{C}}](\cdot|s') \tag{4.27}$$

"SPI PERTURBED BETA CLONE MAX"

Since these are inherently adaptive, data-dependent convex combinations of previously introduced and discussed operators (*cf.* SECTION 4.5.1) we can expect the action value Q_ω learned with ℓ_ω and these hybrid proposal policies to be in the convex hull of the three corners of the simplex depicted in FIGURE 4.5.1, Q^*, Q^β, and Q^{π_θ}. Note, the condition Γ^θ_∂ involves the potential function ρ over $\mathcal{S} \times \mathcal{A}$ defined in EQ 4.10, and, perhaps more critically, depends on $\tilde{a}^\theta_{\text{MAX}} \sim \mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta](\cdot|s')$. Concretely, the *"safe"* proposal policy will act according to $\mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta]$ when $\tilde{a}^\theta_{\text{MAX}}$ is close to being distributed as β (*i.e.* π_θ is close to β), but will act according to $\tilde{\zeta} \in \{\mathcal{T}_{\text{EVAL}}[\beta_{\text{C}}], \mathcal{T}^{\omega',m}_{\text{MAX}}[\beta_{\text{C}}], \mathcal{T}^{\omega',m}_{\text{MAX}}[\beta^\xi_{\text{C}}]\}$ when $\tilde{a}^\theta_{\text{MAX}}$ does not seem to have been sampled from β (*i.e.* π_θ is far from β). In other words, $a' \sim \mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta]$ if ρ

believes $\tilde{a}^{\theta}_{\text{MAX}} \sim \beta$, and $a' \sim \tilde{\zeta} \in \{\mathcal{T}_{\text{EVAL}}[\beta_{\text{C}}], \mathcal{T}^{\omega',m}_{\text{MAX}}[\beta_{\text{C}}], \mathcal{T}^{\omega',m}_{\text{MAX}}[\beta^{\xi}_{\text{C}}]\}$, depending on the chosen strategy, when ρ believes $\tilde{a}^{\theta}_{\text{MAX}} \nsim \beta$. These three *safe* proposal policies have a direct grasp on whether the actor's policy is about to predict out-of-distribution actions, and can act on it by instead opting for a *next* action more likely to be in-distribution, by sampling from an alternate, *safer* proposal distribution derived from a estimated clone of the offline policy β. Making sure ρ's beliefs should be trusted is of independent interest, and its design comes with its own set of challenges. In the experiments reported in this work, we stick to the implementation of ρ reported in SECTION 4.5.1. The role of ρ could be filled by a myriad of density, novelty, or uncertainty estimators. Yet, their effectiveness and impact on learning dynamics and final performance is left out of the scope of this work.

All in all, we have introduced 9 proposal policies ζ to sample the next action a' from, in ℓ_ω: "BETA SARSA", "BETA CLONE", "THETA", "BETA CLONE MAX", "PERTURBED BETA CLONE MAX", "THETA MAX", "SPI BETA CLONE", "SPI BETA CLONE MAX", and finally "SPI PERTURBED BETA CLONE MAX".

Algorithmic changes. The pseudo-code of the algorithm we use to conduct the experiments reported in SECTION 4.5.4 coincide with the one laid out in ALGORITHM 5, except for ℓ_ω that we replace with the following critic loss:

$$\ell_\omega := \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s),s'\sim\rho^\beta(\cdot)}\left[\left(Q_\omega(s,a) - \left(r(s,a,s') + \gamma\,\mathbb{E}_{a'\sim\zeta(\cdot|s')}\left[Q_{\omega'}(s',a')\right]\right)\right)^2\right] \tag{4.28}$$

where the sole change from ALGORITHM 5 is colored in red.

We now report and discuss our experimental findings.

4.5.4 EXPERIMENTAL RESULTS

We rely on the experimental setting thoroughly described in SECTION 4.4.2 to carry out the empirical investigation laid out here.

Notably, albeit perhaps naive in appearance, the results reported in FIGURE 4.5.2 show that proposal policies as simple as "BETA SARSA" perform strikingly well compared to considerably more sophisticated, safe, and adaptive methods (*e.g.* "SPI PERTURBED BETA CLONE MAX") in most environments and datasets — although one need access to *SARSA*-formatted transition in the offline dataset $\mathcal{D}$ to be able to leverage this proposal distribution strategy. Indeed, "BETA SARSA" only performs worse than the baseline strategy "THETA" in a *single* environment (equivalently, in a single dataset), in the top-right sub-plot, and does so considerably. Nevertheless, in the *14* other environment-dataset couples, the empirical performance of the "BETA SARSA" proposal distribution strategy is on par with the baseline "THETA", and even often outperforms it, despite not involving the actor's policy π_θ in

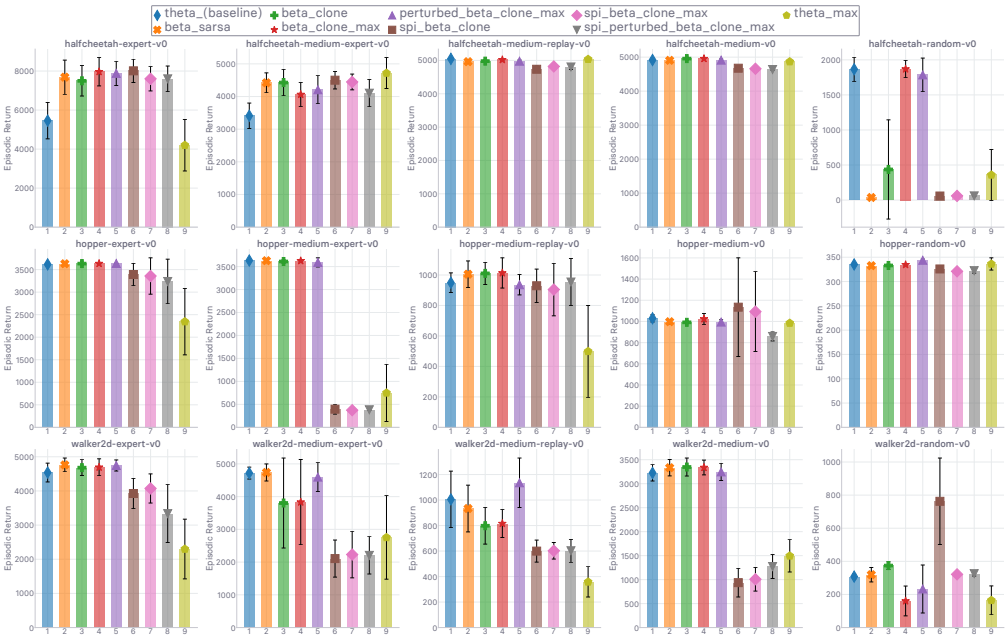


Figure 4.5.2: Empirical comparison of how the proposal distributions introduced in SECTION 4.6.2 impact the final *evaluation* performance of the base algorithm considered. Everything except the proposal policy ζ used to sample the *next* action from is identical. The adjective *final* corresponds to a runtime 12 hours. Best seen in color.

its policy evaluation update of Q_ω via ℓ_ω. On the opposite side of the spectrum in FIGURE 4.5.2, the proposal distribution "THETA MAX" shows the *poorest* results in almost every datasets, which is *in fine* not too surprising considering the higher chances of out-of-distribution actions in the Bellman backup it incurs. As we laid out earlier in SECTION 4.6.2, involving a maximization operator over estimated action-values $Q_\omega(s, a)$ (for a given pair $(s, a) \in \mathcal{S} \times \mathcal{A}$) will have the harmful effect of compounding onto the overestimation bias Q_ω is prone to [TS93], locking the actor's policy onto arbitrarily overestimated action values when bootstrapping Q_ω via Bellman's equation. Such effect is exacerbated in the *offline* RL setting [LKTF20], although this *distributional shift* is still present when further interactions are allowed, in the online RL scenario [FKSL19]. The visual aid depicted in FIGURE 4.5.1 illustrates the faced challenge well: involving a maximization operation to move Q_ω closer to Q^*, while making sure the proposal policy that generates the *next* action to bootstrap with is not too far off being distributed as the offline policy β, underlying the offline dataset $\mathcal{D}$. In addition, we observe in FIGURE 4.5.2 that the proposal distribution "THETA MAX" is significantly *weaker* (in fact, at its *weakest*) when the *coverage* of the offline dataset $\mathcal{D}$ is expected to be poor. Such a criterion is more likely to be satisfied on *"expert"* datasets (*left*-most plots in FIGURE 4.5.2), where the underlying β is expected to have lower entropy than for *"random"* datasets (*right*-most plots). Despite this

trend, "THETA MAX" can still display high return in some isolated cases, as Q_ω *can* suffer from said overestimation bias, yet not predictably or controllably so, and therefore not consistently across the benchmark. As such, attempting to move Q_ω closer to Q^* directly from Q^{π_θ} is objectively not the safest route for the *offline* RL practitioner to take.

When it comes down to the *clone* group ("BETA CLONE", "BETA CLONE MAX", and "PERTURBED BETA CLONE MAX"), and the *SPI* group ("SPI BETA CLONE", "SPI BETA CLONE MAX", and "SPI PERTURBED BETA CLONE MAX"), both arguably are a mixed bag. The proposal distributions of the *clone* group all yield similar returns, *i.e.* none of the methods within the group outperforms the two other consistently across the benchmark. Interestingly, these methods only rarely beat the "BETA SARSA" heuristic, which despite needing *SARSA*-formatted transitions, does not rely on an additional state-conditional generative model of the offline dataset $\mathcal{D}$ — β_C or its perturbed version β_C^ξ — effectively *cloning* β. Such a trade-off can, in practice, be addressed differently depending on how complex the data distribution β is (therefore more difficult to estimate accurately via behavioral cloning [Pom89, Pom90, RBS07, Bag15]), and how feasible it is for the practitioner to gather the dataset to be used offline such that the collected transitions are *sequentially* ordered in *connex* trajectories. The proposal distributions of the *SPI* group — "SPI BETA CLONE", "SPI BETA CLONE MAX", and "SPI PERTURBED BETA CLONE MAX" — are to a certain extent *hybrids* between *a)* "THETA MAX" and *b)* "BETA CLONE", "BETA CLONE MAX", and "PERTURBED BETA CLONE MAX", respectively. The quality of these methods depends not only on the clone β_C or the perturbed clone β_C^ξ (adding respectively *one* and *two* extra function approximators to the global neural architecture), but also on the quality of the density (or novelty, uncertainty, *cf.* SECTION 4.5.1) estimator ρ, which determines from which policy the next action will be sampled. While the proposal strategies for ζ belonging to the *SPI* group perform better than "THETA MAX" on the *"expert"* datasets (left-most column of plots in the grid of FIGURE 4.5.2) while performing worse than the *clone* methods, this pattern is not maintained across every dataset. In fact, a given strategy from the *SPI* group often underperforms both of the strategies it *mixes i.e.* "THETA MAX" and either one of the options from the *clone* group (listed out just above in *b)*) depending on the used variant. Even if these theoretically-safer strategies *can* outperform the others in some environment-dataset scenarios (*e.g.* bottom-right corner sub-plot in FIGURE 4.5.2) it appears that overall it is not worth spending the extra resources to implement them (including the time it takes to tune these extra moving pieces and knobs). All in all, what FIGURE 4.5.2 shows is that involving β in the proposal distribution — be it by using it directly provided *next* actions are available through the offline dataset $\mathcal{D}$ or by *cloning* it and using the clone instead — that is used to generate the bootstrap action in the temporal-difference update *should* be the preferred route to train the action-value Q_ω in offline RL.

Among the proposal distributions that we have formalized in a unified framework and empirically

evaluated in the section, some have a *counterpart* in prior offline RL algorithms introduced in recent years — whether they are presented as primary or secondary contributions in their respective encapsulating works. Note, we consider the policy evaluation step in isolation from the GPI cycle it is embedded in (*cf.* FIGURE 4.1.1). The proposal distribution $\zeta \coloneqq \pi_\theta$ set in "THETA" coincides with the usual *SARSA* update [RN94, Thr95, Sut96, vSvHWW09] (as opposed to Q-learning update [Wat89, WD92]) adopted by most actor-critic architectures whose action-value's loss is based on the *evaluation* version of Bellman's equation [SLH+14, LHP+16, BMHB+18, HZAL18, KFTL19, WTN19, KZTL20, SSB+20, WNZ+20, NDGL20]. By bootstrapping Q_ω with $a' \sim \mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta](\cdot|s')$, $\forall s' \in \mathcal{S}$, the proposal strategy "THETA MAX" sets out to *emulate* a Q-learning update to urge the actor π_θ, by design greedy with respect to the critic Q_ω, to increase the probability density of actions that Q^* views as optimal (as opposed to the usual Q^{π_θ} critic in *SARSA*-like off-policy online actor-critic architectures [LJL+18]). Such desideratum has been sought after in a slew of works released concurrently, among which *Amortized* Q-learning (AQL) [VdWWFMM18] draws the closest resemblance, albeit being an *online* off-policy method (*cf.* SECTION 4.2 for a rundown of said concurrent works by differ by how they *relax* the intractable maximization operation over $\mathcal{A}$ in the Q-learning version of Bellman's equation). Later, "THETA MAX" has been used in the BEAR-QL method [KFTL19] in an *offline* RL context. By replacing π_θ in the latter by a clone β_{C} of the distribution underlying the offline dataset $\mathcal{D}$ such that $a' \sim \mathcal{T}^{\omega',m}_{\text{MAX}}[\beta_{\text{C}}](\cdot|s')$, $\forall s' \in \mathcal{S}$, we obtain the "BETA CLONE MAX" proposal strategy, which one can find as a standalone contribution in the EMaQ method [GSG20]. Further, by replacing π_θ by a *perturbed* clone β^ξ_{C}, such that $a' \sim \mathcal{T}^{\omega',m}_{\text{MAX}}[\beta^\xi_{\text{C}}](\cdot|s')$, $\forall s' \in \mathcal{S}$, we obtain the "PERTURBED BETA CLONE MAX" proposal strategy, which is part of the BCQ method [FMP18] (the perturbed clone β^ξ_{C} is effectively the actor in BCQ, such that $\pi_\theta \coloneqq \beta^\xi_{\text{C}}$ at evaluation time). Finally, BRPO [SCO+20] can be cast as an instance of "SPI BETA CLONE".

So as to complement our analysis on how to better carry out policy evaluation in *offline* RL, we conduct two additional sets of experiments. First, we investigate the effect of Baird's advantage-learning [Bai93, Bai99] as re-adapted to modern objective designs in [BOG+15]. The purpose of Baird's advantage-learning is to *increase* the gap in action-value between optimal and sub-optimal actions, so that the greedy actor is less likely to select sub-optimal action because of misestimation or simply numerical precision. Notably, in offline RL, [WTN19] and [KZTL20] undertook to increase the gap between actions that are close to being distributed as the offline distribution β and actions that seem not to be. Despite being motivated by different desiderata — avoiding sub-optimal action for Baird's advantage-learning, *i.e.* $a \nsim \pi^*(\cdot|s)$; avoiding out-of-distribution actions for works like [WTN19] and [KZTL20] — both Baird's advantage-learning and *Q-constrained* offline RL Q_ω objectives (*e.g.* [WTN19], [KZTL20]) are similar in spirit. Concretely, we add the α-scaled *advantage* $\alpha A^{\pi_\theta}_\omega(s,a)$ to $Q_\omega(s,a)$'s target in the temporal-difference objective ℓ_ω that updates Q_ω over the offline dataset

$\mathcal{D}$ (*i.e.* $s \sim \rho^\beta(\cdot), a \sim \beta(\cdot|s), s' \sim \rho^\beta(\cdot)$). We define the advantage $A_\omega^{\pi_\theta}$ over $\mathcal{D}$ as $A_\omega^{\pi_\theta}(s,a) \coloneqq Q_\omega(s,a) - \mathbb{E}_{\tilde{a}\sim\pi_\theta}[Q_\omega(s,\tilde{a})]$, where the expectation is estimated with the usual unbiased empirical mean. Note, $A_\omega^{\pi_\theta}$ takes values in $\mathbb{R}$, while $\alpha > 0$. Since tuning the scale of bonuses or penalties added to Q_ω's target (*e.g.* [WTN19]) or Q_ω's objective (*e.g.* [KZTL20]) has proved tremendously tedious due to the *stiffness* (*cf.* definition in Section 4.3) of such hyper-parameter, we conducted a grid search over values separated by equal spaces and ranging from 0.1 to 0.9. The latter upper bound is set to such value since that is the highest, still theoretically-principled, value that can be assumed by Baird's advantage-learning scaling coefficient according to [BOG+15] — although it has later been argued otherwise in [LSW19]. Note, our advantage-learning bonus is *only* applied on points from the offline dataset $\mathcal{D}$, as our add-on *concretely* changes ℓ_ω as described in eq. 4.14 into the following objective:

$$\ell_\omega^{\text{AL}} \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s),s'\sim\rho^\beta(\cdot)}\left[\left(Q_\omega(s,a) - \left(r(s,a,s') + \gamma\,\mathbb{E}_{a'\sim\zeta(\cdot|s')}\left[Q_{\omega'}(s',a') + \alpha A_\omega^{\pi_\theta}(s,a)\right]\right)\right)^2\right] \tag{4.29}$$

We study how using the loss defined in eq. 4.29 affects final performance over the grid of α values reported above, and report our empirical findings in Appendix 4.A, Figure 4.A.1. As expected, we observe that there is little to gain from such an add-on, and more perhaps more importantly that there is a lot to lose judging by how *stiff* the new problem is with respect to the newly introduced hyper-parameter α. Consequently, we do not use this revisited version of Baird's advantage-learning in *any* of our experiments except the ablation we have carried out in Appendix 4.A, Figure 4.A.1.

Finally, we investigate the impact of the hyper-parameter m on the agent's performance, where m is involved in the operator $\mathcal{T}_{\text{Max}}^{\omega,m}$ introduced in Section 4.5.1 to be used in the design of the proposal distributions $\mathcal{T}_{\text{Max}}^{\omega',m}[\beta_{\text{C}}]$, $\mathcal{T}_{\text{Max}}^{\omega',m}[\beta_{\text{C}}^\xi]$, and $\mathcal{T}_{\text{Max}}^{\omega',m}[\pi_\theta]$ — "beta clone max", "perturbed beta clone max", and "theta max", respectively. Concretely, m is the number of times we sample actions before selecting the one with the highest value in said operators. In essence, m controls the degree of interpolation with Q^*, as discussed earlier in Section 4.5.3. We report our empirical findings in Appendix 4.B, Figures 4.B.1, 4.B.2, and 4.B.3 respectively.

In short, these figures show that while the "beta clone max" and "perturbed beta clone max" proposal distributions are fairly resilient (and opposed to stiff) to changes in the value of m, "theta max" often displays significant gaps in performance between distinct values of m, in line with our previous discussions about the "theta max" proposal being far more exposed to out-of-distribution actions than "beta clone max" and "perturbed beta clone max". Increasing m increases the chance of involving an arbitrarily overestimated Q_ω value to the set of m values the operator $\mathcal{T}_{\text{Max}}^{\omega,m}$ takes the argmax over, which explains the greater spread in performance for the proposal that does

not involve a mechanism to ensure the actor's policy π_θ remains close to β. As a final note that will hold for the remainder of this work, we did not dedicate a computational budget to design a *best-in-class* behavioral cloning architecture that would learn a better state-conditional generative model of β, and encourage future research endeavors to pursue that route.

4.6 The Generalized Importance-Weighted Regression framework

4.6.1 Generalized constrained policy improvement

In Section 4.5, we undertook the design of several proposal policies ζ to sample the *next* action from in the temporal-difference learning update [Sut84, Sut88, SMSM99] of the critic Q_ω, giving rise to as many variants of Bellman's equation. We analyzed and reported the impact of each of these on the agent's learning dynamics and final asymptotic performance in Section 4.5.4, by changing the proposal policy used in the policy evaluation strategy while keeping the policy improvement subroutine of each policy iteration step identical and fixed. In this section, we do the exact opposite: we vary the proposal distribution used in the new actor update method we introduce, while keeping the policy evaluation method set to the standard *SARSA* actor-critic update ubiquitous in both online and offline RL, "theta". Note, the designed framework nevertheless allows for the use of any proposal strategy introduced in Section 4.5 in policy evaluation while varying the proposal used in the subsequent policy improvement step. As such, the total number of pairs of policy evaluation and improvement strategies evolves *quadratically* as we introduce more evaluation or improvement methods. To limit the number of experiments, we fixed (as just mentioned), the proposal strategy to "theta" on the policy evaluation side, thus exploring ***nine*** different methods. Since some of the possible pairings allow for synergies stronger than others, we claim such study to be an avenue of interest for future work.

We now exhibit how we involve the proposal policies in policy improvement, by *1)* laying out the constrained optimization problem we set out our agent's policy to be a solution, and *2)* deriving a tractable iterative procedure from the designed constrained optimization problem by leveraging a dual formulation, several weak relaxations, where we derive everything *from first principles*. We show that the laid out derivation generalizes the past derivations it takes inspiration from, and therefore *subsumes* the policy improvement formulations of several popular offline and online RL algorithms.

Our derivation is inspired from the derivations of the *almost identical* constrained optimization problem carried out in several *KL-control* works, which we divide in three waves based on when the respective works appeared: it was first reported in REPS [PMA10], RWR [PS07, KOP10], and LAWER

[NP08], then later in AWR [PKZL19], whose reminiscent elements appear in TRPO's derivation as well [SLM+15], to finally reemerge later in the concurrent works CRR [WNZ+20] and AWAC [NDGL20]. Despite sharing most of the mechanisms overlapping in each of these waves, our derivation involves a slightly altered constraint in the initial constrained optimization problem formulation. As such, we solve the said problem analytically from the start, to arrive at a tractable solution taking into account our change in the original formulation. The proposed problem alteration and its provably-adapted and computationally-tractable solution provide a *generalized* framework that allows the practitioner to involve additional constraints to the original *KL-control*-based constrained optimization problem using any proposal distribution introduced and discussed in SECTION 4.5. While REPS [PMA10] and RWR [PS07, KOP10], maximize the expected return $J(\pi)$, LAWER [NP08], CPI [KL02], TRPO [SLM+15], MARWIL [WXH+18], MPO [AST+18], AWR [PKZL19], CRR [WNZ+20] and AWAC [NDGL20] maximize the expected improvement $\eta(\pi)$. In other words, while former group intends to learn policies that maximize the *action-value* from the start state, the latter group cares about the maximization of the *advantage* from the start state: $\eta(\pi) \coloneqq \mathbb{E}_{s\sim\rho^\pi(\cdot),a\sim\pi(\cdot|s)}[A^\pi(s,a)]$.

Nevertheless, since we work under the *offline* RL setting, we only have access to states s coming from the offline dataset $\mathcal{D}$, *i.e.* distributed as $s \sim \rho^\beta(\cdot)$. We therefore define a *surrogate* objective $\tilde{\eta}^\beta(\pi)$ that, by contrast with $\eta(\pi)$, we *can* evaluate in the offline setting: $\tilde{\eta}^\beta(\pi) \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\pi(\cdot|s)}[A^\pi(s,a)]$. The severity of this relaxation depends on how well the state visitation distribution $\rho^\pi(\cdot)$, displayed by π, matches the one observed in the offline dataset, *i.e.* $\rho^\beta(\cdot)$. As such, if π and β lead the agent to the same states such that $\rho^\pi \approx \rho^\beta$, then $\tilde{\eta}^\beta(\pi)$ approximates $\eta(\pi)$ well. The relaxation is then *mild*. Crucially, since we often encourage the learned policy π to remain close to β in offline RL to avoid *out-of-distribution actions* in Q_ω, then the approximation $\tilde{\eta}^\beta(\pi) \approx \eta(\pi)$ is even more likely to be satisfied in the offline setting.

We tie the new iterate of the actor's policy $\pi_{\theta^{\text{new}}}$ to the previous one, $\pi_{\theta^{\text{old}}}$, via the constrained optimization problem that follows (*cf.* SECTION 4.3 for a reminder of how we denote either direction of the KL divergence in this work):

$$\pi_{\theta^{\text{new}}} \coloneqq \underset{\pi\in\mathcal{P}(\mathcal{A})^{\mathcal{S}}}{\operatorname{argmax}} \quad \tilde{\eta}^\beta(\pi) \tag{4.30}$$

$$\text{s.t.} \quad (\forall s \in \mathcal{S}) \quad D_{\overleftarrow{\text{KL}}}^{\zeta}[\pi](s) \leq \delta \tag{4.31}$$

$$(\forall s \in \mathcal{S}) \quad \int_{a\in\mathcal{A}} \pi(a|s)\, da = 1 \tag{4.32}$$

Similarly to how we could not evaluate $\eta(\pi)$ and had to use its relaxation $\tilde{\eta}^\beta(\pi)$ using states from $\mathcal{D}$ as a surrogate objective, we also apply an identical relaxation to the equality constraint. As such, "$(\forall s \in$

$\mathcal{S}$), $D^{\zeta}_{\overleftarrow{\mathrm{KL}}}[\pi](s) \leq \delta$" becomes "$\mathbb{E}_{s\sim\rho^{\beta}(\cdot)}\left[D^{\zeta}_{\overleftarrow{\mathrm{KL}}}[\pi](s)\right] \leq \delta$". Here, instead of attempting to enforce the equality constraint over the entirety of $\mathcal{S}$, we restrict the constraint's field of view to $\mathcal{D}$, the only subspace over which we *can* enforce it. Lastly, we relax the theoretical placeholder of the advantage A^{π} with $A^{\pi_{\theta^{\text{old}}}}_{\omega}$, defined as $A^{\pi_{\theta^{\text{old}}}}_{\omega}(s,a) \coloneqq Q_{\omega}(s,a) - \mathbb{E}_{\tilde{a}\sim\pi_{\theta^{\text{old}}}}[Q_{\omega}(s,\tilde{a})]$, where the expectation is estimated with the usual unbiased empirical mean. The equality constraint — urging π to describe a probability distribution over $\mathcal{A}$, $\forall s \in \mathcal{S}$ — can not be relaxed as we need this property to be distilled into the learned π wholly. After applying all these relaxations, the constrained optimization problem we set out to solve is the following:

$$\pi_{\theta^{\text{new}}} \coloneqq \underset{\pi\in\mathcal{P}(\mathcal{A})^{\mathcal{S}}}{\operatorname{argmax}} \quad \mathbb{E}_{s\sim\rho^{\beta}(\cdot),a\sim\pi(\cdot|s)}\left[A^{\pi_{\theta^{\text{old}}}}_{\omega}(s,a)\right] \tag{4.33}$$

$$\text{s.t.} \quad \mathbb{E}_{s\sim\rho^{\beta}(\cdot)}\left[D^{\zeta}_{\overleftarrow{\mathrm{KL}}}[\pi](s)\right] \leq \delta \tag{4.34}$$

$$(\forall s \in \mathcal{S}) \quad \int_{a\in\mathcal{A}} \pi(a|s)\, da = 1 \tag{4.35}$$

Via *Lagrange-Duality*, we define the following Lagrangian from the relaxed objective laid out in EQ 4.33, subjected to both *a)* the inequality constraint in EQ 4.34 encouraging the learned policy π to be close in reverse KL to the proposal policy ζ, and *b)* the equality constraint in EQ 4.35 ensuring π defines a proper conditional probability distribution. In particular, to deal with the inequality constraint (*cf.* EQ 4.34), we carry out the derivations under the KKT conditions:

$$\begin{aligned}\mathcal{L}(\pi,\lambda_{\mathrm{KL}},\lambda) \coloneqq\ & \mathbb{E}_{s\sim\rho^{\beta}(\cdot),a\sim\pi(\cdot|s)}\left[A^{\pi_{\theta^{\text{old}}}}_{\omega}(s,a)\right] \\ & - \lambda_{\mathrm{KL}}\left[\mathbb{E}_{s\sim\rho^{\beta}(\cdot)}\left[D^{\zeta}_{\overleftarrow{\mathrm{KL}}}[\pi](s)\right] - \delta\right] - \int_{s\in\mathcal{S}} \lambda(s)\left[\int_{a\in\mathcal{A}} \pi(a|s)\, da - 1\right] ds\end{aligned} \tag{4.36}$$

where $\lambda_{\mathrm{KL}} \in (0,\infty)$ is the Lagrange multiplier associated with the *KL*-based inequality constraint (*cf.* EQ 4.34), and $\lambda : \mathcal{S} \to (0,\infty)$ is a function that, for all $s \in \mathcal{S}$, returns a Lagrange multiplier for *each* equality constraint (*cf.* EQ 4.35). By expanding the expectations (and KL divergence) into integrals in EQ 4.36, we trivially obtain the following:

$$\begin{aligned}\mathcal{L}(\pi,\lambda_{\mathrm{KL}},\lambda) \coloneqq\ & \int_{s\in\mathcal{S}} \rho^{\beta}(s) \int_{a\in\mathcal{A}} \pi(a|s)\, A^{\pi_{\theta^{\text{old}}}}_{\omega}(s,a) \\ & - \lambda_{\mathrm{KL}}\left[\int_{s\in\mathcal{S}} \rho^{\beta}(s) \int_{a\in\mathcal{A}} \pi(a|s)\left(\log\pi(a|s) - \log\zeta(a|s)\right) - \delta\right] \\ & - \int_{s\in\mathcal{S}} \lambda(s)\left[\int_{a\in\mathcal{A}} \pi(a|s)\, da - 1\right] ds\end{aligned} \tag{4.37}$$

Then, by taking the first and second derivatives of $\mathcal{L}(\pi,\lambda_{\mathrm{KL}},\lambda)$ as expressed in EQ 4.37 with respect

to $\pi(a|s)$, we get:

$$\frac{\partial \mathcal{L}(\pi, \lambda_{\text{KL}}, \lambda)}{\partial \pi(a|s)} = \rho^{\beta}(s)\, A_{\omega}^{\pi_{\theta^{\text{old}}}}(s,a) - \lambda_{\text{KL}} \rho^{\beta}(s) \big(\log \pi(a|s) + 1 - \log \zeta(a|s) \big) - \lambda(s) \tag{4.38}$$

$$= B - \lambda_{\text{KL}} \rho^{\beta}(s) \log \pi(a|s) \tag{4.39}$$

with $B \coloneqq \rho^{\beta}(s) \left(A_{\omega}^{\pi_{\theta^{\text{old}}}}(s,a) - \lambda_{\text{KL}} \left(1 - \log \zeta(a|s)\right) \right) - \lambda(s)$. It follows that:

$$\frac{\partial^2 \mathcal{L}(\pi, \lambda_{\text{KL}}, \lambda)}{\partial \pi(a|s)^2} = -\frac{\lambda_{\text{KL}} \rho^{\beta}(s)}{\pi(a|s)} \leq 0 \quad \implies \textit{critical points} \text{ of } \mathcal{L}(\pi, \lambda_{\text{KL}}, \lambda) \text{ w.r.t. } \pi(a|s) \text{ are } \textit{maxima}. \tag{4.40}$$

Since $\mathcal{L}(\pi, \lambda_{\text{KL}}, \lambda)$ is *concave* with respect to $\pi(a|s)$ over the studied spaces, we look for the argmax of the objective (*cf.* EQ 4.33) seeking a critical point of the Lagrangian $\mathcal{L}(\pi, \lambda_{\text{KL}}, \lambda)$ along the π dimension, which we name π^*:

$$\left.\frac{\partial \mathcal{L}(\pi, \lambda_{\text{KL}}, \lambda)}{\partial \pi(a|s)}\right|_{\pi^*} = 0 \iff B - \lambda_{\text{KL}} \rho^{\beta}(s) \log \pi^*(a|s) = 0 \iff \log \pi^*(a|s) = \frac{B}{\lambda_{\text{KL}} \rho^{\beta}(s)} \tag{4.41}$$

Hence,

$$\pi^*(a|s) = C(s) \zeta(a|s) \exp\left(\frac{1}{\lambda_{\text{KL}}} A_{\omega}^{\pi_{\theta^{\text{old}}}}(s,a) \right) = C(s) \varphi(a|s) \tag{4.42}$$

$$\text{with} \quad C(s) \coloneqq \exp\left(-\left[1 + \frac{\lambda(s)}{\lambda_{\text{KL}} \rho^{\beta}(s)} \right] \right) \tag{4.43}$$

$$\text{and} \quad \varphi(a|s) \coloneqq \zeta(a|s) \exp\left(\frac{1}{\lambda_{\text{KL}}} A_{\omega}^{\pi_{\theta^{\text{old}}}}(s,a) \right) \tag{4.44}$$

where φ is the *unnormalized* advantage-weighted counterpart of ζ. φ is *not* a PDF, and will need to be normalized to be one. We do so using the equality constraint (*cf.* EQ 4.35) encoding our desideratum for π to be a probability distribution, which *a fortiori* naturally also applies to the point π^* maximizing $\mathcal{L}(\pi, \lambda_{\text{KL}}, \lambda)$:

$$(\forall s \in \mathcal{S}) \quad \int_{a \in \mathcal{A}} \pi^*(a|s)\, da = 1 \iff C(s) \int_{a \in \mathcal{A}} \varphi(a|s)\, da = 1 \iff C(s) = \frac{1}{\varphi(s)} \tag{4.45}$$

where $\varphi(s) \coloneqq \int_{a \in \mathcal{A}} \varphi(a|s)\, da$ is the Bayesian evidence, or partition function. As such, for $\pi^*(a|s) = C(s)\varphi(a|s)$ to define a PDF, we need $C(s)$ to satisfy $C(s) = 1 \,/\, \int_{a \in \mathcal{A}} \left[\zeta(a|s) \exp\left(A_{\omega}^{\pi_{\theta^{\text{old}}}}(s,a) / \lambda_{\text{KL}} \right) \right] da$.

Hence, π^* verifying:

$$\pi^*(a|s) = \frac{1}{\varphi(s)}\zeta(a|s)\exp\left(\frac{1}{\lambda_{\text{KL}}}A_\omega^{\pi_{\theta^{\text{old}}}}(s,a)\right) \tag{4.46}$$

defines a PDF since $\int_{a\in\mathcal{A}}\pi^*(a|s)\,da = 1$. *In fine*, such π^* is the *normalized* advantage-weighted counterpart of the proposal policy ζ — the policy ζ being the *trajectory distribution* in the *KL*-control literature, *e.g.* in [PS07, KOP10, PMA10, Neu11]). As such, we can refer to π^* as defined in EQ 4.46 as the *advantage-weighted proposal policy*. To disambiguate the notations, the advantage-weighted counterpart of the proposal policy ζ will be denoted as ζ_{IW}, *i.e.* $\zeta_{\text{IW}} \coloneqq \pi^*$ (the acronym "IW" standing for *i*mportance-*w*eighted, where *"importance"* here plays the role of universal, unifying placeholder for either *reward* or *advantage* depending on the considered method). As in all the previous work cited in this section for either reporting or building on the derivation of the present derivation (or a variant thereof) we stick to the traditional E-M scheme. Constructing ζ_{IW}, whose assembly procedure is described in EQ 4.46, is nevertheless tedious since computing the evidence $\varphi(s)$ in EQ 4.46 requires an inordinate amount of compute to estimate exactly (*cf.* Bayesian machine learning, energy-based models in particular, assembling Boltzmann distributions in a similar fashion).

4.6.2 Projection options for distributional shift mitigation

Yet, instead of trying to relax said evidence or find a more computationally affordable surrogate, we treat the intractable analytical solution ζ_{IW} (*cf.* EQ 4.46) as an *input* in a subsequent, distinct, unconstrained optimization problem (in line with the E-M procedural paradigm). Said optimization problem is defined as follows:

$$\theta \coloneqq \operatorname*{argmin}_{\theta\in\Theta} \mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[\Delta\big(\pi_\theta(\cdot|s),\zeta_{\text{IW}}(\cdot|s)\big)\Big] \tag{4.47}$$

with Δ being a measure between probability distributions.

In this subsequent problem, we set out to find a tractable decision-making rule by directly *projecting* the intractable advantage-weighted proposal policy ζ_{IW} onto the manifold of parametric policies $\{\pi_\theta \mid \theta \in \Theta\}$ we are able to estimate empirically. By construction, we can therefore hope to afford to compute the solution to this problem. We opt for the KL divergence as our choice of measure Δ to perform such projection. Since this measure is asymmetric, we have two options: we can either perform an I-projection (reverse, exclusive KL), or a M-projection (forward, inclusive KL), whose respective advantages and drawbacks are discussed in detail in the books of MacKay [Mac03], Bishop [Bis06], and Murphy [Mur12]. Consider the projection in KL divergence of the target distribution p onto the set of parametric distributions in which we look for q_θ, where $\theta \in \Theta$. In short, an $\mathcal{M}$-projection (*"$\mathcal{M}$"*

for *M*oment) will have the effect of making q_θ *cover* the modes of p (*"mode-covering"*), caring much about *not* assigning zero density wherever p has non-zero probability (*"zero-avoiding"*), yet not caring much about wrongly assigning non-zero density outside the support of p. Conversely, an I-projection (*"I"* for *I*nformation) will have the effect of making q_θ *seek* the modes of p (*"mode-seeking"*), caring much about *not* assigning *non*-zero density wherever p has zero probability (*"non-zero-avoiding"*, by symmetry), yet not caring much about assigning zero density inside the support of p, missing areas where p has non-zero probability. Said differently, the ultimate priority of an M-projection is to *not miss anything inside* the support of p, while the ultimate priority of an I-projection is to *miss everything outside* the support of p. As such, it is easy to see how using an I-projection (reverse KL) on a target trajectory distribution would have the indirect effect of learning *"cost-averse"* policies, while an M-projection (forward KL) would make policies *"reward-chasing"*. In [GGT15], Section 3, the authors give four reasons as to why using an M-projection as optimization objective might be beneficial. Their third reason posits that the projection resulting from a forward KL objective tends to display a higher entropy than the target distribution, which makes the projected proposal policy a good candidate for importance sampling, in various sub-areas (*e.g.* Monte-Carlo estimation in [GGT15]). This claim is supported by [Mac03], defending that the reverse KL, or I-projection, conversely does *not* yield proposal distributions suited for importance sampling. Besides, higher-entropy policies are naturally equipped with dithering capabilities to trade off with their greedy incentive to maximize Q_ω, enabling them to be reasonably proficient at exploring their environment without needing extra dithering mechanisms.

In offline RL however, high-entropy policies are more likely to evaluate the critic's value Q_ω at out-of-distribution actions *a)* in policy evaluation at training time, provided the actor's policy π_θ is used by the proposal distribution generating the next action in the temporal-difference objective, and *b)* in policy improvement at training time *and* evaluation time. Indeed, in the specific case of offline RL, it is far safer to perform I-projections, ensuring the learned policies are *not* assigning non-zero weight to actions *outside* the support of the target policy involved in the projection. Otherwise (using M-projections), by trying to cover all the modes of the target policy, the projected policy would be urged to *fill in* gaps in between peaks of the target distribution by assigning density where the target assigns none. The propensity to put overshoot the assigned density is particularly detrimental when gaps are numerous, *i.e.* when the target distribution is *not* concave (*e.g.* when the latter is multimodal). As such, there would be a distributional shift between the projected and the target distributions, causing severe instabilities both at training time and evaluation time, given that we are working in the offline setting in which collecting more data via interactions with the world is not allowed. Remaining *in*-distribution is paramount, and keeping the entropy down by leveraging I-projection rather than M-projection appears as the safest option to achieve this desideratum. Besides, since the

agent need only exploit — and not explore — in offline RL, possessing the natural exploration capabilities enabled by following a higher-entropy policy is void of benefit in offline RL, in contrast with online RL. As such, the problem formulation in Eqs 4.30, 4.31, and 4.32 — which has been adopted in a slew of works such as REPS [PMA10], RWR [PS07, KOP10], LAWER [NP08], CPI [KL02], VIP [Neu11], TRPO/PPO [SLM+15, SWD+17] (forward KL constraint instead of reverse KL), MPO [AST+18], AWR [PKZL19], ABM [SSB+20], CRR [WNZ+20] and AWAC [NDGL20] — is particularly well-suited to the *offline* RL setting (under which MARWIL [WXH+18], AWR [PKZL19], ABM [SSB+20], CRR [WNZ+20] and AWAC [NDGL20] are framed) where the cost-aversion encoded via and distilled by I-projections enables the learned agents to prevent straying from the behavior policy into a distributional shift where errors compound. The reverse KL constraint in EQ 4.31 leads to EQ 4.47 via the exhibited derivation (*cf.* beginning of SECTION 4.6.1). Since all of these past methods have gone through or have reused said derivations of similar flavor, they (and we) all face the same *new* objective as reported in EQ 4.47, and are then subject to the same subsequent task consisting of choosing a measure Δ. All of these works have opted for a KL divergence. Based on the arguments posited above, picking the *reverse* KL, an I-projection, seem like the natural choice given how detrimental and unforgiving naively chasing after rewards (like an M-projection would dictate) seems to be — in offline RL above all else. Starting from the objective in EQ 4.47, we now lay out the derivations of said objective *1)* using the forward KL for Δ, and *2)* using the reverse KL for Δ. Our aim is to highlight that, while there is a striking claim for an I-projection,based the discussion that precedes, the M-projection is inordinately easier to compute than the I-projection, leaving us with a trade-off to balance. We begin with the *forward* KL, by unpacking the measure and the expectations into explicit integral form, and injecting EQ 4.46:

$$\mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[\Delta\big(\pi_\theta(\cdot|s),\zeta_{\text{IW}}(\cdot|s)\big)\Big] \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[D_{\overrightarrow{\text{KL}}}^{\zeta_{\text{IW}}}[\pi_\theta](s)\Big] \tag{4.48}$$

$$\implies \quad \theta \coloneqq \operatorname*{argmin}_{\theta\in\Theta} \ \mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[\Delta\big(\pi_\theta(\cdot|s),\zeta_{\text{IW}}(\cdot|s)\big)\Big] \tag{4.49}$$

$$= \operatorname*{argmin}_{\theta\in\Theta} \ -\int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in\mathcal{A}}\zeta_{\text{IW}}(a|s)\log\pi_\theta(a|s)\,da\,ds \tag{4.50}$$

$$= \operatorname*{argmin}_{\theta\in\Theta} \ -\int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in\mathcal{A}}\zeta(a|s)\exp\left(\frac{1}{\lambda_{\text{KL}}}A_\omega^{\pi_\theta}(s,a)\right)\log\pi_\theta(a|s)\,da\,ds \tag{4.51}$$

$$= \operatorname*{argmax}_{\theta\in\Theta} \ \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\zeta(\cdot|s)}\left[\exp\left(\frac{1}{\lambda_{\text{KL}}}A_\omega^{\pi_\theta}(s,a)\right)\log\pi_\theta(a|s)\right] \tag{4.52}$$

Conversely, by opting for the *reverse* KL instead, the problem in EQ 4.46 reduces to the following

problem:

$$\mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[\Delta\big(\pi_\theta(\cdot|s),\zeta_{\text{IW}}(\cdot|s)\big)\Big] \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[D^{\zeta_{\text{IW}}}_{\overleftarrow{\text{KL}}}[\pi_\theta](s)\Big] \tag{4.53}$$

$$\implies \quad \theta \coloneqq \operatorname*{argmin}_{\theta\in\Theta} \ \mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[\Delta\big(\pi_\theta(\cdot|s),\zeta_{\text{IW}}(\cdot|s)\big)\Big] \tag{4.54}$$

$$= \operatorname*{argmin}_{\theta\in\Theta} \int_{s\in\mathcal{S}} \rho^\beta(s) \int_{a\in c\mathcal{A}} \pi_\theta(a|s) \log\left(\zeta(a|s)\exp\left(\frac{1}{\lambda_{\text{KL}}}A^{\pi_\theta}_\omega(s,a)\right)\right) da\, ds$$

$$- \int_{s\in\mathcal{S}} \rho^\beta(s) \int_{a\in c\mathcal{A}} \pi_\theta(a|s)\log\pi_\theta(a|s)\, da\, ds \tag{4.55}$$

$$= \operatorname*{argmin}_{\theta\in\Theta} \ \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\pi_\theta(\cdot|s)}\left[\log\zeta(a|s) + \frac{1}{\lambda_{\text{KL}}}A^{\pi_\theta}_\omega(s,a)\right] + \mathbb{E}_{s\sim\rho^\beta(\cdot)}\big[H\big(\pi_\theta(\cdot|s)\big)\big] \tag{4.56}$$

where $H\big(\pi_\theta(\cdot|s)\big)$ denotes the entropy of π_θ for a given state s (for more detailed derivations, see Appendix 4.C). Directly echoing our previous discussion about the hurdles of high-entropy policies in offline RL, we observe that eq 4.56 directly involves the entropy of the actor's policy $H\big(\pi_\theta(\cdot|s)\big)$ estimated over states from the offline dataset $\mathcal{D}$. Crucially, we see that when designing Δ as an I-projection, the problem of finding the parametric policy that minimizes $\Delta\big(\pi_\theta(\cdot|s),\zeta_{\text{IW}}(\cdot|s)\big)$ over states from the dataset $\mathcal{D}$ reduces to a formulation in eq 4.56 where we need to *minimize* the entropy of π_θ on said distribution of states $s\sim\rho^\beta(\cdot)$. In other words, the I-projection urges the policy π_θ to have the *lowest* bandwidth possible so as to place the least amount of density outside the support of the target distribution, fitting the modes tightly (albeit likely ignoring some of them in non-concave target scenarios, as developed earlier). The I-projection however tends to make the learned policy *collapse* [AST+18]. Despite being somewhat aligned with our conservative desiderata — and setting aside the policy's propensity to collapse since it can be alleviated via regularization with relative ease, the reduction in eq 4.56 faces two hurdles that make the reduced problem tedious to solve efficiently in practice.

First, we need a model of ζ that enables the evaluation of the likelihood of an action at a given state $\zeta(a|s)$ — which might be already readily available depending on how the proposal policy ζ is defined (*cf.* we lay out the ζ options considered in this work in Section 4.5). In the particular case where $\zeta \coloneqq \beta$, *i.e.* sampling actions from ζ simply means picking actions from the offline dataset $\mathcal{D}$, which means we do not have any way to evaluate the likelihood of an action according to β *other than* modelling the offline distribution β underlying the dataset (equivalently, ζ) with a model that enables such evaluation. Notably, one could sidestep the need for a likelihood model estimating probability densities by leveraging a conditional *score* density estimator, whose returned score *can* be used as proxies for said likelihoods. Relaxing the problem even further, via Bayes' rule, one could craft a surrogate

for said conditional score from a joint score over $\mathcal{S} \times \mathcal{A}$ and a score over $\mathcal{S}$, which, on top of being easier to estimate in most cases, would naturally regulate the scale of the assembled conditional score. The density, novelty, or uncertainty estimator formalized as ρ in EQ 4.10 is a suitable candidate to build a proxy for $\zeta(a|s)$ when $\zeta := \beta$. We refer the reader to the discussion surrounding EQ 4.10 in SECTION 4.5 about potential practical candidates for ρ, along with references to works leveraging such estimators.

The second reason why EQ 4.56 can be tedious to estimate is due to the presence of an expectation over samples from the *very* model we set out to update, in both pieces of the operand. In other words, directly implementing the reduced objective of EQ 4.56 means sending gradients backwards through the stochastic sampling unit "$\sim \pi_\theta(\cdot|s)$" to update π_θ — a non-differentiable operator as is. There are nevertheless numerous tricks to bypass this hurdle. Using a reparametrization trick is the most popular option, first popularized as such in the context of variational auto-encoders in [KW14, RMW14] for Gaussian distributions, then extended to a wider class of variational distributions (*e.g.* beta and gamma distributions) in [RTB16], then concurrently adapted to the categorical distribution in [JGP17, MMT17] by leveraging the Gumbel distribution, [Gum54]. These have seen wide adoption in RL since [HWS+15]. Considering that these encompass virtually every distribution usually used in RL to model the learned policy, one rarely need look elsewhere. Still, in more exotic scenarios, one can turn to REBAR [TMM+17], LAX or RELAX [GCW+18], the *straight-through* estimator [BLC13], or the archetype *REINFORCE trick* [Wil92] — as last resort due to high-variance gradients (*cf.* [SHWA15, Sch16] for an in-depth dive into stochastic computational graphs).

By contrast, the objective resulting from the M-projection in EQ 4.52 is burdened by none of the two previous hurdles. We only need to be able to sample from ζ, as opposed to having access to the likelihood $\zeta(a|s)$. Like before, considering the particular case where $\zeta := \beta$, we do not even need access to a sampling unit, since we can directly use state-actions pairs picked from the offline dataset $\mathcal{D}$. By defining Δ as an M-projection, one therefore reduces the problem described in EQ 4.47 (itself reduced from original one *cf.* EQ 4.33) into a strikingly simpler problem (*cf.* EQ 4.52) consisting in maximizing the *re-weighted* likelihood of the actor's policy π_θ over the dataset $\mathcal{D}$. Still, despite being comparatively tedious to estimate *in practice*, the objective resulting from the I-projection in EQ 4.56 might be worth optimizing, due to the greater resilience against out-of-distribution actions it invests the policy with, *in theory*.

Among the past works that had to tackle the projection task in EQ 4.47, REPS [PMA10], RWR [PS07, KOP10], LAWER [NP08], MPO [AST+18], MARWIL [WXH+18], AWR [PKZL19], ABM [SSB+20], CRR [WNZ+20], and AWAC [NDGL20] opted for a M-projection (forward KL), while VIP [Neu11] chose to observe the problem through a variational inference lens and went for an I-

projection (reverse KL), claiming that the cost-aversion induced via I-projections alleviates plenty of issues that are attributed to M-projections (*cf.* our discussion on these projections, earlier in SECTION 4.6.1). Nevertheless, the authors of [Neu11] conclude that the reverse KL operation is considerably more difficult to compute, which our previous discussion of the problem we arrived at in EQ 4.56 corroborates.

In this work, we opt for the use of a *forward* KL divergence, an M-projection, to define Δ in EQ 4.47. Indeed, we deem the trade-off to lean towards computational feasibility and ease of implementation in modern settings, despite the *"reward-chasing"* behavior it can distill in the learned policies, particularly destructive in offline RL. Note, however, we still use a *reverse* KL divergence in the inequality constraint (*cf.* EQ 4.34) of the original optimization problem laid out in EQ 4.33, in spite of using a *forward* KL divergence in the derived problem in EQ 4.47. To sum up, *1)* we formulate a first problem (*cf.* EQS 4.30, 4.31, and 4.32) where the policy is urged to remain close to a proposal policy ζ in *reverse* KL, *2)* we observe that the analytical closed-form solution of this constrained optimization problem is the *importance-weighted* counterpart ζ_{IW} of the proposal policy ζ, *3)* we formulate a second problem (*cf.* EQ 4.47) where the policy is now urged to remain close to the *importance-weighted* proposal policy ζ_{IW} in *forward* KL, and finally *4)* we observe that this second problem reduces to a final formulation that is simple, interpretable, and light on compute. *In fine*, in practice, we update the actor's policy π_θ by minimizing (via gradient descent) the loss ℓ_θ, directly derived from EQ 4.52:

$$\ell_\theta \coloneqq -\mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\zeta(\cdot|s)}\left[\exp\left(\frac{1}{\lambda_{\text{KL}}}A^{\pi_\theta}_\omega(s,a)\right)\log\pi_\theta(a|s)\right] \tag{4.57}$$

4.6.3 Expansion to multiple streams of decisions

The actor's loss ℓ_θ (*cf.* EQ 4.57) we arrived at the end of SECTION 4.6.2 involves the proposal distribution ζ, which we can define from any of the proposal policies we have laid out in SECTION 4.5, under the same handle ζ. In particular, the past works MARWIL [WXH+18], AWR [PKZL19], CRR [WNZ+20], and AWAC [NDGL20] (setting aside here how these methods estimate Q_ω — Monte-Carlo estimation for MARWIL and AWR, TD-learning for CRR and AWAC) all update the actor's policy π_θ using ℓ_θ (*cf.* EQ 4.57), and using the offline policy β as proposal policy ζ, *i.e.* $\zeta \coloneqq \beta$ (*cf.* "BETA SARSA" strategy in SECTION 4.5.3). As discussed in SECTION 4.6.2, when the proposal is β the expectations over state and action in ℓ_θ are implemented in practice simply by taking samples from the offline dataset $\mathcal{D}$, *i.e.* we need not have an explicit handle on ζ, be it for sampling from ζ or for computing a likelihood estimate $\zeta(a|s)$ for a given state-action pair $(s,a) \in \mathcal{S}\times\mathcal{A}$. In addition to *subsuming* the learning rules of MARWIL [WXH+18], AWR [PKZL19], CRR [WNZ+20], and AWAC [NDGL20], the loss ℓ_θ, as depicted in EQ 4.57, also subsumes *both* policy improvement rules

proposed in ABM [SSB+20], where the proposal policy ζ plays the role of *prior* (*cf.* ABM [SSB+20]).

Importantly, as a design choice, we *never* allow gradients to flow backwards through the ζ sampling unit when the parameter vector θ, parametrizing the actor's policy π_θ, are used in the assembly of the proposal policy ζ (if ζ does not use θ, the ζ sampling unit *"$a \sim \zeta(\cdot|s)$"* is out of the computational graph for the actor update anyway). We consequently need not involve stochastic computational graphs techniques, of which we gave an overview earlier in SECTION 4.6.2 when analyzing EQ 4.56. In practice, this means treating the actions sampled via $a \sim \zeta(\cdot|s)$ as *inputs* to the computational graph of the policy improvement update, or to *detach* these samples from the graph. We now consider this as a given and will not involve *stop-gradient* operations in the derivations that follow, whatever ζ contains.

Coming back to how the loss ℓ_θ depicted in EQ 4.57 subsumes the actor update of ABM [SSB+20], we can replicate the one using the *"BM"* prior (*cf.* [SSB+20]) by setting $\zeta := \beta_{\text{C}}$ (*cf.* "BETA CLONE" strategy in SECTION 4.5.3), where β_{C} is a policy resulting from cloning the behavior policy β underlying the dataset $\mathcal{D}$. Furthermore, we can replicate the actor update rule using the *"ABM"* prior (*cf.* [SSB+20]) by modelling ζ with an *auxiliary* actor learned with an n-step TD return [PW96] hybrid between MARWIL/AWR ([WXH+18, PKZL19], pure MC return) and CRR/AWAC ([WNZ+20, NDGL20], 1-step TD return). In such a setting, the auxiliary actor would also use an auxiliary critic, learned via Monte-Carlo estimation, in order to build its own advantage estimate, exclusively used by the *"ABM"* prior.

Moreover, we observe that by aligning ζ with the actor itself (*cf.* "THETA" strategy in SECTION 4.5.3) — more accurately, with a fixed, detached from the graph, copy of the previous actor update, which can be denoted by $\pi_{\theta^{\text{old}}}$, akin to the notations adopted in TRPO [SLM+15] — the inequality constraint depicted in EQ 4.34 becomes $\mathbb{E}_{s\sim\rho^\beta(\cdot)}\left[D_{\overleftarrow{\text{KL}}}^{\pi_{\theta^{\text{old}}}}[\pi_\theta](s)\right] = \mathbb{E}_{s\sim\rho^\beta(\cdot)}\left[D_{\text{KL}}\left(\pi_\theta(\cdot|s)\,||\,\pi_{\theta^{\text{old}}}(\cdot|s)\right)\right]$ when applied to π_θ. This constraint coincide with the one adopted in MPO [AST+18], and had it been a *forward* KL instead of the reverse one, this constraint would have matched the one used by TRPO [SLM+15], and PPO [SWD+17]: $\mathbb{E}_{s\sim\rho^\beta(\cdot)}\left[D_{\overrightarrow{\text{KL}}}^{\pi_{\theta^{\text{old}}}}[\pi_\theta](s)\right] = \mathbb{E}_{s\sim\rho^\beta(\cdot)}\left[D_{\text{KL}}\left(\pi_{\theta^{\text{old}}}(\cdot|s)\,||\,\pi_\theta(\cdot|s)\right)\right]$. In essence, once one can instantiate the problem laid out in TRPO from one's framework, one can also do so — omitting minor irrelevant specificities — for all the methods adopting a *natural gradient* [Ama16, Kak01, PS08] approach, from which TRPO is inspired, such as NPG [Kak01], and CPI [KL02]. All in all, the loss ℓ_θ depicted in EQ 4.57 *already* enables us to instantiate a number of methods from the online and offline RL literature. As such, the loss ℓ_θ is not novel *per se*, but the crafted framework provides a unified view of the current state-of-the-art methods in offline RL (CRR [WNZ+20], and AWAC [NDGL20]), that are readily expressible under the framework for policy improvement we propose in this section. Note, we established a similar unification earlier in

SECTION 4.5, but over a wide range of distinct ways one could perform policy evaluation back then, with thorough empirical support.

What the diagrams of FIGURE 4.5.1 illustrated clearly in the context of policy evaluation is that designing an update rule (equivalently, loss function) for the action-value Q_ω is not a *one-dimensional* problem in *offline* RL like it is in online RL. Re-using the nomenclature introduced in SECTION 4.5, the practitioner in charge of designing the policy evaluation learning update for Q_ω has by construction tight control over where the learned action-value Q_ω is located — and how it will evolve and and travel — over the value simplex depicted in FIGURE 4.5.1. While in online RL (at least in the traditional setting), said practitioner could design a critic's loss that places Q_ω *anywhere* on the closed line segment joining the *SARSA* critic Q^{π_θ} (perfectly consistent with π_θ) and the optimal critic Q^* (perfectly consistent with π^*, and called *"expert"* instead of critic by [LJL+18] to further emphasize the gap in their objectives). As such, the DDPG [LHP+16] critic (among many others like SAC [HZAL18], etc.) is effectively updated as a *SARSA* critic where the next action injected in $\ell_\omega^{\text{SARSA}}$ (*cf.* EQ 4.16) is from the (greedy) actor π_θ, while the actor-critic methods that stemmed from [CB95] attempt for Q_ω to approximate Q^* more directly. Based on the previous sentence, one can easily place the action-value Q_ω, for either summoned algorithm, on the closed line segment joining Q^{π_θ} and Q^* on the value simplex of FIGURE 4.5.1. The offline RL setting introduces Q^β due to the added constraint discouraging the actor from straying from β. The involvement of β, underlying the offline dataset $\mathcal{D}$, has the effect of inflating the previous 1-dimensional closed line segment into said 2-dimensional simplex (*cf.* FIGURE 4.5.1).

Constraining π_θ to remain close to β in reverse KL divergence as encoded by EQ 4.34, albeit instrumental in alleviating out-of-distribution actions at training *and* evaluation time, can (as a side effect) thwart the *true* objective the actor should aim at: converging towards π^* for the task at hand. Such reasoning is vividly echoing the discussion we carried out in SECTION 4.5, which we provided a retake of and pointers to in the previous paragraph. Similarly to how the design of the policy evaluation step in offline RL makes Q_ω follow a certain path (throughout the iterations) on a *value* simplex whose vertices are $\{Q^*, Q^\beta, Q^{\pi_\theta}\}$ (*cf.* FIGURE 4.5.1), the design of the policy *improvement* step in offline RL makes π_θ follow a certain path (throughout the iterations) on a *policy* simplex whose vertices are $\{\pi^*, \beta, \pi_{\theta^{\text{old}}}\}$ (*cf.* FIGURE 4.6.1). Crucially, note, these simplices are asymmetrical: their verticies are not tied by a bijection, *i.e.* there is not a one-to-one mapping linking each vertex of one simplex to its counterpart in the other. Indeed, while the two verticies π^* and β are both coupled with their counterparts Q^* and Q^β respectively by a *"greedifies* $\leftrightarrow$ *evaluates"* relationship, this is not at all the case for $\pi_{\theta^{\text{old}}}$ and Q^{π_θ}. This is due to the fact that there is an extra degree of estimation for the action-value compared to the policy. While the estimated policy is π_θ, the estimated action-value is Q_ω, which is not necessarily designed to be consistent with the estimated actor's policy π_θ.

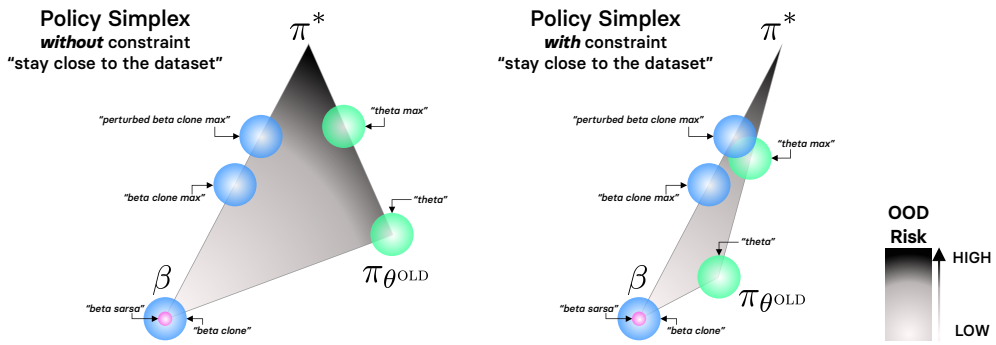


Figure 4.6.1: Abstract representation of the relative positioning of the policies learned using the various proposal distributions ζ laid out in SECTION 4.5 (whose names are depicted on the diagram) to sample actions from in ℓ_θ (*cf.* EQ 4.57). These policies are depicted by disks over the simplex spanned by the optimal policy π^*, the policy followed by the learned actor π_θ, and the policy underlying the offline dataset, β. The diameter of said disks crudely depicts how confident one can be about the placement of the various tackled proposal policies (*cf.* SECTION 4.5) on the abstract simplex. Albeit only roughly estimating the actual geometry of the policy simplex, this diagram can nevertheless help us categorize the different proposal distributions with respect to how they expose to agent (and its value) to *out-of-distribution* (OOD) actions *at training time*, and crucially *at evaluation time*. We omit here the methods reminiscent of safe policy improvement (SPI) approaches as the associated proposal policies change their predictions based on an extra data-dependent condition being fulfilled, making them tedious to place on the simplex. Note, we only consider the use of a *single* proposal policy in $\ell_\theta^{\text{GIWR}}$ (*cf.* EQ 4.58) for the abstract, illustrative purposes of these diagrams. Best seen in color: *pink* signifies that the proposal distribution is β *exactly*, *blue* that the proposal distribution relies on an estimate of the β distribution, and *green* that the proposal policy ζ solely involves the actor π_θ — in other words, the proposal distribution ζ is not derived from the offline dataset $\mathcal{D}$ in any shape or form. We keep the residual denomination *"SARSA"* from SECTION 4.5, EQ "BETA SARSA" to signify that $\zeta := \beta$, for the sake of conceptual symmetry between *evaluation* (*cf.* SECTION 4.5) and *improvement* (*cf.* SECTION 4.6) — despite the fact that the *next* action a' (last *"A"* in *"SARSA"*) plays no functional role in the policy improvement step.

The result $Q_\omega \approx Q^{\pi_\theta}$ can be achieved *only* if the "THETA" strategy (proposed in SECTION 4.5.3) is picked for policy evaluation. The action-value (Q^{π_θ}) perfectly consistent with the actor's policy (π_θ) that navigates the policy simplex (*cf.* FIGURE 4.6.1) is a *vertex* in the value simplex (*cf.* FIGURE 4.5.1), and it is Q_ω that navigates the value simplex (in line with the proposal strategy used for policy evaluation). Note, the policy simplex in FIGURE 4.6.1 is simpler to interpret than the value simplex in the sense that the potential proximity constraint imposed between π_θ and β is directly observable since they both live in the same space as the simplex. This is not the case for the value simplex in FIGURE 4.5.1, for which the entities tied by said proximity constraints (policies) do not live in the same space as the points of the simplex (action-values).

By aligning ζ with β (*cf.* "BETA SARSA" proposal strategy) in EQ 4.34 for the policy improvement step, the actor update will *attract* π_θ towards the "β" corner of the simplex in FIGURE 4.6.1, as it departs from $\pi_{\theta^{\text{old}}}$ and makes a gradient step into the simplex. On the next policy improvement step, the $\pi_{\theta^{\text{old}}}$ vertex will see its location overridden with the freshly obtained π_θ. Note, like in the value simplex, the vertex involving the parameter vector θ in the policy simplex changes *continually* as the agent iterates through the GPI steps, depicted in FIGURE 4.1.1. Similarly, by setting ζ to be $\pi_{\theta^{\text{old}}}$ (*cf.* "THETA" proposal strategy) in EQ 4.34, the actor update will attract π_θ towards the "$\pi_{\theta^{\text{old}}}$" corner of the simplex as it departs from $\pi_{\theta^{\text{old}}}$ and makes a gradient step into the simplex, effectively restricting the amplitude of updates π_θ goes through — as mentioned earlier when describing how our framework can implement the conservative KL constraints akin to natural gradient [Ama16, Kak01, PS08] methods from TRPO [SLM+15], PPO [SWD+17], and MPO [AST+18], but here set in the context of FIGURE 4.6.1.

Crucially, in this work, we want the learned actor policy π_θ to *cover more ground* on the policy simplex, similarly to our coverage of the value simplex in SECTION 4.5 where we did so by considering a slew of proposal policies ζ and using these to generate the *next* action to inject in Bellman's equation (*cf.* ℓ_ω in EQ 4.14). We want to expand the navigation capabilities of π_θ over the policy simplex, and can do so (as we have just laid out earlier in this paragraph) by using various designs of the proposal distribution ζ in the inequality constraint (*cf.* EQ 4.34) at the source of the policy improvement update rule we have just derived in SECTION 4.6.1. In SECTION 4.5, it is tedious to controllably navigate Q_ω over the value simplex since the only entity we have control over with the proposal distribution ζ is the next action $a' \sim \zeta(\cdot|s')$, to use in ℓ_ω. We were able to interpolate between proposal policies by introducing *SPI*-inspired designs (*cf.* SECTION 4.5.3), but these can also only be used to output the next action for Bellman's equation, thwarting the interpolation of proposal strategies by limiting their expressiveness. Besides, making an action-value consistent with several proposal policies by optimizing a combination of temporal-difference losses is a tall order, as it promises to be remarkably unstable. Learning separate values in an ensemble could be an option, but is costly, and out of the scope of this work (*cf.* SECTION 4.3). By contrast, involving several proposal strategies in our policy improvement objective ℓ_θ depicted in EQ 4.57 seems significantly easier to optimize, as it essentially *augments* the dataset over which π_θ must maximize its importance-weighted likelihood. Such data augmentation, rather that trying to find a better representation to favor the resolution of downstream tasks, aims to assist the agent towards a speedier resolution of the task currently at hand by *"squeezing more juice"* out of the available information — $\mathcal{D}$ in our case (notable instances of such data augmentation include the works of [Kae93] and [AWR+17]).

As such, we introduce a new framework, called GIWR (*"giver"*) for *Generalized Importance-Weighted Regression*, whose defining objective $\ell_\theta^{\text{GIWR}}$ involves *families* of proposal policies $Z_n \coloneqq (\zeta_i)_{i\in[1,n]\cap\mathbb{N}}$, and their accompanying scaling coefficients $K_n \coloneqq (\kappa_i)_{i\in[1,n]\cap\mathbb{N}}$ with $\kappa_i > 0$ $(\forall i \in [1, n] \cap \mathbb{N})$. We

define $\ell_\theta^{\text{GIWR}}$ as follows:

$$\ell_\theta^{\text{GIWR}} := -\mathbb{E}_{s\sim\rho^\beta(\cdot)}\left[\sum_{i=1}^{n} \kappa_i \, \mathbb{E}_{a\sim\zeta_i(\cdot|s)}\left[\exp\left(\frac{1}{\lambda_{\text{KL}}} A_\omega^{\pi_\theta}(s,a)\right)\log\pi_\theta(a|s)\right]\right] \tag{4.58}$$

where the temperature hyper-parameter λ_{KL} — shared by all the n contributions to the GIWR actor loss $\ell_\theta^{\text{GIWR}}$ defined in EQ 4.58 — could be made specific per proposal policy $\zeta \in Z_n$. We would then also have a family of n temperatures, one for each ζ in Z_n, making the hyper-parameter sweep considerably more tedious to complete. We therefore opted for simplicity and sticked with the use a single, shared temperature λ_{KL}. Since we conceived the loss in EQ 4.58 as a *multi-objective* inflation of the loss in EQ 4.57, we can interpret GIWR as introducing extra KL inequality constraints following the schema of EQ 4.34, restricting $\mathbb{E}_{s\sim\rho^\beta(\cdot)}\left[D_{\underset{\text{KL}}{\leq}}^{\zeta_k}[\pi_\theta](s)\right] = \mathbb{E}_{s\sim\rho^\beta(\cdot)}\left[D_{\text{KL}}\left(\pi_\theta(\cdot|s)\,||\,\zeta_k(\cdot|s)\right)\right]$ to remain below a certain threshold, for the n proposal distributions ζ_k of the proposal family Z_n. We refer the reader to [RVWD14] and [MARW16] for a survey and overview of the *multi-objective RL* sub-field.

In particular, the work of [AHH+20], dresses the proposed MO-MPO framework as a multi-objective RL framework first and foremost, but is in essence a multi-task RL framework that tackles the tasks at end via a multi-objective formulation, and learns a single policy that must trade off across different *tasks*, or interchangeably, *objectives* (*cf.* SECTION 2.3 of their work [AHH+20]). They build on the premise that the agent is provided with a family of reward signals, a distinct one for each task or objective, and learn an action-value for each. They learn an action distribution (using our terminology, a proposal distribution) for each of these action-values, and combine these distributions along with their associated task-specific values to obtain the next actor iterate. Our proposed objective draws similarities with theirs, as they build on MPO [AST+18] which we showed earlier can be instantiated under our current framework and shares the derivation re-purposed in SECTION 4.6.1 like most of the approaches adopting the *"RL as inference"* paradigm. In this work, by contrast, the agent does not have access to a family of rewards (world framed as a *multi-objective MDP* in [AHH+20]), but only to the rewards collected by β and provided through the offline dataset $\mathcal{D}$ (*cf.* the work of [FMP18] and [LTdC19] for a formulation of *offline* or *dataset-bound* MDP underlying $\mathcal{D}$ and derived from the dynamics traces observed in $\mathcal{D}$). As such, we only learn a single action-value. Plus, despite also involving a proposal family (denoted by Z_n in our work), the proposal distributions are defined in a completely different way: while [AHH+20] has one per reward signal, we conceive the ζ's in Z_n from the bare information available in the offline setting (just $\mathcal{D}$), as strategies that empower the agent to cover the policy and value simplicies (*cf.* FIGURES 4.6.1 and 4.5.1) to a greater extent, so as to achieve optimality faster and more reliably while avoiding the pitfalls of offline RL. Since the GIWR framework allows for the involvement of as many constraints one desires, we can essentially instantiate both a trust-region constraint tying the next iterate to the previous one $\pi_{\theta^{\text{old}}}$ and another constraint forcing

it to be in the vicinity of β in the policy simplex (*cf.* FIGURE 4.6.1). As such, with a proposal family of two defined as $Z = (\zeta_{\text{AWR}}, \pi_{\theta^{\text{old}}})$, where ζ_{AWR} is an *auxiliary actor* learned with n-step TD extension [PW96] of the AWR [PKZL19] algorithm, we can effectively replicate the variant of ABM [SSB+20], called ABM-MPO, reported as achieving the highest performance in said work. Alternatively, by replacing ζ_{AWR} with $\mathcal{T}_{\text{EVAL}}[\beta_{\text{C}}]$ (*cf.* "BETA CLONE" in SECTION 4.5) in the proposal family Z, we get the empirically-weaker BM-MPO method, as reported in [SSB+20]. Nonetheless, these two last methods are reported in [NDGL20] to be outperformed by AWAC [NDGL20] — and consequently also by the concurrent, virtually-identical CRR method [WNZ+20].

These two last methods (along with MARWIL [WXH+18] and AWR [PKZL19] if we set aside how they estimate Q_ω) can be cast as instances of GIWR (*cf.* $\ell_\theta^{\text{GIWR}}$ in EQ 4.58) where the proposal family Z is a *singleton* that contains only β (and the scaling coefficient family K a singleton that trivially contains only 1.0). These achieve state-of-the-art performance in most situations — dynamics of the environment and quality of the dataset loosely being the main differentiating factors, as we have showcased in SECTION 4.4.3. As such, in the experiments we report in this work (*cf.* SECTION 4.6.4), every proposal family Z that we consider contains β, the proposal distribution "BETA SARSA". Not only has this proposal distribution proved to yield excellent performance in AWAC [NDGL20] and CRR [WNZ+20], but it is also trivial to evaluate the expectation with respect to such proposal, as discussed earlier in SECTION 4.6.1, since we need only take data from the offline dataset $\mathcal{D}$. To keep the number of experiments to a reasonable amount without sacrificing the depth of understanding we can get out out of them, we cap the cardinality of Z at 2. Since we set $|Z| \leq 2$ and $\beta \in Z$ in the experiments of SECTION 4.6.4, we can then use "ζ" to unambiguously denote the *other* proposal distribution (distinct from β) in the family Z when $|Z| = 2$. Likewise, we can then use "κ" to unambiguously denote the coefficient that scales the contribution associated with ζ in $\ell_\theta^{\text{GIWR}}$ (*cf.* EQ 4.58), since we scale the contribution associated with β in $\ell_\theta^{\text{GIWR}}$ by 1.0 consistently across experiments. Concretely, as for policy evaluation in SECTION 4.5, we report and discuss our empirical findings for 9 scenarios. The first corresponds to the case where only β is used — $|Z| = 1$, coincides exactly with AWAC [NDGL20] and CRR [WNZ+20] — and plays the role of *baseline* in SECTION 4.6.4. The 8 other competing scenarios correspond to the cases where $Z = (\beta, \zeta)$ and $K = (1.0, \kappa)$, with ζ covering the spectrum of proposal distributions that we introduced in SECTION 4.5 — all except "BETA SARSA", which would be redundant with the first scenario involving only β, for a total of 8 candidates for ζ: "BETA CLONE", "THETA", "BETA CLONE MAX", "PERTURBED BETA CLONE MAX", "THETA MAX", "SPI BETA CLONE", "SPI BETA CLONE MAX", and finally "SPI PERTURBED BETA CLONE MAX". These scenarios were designed to maintain a high degree of symmetry with the previously reported experiments; we could use *any* policy for ζ. Note, the scenario combining "BETA SARSA" and "THETA" coincides with AWAC [NDGL20] or CRR [WNZ+20] in which an extra *MPO*-like trust-region

constraint [AST+18] is plugged in. Importantly, all 8 methods competing with the baseline are *novel*.

We lay out the pseudo-code for SECTION 4.6.4 in ALGORITHM 6.

We now report and discuss our experimental findings.

4.6.4 EXPERIMENTAL RESULTS

Again, we rely on the experimental setting thoroughly described in SECTION 4.4.2 to carry out the empirical investigation laid out here. We remind the reader that, as specified at the beginning of SECTION 4.6.1, we adopt the proposal distribution strategy "THETA" for policy *evaluation* in the empirical investigations performed in this section tackling policy improvement. We first report the empirical evaluation of the experimental scenarios assembled and laid out at the end of SECTION 4.6.3 in FIGURE 4.6.2. For every experiment reported in FIGURE 4.6.2, we set the scaling coefficient κ to 0.2, and we report the counterpart performances for $\kappa \in \{0.1, 0.5\}$ in APPENDIX 4.D, FIGURES 4.D.1 and 4.D.2 respectively.

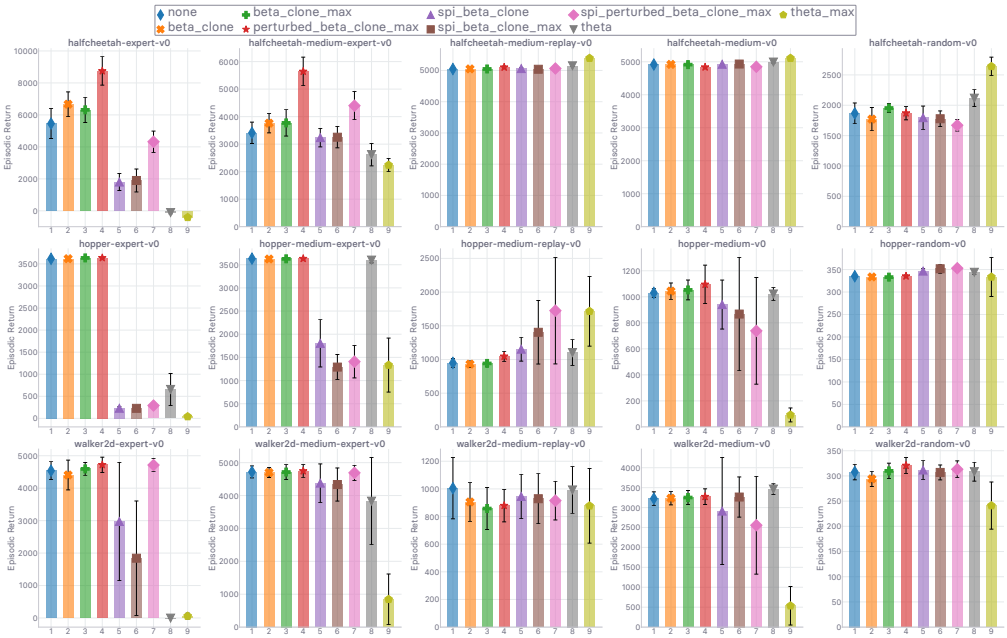


Figure 4.6.2: Empirical comparison of how the proposal distributions introduced in SECTION 4.5.3 impact the final performance of GIWR (*cf.* ALGORITHM 6). Everything except the proposal policy ζ in use is identical. We use $\kappa = 0.2$ as scaling coefficient for the contribution of ζ in EQ 4.58. Runtime is 12 hours. Best seen in color.

FIGURE 4.6.2 tells a story that echoes the one we laid out in SECTION 4.5.4. Similarly, the proposal distributions "THETA" and "THETA MAX" perform very inconsistently across the considered bench-

Algorithm 6: GIWR
with proposal distributions: ζ^{PE} for policy evaluation, $(\zeta_i^{\text{PI}})_{i\in[1,n]\cap\mathbb{N}}$ for policy improvement
$\diamond$: differing from BASE (*cf.* ALGORITHM 5)

init: initialize the random seeds of each framework used for sampling, the random seed of the environment $\mathbb{M}$, the neural function approximators' parameters (θ for the actor's policy π_θ, and ω for the critic's action-value Q_ω), the critic's target network ω' as an exact frozen copy, the offline dataset $\mathcal{D}$.

while *no stopping criterion is met* **do**
  /* Train the agent in $\mathbb{M}^{\text{off}}$ */
  Get a mini-batch of samples from the offline dataset $\mathcal{D}$;
  Perform a gradient *descent* step along $\nabla_\omega \ell_\omega$ (*cf.* below) using the mini-batch;
$\diamond$

$$\ell_\omega := \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\beta(\cdot|s),s'\sim\rho^\beta(\cdot)}\left[\left(Q_\omega(s,a) - \left(r(s,a,s') + \gamma\,\mathbb{E}_{a'\sim\zeta^{\text{PE}}(\cdot|s')}\left[Q_{\omega'}(s',a')\right]\right)\right)^2\right]$$

  where $r(s,a,s')$ was introduced as syntactic sugar in SECTION 4.3;
  (Note, as mentioned early in SECTION 4.6.1, we use $\zeta^{\text{PE}} := \pi_\theta$ in the experiments reported in SECTION 4.6.4);
  Perform a gradient *ascent* step along $\nabla_\theta \mathcal{U}_\theta$ (*cf.* below) using the mini-batch;
$\diamond$

$$\mathcal{U}_\theta := \mathbb{E}_{s\sim\rho^\beta(\cdot)}\left[\sum_{i=1}^{n} \kappa_i\,\mathbb{E}_{a\sim\zeta_i^{\text{PI}}(\cdot|s)}\left[\exp\left(\frac{1}{\lambda_{\text{KL}}}A_\omega^{\pi_\theta}(s,a)\right)\log\pi_\theta(a|s)\right]\right]$$

  where $A_\omega^{\pi_\theta}(s,a) := Q_\omega(s,a) - \mathbb{E}_{\tilde{a}\sim\pi_\theta}[Q_\omega(s,\tilde{a})]$, κ_i are scaling coefficients, and λ_{KL} is a temperature;
  Update the target network ω' using the new ω;
  /* Evaluate the agent in $\mathbb{M}$ */
  if *evaluation criterion is met* **then**
    foreach *evaluation step per iteration* **do**
      Evaluate the empirical return of π_θ in $\mathbb{M}$ (*cf.* evaluation protocol in SECTION 4.4.2);
    end
  end
end

mark of datasets, overall being the two worst choices of proposal policices ζ — in the setting we set out to work with in SECTION 4.6.3, *i.e.* $|Z| \leq 2$ and $\beta \in Z$. We remind the reader that we gave thorough interpretations for every proposal distributions introduced in SECTION 4.5.3 for policy evaluation, as we re-purposed them in the context of policy *improvement*, and refer the reader to these discussions. In short, we observe in FIGURE 4.6.2 that, by encouraging the actor's policy π_θ to stay close to $\mathcal{T}_{\text{EVAL}}[\pi_{\theta^{\text{old}}}]$ or $\mathcal{T}^{\omega,m}_{\text{MAX}}[\pi_{\theta^{\text{old}}}]$ in reverse KL divergence (*cf.* EQ 4.34 in SECTION 4.6.2), in addition to remaining close to β (*cf.* "BETA SARSA") in reverse KL since $\beta \in Z$ as per our experimental design choices, "THETA" and "THETA MAX" (respectively) yield terrible returns in most scenarios, while improving upon the baseline only in rare isolated cases. Note, we do not enforce these constraints *explicitly*, as we optimize the *reduction* that we derived in SECTION 4.6.2 and extended in SECTION 4.6.3. Indeed, despite being essentially equivalent to AWAC [NDGL20] or CRR [WNZ+20] (*blue* color in FIGURE 4.6.2) in which an extra *MPO*-like trust-region constraint [AST+18] is plugged in, seem not to be as conservative and *safe* in terms of how the actor's policy navigates the simplex depicted in FIGURE 4.6.1 as one might expect. The value of κ might be the culprit here, and a deeper hyper-parameter sweep for κ could be key to strike the *right* trade-off of safety against destructive policy updates. This hypothesis is to a certain extent corroborated by FIGURE 4.D.1 in APPENDIX 4.D, where the use of "THETA" in GIWR seem to yield considerably better returns in environments where performance seemed to be disastrous (*e.g.* in the *expert*-grade datasets). Lower values of κ *can* mitigate the dips in performance caused by destructively big updates in parameter space, based on how much the returns drop for the "THETA" and "THETA MAX" proposal distributions in APPENDIX 4.D FIGURE 4.D.2 ($\kappa = 0.1$), compared to in APPENDIX 4.D FIGURE 4.D.1 ($\kappa = 0.5$). These two heuristics seem not to be worth introducing from an offline RL practitioner's standpoint, judging by how sensitive — or *stiff*, as we characterized earlier in SECTION 4.3 — these methods are with respect to the value of κ, especially in the *expert*-grade datasets. Still, it was shocking to observe just how well "THETA MAX" performs in the *random* dataset of the `halcheetah` environment within the tackled benchmark (top-right in FIGURES 4.6.2, 4.D.1, and 4.D.2).

As we observed earlier in SECTION 4.5.4, and adopting the terminology introduced then, the proposal distributions from the *SPI* group are *often* severely hindered (yet not *always*) by the weak performance of the $\mathcal{T}^{\omega',m}_{\text{MAX}}$ operator in "THETA MAX" which they coincide with by design when $\rho(s, \tilde{a}^{\theta}_{\text{MAX}}) \geq \delta$ (*cf.* EQ 4.8 for the definition of the *SPI* operator template, and EQ 4.10 for the definition and surrounding discussion on the design choices related to *rho*). Despite achieving higher returns overall than their "THETA MAX" component, the proposal distributions from the *SPI* group are overall outperformed by their counterparts proposal policies in the *clone* group, with a gap in performance seemingly stemming from how *mediocre* "THETA MAX" is in the considered dataset. As in FIGURE 4.6.2, this behavior is observed when $\kappa \in \{0.1, 0.5\}$ too, as exhibited in FIGURES 4.D.1 and 4.D.2, reported

in Appendix 4.D. As such, the methods withing the *SPI* group achieve performance *consistently* ranked in between "theta max" and their counterparts in the *clone* group, but only rarely reach the return accumulated by the best of the two methods between which they are attempting to strike a trade-off. Since such balance is fully determined by $\Gamma^{\theta}_{\delta}(s) \coloneqq \mathbb{1}\left[\rho\left(s, \tilde{a}^{\theta}_{\text{MAX}}\right) \geq \delta\right]$, one might be able to strike a better trade-off (achieve *"best of both worlds"* results) by fine-tuning the threshold δ for the given dataset-environment couple, and exploring a wider variety of designs for the potential function ρ over $\mathcal{S} \times \mathcal{A}$ (*cf.* eq 4.10). Nevertheless, Figures 4.6.2, 4.D.1, and 4.D.2 show that in most of the considered datasets and environments, our design choices enable the proposal distribution in the *SPI* group to make good and *safe* (*cf.* Section4.2) compromises.

Finally, Figures 4.6.2, 4.D.1, and 4.D.2 show that the proposal policies in the *clone* group (that is, "beta clone", "beta clone max", and "perturbed beta clone max") are positively assisting the baseline methods in *every* environment where it struggled in the first place against the other *state-of-the-art* offline RL methods (*cf.* Section 4.4.3). Importantly, these add-ons do *not* harm the baseline while enhancing it in the environments where it was lagging behind. While "beta clone", "beta clone max" do not improve upon the baseline by a significant margin, "perturbed beta clone max" widens said margin to a greater extent across the benchmark, and especially in the dataset-environment couples in which the baseline showed signs of struggle in Section 4.4.3. *In fine*, involving the perturbation model ξ on top of a clone β of the behavioral distribution β_{C} *and* leveraging the $\mathcal{T}^{\omega',m}_{\text{MAX}}$ operator to build the proposal distribution described in eq 4.12 achieves the best results, by a large margin relative to the other proposal strategies in the *clone* family in the dataset and environments where the baseline needs it most. Note, this proposal distribution coincides with the actor learned in BCQ [FMP18] — yet, Q_{ω} is *decoupled*, *cf.* Section 4.5.3, and *not* learned as in BCQ [FMP18], which would correspond to performing policy evaluation with the proposal distribution "perturbed beta clone max". We observe identical results in Figures 4.D.1 and 4.D.2 in Appendix 4.D.

In line with these findings, the strategy that displays the highest performance among the *SPI* group is the one whose counterpart in the *clone* group is "perturbed beta clone max" — which outperforms *every* other method as we have just stressed. Such an observation is not surprising but attests to the consistency and robustness of the proposal heuristics we have put into place. In the same vein, we also observe that the GIWR framework we here introduce is *not* stiff (*cf.* Section 4.3) with respect to the choice of κ, as depicted in Figures 4.D.1 and 4.D.2 where the ranking of methods is essentially identical to the one observed in Figure 4.6.2. While being robust with respect to κ, we see from these plots describing the performed sweep that increasing the value of κ increases the return of the best performing methods further for *expert* datasets, while not having neither unexpectedly positive nor unexpectedly negative effect in the non-*expert* datasets. All in all, the GIWR framework is *robust*

in that respect.

We place the best performing studied variant of GIWR, the one using "PERTURBED BETA CLONE MAX" for ζ, among the other *state-of-the-art* offline RL baselines introduced in SECTION 4.4.1 and compared empirically in SECTION 4.4.3, with $\kappa = 0.2$, in FIGURE 4.6.3. We omit SAC [HZAL18] and our version of D4PG [BMHB+18] judging by how poorly they performed in the analysis we carried out and laid out in SECTION 4.4.3. While FIGURE 4.6.3 does not provide new information *per se*, it puts things in perspective as for how GIWR enables us to close the gap between the chosen baseline (*cf.* ALGORITHM 5) and its competition in the environments in which it lagged behind. For instance, in the second plot of the grid in FIGURE 4.6.3 (first row, second column), CRR displays the *eighth* highest return, while GIWR achieves the *second* highest return — behind BCQ [FMP18] which *underperforms* both CRR and our GIWR instance in *13 out of the 15* datasets of the suite. Note, we could fall back to the *next-in-line* best performing model, AWR [PKZL19], simply by appropriating Q_ω via Monte-Carlo estimation instead of temporal-difference learning, for the $\beta \in Z$ component of the GIWR loss (*cf.* EQ 4.58), or for every distribution of the proposal family $Z = (\beta, \zeta)$. The obtained results confirm our intuition, laid out in SECTION 4.6.3 and illustrated in the simplices of FIGURE 4.5.1 and FIGURE 4.6.1: it *can* be highly beneficial to *directly* urge π_θ to approach π^*, and by using "THETA" as proposal distribution in policy *evaluation* as we do in this section, to indirectly urge Q_ω to Q^* — essentially skewing the *SARSA* update [RN94, Thr95, Sut96, vSvHWW09] consisting in using "THETA" as proposal distribution in policy evaluation into a Q-learning update [Wat89, WD92], *while* dealing with the hindrance of distributional shift in offline RL.

By far the most crucially appealing feature of our framework is that the practitioner *need not* take any risky decisions when it comes down to the design of the policy improvement rule, as we have shown that we can increase the performance of the *state-of-the-art* offline RL baseline in specific datasets (*e.g.* in the *expert* ones) without hurting its performance in the remainder of the suite. This *"best of both worlds"* trade-off has not been struck by any other method preceding GIWR, as we have shown profusely in SECTION 4.4.4. By leveraging *implicit* I-projections while being heavily modular, GIWR enables practitioner to built policy improvement update rules that can fit their use cases while being safely shielded from *spurious* inductive biases most baselines *aggressively* inject in their model.

As a final note, one must not forget that in our approximate dynamic programming setting, the generalized policy iteration scheme tackled in this work and illustrated in FIGURE 4.1.1 for both online and offline RL *entangles* a policy evaluation step with a policy improvement step an an *alternating*, *iterative* process. One seemingly small disparity between evaluation (*cf.* 4.5) or improvement (*cf.* 4.6) update rules can cause considerable *ripple effects* on the whole compound procedure. In this work, we have tackled its *in-depth* and *in-breadth* study in the *offline* RL setting, further burdened with

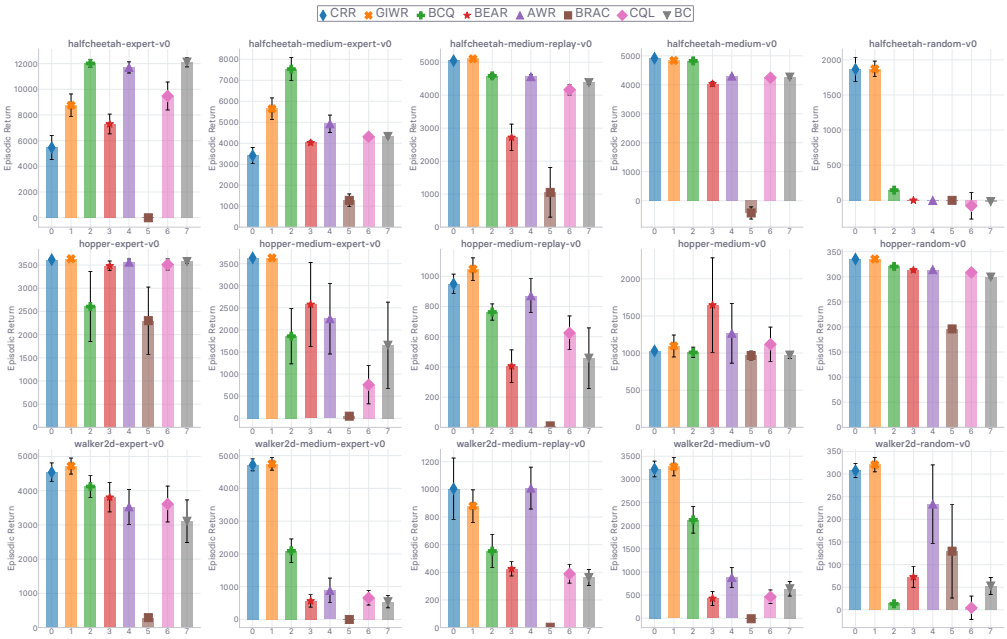


Figure 4.6.3: Empirical comparison between the best performing studied variant of GIWR, the one using "PERTURBED BETA CLONE MAX" for ζ, and the other *state-of-the-art* offline RL baselines introduced in SECTION 4.4.1 and compared empirically in SECTION 4.4.3. We use $\kappa = 0.2$ for ζ in EQ 4.58. Runtime is 12 hours. Best seen in color.

additional points of failure such as distributional shift, due to the inability for the agent to collect more training data as in online RL.

4.7 CONCLUSION

Our *first* contribution consisted in the re-implementation of the main state-of-the-art offline RL baselines under a fair, unified, highly factorized, and open-sourced framework and accompanying codebase. We attribute the success and failure of these baselines over the spectrum of considered datasets and environments to how *biased* the agent is made (by the offline method) towards positing the optimality of the policy underlying the provided offline dataset for the given task. Approaches that perform well on one end of the spectrum (*e.g.* with expert-grade datasets) typically achieve deterringly low returns on the other end of the spectrum (random-grade datasets), and *vice versa*. Understandably, the hyper-parameters that control the bias injection are the hardest to tune, across the entire range of methods. We looked for the method that achieved the best over the spectrum of considered dataset qualities, and therefore took a advantage-weighted regression template as base. We first studied how this method — well-behaved on the *low*-quality end of the spectrum, subpar relative to

the competing baselines on the *high*-quality end — reacted to the purposeful injection of optimality inductive bias, and how it impacted final performance. Via a toy extension of the base method, we showed just how brutally detrimental the usual direct injection of bias can be on the achieved levels of return when the offline dataset is *sub*-optimal. This empirical evidence constitutes the *second* contribution of this work. As our *third* and *fourth* contributions respectively, we propose generalizations of the policy evaluation and improvement steps, involving 9 distinct proposal distributions over actions. The dual generalization contribution effectively revisits the generalized policy iteration scheme for the offline regime, setting out to understand how to design an offline RL method that enables the agent to ***close in on optimality, while remaining shielded from the distributional shift hindering offline methods***. Notably, in policy evaluation, we showed that (provided the extra information) even a method as simple as *SARSA* with respect to the offline distribution yields surprisingly good and robust results. In policy improvement, the proposed *novel* GIWR framework enables the practitioner to craft the objective that suits the desired level of awareness about the quality of the dataset. The closer to optimality, the more bias should be injected. Contrary to previous works (re-implemented and empirically compared as our first contribution) and the toy extension studied purposely through the lens of inductive bias injection (second contribution), we *can* get gains on one end of the spectrum without hurting performance on the other end. We consistently highlight which proposals seem to perform best in evaluation and improvement respectively, and advocate for their usage in practical scenarios since they enable *improvements without compromise*, despite not being aware of how sub-optimal the offline distribution actually is. The involvement of privileged information about the quality of the dataset (telling *"how optimal"* the underlying policy is) to guide the learning process of offline RL agent is a promising research direction that is likely to have a considerable impact in safety-critical systems — *e.g.* robotics, autonomous driving, healthcare applications.

Appendix

4.A Baird's advantage-learning investigation

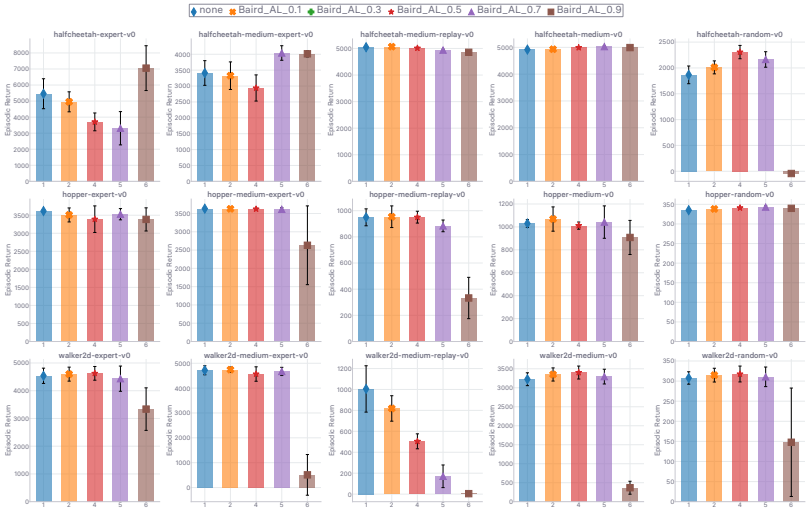


Figure 4.A.1: Empirical evaluation of the use of Baird's advantage-learning bonus (*cf.* 4.29), and sweep over the associated scaling coefficient α. Runtime is 12 hours. Best seen in color.

4.B Proposal involving $\mathcal{T}^{\omega,m}_{\text{Max}}$ in policy evaluation

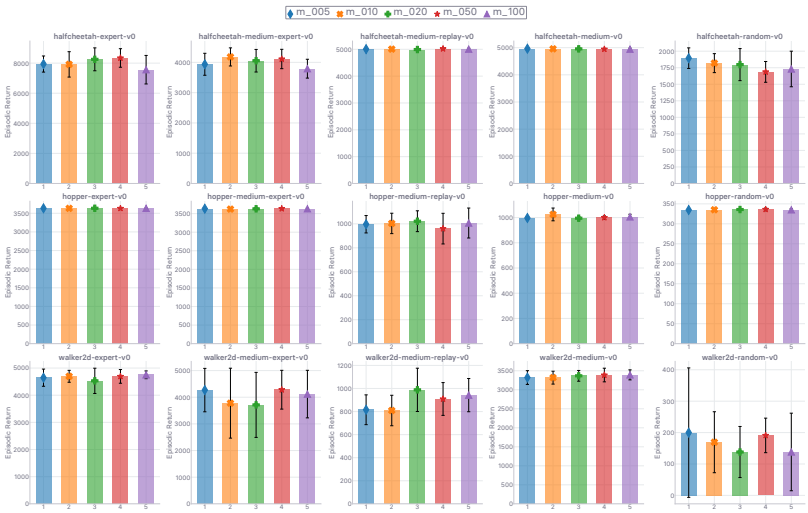


Figure 4.B.1: Sweep over the number of samples m used in the operator $\mathcal{T}^{\omega',m}_{\text{Max}}[\beta_c]$ (*cf.* Section 4.5.1, "beta clone max"). Runtime is 12 hours. Best seen in color.

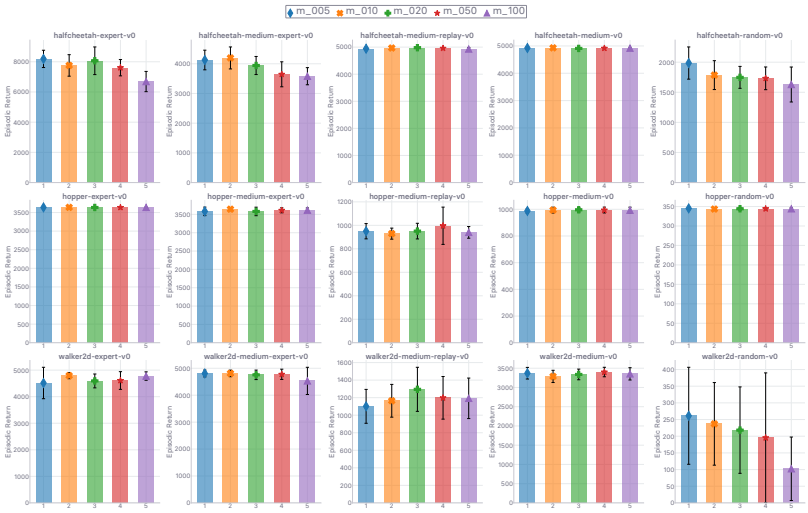


Figure 4.B.2: Sweep over the number of samples m used in the operator $\mathcal{T}^{\omega',m}_{\text{MAX}}[\beta^{\xi}_{\text{C}}]$ (*cf.* SECTION 4.5.1, "PERTURBED BETA CLONE MAX"). Runtime is 12 hours. Best seen in color.

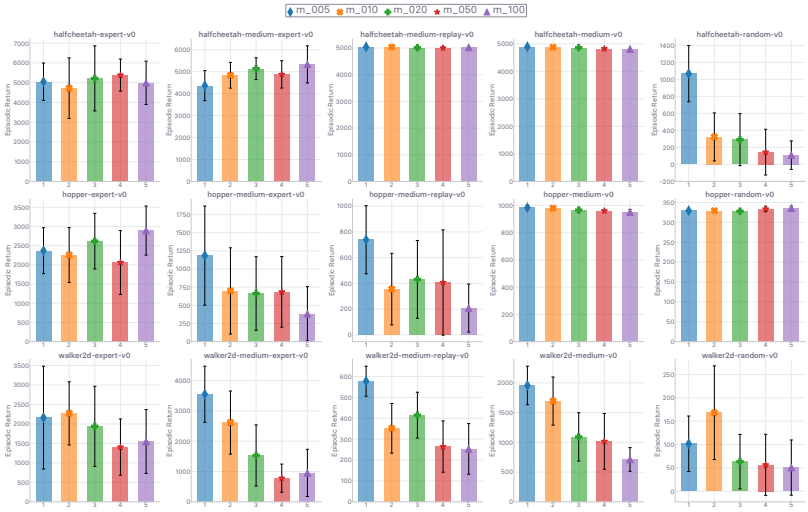


Figure 4.B.3: Sweep over the number of samples m used in the operator $\mathcal{T}^{\omega',m}_{\text{MAX}}[\pi_\theta]$ (*cf.* SECTION 4.5.1, "THETA MAX"). Runtime is 12 hours. Best seen in color.

4.C Policy improvement objective derivation

We begin with the *forward* KL, by unpacking the measure and the expectations into explicit integral form, and injecting eq. 4.46:

$$\mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[\Delta\big(\pi_\theta(\cdot|s),\zeta_{\text{IW}}(\cdot|s)\big)\Big] \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[D^{\zeta_{\text{IW}}}_{\overrightarrow{\text{KL}}}[\pi_\theta](s)\Big] \tag{4.59}$$

$$= \int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in c\mathcal{A}}\zeta_{\text{IW}}(a|s)\big(\log\zeta_{\text{IW}}(a|s)-\log\pi_\theta(a|s)\big)\,da\,ds \tag{4.60}$$

$$= \int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in c\mathcal{A}}\zeta_{\text{IW}}(a|s)\log\zeta_{\text{IW}}(a|s)\,da\,ds - \int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in c\mathcal{A}}\zeta_{\text{IW}}(a|s)\log\pi_\theta(a|s)\,da\,ds \tag{4.61}$$

$$\implies \quad \theta \coloneqq \operatorname*{argmin}_{\theta\in\Theta}\ \mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[\Delta\big(\pi_\theta(\cdot|s),\zeta_{\text{IW}}(\cdot|s)\big)\Big] \tag{4.62}$$

$$= \operatorname*{argmin}_{\theta\in\Theta}\ -\int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in c\mathcal{A}}\zeta_{\text{IW}}(a|s)\log\pi_\theta(a|s)\,da\,ds \tag{4.63}$$

$$= \operatorname*{argmin}_{\theta\in\Theta}\ -\int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in c\mathcal{A}}\zeta(a|s)\exp\left(\frac{1}{\lambda_{\text{KL}}}A^{\pi_\theta}_\omega(s,a)\right)\log\pi_\theta(a|s)\,da\,ds \tag{4.64}$$

$$= \operatorname*{argmax}_{\theta\in\Theta}\ \mathbb{E}_{s\sim\rho^\beta(\cdot),a\sim\zeta(\cdot|s)}\left[\exp\left(\frac{1}{\lambda_{\text{KL}}}A^{\pi_\theta}_\omega(s,a)\right)\log\pi_\theta(a|s)\right] \tag{4.65}$$

Conversely, by opting for the *reverse* KL instead, the problem in eq. 4.46 reduces to the following problem:

$$\mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[\Delta\big(\pi_\theta(\cdot|s),\zeta_{\text{IW}}(\cdot|s)\big)\Big] \coloneqq \mathbb{E}_{s\sim\rho^\beta(\cdot)}\Big[D^{\zeta_{\text{IW}}}_{\overleftarrow{\text{KL}}}[\pi_\theta](s)\Big] \tag{4.66}$$

$$= \int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in c\mathcal{A}}\pi_\theta(a|s)\big(\log\zeta_{\text{IW}}(a|s)-\log\pi_\theta(a|s)\big)\,da\,ds \tag{4.67}$$

$$= \int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in c\mathcal{A}}\pi_\theta(a|s)\log\zeta_{\text{IW}}(a|s)\,da\,ds - \int_{s\in\mathcal{S}}\rho^\beta(s)\int_{a\in c\mathcal{A}}\pi_\theta(a|s)\log\pi_\theta(a|s)\,da\,ds \tag{4.68}$$

$$\implies \quad \theta := \operatorname*{argmin}_{\theta \in \Theta} \; \mathbb{E}_{s \sim \rho^\beta(\cdot)} \Big[\Delta\big(\pi_\theta(\cdot|s), \zeta_{\text{IW}}(\cdot|s)\big) \Big] \tag{4.69}$$

$$= \operatorname*{argmin}_{\theta \in \Theta} \; \int_{s \in \mathcal{S}} \rho^\beta(s) \int_{a \in \mathcal{A}} \pi_\theta(a|s) \log \left(\zeta(a|s) \exp\left(\frac{1}{\lambda_{\text{KL}}} A^{\pi_\theta}_\omega(s,a) \right) \right) da \, ds$$

$$- \int_{s \in \mathcal{S}} \rho^\beta(s) \int_{a \in \mathcal{A}} \pi_\theta(a|s) \log \pi_\theta(a|s) \, da \, ds \tag{4.70}$$

$$= \operatorname*{argmin}_{\theta \in \Theta} \; \mathbb{E}_{s \sim \rho^\beta(\cdot), a \sim \pi_\theta(\cdot|s)} \left[\log \zeta(a|s) + \frac{1}{\lambda_{\text{KL}}} A^{\pi_\theta}_\omega(s,a) \right] + \mathbb{E}_{s \sim \rho^\beta(\cdot)} \big[H\big(\pi_\theta(\cdot|s)\big) \big] \tag{4.71}$$

where $H\big(\pi_\theta(\cdot|s)\big)$ denotes the entropy of π_θ for a given state s.

4.D Generalized Importance-Weighted Regression sweep

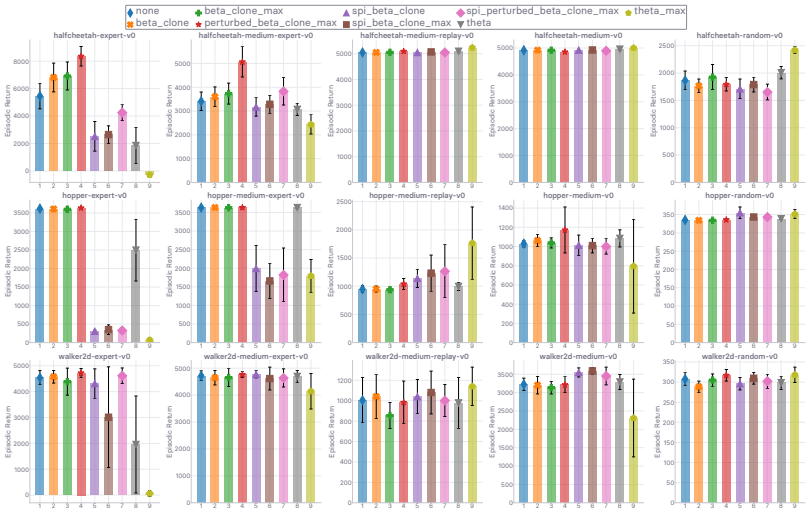


Figure 4.D.1: Empirical comparison of how the proposal distributions introduced in Section 4.5.3 impact the final performance of GIWR (*cf.* Algorithm 6). Everything except the proposal policy ζ in use is identical. We use $\kappa = 0.1$ as scaling coefficient for the contribution of ζ in eq. 4.58. Runtime is 12 hours. Best seen in color.

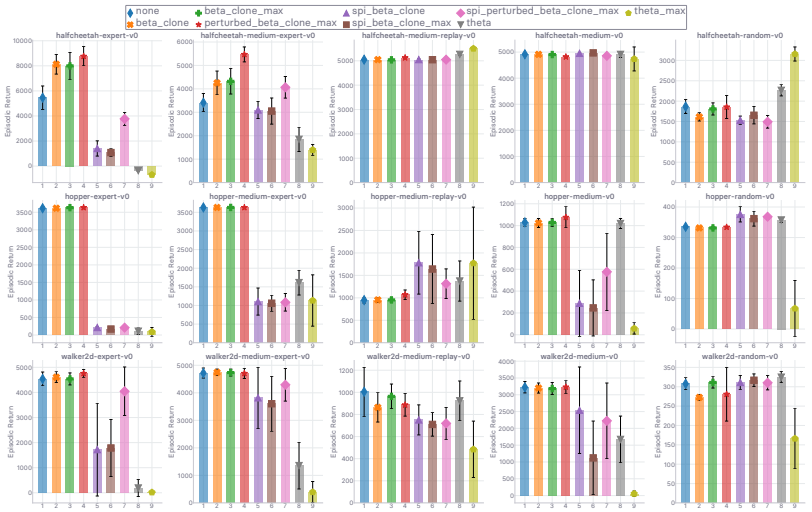


Figure 4.D.2: Empirical comparison of how the proposal distributions introduced in SECTION 4.5.3 impact the final performance of GIWR (*cf.* ALGORITHM 6). Everything except the proposal policy ζ in use is identical. We use $\kappa = 0.5$ as scaling coefficient for the contribution of ζ in EQ. 4.58. Runtime is 12 hours. Best seen in color.

4.E TEMPERATURE SWEEP IN AWR

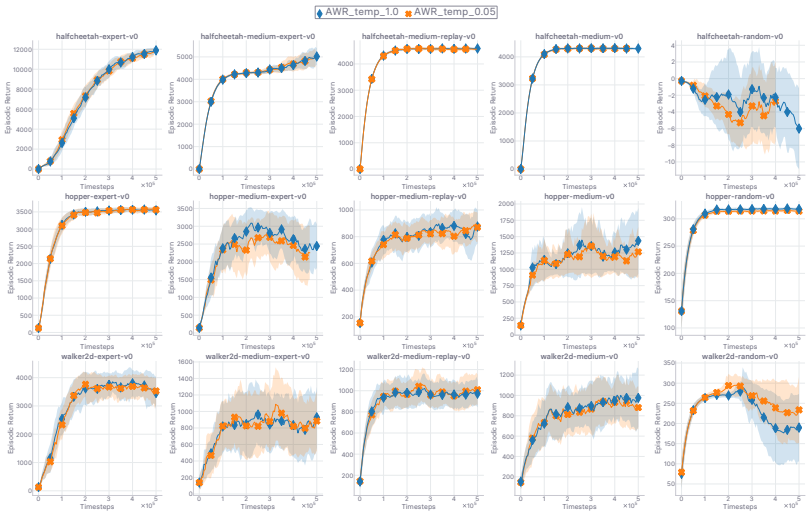


Figure 4.E.1: Sweep over the temperature τ used in the advantage-based exponential weights objective of AWR. Note some sets of runs (*e.g.* top-right sub-plot) terminated early due to an issue on our computational infrastructure. Since the results were conveying the message we wanted to communicate (the temperature has little to no impact on performance), we did not deem it necessary to re-run these experiments. Runtime is 12 hours. Best seen in color.

Chapter 5

Final thoughts

We have studied two ways of teaching an artificial agent to learn to interact with a world from which it does not get direct feedback on its evolving decision process. In imitation learning, the decision maker receives demonstrations from an expert, assumed to behave optimally; it can interact with its environment but does not receive the reward response upon execution. The decision-maker does not know how its own behavior would have been rated by the environment throughout the learning process, and assumes that the demonstrations it is provided with are what said environment would rate as optimal. We have treated this setting in Chapters 2 and 3. Similarly, when reinforcement learning is brought offline, the decision maker is never explicitly told how well it does at completing the interactive task because it is never allowed to interact with its environment in the first place. Instead of being rated directly about its own behavior, it is made aware of how another agent was rated by the environment when attempting to complete the same task. Note, in imitation learning, the demonstrations are not annotated with rewards like in offline RL, hence leaving the decision maker with the sole choice of having to interpret the demonstrations as optimal. In offline RL however, the provided traces of interactions from another agent are rated with rewards, and as such, the content of the the provided offline dataset need not be interpreted as optimal. In Chapters 4, we have treated this setting and shown that seeing the datasets as optimal (despite being annotated with rewards that would enable a more agnostic approach with respect to the offline dataset's optimality) leads to poor performance compared to agnostic approaches when the dataset contains sub-optimal data.

Learning an interactive strategy without direct feedback on the decisions made, a setting we identified as an instance of counterfactual learning, has proven challenging, and is by no means a solved problem. As soon as the decision maker has limited access to the environment (be it the lack of direct reward feedback upon interaction, or the inability to interact altogether) its ability to learn to complete an

interactive task in the world is often significantly impaired. Leveraging datasets of logged interactions captured from another agent exposes the agents to hindering stability issues due to distributional mismatches occurring during the learning process. Still, considering the ubiquity and mass production of various types of sensors, aggregating data into feature-rich datasets of logged interactions is convenient and cheap. It is also far safer than the online (artificial counterpart of *"in-person"*), interactive alternative: learning by acting in the real world. The exploration involved in reinforcement learning quickly makes the entire learning process hazardous both for the artificial decision maker (*e.g.* autonomous vehicle, robot) and its environment (*e.g.* pedestrians, healthcare patients). Looking for a digital surrogate to real-world interactions, the RL practitioner has naturally turned to training decision maker in simulation. Simulators offer valuable flexibility when it comes to exposing the agent to a wide and diverse array of situations that the agent would only rarely be faced by in the real world (*e.g* unusual pavement coloration, signs being held by road workers in the occurrence of a road collision).

High-fidelity, photo-realistic simulators that would best prepare the agent to face the real world, are nevertheless incredibly difficult and finicky to craft. Besides, they are usually used in tandem with augmented datasets of logged interactions from a multitude of agents. Challenges of such combination of simulation and (augmented) real-world data include: *can the crafted simulator represent the data captured from the real world in simulation without mismatch? How should the reward function underlying the observed behavior be designed to entice an agent learned in simulation to reproduce the same behavior? How does one enable the artificial agent to deal with hazardous situations if everything that has ever being recorded corresponds to safe behaviors? How should one bake in the multi-agent and social aspect of real-world interactions in simulation?* The social aspect of most real-world scenarios (in which RL seems to be the right paradigm to teach an AI) is particularly tedious to deal with considering how abstract concepts such as self-awareness enabling the agent to recognize its counterparts in the world can not be easily inferred from sensory recordings. By being able to identify an entity moving in its environment as another learning agent, like itself, the agent would be able to forecast what this external agent would do if it were to follow the main agent's decision process. By being able to forecast how such multi-agent environment might evolve in the next few seconds or minutes, the agent is likely to make better decisions overall. As such, designing social agents able to predict how other actors might act and react to any of the other actors' decisions is an important line of research to pursue, which can take advantage of the relational learning capabilities of recent attention-based research advances. Examples of such endeavors include [BKSB13, AGR+16, LZL+17, TZLB17, VMO17, LVD+18, GJFF+18, KSMM+19, ZLV+20, NELJ20], where self-driving is focused most.

We can expect high-fidelity simulators to become better and better as time goes on, closing the gap between learning in simulation and real-world interaction. We can notably expect these to encode real-world physics with an increasing accuracy, to the point where the physics might even be baked

into the decision maker itself. An example of the exploitation of such physics-originated properties is the explicit use of clothoids in [ZLS+20], the curves described by vehicles when turning the steering wheel at constant angular speed. These curves are instrumental in high-speed road design to join roads where the steering wheel is locked at a zero angle (the trajectory describes a straight line) and roads where the steering wheel is locked at a non-zero angle (the trajectory describes a circle). Clothoids are used because they translate to the most comfortable transition between these two locked steering wheel positions for the driver. They are also known as Euler spirals, or Cornu spirals is optics.

As we close the gap between simulated and real-world reinforcement learning, it will prove increasingly valuable to adopt a multi-agent lens and provide the agent (or agents) with mechanisms that enable it (or them) to display social traits and manifest relational reasoning. Decentralized multi-agent RL, where every involved decision maker is independent, has the best horizontal scalability (compared to its centralized counterpart), but is also hindered by an extra challenge: from any of the independent agent's perspective, the environment (containing the other independent agents) is constantly changing. The non-stationarity of the respective environments every independent agent interacts with makes the task considerably more tedious to tackle empirically and violates the Markov property on which most convergence guarantees rely. It is common to base the agent's decisions on a history of states (or on a summary thereof) in such scenarios, to palliate the non-stationarity of the environment's dynamics from any agent's point of view. Sequential or memory-augmented models might provide the agent with the ability to infer the other agents' intents when the current state is not a sufficient statistic of the future. We can therefore expect progress in general sequential modeling to assist progress in these challenging multi-agent settings. As the simulated environments adopt structures that are increasingly more truthful in complexity to the real world, which is inherently non-stationarity, research progress on transferring models learned in simulation in the real world (*"sim-to-real"*) will without a doubt play a significant role in closing the simulation-reality gap over time.

Still, once we succeed in transferring agents trained in simulation into the real world, leveraging both captured data and high-fidelity simulators, the taught artificial agent must be adopted by the human user and be appealing to interact with. Humans effectively are agents in the multi-agent system, and as such, aspects from human-computer interfaces must be considered to integrate learned RL agent in the daily activities of the human end users. Imitation learning can play a significant role in making interactions with artificial agents more natural and human-like, as given an account of in [Liv06, TBDLM10], tackling *believability* in the context of AI in games. Both conversational and embodied agents can be made more believable, human-like, and therefore seemingly trustworthy by enabling the artificial decision maker to manifest interpersonal *synchrony* (rhythmic coordination of actions, emotions, speech) with its human user, as synchrony plays an essential role in the way humans bond.

Bibliography

[AB17] Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[ABW+19] Anurag Ajay, Maria Bauza, Jiajun Wu, Nima Fazeli, Joshua B Tenenbaum, Alberto Rodriguez, and Leslie P Kaelbling. Combining Physical Simulators and Object-Based Networks for Control. April 2019.

[AC16] Dario Amodei and Jack Clark. Faulty Reward Functions in the Wild. `https://openai.com/blog/faulty-reward-functions/`, December 2016. Accessed: NA-NA-NA.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. January 2017.

[ACBFS95] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The non-stochastic multi-armed bandit problem. *Symposium on Foundations of Computer Science*, 1995.

[AGO19] Peter Auer, Pratik Gajane, and Ronald Ortner. Adaptively Tracking the Best Bandit Arm with an Unknown Number of Distribution Changes. In *Conference on Learning Theory (COLT)*, 2019.

[AGR+16] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[AHH+20] Abbas Abdolmaleki, Sandy H Huang, Leonard Hasenclever, Michael Neunert, H Francis Song, Martina Zambelli, Murilo F Martins, Nicolas Heess, Raia Hadsell, and Martin Riedmiller. A Distributional View on Multi-Objective Policy Optimization. May 2020.

[AK16a] Sherief Abdallah and Michael Kaisers. Addressing environment non-stationarity by repeating Q-learning updates. *Journal of Machine Learning Research (JMLR)*, 17(1):1582–1612, January 2016.

[AK16b] Oren Anava and Zohar Karnin. Multi-armed Bandits: Competing with Optimal Sequences. In *Neural Information Processing Systems (NeurIPS)*, 2016.

[AKA19] Joshua Achiam, Ethan Knight, and Pieter Abbeel. Towards Characterizing Divergence in Deep Q-Learning. March 2019.

[AKA+20] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: Offline Primitive Discovery for Accelerating Offline Reinforcement Learning. October 2020.

[Ama16] Shun-Ichi Amari. *Information Geometry and Its Applications*. Applied Mathematical Sciences. Springer Japan, 2016.

[AN04] Pieter Abbeel and Andrew Y Ng. Apprenticeship Learning via Inverse Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2004.

[AOS+16] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. June 2016.

[AS97] Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In *International Conference on Machine Learning (ICML)*, volume 97, pages 12–20, 1997.

[ASM07] András Antos, Csaba Szepesvári, and Rémi Munos. Fitted Q-iteration in continuous action-space MDPs. In *Neural Information Processing Systems (NeurIPS)*, volume 20, 2007.

[ASN20] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An Optimistic Perspective on Offline Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2020.

[AST+18] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a Posteriori Policy Optimisation. In *International Conference on Learning Representations (ICLR)*, 2018.

[AWR+17] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. July 2017.

[AYBS13] Yasin Abbasi-Yadkori, Peter L Bartlett, and Csaba Szepesvari. Online Learning in Markov Decision Processes with Adversarially Chosen Transition Probability Distributions. March 2013.

[BACM17] Nir Baram, Oron Anschel, Itai Caspi, and Shie Mannor. End-to-End Differentiable Adversarial Imitation Learning. In *International Conference on Machine Learning (ICML)*, pages 390–399, 2017.

[Bag15] J Andrew Bagnell. An invitation to imitation. Technical report, Carnegie Mellon, Robotics Institute, Pittsburgh, 2015.

[Bai93] Leemon C Baird, III. Advantage Updating. Technical report, Wright Laboratory, 1993.

[Bai99] Leemon C Baird, III. *Reinforcement Learning Through Gradient Descent*. PhD thesis, 1999.

[Bar13] Andrew G Barto. Intrinsic Motivation and Reinforcement Learning. In *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 17–47. Springer Berlin Heidelberg, 2013.

[BCDS08] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot Programming by Demonstration. In Siciliano Bruno and Khatib Oussama, editors, *Springer Handbook of Robotics*, pages 1371–1394. Springer Berlin Heidelberg, 2008.

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. June 2016.

[BD62] Richard E Bellman and Stuart E Dreyfus. *Applied Dynamic Programming*. Princeton University Press, 1962.

[BDM17] Marc G Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning. July 2017.

[BDTD+16] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. April 2016.

[Bel57] Richard E Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[Ber00] Dimitri P Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2000.

[BESK18] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. October 2018.

[BGB20] Jacob Buckman, Carles Gelada, and Marc G Bellemare. The Importance of Pessimism in Fixed-Dataset Policy Optimization. September 2020.

[BGZ14] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic Multi-Armed-Bandit Problem with Non-stationary Rewards. In *Neural Information Processing Systems (NeurIPS)*, 2014.

[Bie20] Lukas Biewald. Experiment Tracking with Weights and Biases, 2020.

[Bis95] Chris M Bishop. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Comput.*, 7(1):108–116, January 1995.

[Bis06] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BIW+17] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. September 2017.

[BK19] Lionel Blondé and Alexandros Kalousis. Sample-Efficient Imitation Learning via Generative Adversarial Nets. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. July 2016.

[BKSB13] W Bradley Knox, Peter Stone, and Cynthia Breazeal. Training a Robot via Human Feedback: A Case Study. In *Social Robotics*, pages 460–470. Springer, Cham, October 2013.

[Bla62] David Blackwell. Discrete Dynamic Programming. *Ann. Math. Statist.*, 33(2):719–726, June 1962.

[BLC13] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. August 2013.

[BMHB+18] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, T B Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed Distributional Deterministic Policy Gradients. In *International Conference on Learning Representations (ICLR)*, 2018.

[BMSS16] André Barreto, Rémi Munos, Tom Schaul, and David Silver. Successor Features for Transfer in Reinforcement Learning. June 2016.

[BOG+15] Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip S Thomas, and Rémi Munos. Increasing the Action Gap: New Operators for Reinforcement Learning. In *Conference on Artificial Intelligence (AAAI)*, 2015.

[BPMP17] Diana Borsa, Bilal Piot, Rémi Munos, and Olivier Pietquin. Observational Learning by Reinforcement Learning. In *Neural Information Processing Systems (NeurIPS)*, June 2017.

[BPQC+13] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: the example of computational advertising. *Journal of Machine Learning Research*, 2013.

[BSGL07] Shalabh Bhatnagar, Richard S Sutton, Mohammad Ghavamzadeh, and Mark Lee. Incremental Natural Actor-Critic Algorithms. In *Neural Information Processing Systems (NeurIPS)*, 2007.

[BSO+16] Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. June 2016.

[BT96] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996.

[BXCVdP18] Glen Berseth, Cheng Xie, Paul Cernek, and Michiel Van de Panne. Progressive Reinforcement Learning with Distillation for Multi-Skilled Motion Control. In *International Conference on Learning Representations (ICLR)*, 2018.

[CB95] Robert H Crites and Andrew G Barto. An Actor/Critic Algorithm that Equivalent to Q-Learning. In *Neural Information Processing Systems (NeurIPS)*, 1995.

[CCN+19] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, Oleg Sushkov, David Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. September 2019.

[CG87] Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.

[CLLW19] Yifang Chen, Chung-Wei Lee, Haipeng Luo, and Chen-Yu Wei. A New Algorithm for Non-stationary Contextual Bandits: Efficient, Optimal, and Parameter-free. In *Conference on Learning Theory (COLT)*, 2019.

[CML+18] F Codevilla, M Müller, A López, V Koltun, and A Dosovitskiy. End-to-End Driving Via Conditional Imitation Learning. In *International Conference on Robotics and Automation (ICRA)*, pages 4693–4700, May 2018.

[CSKX15] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *International Conference on Computer Vision (ICCV)*, pages 2722–2730, 2015.

[CSLZ19a] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Learning to Optimize under Non-Stationarity. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[CSLZ19b] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Reinforcement Learning under Drift. June 2019.

[CWW19] Maciek Chociej, Peter Welinder, and Lilian Weng. ORRB – OpenAI Remote Rendering Backend. June 2019.

[DAS+17] Yan Duan, Marcin Andrychowicz, Bradly C Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-Shot Imitation Learning. In *Neural Information Processing Systems (NeurIPS)*, 2017.

[DCH+16] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking Deep Reinforcement Learning for Continuous Control. In *International Conference on Machine Learning (ICML)*, pages 1329–1338, June 2016.

[DCJ+18] Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting Auxiliary Losses Using Gradient Similarity. December 2018.

[Den67] Eric V Denardo. Contraction Mappings in the Theory Underlying Dynamic Programming. *SIAM Rev.*, 9(2):165–177, 1967.

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised Visual Representation Learning by Context Prediction. In *International Conference on Computer Vision (ICCV)*, May 2015.

[DGS14] Travis Dick, Andras Gyorgy, and Csaba Szepesvari. Online Learning in Markov Decision Processes with Changing Cost Sequences. In *International Conference on Machine Learning (ICML)*, 2014.

[Dic80] Anthony Dickinson. *Contemporary Animal Learning Theory*. Cambridge University Press, 1980.

[DKNU+20] Will Dabney, Zeb Kurth-Nelson, Naoshige Uchida, Clara Kwon Starkweather, Demis Hassabis, Rémi Munos, and Matthew Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675, January 2020.

[DL77] Stuart E Dreyfus and Averill M Law. *Art and Theory of Dynamic Programming*. Academic Press, USA, 1977.

[DOSM18] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit Quantile Networks for Distributional Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2018.

[DPBB17] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp Minima Can Generalize For Deep Nets. In *International Conference on Machine Learning (ICML)*, 2017.

[DPS12] T Degris, P M Pilarski, and R S Sutton. Model-Free reinforcement learning with continuous action in practice. In *American Control Conference (ACC)*, pages 2177–2182, June 2012.

[DRBM17] Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional Reinforcement Learning with Quantile Regression. October 2017.

[DSBBE06] Bruno C Da Silva, Eduardo W Basso, Ana L C Bazzan, and Paulo M Engel. Dealing with Non-Stationary Environments using Context Detection. In *International Conference on Machine Learning (ICML)*, 2006.

[DWS12] Thomas Degris, Martha White, and Richard S Sutton. Off-Policy Actor-Critic. In *International Conference on Machine Learning (ICML)*, 2012.

[EdKM05] Eyal Even-dar, Sham M Kakade, and Yishay Mansour. Experts in a Markov Decision Process. In *Neural Information Processing Systems (NeurIPS)*, 2005.

[EGW05] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research (JMLR)*, 6:503–556, 2005.

[EKO+17] Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. Reinforcement Learning with a Corrupted Reward Channel. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[FAP+17] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy Networks for Exploration. June 2017.

[FB10] Thomas Furmston and David Barber. Variational methods for reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[FCAL16] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models. November 2016.

[FCAO18] Chris Finlay, Jeff Calder, Bilal Abbasi, and Adam Oberman. Lipschitz regularized Deep Neural Networks generalize and are adversarially robust. August 2018.

[FHDV20] Joseph Futoma, Michael C Hughes, and Finale Doshi-Velez. POPCORN: Partially Observed Prediction COnstrained ReiNforcement Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[FHGS19] J Fernando Hernandez-Garcia and Richard S Sutton. Understanding Multi-Step Deep Reinforcement Learning: A Systematic Study of the DQN Target. January 2019.

[FKN+20] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. April 2020.

[FKSL19] Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing Bottlenecks in Deep Q-learning Algorithms. In *International Conference on Machine Learning (ICML)*, 2019.

[FLL18] Justin Fu, Katie Luo, and Sergey Levine. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[FMP18] Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. December 2018.

[FMWE10] Raphael Fonteneau, Susan A Murphy, Louis Wehenkel, and Damien Ernst. Model-Free Monte Carlo–like Policy Evaluation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[FMWE13] Raphael Fonteneau, Susan A Murphy, Louis Wehenkel, and Damien Ernst. Batch Mode Reinforcement Learning based on the Synthesis of Artificial Trajectories. *Ann. Oper. Res.*, 208(1):383–416, September 2013.

[FPT15] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the Noise in Reinforcement Learning via Soft Updates. December 2015.

[FRL+18] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M Dai, Shakir Mohamed, and Ian Goodfellow. Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence At Every Step. In *International Conference on Learning Representations (ICLR)*, 2018.

[FT91] C T Farley and C R Taylor. A mechanical trigger for the trot-gallop transition in horses. *Science*, 253(5017):306–308, July 1991.

[FvHM18] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning (ICML)*, 2018.

[FW06] David J Foster and Matthew A Wilson. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084):680–683, March 2006.

[GAA+17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. In *Neural Information Processing Systems (NIPS)*, 2017.

[GCW+18] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations (ICLR)*, 2018.

[GDL16] David Sierra González, Jilles Steeve Dibangoye, and Christian Laugier. High-speed highway scene prediction based on driver models learned from demonstrations. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 149–155, November 2016.

[GDL+17] Abhishek Gupta, Coline Devin, Yuxuan Liu, Pieter Abbeel, and Sergey Levine. Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[GF15] Javier Garcia and Fernando Fernandez. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research (JMLR)*, 2015.

[GG19] Shani Gamrian and Yoav Goldberg. Transfer Learning for Related Reinforcement Learning Tasks via Image-to-Image Translation. In *International Conference on Machine Learning (ICML)*, 2019.

[GGT15] Shixiang Gu, Zoubin Ghahramani, and Richard E Turner. Neural Adaptive Sequential Monte Carlo. In *Neural Information Processing Systems (NeurIPS)*, 2015.

[GJFF+18] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. March 2018.

[GJK+19] Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nat. Med.*, 25(1):16–18, January 2019.

[GJM+18] Omer Gottesman, Fredrik Johansson, Joshua Meier, Jack Dent, Donghun Lee, Srivatsan Srinivasan, Linying Zhang, Yi Ding, David Wihl, Xuefeng Peng, Jiayu Yao, Isaac Lage, Christopher Mosch, Li-Wei H Lehman, Matthieu Komorowski, Matthieu Komorowski, Aldo Faisal, Leo Anthony Celi, David Sontag, and Finale Doshi-Velez. Evaluating Reinforcement Learning Algorithms in Observational Health Settings. May 2018.

[GKM+20] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset. April 2020.

[GLDS20] Zhicheng Gu, Zhihao Li, Xuan Di, and Rongye Shi. An LSTM-Based Autonomous Driving Model Using Waymo Open Dataset. February 2020.

[GLG+16] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic. In *International Conference on Learning Representations (ICLR)*, November 2016.

[GLSL16] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous Deep Q-Learning with Model-based Acceleration. March 2016.

[GM11] Aurélien Garivier and Eric Moulines. On Upper-Confidence Bound Policies for Switching Bandit Problems. In *Algorithmic Learning Theory (ALT)*, pages 174–188. Springer Berlin Heidelberg, 2011.

[GOA18] Pratik Gajane, Ronald Ortner, and Peter Auer. A Sliding-Window Algorithm for Markov Decision Processes with Arbitrarily Changing Rewards and Transitions. May 2018.

[Goo17] Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. April 2017.

[Gor95] Geoffrey J Gordon. Stable Function Approximation in Dynamic Programming. In Armand Prieditis and Stuart Russell, editors, *International Conference on Machine Learning (ICML)*, pages 261–268, San Francisco (CA), 1995.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Neural Information Processing Systems (NIPS)*, 2014.

[GPL+20] Jianfeng Gao, Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Heung-Yeung Shum. Robust Conversational AI with Grounded Text Generation. September 2020.

[GSG20] Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. EMaQ: Expected-Max Q-Learning Operator for Simple Yet Effective Offline and Online RL. July 2020.

[GSZ+20] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[Gum54] Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*. U. S. Govt. Print. Office, 1954.

[GŽB+14] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A Survey on Concept Drift Adaptation. *ACM Computing Surveys (CSUR)*, 2014.

[Har97] Daishi Harada. Reinforcement Learning with Time. In *Conference on Artificial Intelligence (AAAI)*, 1997.

[HB20] Danny Hernandez and Tom B Brown. Measuring the Algorithmic Efficiency of Neural Networks. May 2020.

[HBLH19] Jonathan J Hunt, Andre Barreto, Timothy P Lillicrap, and Nicolas Heess. Composing Entropic Policies using Divergence Correction. In *International Conference on Machine Learning (ICML)*, 2019.

[HCS+17] Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph Lim. Multi-Modal Imitation Learning from Unstructured Demonstrations using Generative Adversarial Nets. In *Neural Information Processing Systems (NIPS)*, 2017.

[HE16] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. In *Neural Information Processing Systems (NIPS)*, 2016.

[Hec79] James J Heckman. Sample Selection Bias as a Specification Error. *Econometrica*, 47(1):153–161, 1979.

[HGE16] Jonathan Ho, Jayesh K Gupta, and Stefano Ermon. Model-Free Imitation Learning with Policy Optimization. In *International Conference on Machine Learning (ICML)*, 2016.

[HIB+18] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning that Matters. In *AAAI Conference on Artificial Intelligence*, 2018.

[HLG05] James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data. In *IJCAI Workshop "Knowledge and Reasoning in Practical Dialogue Systems"*, 2005.

[HLSP17] Frank S He, Yang Liu, Alexander G Schwing, and Jian Peng. Learning to Play in a Day: Faster Deep Reinforcement Learning by Optimality Tightening. In *International Conference on Learning Representations (ICLR)*, 2017.

[HMC+21] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. NeuralSim: Augmenting Differentiable Simulators with Neural Networks. In *International Conference on Robotics and Automation (ICRA)*, 2021.

[HMvH+17] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. October 2017.

[HMZ+17] David Held, Zoe McCarthy, Michael Zhang, Fred Shentu, and Pieter Abbeel. Probabilistically Safe Policy Transfer. May 2017.

[How71] Ronald A Howard. *Dynamic Probabilistic Systems: Semi-Markov and Decision Processes, Volume 2*. Wiley, John, New York, 1971.

[HR11] Roland Hafner and Martin Riedmiller. Reinforcement learning in feedback control. *Mach. Learn.*, 84(1-2):137–169, July 2011.

[HRS15] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. September 2015.

[HRX+20] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer. November 2020.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Comput*, 9(1):1–42, January 1997.

[HS17] Josiah P Hanna and Peter Stone. Grounded Action Transformation for Robot Learning in Simulation. In *AAAI Conference on Artificial Intelligence*, 2017.

[HT81] Donald F Hoyt and C Richard Taylor. Gait and the energetics of locomotion in horses. *Nature*, 292(5820):239–240, July 1981.

[HTAL17] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement Learning with Deep Energy-Based Policies. February 2017.

[HWS+15] Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning Continuous Control Policies by Stochastic Value Gradients. In *Neural Information Processing Systems (NIPS)*, 2015.

[HXS+20] Junning Huang, Sirui Xie, Jiankai Sun, Qiurui Ma, Chunxiao Liu, Dahua Lin, and Bolei Zhou. Learning a Decision Module by Imitating Driver's Control Behaviors. In *Conference on Robot Learning (CoRL)*, 2020.

[HXT+20] Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to Walk in the Real World with Minimal Human Effort. February 2020.

[HZAL18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning (ICML)*, 2018.

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. February 2015.

[Iva70] A G Ivakhnenko. Heuristic self-organization in problems of engineering cybernetics. *Automatica*, 6(2):207–219, March 1970.

[JGP17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations (ICLR)*, 2017.

[JMC+16] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement Learning with Unsupervised Auxiliary Tasks. November 2016.

[JOA10] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal Regret Bounds for Reinforcement Learning. *Journal of Machine Learning Research (JMLR)*, 11:1563–1600, 2010.

[JPS+16] Alistair E W Johnson, Tom J Pollard, Lu Shen, Li-Wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):160035, May 2016.

[JSG+20] Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric Dialog Training via Offline Reinforcement Learning. October 2020.

[JWK+18] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks. December 2018.

[KAD+19] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[Kae93] Leslie Pack Kaelbling. Learning to Achieve Goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1993.

[KAHK17] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. How to train your DRAGAN. May 2017.

[Kak01] Sham M Kakade. A Natural Policy Gradient. In *Neural Information Processing Systems (NIPS)*, pages 1531–1538, 2001.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. December 2014.

[KFTL19] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[KGO12] Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Mach. Learn.*, 87(2):159–182, May 2012.

[KHJ+19] A Kendall, J Hawke, D Janz, P Mazur, D Reda, J Allen, V Lam, A Bewley, and A Shah. Learning to Drive in a Day. In *International Conference on Robotics and Automation (ICRA)*, pages 8248–8254, May 2019.

[Kim07] Hajime Kimura. Reinforcement learning in multi-dimensional state-action space using random rectangular coarse coding and gibbs sampling. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, September 2007.

[KIP+18] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. June 2018.

[Kir04] Donald E Kirk. *Optimal Control Theory: An Introduction*. Courier Corporation, January 2004.

[KK17] Alex Kuefler and Mykel J Kochenderfer. Burn-In Demonstrations for Multi-Modal Imitation Learning. October 2017.

[KL02] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 2, pages 267–274, 2002.

[KLZ+18] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The GAN Landscape: Losses, Architectures, Regularization, and Normalization. July 2018.

[KMN+17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *International Conference on Learning Representations (ICLR)*, 2017.

[KOP10] J Kober, E Oztop, and J Peters. Reinforcement learning to adjust robot movements to new situations. In *Robotics: Science and Systems (RSS)*. Robotics: Science and Systems Foundation, June 2010.

[KP08] Jens Kober and Jan Peters. Policy Search for Motor Primitives in Robotics. In *Neural Information Processing Systems (NeurIPS)*, 2008.

[KRNJ20] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL : Model-Based Offline Reinforcement Learning. May 2020.

[KSMM+19] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S Hamid Rezatofighi, and Silvio Savarese. Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[KT00] Vijay R Konda and John N Tsitsiklis. Actor-Critic Algorithms. In *Neural Information Processing Systems (NIPS)*, pages 1008–1014, 2000.

[KT03] Vijay Konda and John N Tsitsiklis. On Actor-Critic Algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, January 2003.

[KVDP12] Andrej Karpathy and Michiel Van De Panne. Curriculum learning for motor skills. In *Advances in Artificial Intelligence (Canadian Conference on Artificial Intelligence)*, pages 325–330, May 2012.

[KW14] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[KZLA16] Gregory Kahn, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. PLATO: Policy Learning using Adaptive Trajectory Optimization. March 2016.

[KZTL20] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. June 2020.

[LAS92] T Ljungberg, P Apicella, and W Schultz. Responses of monkey dopamine neurons during learning of behavioral reactions. *J. Neurophysiol.*, 67(1):145–163, January 1992.

[LGR12] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch Reinforcement Learning. In Marco Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 45–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[LHP+16] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[Lin92] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8(3):293–321, May 1992.

[Lin93] Long-Ji Lin. *Reinforcement learning for robots using neural networks*. PhD thesis, 1993.

[Liv06] Daniel Livingstone. Turing's Test and Believable AI in Games. *Comput. Entertain.*, 4(1), January 2006.

[LJL+18] Sungsu Lim, Ajin Joseph, Lei Le, Yangchen Pan, and Martha White. Actor-Expert: A Framework for using Q-learning in Continuous Action Spaces. In *NeurIPS Workshop "Deep Reinforcement Learning"*, October 2018.

[LKM+17] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs Created Equal? A Large-Scale Study. November 2017.

[LKTF20] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. May 2020.

[LP03] Michail G Lagoudakis and Ronald Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, 2003.

[LPK11] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear Inverse Reinforcement Learning with Gaussian Processes. In *Neural Information Processing Systems (NIPS)*, pages 19–27, 2011.

[LPPM+20] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter Selection for Offline Reinforcement Learning. July 2020.

[LR85] T L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1):4–22, March 1985.

[LR10] S Lange and M Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2010.

[LR19] Erwan Lecarpentier and Emmanuel Rachelson. Non-Stationary Markov Decision Processes, a Worst-Case Approach using Model-Based Reinforcement Learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[LSE17] Yunzhu Li, Jiaming Song, and Stefano Ermon. InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations. In *Neural Information Processing Systems (NIPS)*, 2017.

[LSW19] Yingdong Lu, Mark Squillante, and Chai Wah Wu. A Family of Robust Stochastic Operators for Reinforcement Learning. In *Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[LTdC19] Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe Policy Improvement with Baseline Bootstrapping. In *International Conference on Machine Learning (ICML)*, 2019.

[LVD+18] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning Deep Generative Models of Graphs. March 2018.

[LWAL18] Haipeng Luo, Chen-Yu Wei, Alekh Agarwal, and John Langford. Efficient Contextual Bandits in Non-stationary Worlds. In *Conference on Learning Theory (COLT)*, 2018.

[LXM13] Shiau Hong Lim, Huan Xu, and Shie Mannor. Reinforcement Learning in Robust Markov Decision Processes. In *Neural Information Processing Systems (NeurIPS)*, 2013.

[LXT+18] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the Loss Landscape of Neural Nets. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[LZL+17] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent Reinforcement Learning in Sequential Social Dilemmas. February 2017.

[Mac75] N J Mackintosh. Blocking of conditioned suppression: role of the first compound trial. *J. Exp. Psychol. Anim. Behav. Process.*, 1(4):335–345, October 1975.

[Mac83] Nicholas John Mackintosh. *Conditioning and associative learning*. Clarendon Press Oxford, 1983.

[Mac03] David J C MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, September 2003.

[MARW16] Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. Multi-Objective Deep Reinforcement Learning. October 2016.

[MD18] Marcelo Gomes Mattar and Nathaniel D Daw. Prioritized memory access explains planning and hippocampal replay. May 2018.

[MDS96] P R Montague, P Dayan, and T J Sejnowski. A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *J. Neurosci.*, 16(5):1936–1947, March 1996.

[MGN18] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which Training Methods for GANs do actually Converge? In *International Conference on Machine Learning (ICML)*, 2018.

[MHN13] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *International Conference on Machine Learning (ICML)*, 2013.

[MIJD17] Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete Sequential Prediction of Continuous Actions for Deep RL. May 2017.

[Min61] Marvin Minsky. Steps toward Artificial Intelligence. *Proceedings of the IRE*, 49(1):8–30, January 1961.

[MKH19] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Neural Information Processing Systems (NeurIPS)*, 2019.

[MKKY18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[MKL+20] Louis Monier, Jakub Kmec, Alexandre Laterre, Thomas Pierrot, Valentin Courgeau, Olivier Sigaud, and Karim Beguir. Offline Reinforcement Learning Hands-On. In *NeurIPS Workshop "Offline Reinforcement Learning"*, 2020.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. December 2013.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[MMT17] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations (ICLR)*, 2017.

[MPD02] José del R Millán, Daniele Posenato, and Eric Dedieu. Continuous-Action Q-Learning. *Mach. Learn.*, 49(2):247–265, November 2002.

[MPLN17] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The Oxford RobotCar dataset. *Int. J. Rob. Res.*, 36(1):3–15, January 2017.

[MPV+16] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to Navigate in Complex Environments. November 2016.

[MRB+19] Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. IRIS: Implicit Reinforcement without Interaction at Scale for Learning Control from Offline Robot Manipulation Data. November 2019.

[MRG03] Shie Mannor, Reuven Rubinstein, and Yohai Gat. The Cross Entropy method for Fast Policy Search. In *International Conference on Machine Learning (ICML)*, 2003.

[MSS18] Ashish Mehta, Adithya Subramanian, and Anbumani Subramanian. Learning End-to-end Autonomous Driving using Guided Auxiliary Supervision. August 2018.

[Mur12] Kevin P Murphy. *Machine Learning, a Probabilistic Perspective*. Adaptive Computation and Machine Learning. MIT Press, 2012.

[NBS10] S Niekum, A G Barto, and L Spector. Genetic Programming for Reward Function Search. *IEEE Trans. Auton. Ment. Dev.*, 2(2):83–90, June 2010.

[NDGL20] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating Online Reinforcement Learning with Offline Datasets. June 2020.

[NEG05] Arnab Nilim and Laurent El Ghaoui. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Oper. Res.*, 53(5):780–798, October 2005.

[NELJ20] Kamal Ndousse, Douglas Eck, Sergey Levine, and Natasha Jaques. Multi-agent Social Reinforcement Learning Improves Generalization. October 2020.

[Neu11] Gerhard Neumann. Variational inference for policy search in changing situations. In *International Conference on Machine Learning (ICML)*, pages 817–824, 2011.

[NHR99] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML)*, pages 278–287, 1999.

[NP08] Gerhard Neumann and Jan Peters. Fitted Q-iteration by advantage weighted regression. In *Neural Information Processing Systems (NeurIPS)*, 2008.

[NPH+18] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta Learn Fast: A New Benchmark for Generalization in RL. April 2018.

[NR00] Andrew Y Ng and Stuart J Russell. Algorithms for Inverse Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, pages 663–670, 2000.

[OAB+18] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning Dexterous In-Hand Manipulation. August 2018.

[OBB+19] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning. December 2019.

[OBPVR16] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep Exploration via Bootstrapped DQN. February 2016.

[OBvdOM17] Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Remi Munos. Count-Based Exploration with Neural Density Models. March 2017.

[OGA19] Ronald Ortner, Pratik Gajane, and Peter Auer. Variational Regret Bounds for Reinforcement Learning. May 2019.

[Ope19] OpenAI. Solving Rubik's Cube with a Robot Hand. 2019.

[OS02] Dirk Ormoneit and Śaunak Sen. Kernel-Based Reinforcement Learning. *Mach. Learn.*, 49(2):161–178, November 2002.

[PBS16] Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[PCG16] Marek Petrik, Yinlam Chow, and Mohammad Ghavamzadeh. Safe Policy Improvement by Minimizing Robust Baseline Regret. July 2016.

[PCZ+20] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning Agile Robotic Locomotion Skills by Imitating Animals. April 2020.

[PGCFB11] Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Trans. Speech Lang. Process.*, 7(3):1–21, June 2011.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[PHD+18] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter Space Noise for Exploration. In *International Conference on Learning Representations (ICLR)*, 2018.

[PJ92] B T Polyak and A B Juditsky. Acceleration of Stochastic Approximation by Averaging. *SIAM J. Control Optim.*, 30(4):838–855, July 1992.

[PKT+18] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow. October 2018.

[PKZL19] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. 2019.

[PMA10] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative Entropy Policy Search. In *AAAI Conference on Artificial Intelligence*, 2010.

[Pom89] Dean Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Neural Information Processing Systems (NIPS)*, pages 305–313, 1989.

[Pom90] Dean Pomerleau. Rapidly Adapting Artificial Neural Networks for Autonomous Navigation. In *Neural Information Processing Systems (NIPS)*, pages 429–435, 1990.

[PPB19] Sindhu Padakandla, K J Prabuchandran, and Shalabh Bhatnagar. Reinforcement Learning in Non-Stationary Environments. May 2019.

[PS07] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine Learning (ICML)*, New York, New York, USA, 2007. ACM Press.

[PS08] Jan Peters and Stefan Schaal. Natural Actor-Critic. *Neurocomputing*, 71(7):1180–1190, March 2008.

[PTLK18] Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time Limits in Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2018.

[Put94] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[PV16] David Pfau and Oriol Vinyals. Connecting Generative Adversarial Networks and Actor-Critic Methods. October 2016.

[PVS05] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural Actor-Critic. In *Machine Learning: ECML 2005*, pages 280–291. Springer Berlin Heidelberg, 2005.

[PW96] Jing Peng and Ronald J Williams. Incremental Multi-Step Q-Learning. *Mach. Learn.*, 22(1-3):283–290, January 1996.

[QJN+18] Deirdre Quillen, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, and Sergey Levine. Deep Reinforcement Learning for Vision-Based Robotic Grasping: A Simulated Comparative Evaluation of Off-Policy Methods. February 2018.

[RA98] Jette Randløv and Preben Alstrøm. Learning to Drive a Bicycle Using Reinforcement Learning and Shaping. In *International Conference on Machine Learning (ICML)*, volume 98, pages 463–471, 1998.

[RAA19] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. November 2019.

[RAW+18] Scott Reed, Yusuf Aytar, Ziyu Wang, Tom Paine, Aäron van den Oord, Tobias Pfaff, Sergio Gomez, Alexander Novikov, David Budden, and Oriol Vinyals. Visual Imitation With a Minimal Adversary. Technical report, Deepmind, 2018.

[RB10] Stéphane Ross and J Andrew Bagnell. Efficient Reductions for Imitation Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[RBS07] Nathan Ratliff, J Andrew Bagnell, and Siddhartha S Srinivasa. Imitation learning for locomotion and manipulation. In *IEEE-RAS International Conference on Humanoid Robots*, pages 392–397, November 2007.

[RCG+15] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy Distillation. November 2015.

[RGB11] Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[RHP+18] Joshua Romoff, Peter Henderson, Alexandre Piché, Vincent Francois-Lavet, and Joelle Pineau. Reward Estimation for Variance Reduction in Deep Reinforcement Learning. In *Conference on Robot Learning (CoRL)*, 2018.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[Rie05] Martin Riedmiller. Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *European Conference on Machine Learning (ECML)*, pages 317–328. Springer Berlin Heidelberg, 2005.

[RLNH17] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing Training of Generative Adversarial Networks through Regularization. In *Neural Information Processing Systems (NeurIPS)*, 2017.

[RML18] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep Imitative Models for Flexible Inference, Planning, and Control. October 2018.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning (ICML)*, 2014.

[RN94] G A Rummery and M Niranjan. On-Line Q-Learning Using Connectionist Systems. Technical report, 1994.

[Rob52] Herbert Robbins. Some Aspects of the Sequential Design of Experiments. *Bull. Amer. Math. Soc.*, 58(5):527–535, 1952.

[Ros83] Sheldon M Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, 1983.

[RTB16] Francisco J R Ruiz, Michalis K Titsias, and David M Blei. The Generalized Reparameterization Gradient. In *Neural Information Processing Systems (NIPS)*, pages 460–468, October 2016.

[RTV13] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference (extended abstract). In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[Rub99] Reuven Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodol. Comput. Appl. Probab.*, 1(2):127–190, September 1999.

[Rus92] John Rust. *Do people behave according to Bellman's principle of optimality?* The Hoover Institution, Stanford University, May 1992.

[Rus94] John Rust. Structural estimation of markov decision processes. In *Handbook of Econometrics*, volume 4, chapter 51, pages 3081–3143. Elsevier, January 1994.

[Rus98] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Conference on Learning Theory (COLT)*, 1998.

[RVC19] Yoan Russac, Claire Vernade, and Olivier Cappé. Weighted Linear Bandits for Non-Stationary Environments. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[RVR+16] Andrei A Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-Real Robot Learning from Pixels with Progressive Nets. October 2016.

[RVWD14] Diederik Marijn Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A Survey of Multi-Objective Sequential Decision-Making. February 2014.

[SAPY17] Ahmad E L Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep Reinforcement Learning framework for Autonomous Driving. *Electronic Imaging*, 2017(19):70–76, 2017.

[SB98] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[SBS08] Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship Learning Using Linear Programming. In *International Conference on Machine Learning (ICML)*, pages 1032–1039, 2008.

[Sch97] Stefan Schaal. Learning from Demonstration. In *Neural Information Processing Systems (NeurIPS)*, 1997.

[Sch10] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2010.

[Sch16] John Schulman. *Optimizing Expectations: From Deep Reinforcement Learning to Stochastic Computation Graphs*. PhD thesis, University of California, Berkeley, 2016.

[SCO+20] Sungryull Sohn, Yinlam Chow, Jayden Ooi, Ofir Nachum, Honglak Lee, Ed Chi, and Craig Boutilier. BRPO: Batch Residual Policy Optimization. February 2020.

[SDM97] W Schultz, P Dayan, and P R Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, March 1997.

[SDZ+19] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. AVID: Learning Multi-Stage Tasks via Pixel-Level Translation of Human Videos. December 2019.

[SG86] Jeffrey C Schlimmer and Richard H Granger, Jr. Incremental Learning from Noisy Data. *Mach. Learn.*, 1986.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[SHS+17] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. December 2017.

[SHWA15] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient Estimation Using Stochastic Computation Graphs. June 2015.

[Sil15] David Silver. Lecture 2: Markov Decision Processes, 2015.

[SKD+19] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. December 2019.

[Ski48] B F Skinner. Superstition in the pigeon. *J. Exp. Psychol.*, 38(2):168–172, April 1948.

[SLB09] Satinder Singh, Richard L Lewis, and Andrew G Barto. Where Do Rewards Come From? 2009.

[SLH+14] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. In *International Conference on Machine Learning (ICML)*, pages 387–395, January 2014.

[SLM+15] John Schulman, Sergey Levine, Philipp Moritz, Michael I Jordan, and Pieter Abbeel. Trust Region Policy Optimization. In *International Conference on Machine Learning (ICML)*, February 2015.

[SLZ+20] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-Driven Behavioral Priors for Reinforcement Learning. November 2020.

[SMAD16] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own Reward: Self-Supervision for Reinforcement Learning. December 2016.

[SMG13] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. December 2013.

[SML+16] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

[SMSM99] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Neural Information Processing Systems (NIPS)*, pages 1057–1063, 1999.

[SPZT18] Liting Sun, Cheng Peng, Wei Zhan, and Masayoshi Tomizuka. A Fast Integrated Planning and Control Framework for Autonomous Driving via Imitation Learning. In *ASME, Series "Dynamic Systems and Control Conference"*. American Society of Mechanical Engineers Digital Collection, November 2018.

[SQAS16] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. In *International Conference on Learning Representations (ICLR)*, November 2016.

[SS08] Umar Syed and Robert E Schapire. A Game-Theoretic Approach to Apprenticeship Learning. In *Neural Information Processing Systems (NIPS)*, pages 1449–1456, 2008.

[SSB+20] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep Doing What Worked: Behavioral Modelling Priors for Offline Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[Sut84] Richard S Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 1984.

[Sut88] Richard S Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3(1):9–44, August 1988.

[Sut96] Richard S Sutton. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. In D Touretzky, M C Mozer, and M Hasselmo, editors, *Neural Information Processing Systems (NeurIPS)*, volume 8, pages 1038–1044. MIT Press, 1996.

[Sut01] Richard S Sutton. Comparing policy-gradient algorithms. `http://incompleteideas.net/papers/SSM-unpublished.pdf`, 2001. Accessed: 2021-6-22.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Klimov Oleg. Proximal Policy Optimization Algorithms. July 2017.

[SYK19] Fumihiro Sasaki, Tetsuya Yohira, and Atsuo Kawaguchi. Sample-Efficient Imitation Learning for Continuous Control. In *International Conference on Learning Representations (ICLR)*, 2019.

[SZ97] Nestor A Schmajuk and B Silvano Zanutto. Escape, Avoidance, and Imitation: A Neural Network Approach. *Adapt. Behav.*, 6(1):63–129, June 1997.

[TBDLM10] Fabien Tencé, Cédric Buche, Pierre De Loor, and Olivier Marc. The Challenge of Believability in Video Games: Definitions, Agents Models and Imitation Learning. September 2010.

[TBS10] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A Generalized Path Integral Control Approach to Reinforcement Learning. *Journal of Machine Learning Research (JMLR)*, 11(Nov):3137–3181, 2010.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, October 2012.

[TFR+17] J Tobin, R Fong, A Ray, J Schneider, W Zaremba, and P Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, September 2017.

[THF+16] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. November 2016.

[Thr95] Sebastian Thrun. An approach to learning mobile robot navigation. *Rob. Auton. Syst.*, 15(4):301–319, October 1995.

[TMM+17] George Tucker, Andriy Mnih, Chris J Maddison, Dieterich Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Neural Information Processing Systems (NIPS)*, 2017.

[TS93] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[TS97] David S Touretzky and Lisa M Saksida. Operant Conditioning in Skinnerbots. *Adapt. Behav.*, 5(3-4):219–247, January 1997.

[TS09] M E Taylor and P Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research (JMLR)*, 2009.

[TTG15] Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High Confidence Policy Improvement. In Francis Bach and David Blei, editors, *International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 2380–2388, Lille, France, 2015. PMLR.

[TZC+18] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. April 2018.

[TZLB17] Lei Tai, Jingwei Zhang, Ming Liu, and Wolfram Burgard. Socially-compliant Navigation through Raw Depth Inputs with Generative Adversarial Imitation Learning. October 2017.

[UO30] G E Uhlenbeck and L S Ornstein. On the Theory of the Brownian Motion. *Phys. Rev.*, 36(5):823–841, September 1930.

[VBC+19] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P Agapiou, Max Jaderberg, Alexander S Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, October 2019.

[VdWWFMM18] Tom Van de Wiele, David Warde-Farley, Andriy Mnih, and Volodymyr Mnih. Q-Learning in enormous action spaces via amortized approximate maximization. In *NeurIPS Workshop "Deep Reinforcement Learning"*, 2018.

[vH10] Hado van Hasselt. Double Q-learning. In *Neural Information Processing Systems (NeurIPS)*, 2010.

[vHDS+18] Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep Reinforcement Learning and the Deadly Triad. December 2018.

[vHGH+16] Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In *Neural Information Processing Systems (NeurIPS)*, February 2016.

[vHGS15] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. In *AAAI Conference on Artificial Intelligence*, pages 2094–2100, September 2015.

[VHS+17] Matej Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. July 2017.

[VLJY19] Cameron Voloshin, Hoang M Le, Nan Jiang, and Yisong Yue. Empirical Study of Off-Policy Policy Evaluation for Reinforcement Learning. November 2019.

[VMO17] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social Attention: Modeling Attention in Human Crowds. October 2017.

[vSvHWW09] H van Seijen, H van Hasselt, S Whiteson, and M Wiering. A theoretical and empirical analysis of Expected Sarsa. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 177–184, March 2009.

[VTKP09] Nikos Vlassis, Marc Toussaint, Georgios Kontes, and Savas Piperidis. Learning model-free robot control by a Monte Carlo EM algorithm. *Auton. Robots*, 27(2):123–130, August 2009.

[WADW18] Anton Orell Wiehe, Nil Stolt Ansó, Madalina M Drugan, and Marco A Wiering. Sampled Policy Gradient for Learning to Play the Game Agar.io. September 2018.

[Wal47] Abraham Wald. *Sequential Analysis*. John Wiley, New York, 1947.

[Wat89] Christopher J C H Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, London, May 1989.

[WBH+16] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample Efficient Actor-Critic with Experience Replay. November 2016.

[WCAD19] Ruohan Wang, Carlo Ciliberto, Pierluigi Amadori, and Yiannis Demiris. Random Expert Distillation: Imitation Learning via Expert Policy Support Estimation. In *International Conference on Machine Learning (ICML)*, 2019.

[WD92] Christopher J C H Watkins and Peter Dayan. Technical Note: Q-Learning. *Mach. Learn.*, 8(3):279–292, May 1992.

[Web94] A R Webb. Functional approximation by feed-forward networks: a least-squares approach to generalization. *IEEE Trans. Neural Netw.*, 5(3):363–371, 1994.

[Wer74] Paul Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, 1974.

[Wie48] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. MIT Press, 1948.

[Wil92] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, May 1992.

[WLBF18] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. NerveNet: Learning Structured Policy with Graph Neural Networks. In *International Conference on Learning Representations (ICLR)*, February 2018.

[WMR+17] Ziyu Wang, Josh Merel, Scott Reed, Greg Wayne, Nando de Freitas, and Nicolas Heess. Robust Imitation of Diverse Behaviors. In *Neural Information Processing Systems (NIPS)*, 2017.

[WNZ+20] Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic Regularized Regression. June 2020.

[WTN19] Yifan Wu, George Tucker, and Ofir Nachum. Behavior Regularized Offline Reinforcement Learning. November 2019.

[WXH+18] Qing Wang, Jiechao Xiong, Lei Han, Peng Sun, Han Liu, and Tong Zhang. Exponentially Weighted Imitation Learning for Batched Historical Data. In *Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2018.

[XD19] Danfei Xu and Misha Denil. Positive-Unlabeled Reward Learning. November 2019.

[XM07] Huan Xu and Shie Mannor. The Robustness-Performance Tradeoff in Markov Decision Processes. In *Neural Information Processing Systems (NeurIPS)*, 2007.

[YLCT20] E Yurtsever, J Lambert, A Carballo, and K Takeda. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access*, 8:58443–58469, 2020.

[YM09a] J Y Yu and S Mannor. Arbitrarily modulated Markov decision processes. In *Conference on Decision and Control (CDC)*, pages 2946–2953, December 2009.

[YM09b] J Y Yu and S Mannor. Online learning in Markov decision processes with arbitrarily changing rewards and transitions. In *International Conference on Game Theory for Networks*, pages 314–322, May 2009.

[YN21] Mengjiao Yang and Ofir Nachum. Representation Matters: Offline Pretraining for Sequential Decision Making. February 2021.

[YRZ+20] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. Adversarial Robustness Through Local Lipschitzness. March 2020.

[YS19] Tiancheng Yu and Suvrit Sra. Efficient Policy Learning for Non-Stationary MDPs under Adversarial Manipulation. July 2019.

[YTY+20] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based Offline Policy Optimization. May 2020.

[ZLS+20] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end Interpretable Neural Motion Planner. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[ZLV+20] Ming Zhou, Jun Luo, Julian Villela, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, Aurora Chongxi Huang, Ying Wen, Kimia Hassanzadeh, Daniel Graves, Dong Chen, Zhengbang Zhu, Nhat Nguyen, Mohamed Elsayed, Kun Shao, Sanjeevan Ahilan, Baokuan Zhang, Jiannan Wu, Zhengang Fu, Kasra Rezaee, Peyman Yadmellat, Mohsen Rohani, Nicolas Perez Nieves, Yihan Ni, Seyedershad Banijamali, Alexander Cowen Rivers, Zheng Tian, Daniel Palenicek, Haitham Bou Ammar, Hongbo Zhang, Wulong Liu, Jianye Hao, and Jun Wang. SMARTS: Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. In *Conference on Robot Learning (CoRL)*, 2020.

[ZMBD08] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.

[ZML17] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based Generative Adversarial Network. In *International Conference on Learning Representations (ICLR)*, 2017.

[ZRN+19] Konrad Zolna, Scott Reed, Alexander Novikov, Sergio Gomez Colmenarej, David Budden, Serkan Cabi, Misha Denil, Nando de Freitas, and Ziyu Wang. Task-Relevant Adversarial Imitation Learning. October 2019.

[ZSRE17] Li Zhou, Kevin Small, Oleg Rokhlenko, and Charles Elkan. End-to-End Offline Goal-Oriented Dialog Policy Learning via Policy Gradient. In *NeurIPS Workshop "Conversational AI"*, December 2017.