

# Sample-based Planning and Learning with Function Approximation

Tor Lattimore and Csaba Szepesvári

*Abstract.* We give a short tutorial on sample-based planning and learning with function approximation for reinforcement learning. The focus is on explaining the standard setups and algorithm design principles. The core algorithmic ideas explained are approximate backwards induction, the (Bellman) eluder dimension and the optimism principle.

*Key words and phrases:* Reinforcement learning, planning.

## 1. INTRODUCTION

Reinforcement learning (RL) focuses on learning optimal actions in unfamiliar environments. This paper investigates learning and planning using samples from an unknown stochastic environment. Notably, we consider scenarios where these samples are collected by the learner/agent interacting with the environment, in contrast to offline RL where data is provided beforehand. The literature on reinforcement learning (RL) has expanded significantly in recent decades, necessitating a narrower scope even when focusing on sample-based planning and learning. While we aim to acknowledge relevant literature, this is not a comprehensive survey. Our objective is to elucidate fundamental RL paradigms and algorithm design principles, capturing what we believe are the most valuable ideas in RL through a theoretical lens. Primarily, we concentrate on the *theory* of planning and learning, aiming to capture the most valuable ideas in RL through a theoretical lens. While this focus unfortunately necessitates omitting the vast literature on algorithms that aim at practical performance improvements but lack a strong theoretical foundation, we hope this approach will best serve our goal of enhancing the reader's understanding. We aim to explain high-level themes in modern RL theory and provide intuitive understanding of similarities and differences between various settings, reflecting our personal preferences and expertise. Additionally, we offer detailed explanations of specific algorithms and analyses to facilitate deeper comprehension.

The fundamental objective of reinforcement learning is to learn a near-optimal policy while interacting with an environment. Here, by a policy we mean a function

that determines how the agent chooses actions (potentially stochastically, and also history dependent). There are, however, many ways to model environments and to formalise the learning problem. For example, one might care about finding a near-optimal policy for a given state only or for all states. There are even many ways to define what is meant by near-optimality. Generally speaking a policy  $\pi$  is near-optimal in state  $s$  if its value  $v^\pi(s)$  is close to the optimal value  $v^*(s)$ , where in this work suboptimality is measured by their difference,  $v^*(s) - v^\pi(s)$  (we will give definitions in the next section). There are multiple ways to define value functions, including using discounting, average reward or cumulative reward. The latter two definitions need restrictions on the environments for even to be well-defined.

There are also differences in how the learner is able to interact with the environment. The most archetypical interaction protocol is the *online interaction protocol* where the learner starts in some state, takes actions and transitions to next states and so-on, while observing data along the trajectory that is obtained. This is the situation we find ourselves in as agents interacting with the world. Another common interaction protocol occurs when the learner has access to a *generative model*. In this situation the learner can query the model in any state (this also means that the agent knows what the possible states are) and obtains a sampled next state and reward. The interaction protocol plays an influential role in the hardness of the problem and the complexity of algorithm design. Exploring the options and trade-offs is one of the goals of this article.

Lastly, while early theoretical studies in reinforcement learning focused on unstructured problems, attention has now shifted towards understanding *function approximation*, where the learner has some structured information about the behaviour of the environment. For example, the optimal value function might be linearly representable in terms of some given state/action features. The reason for introducing structure is simple: The vast majority of

---

*Tor Lattimore is a research scientist at Google DeepMind, United Kingdom (e-mail: [lattimore@google.com](mailto:lattimore@google.com)). Csaba Szepesvári is research scientist at Google DeepMind and a professor at the Department of Computing Science, University of Alberta, Canada (e-mail: [szepi@google.com](mailto:szepi@google.com)).*

practically interesting RL problems have enormous state spaces and lower bounds for unstructured problems of this kind show that learning in such setups takes time that scales linearly or worse with the number of states, and as such is hopeless in the worst case. By introducing structure, learning sometimes becomes possible. The challenge then shifts to understanding what kinds of structure is (a) reasonable and (b) leads to practical and efficient algorithms with guarantees.

### 1.1 What is covered and prerequisites

We start with the formal details of Markov decision processes, which define the reinforcement learning environment. In Section 3 we introduce the interaction protocols that determine how the learner interacts with the environment. Because RL is about learning an unknown environment, a single MDP does not define an RL problem. Instead, an RL problem is defined by a class of such MDPs. As we explain in Section 4, the desire to introduce structure through the use of function approximation naturally leads to different interesting classes, or, more generally, measures that quantify the suitability of an MDP to algorithms that are to be designed to use function approximation in a certain way. Next we explain standard principles for algorithm design, focusing on generalisation to unseen states and the optimism in the face of uncertainty principle (Section 5). The remaining sections give examples of algorithms for three different settings. We introduce these at the end of Section 5 where we have more context.

In terms of prerequisites, the manuscript is reasonably self contained. We make use of standard results from the theory of concentration of measure [11], but nothing much more sophisticated than Bernstein’s and Azuma’s inequalities. Some familiarity with MDPs and reinforcement learning notation and theory will no doubt be useful. Yet we do not need any sophisticated machinery. For example, we will not need any of the more complicated theory involving infinite-horizon average cost/reward MDPs, nor will we need any serious measure theory. All results are stated for finite/countable spaces, with the understanding that this could be generalised without much difficulty should the need arise. Some standard books on reinforcement learning are [44, 46], but the most relevant text for our paper is the recent manuscript by Agarwal et al. [6].

## 2. NOTATION

We use  $\|\cdot\|$  for the standard euclidean norm and  $\langle \cdot, \cdot \rangle$  for the corresponding inner product.  $\|\cdot\|_1$  and  $\|\cdot\|_\infty$  are the 1-norm and  $\infty$ -norm, respectively. Functions with a finite domain are often identified with vectors. When  $f, h : X \rightarrow \mathbb{R}$  are real-valued functions on a finite space  $X$  we let  $\langle f, g \rangle = \sum_{x \in X} f(x)g(x)$  and  $\|f\|_1 = \sum_{x \in X} |f(x)|$  and  $\|f\|_\infty = \sup_{x \in X} |f(x)|$  where the latter definition is valid even if  $X$  is infinite. The indicator function is

1. The closed euclidean ball of radius  $r$  is  $\mathbb{B}_r = \{x \in \mathbb{R}^d : \|x\| \leq r\}$ . For  $X$  a set with a suitable measurability structure, we let  $\Delta_X$  be the space of probability measures over  $X$ . When  $X$  is finite or countable, we identify  $\Delta_X$  with the probability mass functions over  $X$ . When  $X$  is the set of reals  $\mathbb{R}$ , this is the standard set of probability measures over  $\mathbb{R}$  when  $\mathbb{R}$  is equipped with the  $\sigma$ -algebra induced by the open subsets of  $\mathbb{R}$ . More generally,  $\Delta_X$  is the set of probability measures on the Borel  $\sigma$ -algebra of topological space  $X$ . Below we use this with  $X = \mathbb{R} \times S$  with  $S$  finite. In this case, the topology is the product topology on  $X$  derived from the usual topology on  $\mathbb{R}$  and the discrete topology on  $S$ . Readers not familiar with measure theory may think of  $\mathbb{R}$  being replaced with a large discrete set, in which case, the probability distributions over  $X$  are identifiable with the all familiar probability mass functions. All in all, nothing unusual is happening here. Given a set of real-valued functions  $\mathcal{F}$ , let  $\text{sym}(\mathcal{F}) = \{f - g : f, g \in \mathcal{F}\}$  be the symmetrisation of  $\mathcal{F}$ .

### 2.1 Markov decision problems

A finite Markov decision problem (MDP) is given by a tuple  $(S, A, \mathbf{T})$  where  $S$  and  $A$  are finite sets of states and actions, respectively,  $\mathbf{T} : S \times A \rightarrow \Delta_{\mathbb{R} \times S}$  is the transition kernel. An MDP induces a discrete time decision process that describes the interaction of an agent and an environment: If in step  $t$  of the process, the environment state is  $s_t \in S$  and the agent takes action  $a_t \in A$  then this action causes the next state of the environment to move to  $s_{t+1} \in S$ , while a reward of  $r_{t+1} \in \mathbb{R}$  is emitted, where  $(r_{t+1}, s_{t+1}) \sim \mathbf{T}(s_t, a_t)$  given the past. In words: the distribution of  $(r_{t+1}, s_{t+1})$  given the past is  $\mathbf{T}(s_t, a_t)$ . The agent generally observes the states and the associated rewards as they are generated and the agent’s goal is to take actions so as to maximise the sum of rewards over long horizons. The assumption that states are observable may look strong. While this may be true, the much bigger concern is that the state space itself will generally be huge, so if anything, it does not help that the observations include the states. In particular, one main goal in RL is to develop algorithms whose reliability and cost does not depend on the number of states. With this goal in mind, the fact that states are observed will be less concerning.

We extract the reward kernel  $\mathbf{R} : S \times A \rightarrow \Delta_{\mathbb{R}}$  and the state transition kernel  $\mathbf{P} : S \times A \rightarrow \Delta_S$  from  $\mathbf{T}$ :  $\mathbf{R}(\cdot | s, a) = \mathbf{T}(\cdot \times S | s, a)$  and  $\mathbf{P}(\cdot | s, a) = \mathbf{T}(\mathbb{R} \times \cdot | s, a)$  for all  $(s, a)$  pairs, so that, in particular  $r_{t+1} \sim \mathbf{R}(s_t, a_t)$  and  $s_{t+1} \sim \mathbf{P}(s_t, a_t)$ . We will assume for the entire manuscript that the reward distribution is supported on  $[0, 1]$ :

**ASSUMPTION 2.1.**  $\mathbf{R}(\cdot | s, a)$  is supported on  $[0, 1]$  for all  $s, a \in S \times A$ .

This assumption is quite normal in the RL literature, though a number of authors study the more subtle assumption that cumulative rewards are bounded during any episode [31, 32, 49, 57, 60]. One can replace boundedness with an assumption that the reward distributions are sub-gaussian with a common global parameter and nothing much would change in the story that follows. In principle, one could also study more sophisticated assumptions on the reward distribution, but this is generally not thought to be the main source of difficulty. In fact, many authors assume the mean reward is known and focus only on learning the state transition dynamics.

Belatedly, we restrict our attention to finite state and action spaces because this is sufficient to develop intuition without devoting space to the technical complexities introduced by infinite spaces and the reader is advised to think of problems where the state-space (and sometimes also the action-space) is enormous. Technically, all the main results will continue to hold for infinite state spaces under suitable mild assumptions.

Also for the sake of simplicity and to enhance clarity, in this paper we focus on *finite horizon* MDPs where the horizon is given by a positive integer  $H$ . In this case, the performance of an agent is measured by the cumulative reward they collect while interacting with the environment for exactly  $H$  steps. Given a policy  $\pi$  and state  $s \in S$ , the value  $v^\pi(s)$  is defined as the expected cumulative reward collected when the policy is started from that state and interacts with the MDP for  $H$  steps. Given an MDP (with our constraints), there is always at least one *optimal policy* that achieves the best possible (optimal) value,  $v^*(s)$ , from any initial state  $s \in S$ . Of special interest will be policies that choose actions based only on the identity of most recent state visited and in a deterministic fashion. Policies of this kind are called *memoryless* and *deterministic* and they are identified with a map  $\pi : S \rightarrow A$ . Following  $\pi$  then means taking action  $\pi(s)$  when in state  $s$ . The space of all memoryless deterministic policies is denoted by  $\Pi$ . Memoryless stochastic policies map states to distributions over actions.

We chose to present results for the finite horizon case because this setting is in some ways simpler, which brings much needed clarity. At the same time, the principles and results derived in the finite horizon setting tend to generalise to other settings, though not without hard work and occasionally some nuance. Although it buys much simplicity, there is also one complexity arising from the fixed horizon setting. The problem is that when the horizon is finite, value functions as defined above are not that useful on their own. The reason for this is that in this case the value that will be incurred in some step  $t$  of the  $H$  step process will depend on the value of  $t$  (equivalently, on the number of outstanding steps,  $H - t$ ). This would call for introducing  $h$  step value functions with  $0 \leq h \leq H$ , which

complicates notation. To avoid this, we make the assumption that the state space has a layered structure: The layers  $S_h$  with  $h = 0, \dots, H$  form a chain such that, except for the last layer, taking a step in layer  $S_h$  always results in a state that belongs to the next layer,  $S_{h+1}$ . This way, we save ourselves from the need to introduce a dependence on the time elapsed, or time remaining out of the total  $H$  steps that a policy is executed for. Starting in a later layer simply means that the effective number of time steps remaining till the end is reduced. Every  $H$  step problem can be turned into an  $H$  step (or even infinite step) problem with a layered structure by just creating identical copies of the state space, as we explain later.

**ASSUMPTION 2.2.** *The state space is a disjoint union  $S = \cup_{h=0}^H S_h$  such that:*

- (i)  $\mathbf{P}(\cdot|s, a)$  is supported on  $S_{h+1}$  whenever  $h < H$  and  $s \in S_h$ ,
- (ii)  $S_H = \{\perp\}$  and the state  $\perp$  is absorbing and un-rewarding:  $\mathbf{P}(\perp|\perp, a) = 1$  and  $r(\perp, a) = 0$  for all  $a \in A$ .

## 2.2 Value functions

We are now in a position to define the value and  $q$ -value functions associated with a policy. Given a memoryless deterministic policy  $\pi$ , the value function of  $\pi$  satisfies the Bellman equations:  $q^\pi(\perp, a) = 0$  and for all  $(s, a) \in S \times A$ ,

$$q^\pi(s, a) = r(s, a) + \sum_{s' \in S} \mathbf{P}(s'|s, a) q^\pi(s', \pi(s')),$$

$$v^\pi(s) = q^\pi(s, \pi(s)).$$

Here,  $q^\pi(s, a)$  gives the expected total reward when starting at  $s$ , first taking action  $a$  and then following  $\pi$ . We call  $q^\pi(s, a)$  the *action-value* of  $\pi$  at  $(s, a)$ . Analogously to  $v^*$ , one can also define the *optimal action-value function*  $q^* : S \times A \rightarrow \mathbb{R}$  such that  $q^*(s, a)$  gives the best possible action-value when any policy  $\pi$  is allowed. Both  $v^*$  and  $q^*$  are well defined because  $H$  and  $A$  are finite and the mean rewards are assumed to be in  $[0, 1]$ . Thanks to the layered structure and because  $A$  is finite, a memoryless deterministic optimal policy will always exist. (Without the layered structure, the optimal policy would also depend on the number of steps that was elapsed, or remains up to the end. The layered structure allows us to overcome this because it does not matter anymore whether a policy is run for  $H$ , or  $H + t$  steps with  $t > 0$ . Technically, the reason memoryless policies suffice is because in our layered structure MDPs, the finite horizon criteria matches the infinite horizon total expected (undiscounted) reward criterion.) It follows that for any  $(s, a) \in S \times A$ ,

$$v^*(s) = \max_{\pi \in \Pi} v^\pi(s) \quad \text{and} \quad q^*(s, a) = \max_{\pi \in \Pi} q^\pi(s, a).$$

The *Bellman optimality equation* states that  $v^*$  and  $q^*$  satisfy  $v^*(\perp) = 0$  and

$$(1) \quad \begin{aligned} q^*(s, a) &= r(s, a) + \sum_{s' \in S} \mathbf{P}(s'|s, a) v^*(s') \\ v^*(s) &= \max_{a \in A} q^*(s, a) \end{aligned}$$

for all  $(s, a) \in S \times A$ , simultaneously. Introduce  $A^*(s) = \{a \in A : q^*(s, a) = \max_{a' \in A} q^*(s, a')\}$ . Then, any memoryless policy  $\pi : S \rightarrow \Delta_A$  such that  $\pi$  chooses actions from  $A^*(s)$  at each state  $s \in S$  is optimal:  $v^\pi = v^*$ . Accordingly, actions in  $A^*(s)$  are called optimal at state  $s$  and the set itself is called the set of optimal actions.

As promised, let us explain how an MDP where the layered structure is not present can be turned into one where it is present. For this, just replace the state-space  $S$  with  $S \times \{0, \dots, H-1\} \cup \{\perp\}$ , letting  $S_k = S \times \{k\}$  for  $k = 0, \dots, H-1$  and then adapt the transition kernel in the obvious way. This transformation loses the information that the transition kernel is the “same” between each layers: if we want to retain this information, we may consider just those layered structure MDPs where this regularity assumption is satisfied. This is known as the *time-homogeneous* setting.

The assumption that the MDP is layered comes with a non-trivial side-effect: In such an MDP, after  $H$  steps, the state becomes equal to  $\perp$ . Hence, if we want to study algorithms that discover a good policy without knowing the MDP, but by collecting data from interacting with the MDP, we need to allow the learner to *restart* the interaction after  $H$  steps. The ability to restart, or *reset* (perhaps randomly) the state, is nontrivial. For the planning problems we consider, this is less of an issue, but in online learning when the agent interacts with a physical system, restarting the system is often not an option.

So what criterion to consider when there is no obvious end to the interactions? Such infinite horizon settings introduce several complications. Most notably, defining the value function as the expected cumulative reward over an infinite horizon may be undefined, or could result in infinite values. There are three standard choices:

- (i) Restrict the class of MDPs such that this does not happen, such as our layered class, or some generalisation of it;
- (ii) Define the value of a policy to be its *average reward* over an infinite time horizon; or
- (iii) geometrically discount the rewards.

All these choices have their pros and cons. Generally speaking the average reward model requires a greater level of mathematical sophistication and assumptions, while discounting is spiritually more closely related to the finite horizon model that we focus on. As hinted upon earlier, there is a cottage industry of work showing that results established for one type of value function also hold for others.

## 2.3 Occupancy measures and trajectories

Given a policy  $\pi$  let  $\mathbb{P}^\pi$  be the probability measure on  $(S \times A \times \mathbb{R})^H$  induced by the interaction between  $\pi$  and the MDP over a single episode. We also let  $\mathbb{P}_{SA}^\pi(s, a) = \mathbb{P}^\pi(\cup_{h=0}^{H-1} \mathbf{1}(s_h = s, a_h = a))$ , which is called the occupancy measure. Following our convention of treating functions as vectors, given a function  $f : S \times A \rightarrow \mathbb{R}$ ,  $\langle \mathbb{P}_{SA}^\pi, f \rangle = \mathbb{E}_\pi[\sum_{h=0}^{H-1} f(s_h, a_h)]$  where  $\mathbb{E}_\pi$  is the expectation with respect to  $\mathbb{P}^\pi$ .

## 3. INTERACTION PROTOCOLS

The interaction protocol defines the rules governing the interaction between the learner and the environment (MDP). The right interaction protocol depends on the application and the information available. Roughly speaking, there are four typical scenarios:

- *Online interface*: The most fundamental setup is the *fully online RL* where the learner starts from some initial state in the MDP, plays actions until the final layer is reached and then resets to another initial state for another episode.
- *Global simulator interface*: Sometimes, one has access to a simulator that given any state/action pair  $s, a$  returns a reward/state pair  $r, s'$  sampled from transition kernel. In the global simulator access model the learner can query the reward and next-state transition kernels for any state/action pair, which implicitly assumes that  $S$  and  $A$  are known in advance.
- *Local simulator interface*: In this model the learner maintains a set of observed states that contains only the initial state when the interaction starts. Then, in any round the learner can query the reward and next-state transition kernels for any action  $a \in A$  and any state  $s$  in the set of states observed so far. The observed next-state is added to the set of visited states. Local simulator access is less restrictive than online RL because the learner can query the same state/action pair  $(s, a)$  multiple times without having to navigate to  $s$ , but is more restrictive than global simulator access because the learner is only allowed to query state/action pairs at states it has already visited.
- *Offline RL*: In offline RL the learner plays a passive role. All the data has already been collected by some other policy and the learner’s role is to compute some quantity, such as the optimal policy or the value of some predetermined policy. Offline RL is not the topic of this article and is not discussed further.

Each of these protocols have genuine applications. Online RL needs little explanation. A learner interacting with the real world has no choice but to use this interface. Sometimes online RL is deemed risky and the user of RL algorithms opts to either collecting data from the environment

(leading to offline RL), or to building a simulator that is then used to get a good policy. Simulator access of both kinds also arises naturally in many problems. For example, consider the problem of learning to drive a fast lap in a racing simulator. In principle it might be possible to initialise the simulator in any state, but in practice even identifying the set of feasible states is effectively intractable. What is straightforward, however, is restarting the game at any previously observed state. This corresponds to the local simulator access interface above. On the other hand, in simple single-player games the learner may well have a representation for all the states and be able to query the simulator (which coincides with the environment in this case) at any state/action pair. Of course, one can freely mix and match all these setups. In this paper when considering simulator access we only consider the problem of getting a good policy for the simulator as the environment. That is, we will not consider the effect of the mismatch between a simulator and the environment, but just note here that, as expected, the performance loss due to this mismatch can be controlled by making the simulator more realistic. A less obvious but common practice is to make the simulation richer, more demanding (e.g., by randomising certain aspect of the simulation), so that the policies learned become more broadly applicable, which, ideally, translates to better performance at deployment. In the remainder of this section we expand on the individual interaction protocols and the associated natural performance metrics.

### 3.1 Online RL interaction protocol

In the finite-horizon MDP setting the learner interacts with an MDP for  $T$  episodes of length  $H$ . Without the loss of generality, at the start of episode  $t$  the learner chooses a policy  $\pi_t \in \Pi$ , which is used to generate a trajectory  $s_0, a_0, r_1, s_1, a_1, \dots, r_H, s_H, a_H$  where  $s_0^t = s_{\text{init}}$  is a distinguished initial state and  $a_h^t = \pi_t(s_h^t)$  and  $(r_{h+1}^t, s_{h+1}^t)$  is sampled from  $\mathbf{T}(\cdot | s_h^t, a_h^t)$ . The most common performance metric is the *regret*, which is the shortfall in expected cumulative reward collected by the learner due to not playing the optimal policy:

$$\text{Reg}_T = \sum_{t=1}^T [v^*(s_{\text{init}}) - v^{\pi_t}(s_{\text{init}})] .$$

Regret is preferred to total value collected because, as opposed to total value collected, worst-case regret over a class of environments remains a meaningful metric even if there is a trivial environment in the class where all actions give zero (or near zero) reward in all states. As opposed to total value, regret measures skill (lack of skill to be precise).

The regret is random because the policy  $\pi_t$  depends on the data collected in episodes 1 to  $t - 1$ . A typical theorem shows that for some algorithm  $\text{Reg}_T$  is bounded by

such and such with high probability, though bounds in expectation are also fairly common. Here, the focus is often how fast the regret grows as a function of  $T$ : A linear growth shows that the learner is ineffective, while sublinear growth shows that the learner is improving (learns) over time. Generally speaking, when the rewards/transitions are stochastic one should expect the regret to be at least  $\Omega(\sqrt{T})$ . In many classical settings this is also the upper bound, though in non-parametric settings one should normally expect the regret to be larger. Another standard metric is the number of  $\varepsilon$ -suboptimal episodes. Given  $\varepsilon > 0$ , let  $N_\varepsilon = \sum_{t=1}^\infty \mathbf{1}(v^*(s_{\text{init}}) - v^{\pi_t}(s_{\text{init}}) \geq \varepsilon)$ . A bound on  $N_\varepsilon$  is sometimes called a *mistake bound*, or, *sample complexity* (the word complexity is used similarly to how we talk about time-complexity of algorithms). Of course  $N_\varepsilon$  is generally a random variable that must be bounded with certain probabilistic quantifiers. An algorithm that depends on parameters  $\varepsilon > 0$  and  $\delta \in (0, 1)$  is *probably approximately correct* (PAC) if  $\mathbb{P}(N_\varepsilon \geq B(1/\varepsilon, 1/\delta)) \leq \delta$  for all MDPs in some class of interest and  $B$  is polynomial in its arguments. An algorithm only depending on  $\delta \in (0, 1)$  is called uniformly PAC if  $\mathbb{P}(N_\varepsilon \geq B(1/\varepsilon, 1/\delta)) \leq \delta$  holds for all  $\varepsilon > 0$ . Generally speaking  $B$  only depends on  $\log(1/\delta)$ . These definitions differ slightly from the PAC learning concept in statistical learning theory where the algorithm depends on neither  $\varepsilon$  nor  $\delta$ . Yet another possibility is to use the *simple regret*, defined in the next section. Sometimes, sample complexity is defined in connection to simple regret as the number of interactions until the simple regret (with high probability) is pushed below a limit. We outline relations between these concepts at the end of this section.

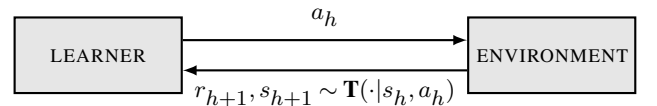


FIG 1: Data flow in online RL.

### 3.2 Local or global planning

The algorithms that interact with a simulator are called planners: They plan either an action, or a way of acting (policy). In the former case, the planner is called whenever a new state is encountered (local planning), while in the latter case, the planner is used in a one-shot manner (global planning). That is, in the latter case the planner first interacts with the simulator for a finite number of steps, after which it computes some internal data structure, such as the components of a parameter vector, which is then what is used to come up with actions in response to states encountered, but with no further interactions with the simulator. We call this global planning,

because the parameter vector computed must contain information about how to behave at any state, globally. In the local planning setting it is natural for the planners to keep information between their calls. While this generally improves performance, the benefit of doing so may be marginal in that it usually does not change whether a problem can be solved efficiently (with polynomial number of interactions per call in the relevant quantities). Obviously, global planning is more demanding than local planning and sometimes global planning is significantly harder than local planning. In what follows, we will revert to using the word learner even when the environment is replaced with a simulator. We do this because a distinction between the two does not appear to be useful. In particular, even a planner is learning about the simulator (or environment) as it interacts with it.

### 3.3 Global access interface

In the global access interaction protocol the learner knows the state and action spaces  $\mathcal{S}$  and  $\mathcal{A}$ , but not the transition kernel. Like in the online model the learner interacts sequentially with the environment. What changes is that now the learner can query the environment at *any* state/action pair and receives a sample from the corresponding transition kernel. Formally, the learner and environment interact over  $t = 1$  to  $T$  rounds. In round  $t$  the learner chooses a state/action pair  $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$  and receives a reward/next-state pair  $r_t, s'_t$  sampled from  $\mathbf{T}(\cdot | s_t, a_t)$ . By the end of the interaction the learner either outputs an action  $\hat{a}$  (local planning) or policy  $\hat{\pi}$  (global planning). The cumulative regret no longer makes sense and is usually replaced by some form of instantaneous regret (also called simple regret), which is defined in one of the following ways:

$$\text{spReg}(s) = v^*(s) - v^{\hat{\pi}}(s) \quad \text{spReg}^* = \max_{s \in \mathcal{S}} \text{spReg}(s)$$

$$\text{saReg}(s) = v^*(s) - q^*(s, \hat{a}) \quad \text{saReg}^* = \max_{s \in \mathcal{S}} \text{saReg}(s).$$

The motivation to introduce the first two quantities should be clear. If a local planner controls  $\text{saReg}(s)$  for any state  $s$  it encounters, it is not hard to see that it can be used to generate trajectories of a near-optimal policy. Generally speaking a learner will be called *sound* if the constraints on its output are satisfied with probability at least  $1 - \delta$ . Sometimes the number of queries that the learner issues is also randomised and then the usual objective is to either guarantee that the number of queries to the simulator is small with high probability, or that the expected number of queries is small, while also demanding that the learner is sound. We will see an example of a learner interacting with an MDP via the global access model and global planning setting in Section 6.

### 3.4 Local access interface

In many cases the learner does not even know the state space but only knows the initial state and has access to a simulator that accepts a known state and any action, and returns a sample from the reward/next-state kernels. A learner with a simulator of this kind can slowly develop a tree from the initial state. The local access interface provides the learner with less flexibility than the global access model and some objectives that are achievable with the latter may not be possible with the former. For example, if there exist states that cannot be reached from the initial state, then in general the local access interface does not provide enough information for the learner to find a globally optimal policy. We restrict our attention to the problem of using as few queries as possible to find a near-optimal policy for the initial state. Formally, the learner is initialised with a set  $V_1 = \{s_{\text{init}}\}$  and interacts with the environment over rounds  $t = 1, \dots, T$  where  $T$  is possibly random. In round  $t$  the learner selects a state  $s_t \in V_t$  and an action  $a_t \in \mathcal{A}$ . The learner then receives  $r_t, s'_t \sim \mathbf{T}(\cdot | s_t, a_t)$  and the set of visited states is updated to  $V_{t+1} = V_t \cup \{s'_t\}$ . Once the interaction has ended, the learner/planner outputs an action  $\hat{a} \in \mathcal{A}$  (local planning) or policy  $\hat{\pi}$  (global planning). The performance metric is the same as in the global access interface except that now it would be unusual to bound  $\text{saReg}^*$  or  $\text{spReg}^*$  due to the problem of reachability with the local access interface. In Section 9 we give an example algorithm that uses the local access interface in a local planning setting.

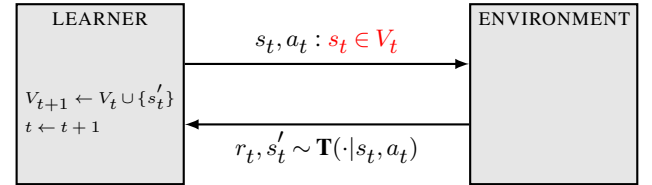


FIG 2: The local access model. The learner is initialised with  $s_1 = s_{\text{init}} \in \mathcal{S}$  and visited states  $V = \{s_{\text{init}}\}$ . In the finite horizon setting, in step  $t$  the learning must choose a state from  $V_t \setminus S_H$ .

### 3.5 Remarks and references

- A high probability bound on the regret does not imply a bound on the mistake count  $N_\varepsilon$  because low regret is achievable by a learner that makes poor decisions infinitely often. Concretely, if  $v^*(s_{\text{init}}) - v^{\pi_t}(s_{\text{init}}) = \mathbf{1}(t \in \{1, 2, 4, 9, 16, \dots\})$ , then the regret is  $\Theta(\sqrt{T})$  but,  $N_\varepsilon = \infty$  for any  $\varepsilon < 1$ .
- Bounds on the mistake counts do imply bounds on the regret. In particular, uniform-PAC algorithms (which, notably, can *not* take  $\varepsilon$  as an input), enjoy this property. In some cases, the regret obtained can be nearly optimal [16]. Corollary 7.1 gives the general recipe of how

to convert these uniform-in- $\varepsilon$  error bounds into regret bounds. To explore the relationships between variations of the definitions given here, see [16], which also gives a historical overview of the definitions.

- In the online setting still, a bound on the regret implies a bound on the simple regret for the “mixture policy” that is obtained by choosing uniformly at random an index in  $[T] = \{1, \dots, t\}$  and then following  $\pi_t$  until the end of the episode.
- The *sample complexity* of a learner in the global/local access models is the number of queries made to the model, which in some cases is random.
- An interesting case that shows the big gap between local and global planning is to contrast the results of two papers, one of which considers local planning, the other global planning. Both are concerned with planning for MDPs where the state space is the  $d$ -dimensional hypercube, the transitions and rewards are smooth in an appropriate sense. On the one hand, Chow and Tsitsiklis [15] show that *any* global planner needs to use  $\Omega((1/\varepsilon)^d)$  samples to find a uniform  $\varepsilon$ -approximation to the value function, while Rust [40] and a follow-up by one of the authors of this paper [45] show that for the same class of MPDs,  $O(\text{poly}(1/\varepsilon, d))$  samples are sufficient if one is interested in getting an  $\varepsilon$ -good estimate of the value at some arbitrarily fixed state. Intriguingly, that the exponential speedup comes from the distinction between local and global planning was only noticed much later.

#### 4. FUNCTION APPROXIMATION

Many early works on theoretical RL treated  $S$  and  $A$  as unstructured sets. Knowledge of the reward/next-state distributions in a given state provide no information about those in another state. While this enormously simplifies analysis and can provide basic insights into the issues that arise when designing algorithms, it is not compatible with practical applications where the state spaces are enormous, but both states and actions are highly structured and an effective learner must generalise across the state/action spaces. The most general way to allow the learner to generalise is to introduce a restricted set of MDPs  $\mathcal{M}$  and assume the true MDP lies in  $\mathcal{M}$ . Let us give some examples:

- (i) *Unstructured MDPs*:  $\mathcal{M}$  is the set of all MDPs on with state-space  $S$  and action space  $A$ .
- (ii) *Linear MDPs*: Given a feature map  $\phi : S \times A \rightarrow \mathbb{R}^d$ , let  $\mathcal{M}$  be the set of all MDPs such that there exists  $\mu : S \rightarrow \mathbb{R}^d$  and  $\theta \in \mathbb{R}^d$  such that

$$r(s, a) = \langle \phi(s, a), \theta \rangle \quad \mathbf{P}(\cdot | s, a) = \langle \phi(s, a), \mu(\cdot) \rangle.$$

- (iii) *Linear value functions*: Given a feature map  $\phi : S \rightarrow \mathbb{R}^d$ , let  $\mathcal{M}$  be the set of all MDPs such that for all  $\pi \in \Pi$  there exists a  $\theta \in \mathbb{R}^d$  such that  $v^\pi(s) = \langle \theta, \phi(s) \rangle$  for all  $s \in S$ .
- (iv) *Linear action-value functions*: Given a feature map  $\phi : S \times A \rightarrow \mathbb{R}^d$ , let  $\mathcal{M}$  be the set of all MDPs such that for all  $\pi \in \Pi$  there exists a  $\theta \in \mathbb{R}^d$  such that  $q^\pi(s, a) = \langle \phi(s, a), \theta \rangle$  for all  $(s, a) \in S \times A$ .
- (v) *Linear optimal value function*: Given a feature map  $\phi : S \rightarrow \mathbb{R}^d$ , let  $\mathcal{M}$  be the set of MDPs such that there exists a  $\theta \in \mathbb{R}^d$  such that  $v^*(s) = \langle \phi(s), \theta \rangle$  for all  $s \in S$ .
- (vi) *Linear optimal action-value function*: Given a feature map  $\phi : S \times A \rightarrow \mathbb{R}^d$ , let  $\mathcal{M}$  be the set of MDPs such that there exists a  $\theta \in \mathbb{R}^d$  such that  $q^*(s) = \langle \phi(s, a), \theta \rangle$  for all  $(s, a) \in S \times A$ .

There are some obvious relations between the classes. For example, linear MDPs have linear action-value functions and unstructured MDPs can be represented as linear MDPs by introducing  $|S \times A|$ -dimensional features. Moreover  $(ii) \subset (iv) \subset (vi)$  and  $(iii) \subset (v)$ . Interestingly the class of environments with linear action-value functions is ‘nearly’ the same as the class of linear MDPs, as explained by [51]. One can imagine many other variations on the classes listed above. There is interest in considering nonlinear (or general) function approximation. For example, the nonlinear version of the linear MDP class can be formulated as follows: Let  $\mathcal{F}$  be a set of functions from  $S \times A$  to the reals. Let  $\mathcal{M}$  be the set of MDPs such that  $r \in \mathcal{F}$  and for any  $s' \in S$ ,  $\mathbf{P}(s' | \cdot) \in \mathcal{F}$ . Another general change is to introduce unrealisability (in statistics, misspecification). For the linear value function setting, the definition would then be as follows. Fix  $\phi : S \rightarrow \mathbb{R}^d$  and  $\varepsilon > 0$ . Then,  $\mathcal{M}$  is the set of MDPs such that for all  $\pi \in \Pi$  there exists a  $\theta \in \mathbb{R}^d$  such that  $|\langle \theta, \phi(s) \rangle - v^\pi(s)| \leq \varepsilon$  for all  $s \in S$ .

There are two standard theoretical questions associated with function approximation.

- (i) Given a class  $\mathcal{M}$ , what can be said about the regret or sample complexity for a given interaction model and can the optimal values of these be (approximately) achieved in a computationally efficient manner?
- (ii) Is there a complexity measure  $C(\mathcal{M})$  such that the regret or sample complexity can be expressed in terms of  $C(\mathcal{M})$ ? Of course one should hope for upper and lower bounds, and for algorithms that can be implemented efficiently given suitable oracle access to operations on  $\mathcal{M}$ . For linear value function approximation with feature maps taking values in  $\mathbb{R}^d$ , the candidate complexity measures are polynomial in  $d$ ,  $H$ , possibly  $|A|$ , but independent of  $|S|$ .

## 5. ALGORITHMIC PRINCIPLES

All algorithms for reinforcement learning need to collect information about the environment. The basic challenge is where to look for information that will “make a difference” given what is already known. While, say, in a linear value function approximation setting, there are only  $d$  parameters to learn, there are many states (and potentially many actions) and it is *a priori* unclear at which states to collect information about the MDP. Care is needed so that information is available that allows to keep errors under control *even at states that are never explored*. Sometimes, this is referred to as the *distribution shift problem*: When a learnt policy is deployed, it very well may visit states that were never encountered before and for it to perform well, the models that are driving the policy need to work well. Controlling these *extrapolation errors* is one of the key problems in RL.

The most successful approach to uncover states that help the learner to keep errors under control is to use the *principle of optimism in the face of uncertainty*. This principle is often described in the context of online RL, but it is equally useful under other performance metrics and interaction models. Yet, it is easiest to describe it in the online setting. Here, the idea is to maintain a confidence set that contains all plausible models of the unknown MDP and act according to the optimal policy associated with the MDP for which the value function is maximised. This method works because one of two things happens:

- (i) The policy is near-optimal and the learner is happy; or
- (ii) The policy is not near-optimal, in which case the learner will observe (in expectation) a reward that is less than the plausibly optimal MDP promised, which means the learner gains information.

Intuitively, with a worse policy, one should gain more information about the MDP, but whether this leads to an overall sample-efficient learner needs careful analysis. Furthermore, while optimism is often useful, optimism can in fact be suboptimal. This happens when actions that are known (at some stage of learning) to be suboptimal should be played to gain critical information.

As noted earlier, the optimism principle can still be useful when the learner’s objective is only to find a near-optimal policy but not to simultaneously play good actions. Generally, however, the benefits are less apparent in these situations and a more natural heuristic is to select actions that lead to information gain. We see an example of this in Section 6. Lastly, there exist more sophisticated methods that balance exploration and exploitation by explicitly comparing information gain and (expected) regret [20, 21, 35, 36]. These methods are the most powerful, but tend to be less well-behaved computationally or challenging to analyse.

## 5.1 Remarks and references

Like all significant ideas, the idea that optimism can be used to trade off exploration and exploitation has been re-discovered a few times. Perhaps the first account is due to Kumar and Becker [27], who introduced “cost-biased parameter estimation” to favour those parameters that lead to a smaller cost in control problems. A few years later, Lai and Robbins [28] published their seminal paper which introduced a big part of the important machinery. Their goal was to study asymptotic optimality. Somewhat later, in the reinforcement learning literature, Campi [14], starting from the work of Kumar and Becker [27], introduces the “bet on the best” principle, which is exactly the principle of optimism: one should always choose the controller for the model which is still plausible given the data and gives rise to the best values (for a more detailed account of this principle, see [10]). Kearns and Singh [25] describe an algorithm that uses optimism to control the number of times large errors are encountered in online learning in a finite discounted MDP. To make it to a full circle, the recent methods mentioned that compare information gain and (expected) regret can be seen as implementing the cost-biased parameter estimation idea.

That extrapolation errors need to be under control is most clear when in the presence of misspecification errors. In this case, Du et al. [18] showed that there is no hope in learning a good policy with polynomial sample complexity as a function of  $d$ , the number of dimensions,  $H$ , the planning horizon and  $1/\varepsilon$ , where  $\varepsilon$  is the desired accuracy. In the next section we look into this case, and present a positive result for a relaxed (and thus more realistic) criterion.

Finally, we collected a number of results in Table 1. These cover only linear function approximation, and are meant to illustrate the type of results that can be achieved in the various settings.

## 5.2 The organisation of the rest of the paper

The next section (Section 6) tells a story about how to deal with the fact that learners need to generalise to unseen states in the context of linear function approximation, in particular, when all action-value functions lie in the span of the features. As it turns, in this case there is a way to compute a good approximation to the optimal action value function by means of an approximate version of backwards induction. For this to work, all we have to do is to give the learner the ability to learn about the MDP at a few, but well chosen states. While, for the sake of simplicity, this section leaves open of just how to find such a set of states, this question is addressed in the next sections in different contexts.

First, we introduce these tools in Section 7 for *bandit problems*. These are problems where there is a single

| REF  | SAMPLE COMPLEXITY                                                | COMPUTATION | SETTING                      | APPROXIMATION CLASS                  | ACCESS MODEL |
|------|------------------------------------------------------------------|-------------|------------------------------|--------------------------------------|--------------|
| [29] | $\text{poly}(d, H, \frac{1}{\epsilon})$                          | poly time*  | discounted or finite horizon | $q^\pi$ approx. linear $\forall \pi$ | global       |
| [43] | $\text{poly}(d, H, \frac{1}{\epsilon})$                          | poly time*  | discounted                   | $v^*$ approx. linear <sup>†</sup>    | global       |
| [58] | $\text{poly}(d, H, \frac{1}{\epsilon})$                          | poly time*  | finite horizon               | $q^*$ approx. linear <sup>†</sup>    | global       |
| [56] | $\text{poly}(d,  A , H, \frac{1}{\epsilon})$                     | poly time   | discounted                   | $q^\pi$ approx. linear $\forall \pi$ | local        |
| [53] | $\text{poly}\left(\left(\frac{dH}{\epsilon}\right)^{ A }\right)$ | inefficient | discounted or finite horizon | $v^*$ approx. linear                 | local        |
| [24] | $\text{poly}\left(d, H, \frac{1}{\epsilon}\right)$               | poly time   | finite horizon               | linear MDP                           | online       |
| [4]  | $\text{poly}\left(d, \log  A , H, \frac{1}{\epsilon}\right)$     | poly time   | finite horizon               | linear MDP                           | online       |
| [17] | $\text{poly}\left(d, H, \frac{1}{\epsilon}\right)$               | inefficient | finite horizon               | linear MDP <sup>†</sup>              | online       |
| [51] | $\text{poly}\left(d, H, \frac{1}{\epsilon}\right)$               | inefficient | finite horizon               | $q^\pi$ linear $\forall \pi$         | online       |

\* all of these methods are only polynomial time subject to the learner being given a ‘core’ set of state/action pairs from which to extrapolate. For example, [29] needs a near-optimal design over the state-space, while the other works use related concepts.

† both of these works make strong additional assumptions on the feature vectors to achieve the given sample complexity.

TABLE 1

RL with linear function approximation. All logarithmic factors in the sample-complexity bounds are dropped.

state, but there are many actions and the job is to generalise across them. As we will see, the tools developed here will work nicely in the more complicated settings when there are many states to generalise over. Section 8 showcases an approach where we consider nonlinear (to be precise, not necessarily linear) function approximation. The specific setting is when the optimal action-value function belong to our chosen set of functions  $\mathcal{F}$ . The algorithm aims to find the optimal action-value function by solving the Bellman optimality equation directly. This is nontrivial since these are non-linear equations. The results are derived under the assumption that  $\mathcal{F}$  is closed under the application of the Bellman optimality operator. As it turns out, when  $\mathcal{F}$  contains functions that are linearly combining some basis functions, the closedness assumption can be dropped, if we consider approximating the optimal value function, rather than the optimal action-value function. This is the topic of Section 9, which is the penultimate section of the article.

## 6. LINEAR ACTION-VALUE FUNCTIONS WITH A GENERATIVE MODEL

The most permissive interaction protocol is when the learner is given a simulator that implements the global access interface, and this is where we start. To begin, we study the setting where the action-values for every policy are a linear function of a state/action feature vector. We make the strong assumption that the action-value function associated with all policies is linearly realisable using a given feature map:

**ASSUMPTION 6.1.** *The learner has access to a feature map  $\phi : S \times A \rightarrow \mathbb{R}^d$  such that:*

- (i) *For all policies  $\pi \in \Pi$  there exists a  $\theta_\pi \in \mathbb{R}^d$  such that  $q^\pi(s, a) = \langle \theta_\pi, \phi(s, a) \rangle$  for all  $(s, a) \in S \times A$ .*
- (ii)  *$\text{span}(\phi(s, a) : (s, a) \in S \times A) = \mathbb{R}^d$ .*

The assumption on the span of the feature vectors is for convenience only. If this does not hold, then certain matrices will not be invertible and the analysis must be carried out in a suitable subspace. Because we have assumed simulator access, the natural objective is to design an algorithm that finds a near-optimal policy using as few queries to the simulator as possible. The idea is to combine three key ingredients: approximate backwards induction, rollouts and linear regression.

### 6.1 Approximate backwards induction

Approximate backwards induction is a process for constructing a sequence of policies  $\pi_H, \dots, \pi_0$  where  $\pi_H$  is arbitrary and  $\pi_k$  is near-optimal for all states  $s \in \cup_{i=k}^H S_i$ . Given  $k \geq 1$ , a policy  $\pi_k$ , approximate backwards induction computes  $\pi_{k-1}$  using the following two-step procedure:

- (i) Interact with the environment/simulator to find a function  $q_k : S \times A \rightarrow \mathbb{R}$  that approximates the value function  $q^{\pi_k}$ .
- (ii) Let  $\pi_{k-1}$  be defined by

$$(2) \quad \pi_{k-1}(s) = \begin{cases} \arg \max_{a \in A} q_k(s, a) & \text{if } s \in S_{k-1} \\ \pi_k(s) & \text{otherwise.} \end{cases}$$

Note that while  $q_k$  is defined on the whole state space, only its behaviour on  $S_{k-1}$  matters. The quality of the sequence of policies depends on how well  $q_k$  approximates  $q^{\pi_k}$  here. For  $q : S \times A$ , let's denote by  $q|_{S_{k-1} \times A}$  the restriction of  $q$  to  $S_{k-1} \times A$ .

**THEOREM 6.2.** *Suppose that  $\pi_H, \dots, \pi_0$  are defined by (2) with  $\pi_H$  arbitrary. Then for all  $s \in S$ ,*

$$v^{\pi_0}(s) \geq v^*(s) - 2 \sum_{k=1}^H \varepsilon_k,$$

where for  $1 \leq k \leq H$ ,  $\varepsilon_k = \|(q_k - q^{\pi_k})|_{S_{k-1} \times A}\|_{\infty}$ .

**PROOF.** We will prove by induction that for any  $0 \leq k \leq H$ ,  $i \geq k$  and  $s \in S_i$ ,

$$(3) \quad v^{\pi_k}(s) \geq v^*(s) - 2 \sum_{j=i+1}^H \varepsilon_j.$$

The base case where  $k = H$  is satisfied trivially, since for any policy  $\pi$ ,  $v^{\pi}(s) = v^*(s)$  for all  $s \in S_H = \{\perp\}$ . Suppose  $k \geq 1$ , the above display holds for  $\pi_k$  and let  $\pi_{k-1}$  be constructed using (2). By definition  $\pi_{k-1} = \pi_k$  on  $\cup_{i=k}^H S_i$ , hence, for any state  $s$  from this set,  $v^{\pi_{k-1}}(s) = v^{\pi_k}(s)$ . Chaining this with (3) gives the result for any  $i \geq k$ . It remains to show the result for  $i = k-1$ , i.e., that for all  $s \in S_{k-1}$ ,

$$v^{\pi_{k-1}}(s) \geq v^*(s) - 2 \sum_{j=k}^H \varepsilon_j.$$

By the definition of the value function and the relation between  $\pi_k$  and  $\pi_{k-1}$ ,

$$\begin{aligned} v^{\pi_{k-1}}(s) &= q^{\pi_{k-1}}(s, \pi_{k-1}(s)) \\ &= q^{\pi_k}(s, \pi_{k-1}(s)) \quad (\pi_k, \pi_{k-1} \text{ agree on } S_k \cup \dots \cup S_H) \\ &\geq q_k(s, \pi_{k-1}(s)) - \varepsilon_k \\ &= \max_{a \in A} q_k(s, a) - \varepsilon_k \quad (\text{def. of } \pi_{k-1}) \\ &\geq \max_{a \in A} q^{\pi_k}(s, a) - 2\varepsilon_k \\ &= \max_{a \in A} \left[ r(s, a) + \sum_{s' \in S_k} \mathbf{P}(s'|s, a) v^{\pi_k}(s') \right] - 2\varepsilon_k \\ &\geq \max_{a \in A} \left[ r(s, a) + \sum_{s' \in S_k} \mathbf{P}(s'|s, a) v^*(s') \right] - 2 \sum_{j=k}^H \varepsilon_j \\ &= v^*(s) - 2 \sum_{j=k}^H \varepsilon_j, \end{aligned}$$

where in the one but last line we used (3) again.  $\square$

## 6.2 Rollouts

In order to implement approximate backwards induction we need a way to estimate  $q^{\pi}$  for a given policy  $\pi$ . This seems like a daunting task because  $q^{\pi}$  is an entire function. Rollouts provide a way to estimate  $q^{\pi}(s, a)$  for any given state/action pair. Let  $(s, a) \in S \times A$  and  $\pi$  be fixed. Given access to a global simulator, the learner can sample a trajectory starting at state/action pair  $(s, a)$  and subsequently following policy  $\pi$  until the end of the episode. Formally, a rollout of  $\pi$  from  $(s, a)$  when  $s \in S_k$  is a random trajectory  $T = (s_k, a_k, r_{k+1}, s_{k+1}, a_{k+1}, \dots, s_H, a_H, r_H)$  where  $(s_k, a_k) = (s, a)$  and  $a_i = \pi(s_i)$  for  $i > k$  and  $(r_{i+1}, s_{i+1}) \sim \mathbf{T}(\cdot|s_i, a_i)$ .

Given a trajectory  $T = (s_k, a_k, r_{k+1}, \dots, r_H)$ , let  $R(T) = \sum_{i=k}^H r_i$  be the cumulative reward along the trajectory, which is also known as the (random) *return* of  $\pi$  from  $(s, a)$ . The learner can estimate  $q^{\pi}(s, a)$  by sampling independent trajectories  $T_1, \dots, T_m$  and computing the average of the resulting returns:

$$q(s, a) = \frac{1}{m} \sum_{i=1}^m R(T_i).$$

Combining Assumption 2.1, which states that the rewards are bounded in  $[0, 1]$ , with Hoeffding's inequality [11] shows that with probability at least  $1 - \delta$ ,

$$(4) \quad |q(s, a) - q^{\pi}(s, a)| \leq H \sqrt{\frac{\log(1/\delta)}{2m}}.$$

Summarising, given any policy  $\pi$  and state/action  $(s, a)$  pair, the learner can estimate  $q^{\pi}(s, a)$  using rollouts. Note that in the online interaction protocol the learner can only sample trajectories from the initial state. On the other hand, with the global simulator interface the learner can sample a trajectory starting from any state, while with the local simulator interface the learner can sample rollouts from any previously visited state. For future reference, given  $(s, a) \in S \times A$  and a policy  $\pi$ , we let  $\mathbf{R}_{\Sigma}^{\pi}(\cdot|s, a)$  denote the *distribution* of the random return of  $\pi$  when started from  $(s, a)$ .

## 6.3 Linear extrapolation

Having noted that the learner can estimate  $q^{\pi}(s, a)$  using rollouts, we now explore how these estimates can be used to estimate the entire function  $q^{\pi}$ . In principle one could use rollouts starting at all state/action pairs, but the sample complexity of any algorithm implementing this idea will scale with the size of the state-space. By making use of the linear structure it is possible to estimate  $q^{\pi}$  on states that have never been visited by the learner. Given a distribution  $\rho$  on  $S \times A$  let its support be  $\text{spt}(\rho) = \{s, a : \rho(s, a) > 0\}$ . Let  $q : \text{spt}(\rho) \rightarrow \mathbb{R}$  be an

estimate of  $q^\pi$  on  $\text{spt}(\rho)$  and let  $\Delta \geq 0$  be such that for all  $(s, a) \in \text{spt}(\rho)$ ,

$$(5) \quad \Delta(s, a) \triangleq q(s, a) - q^\pi(s, a) \in [-\Delta, \Delta].$$

Let  $\theta_\pi \in \mathbb{R}^d$  be a vector such that  $q^\pi(s, a) = \langle \theta_\pi, \phi(s, a) \rangle$ . By assumption, such a vector exists. We estimate  $\theta_\pi$  using

$$(6) \quad \hat{\theta}_\pi = D_\rho^{-1} \sum_{(s,a) \in \text{spt}(\rho)} \rho(s, a) q(s, a) \phi(s, a),$$

where

$$D_\rho = \sum_{(s,a) \in \text{spt}(\rho)} \rho(a) \phi(s, a) \phi(s, a)^\top$$

is the design matrix associated with  $\rho$ , which is assumed to be nonsingular.

**PROPOSITION 6.1.** *Using the notation above, for any  $\eta \in \mathbb{R}^d$ ,*

$$\left| \langle \hat{\theta}_\pi - \theta_\pi, \eta \rangle \right| \leq \Delta \|\eta\|_{D_\rho^{-1}}.$$

**PROOF.** Let  $\eta \in \mathbb{R}^d$  be any vector. By algebra,  $\theta_\pi = D_\rho^{-1} \sum_{(s,a) \in \text{spt}(\rho)} \rho(s, a) q^\pi(s, a) \phi(s, a)$ . Using this, we bound the error of the estimator  $\hat{\theta}_\pi$  in direction  $\eta$  as follows:

$$\begin{aligned} \left| \langle \hat{\theta}_\pi - \theta_\pi, \eta \rangle \right| &= \left| \sum_{(s,a) \in \text{spt}(\rho)} \rho(s, a) \Delta(s, a) \langle D_\rho^{-1} \phi(s, a), \eta \rangle \right| \\ &\leq \Delta \sum_{(s,a) \in \text{spt}(\rho)} \rho(s, a) |\langle D_\rho^{-1} \phi(s, a), \eta \rangle| \\ &\leq \Delta \sqrt{\sum_{(s,a) \in \text{spt}(\rho)} \rho(s, a) \langle D_\rho^{-1} \phi(s, a), \eta \rangle^2} \\ &= \Delta \sqrt{\|\eta\|_{D_\rho^{-1}}^2}, \end{aligned}$$

where in the first inequality we used the definition of  $\Delta$  from (5) and the second we used Cauchy-Schwarz.  $\square$

Proposition 6.1 clarifies what is needed from  $\rho$ . By assumption,  $q^\pi(s, a) = \langle \theta_\pi, \phi(s, a) \rangle$  for all state/action pairs. Hence, in order to estimate  $q^\pi(s, a)$  for all state/action pairs it suffices to find a distribution  $\rho$  such that  $\|\phi(s, a)\|_{D_\rho^{-1}}^2$  is well-controlled for all  $s, a$ . Also note that in order to ensure that error  $\Delta$  in (5) is small we need to evaluate  $q^\pi$  only on  $s, a \in \text{spt}(\rho)$ , which suggests we should find a  $\rho$  such that  $|\text{spt}(\rho)|$  is small. To achieve this, we borrow a celebrated result from the theory of experimental design:

**PROPOSITION 6.2** ([26, 47]). *Let  $X \subset \mathbb{R}^d$  be compact and assume that  $\text{span}(X) = \mathbb{R}^d$ . There exists a distribution  $\rho$  on  $X$  such that*

- (i)  $\max_{x \in X} \|x\|_{D_\rho^{-1}} \leq \sqrt{2d}$  with  $D_\rho = \sum_{x \in \text{spt}(\rho)} \rho(x) x x^\top$ .
- (ii)  $|\text{spt}(\rho)| \leq 4d \log \log(d) + 16$ .

Summarising, letting  $X = \{\phi(s, a) : (s, a) \in \mathcal{S} \times \mathcal{A}\}$ , the learner can find a distribution  $\rho$  satisfying both of the above and then perform rollouts on all  $(s, a) \in \text{spt}(\rho)$  to construct estimators  $q(s, a)$  for  $(s, a) \in \text{spt}(\rho)$ . Then the entire function  $q^\pi$  can be estimated by  $q(s, a) = \langle \hat{\theta}_\pi, \phi(s, a) \rangle$  and the calculations above ensure that  $\|q - q^\pi\|_\infty \leq \Delta \sqrt{2d}$ .

## 6.4 Approximate backwards induction

We now have the necessary tools to implement approximate backwards induction with access to a global simulator (Algorithm 1). The algorithm accepts as input an arbitrary initial policy, a distribution  $\rho \in \Delta_{\mathcal{S} \times \mathcal{A}}$  that should ideally satisfy the conclusions of Proposition 6.2 and a natural number  $m$  that is the number of rollouts to perform when estimating action-values. One may also want to introduce a variant that “localises”  $\rho$  to the layers; we omit this refinement, which, may be valuable in practice but offers no theoretical advantage in the worst-case.

```

1  args:  $\pi_H \in \Pi$ ,  $\rho \in \Delta_{\mathcal{S} \times \mathcal{A}}$ ,  $m$ 
2  compute  $D_\rho^{-1} = \sum_{(s,a) \in \text{spt}(\rho)} \rho(s, a) \phi(s, a) \phi(s, a)^\top$ 
3  for  $k = H$  to 1:
4    for  $(s, a) \in \text{spt}(\rho)$ :
5      sample  $(R_i)_{i=1}^m$  iid from  $\mathbf{R}_{\Sigma^k}^\pi(\cdot | s, a)$ 
6       $q_{k, \text{spt}}(s, a) = \frac{1}{m} \sum_{i=1}^m R_i$ 
7       $\hat{\theta}_k = D_\rho^{-1} \sum_{(s,a) \in \text{spt}(\rho)} q_{k, \text{spt}}(s, a) \phi(s, a)$ 
8       $\pi_{k-1}(s) = \begin{cases} \arg \max_{a \in \mathcal{A}} \langle \phi(s, a), \hat{\theta}_k \rangle & \text{if } s \in \mathcal{S}_{k-1} \\ \pi_k(s) & \text{otherwise} \end{cases}$ 
9  return  $\pi_0$ 

```

**ALGORITHM 1:** Approximate backwards induction for  $q^\pi$ -linearly realisable MDPs

The following theorem bounds the sample complexity of Algorithm 1.

**THEOREM 6.3.** *Under Assumption 6.1, suppose that Algorithm 1 is run with*

$$m = \frac{dH^4}{\epsilon^2} \log \left( \frac{|\text{spt}(\rho)|H}{\delta} \right).$$

*Then, with probability at least  $1 - \delta$ ,  $v^{\pi_H}(s) \geq v^*(s) - \epsilon$  for all  $s \in \mathcal{S}$  and the number of queries to the simulator is at most*

$$\frac{|\text{spt}(\rho)|dH^6}{\epsilon^2} \log \left( \frac{|\text{spt}(\rho)|H}{\delta} \right).$$

REMARK. When the distribution  $\rho$  satisfies the conclusions of Proposition 6.2, then  $|\text{spt}(\rho)| = O(d \log \log(d))$  and the sample complexity of Algorithm 1 is

$$O\left(\frac{d^2 H^6}{\varepsilon^2} \log \log(d) \log\left(\frac{dH}{\delta}\right)\right),$$

which, importantly, is independent of the size of the state-space.

PROOF OF THEOREM 6.3. To begin we control the number of queries to the simulator. Every sample from  $\mathbf{R}_{\Sigma}^{\pi_k}(s, a)$  makes use of at most  $H$  queries to the simulator. Hence, in every iteration the algorithm makes at most  $|\text{spt}(\rho)|mH$  queries to the simulator and since there are  $H$  iterations the number of queries to the simulator during the entire interaction is at most

$$|\text{spt}(\rho)|mH^2 = O\left(\frac{|\text{spt}(\rho)|dH^6}{\varepsilon^2} \log\left(\frac{1}{\delta}\right)\right).$$

Next we prove that  $v^{\pi_H}$  is near-optimal. Let  $B = |\text{spt}(\rho)|H$ , which is the number of state/action pairs for which Algorithm 1 uses rollouts to estimate action-values. By (4) and a union bound, with probability at least  $1 - \delta$ , for all  $1 \leq k \leq H$ ,

$$\max_{s, a \in \text{spt}(\rho)} |q_{k, \text{spt}}(s, a) - q^{\pi_k}(s, a)| \leq H \sqrt{\frac{\log(B/\delta)}{2m}} \triangleq \Delta.$$

Assume for the remainder that this event holds. By Proposition 6.1, with  $q_k(s, a) = \langle \theta_k, \phi(s, a) \rangle$ ,

$$\begin{aligned} \max_{s, a \in S \times A} |q_k(s, a) - q^{\pi_k}(s, a)| &\leq \Delta \sqrt{\|\phi(s, a)\|_{D_\rho^{-1}}^2} \\ &\leq \Delta \sqrt{2d}. \end{aligned}$$

Finally, by Theorem 6.2, for any  $s \in S$

$$v^{\pi_H}(s) \geq v^*(s) - \Delta H \sqrt{2d} = \varepsilon,$$

where we used the definitions of  $\Delta$  and the number of rollouts  $m$ .  $\square$

## 6.5 Misspecified linear model

Theorem 6.3 provides a sample complexity bound for approximate backwards induction when all  $q^\pi$  is linearly realisable for all policies  $\pi$ . One might wonder what happens if the action-value functions are only nearly linearly realisable.

ASSUMPTION 6.4. *There exists an  $\varepsilon_{\text{approx}} > 0$  such that for all policies  $\pi \in \Pi$  there exists a  $\theta_\pi \in \mathbb{R}^d$  such that*

$$\max_{s, a \in S \times A} |q^\pi(s, a) - \langle \phi(s, a), \theta_\pi \rangle| \leq \varepsilon_{\text{approx}}.$$

Furthermore,  $\text{span}(\phi(s, a) : s, a \in S \times A) = \mathbb{R}^d$ .

As before, the span assumption is only for convenience. Algorithm 1 still yields a sample-complexity bound without modification but the bound and analysis need to be modified slightly.

THEOREM 6.5. *Suppose that Algorithm 1 is run with distribution  $\rho$  satisfying the conclusions of Proposition 6.2 and for  $\varepsilon > 0$ ,*

$$m = \frac{dH^2 \log(H|\text{spt}(\rho)|/\delta)}{\varepsilon^2}.$$

*Then, with probability at least  $1 - \delta$ ,*

$$v^{\pi_H}(s) \geq v^*(s) - 2H \left( \varepsilon_{\text{approx}}(1 + \sqrt{2d}) + \varepsilon \right).$$

*Furthermore, the number of queries to the simulator is at most*

$$\frac{d|\text{spt}(\rho)|H^4 \log(|\text{spt}(\rho)|H/\delta)}{\varepsilon^2}.$$

Comparing Theorem 6.5 to Theorem 6.3, notice that the latter theorem shows that with high probability the learned policy is  $\varepsilon$ -optimal, while with misspecification the error depends on the horizon, the dimension and the approximation error. The contribution of the approximation errors does not vanish with more rollouts, which means there is no reason to choose  $\varepsilon \leq \varepsilon_{\text{approx}} \sqrt{d}$ , since this only increases the number of queries but only improves the policy by a constant factor even as  $m \rightarrow \infty$ . With  $\varepsilon = \varepsilon_{\text{approx}} \sqrt{d}$ , Algorithm 1 uses

$$O\left(\frac{dH^4 \log \log(d) \log(1/\delta)}{\varepsilon_{\text{approx}}^2}\right)$$

queries to the generative model and with high probability returns a policy  $\pi_H$  such that

$$v^{\pi_H}(s) \geq v^*(s) - 8H\varepsilon_{\text{approx}} \sqrt{d}.$$

PROOF SKETCH OF THEOREM 6.5. The only difference relative to Theorem 6.3 is that the linear extrapolation works differently. As in Theorem 6.3 the number of queries to the simulator is at most  $m|\text{spt}(\rho)|H$ . Also following the proof of Theorem 6.3, with probability at least  $1 - \delta$  for all  $1 \leq k \leq H$ ,

$$\max_{s, a \in \text{spt}(\rho)} |q_{k, \text{spt}}(s, a) - q^{\pi_k}(s, a)| \leq H \sqrt{\frac{\log(B/\delta)}{2m}} \triangleq \Delta,$$

where  $B = H|\text{spt}(\rho)|$ . For the remainder assume this event occurs. The main difference is that Proposition 6.1 needs modification. In particular, we now have for all  $\eta \in \mathbb{R}^d$

$$|\langle \theta_k - \theta_{\pi_k}, \eta \rangle| \leq (\Delta + \varepsilon_{\text{approx}}) \|\eta\|_{D_\rho^{-1}}.$$

Hence, since  $\rho$  was chosen to satisfy the conclusions of Proposition 6.2, for any state/action pair  $s, a$ ,

$$\begin{aligned} |q_k(s, a) - q^{\pi_k}(s, a)| &= |\langle \theta_k, \phi(s, a) \rangle - q^{\pi_k}(s, a)| \\ &\leq |\langle \theta_k - \theta_{\pi_k}, \phi(s, a) \rangle| + \varepsilon_{\text{approx}} \\ &\leq \Delta \sqrt{2d} + \varepsilon_{\text{approx}} (1 + \sqrt{2d}) \\ &= \varepsilon + \varepsilon_{\text{approx}} (1 + \sqrt{2d}). \end{aligned}$$

Now, Theorem 6.2 yields the claim.  $\square$

Theorem 6.5 shows that learning is possible with misspecified linear models. But there is an unpleasant aspect to the result because the even by sending  $m \rightarrow \infty$  the learner does not identify an optimal policy. Since  $S$  and  $A$  are finite the learner can obviously query the generative model at all state/action pairs many times and identify a near-optimal policy to any accuracy, but the number of queries required to do this will be  $\text{poly}(S, A)$ , which is usually unacceptable. What one might hope for is that with  $\text{poly}(d, H, 1/\varepsilon_{\text{approx}})$  queries it might be possible to find a policy with error  $O(\varepsilon_{\text{approx}})$ , since this is the misspecification level. But the bound in Theorem 6.5 suggests the limit of Algorithm 1 is  $O(\varepsilon_{\text{approx}} H \sqrt{d})$ . Interestingly, one can show that removing the  $\sqrt{d}$  is essentially not possible without using exponentially many queries (in the dimension), which is a beautiful application of the Johnson-Lindenstrauss lemma [18].

## 6.6 Remarks and references

- Except for minor differences, approximate backwards induction was proposed and analysed in the offline setting by Bagnell et al. [8] under the name “policy search by dynamic programming” (dynamic programming is often used synonymously to backwards induction).
- All the fundamental ideas in this section generalise to the discounted setting, in most cases with a more-or-less identical argument and where  $H$  is replaced by the *effective horizon*, which is the number of interactions before the all future rewards are discounted to a negligible amount and is  $1/(1 - \gamma) \log(1/\varepsilon)$ . The reader is recommended to check out the work of Scherrer [41] on this topic.
- Rather than approximate backwards induction, it is tempting to use an approximate form of *policy iteration*. This corresponds to eliminating the second case in (2) by setting  $\pi_{k-1}(s) = \arg \max_{a \in A} q_k(s, a)$  for all  $s \in S$ , including states in  $S_k \cup S_H$ . It is not hard to see that in this case Theorem 6.2 still holds but with a worse error bound. The difference is easiest to explain when  $\varepsilon = \varepsilon_1 = \dots = \varepsilon_H$ . In this case, for approximate backwards induction the error becomes  $O(H\varepsilon)$ , while for approximate policy iteration it is  $O(H^2\varepsilon)$ . The reason

for the worse, quadratic horizon dependence is as follows: While for approximate backward induction, for any  $s \in S_k \cup \dots \cup S_H$ , setting  $a = \pi_{k-1}(s)$  we have  $q^{\pi_{k-1}}(s, a) = q^{\pi_k}(s, a)$ , this property only holds up to an additive error in the case of approximate policy iteration. As it turns out, the quadratic dependence on the horizon here cannot be improved.

In the discounted setting, a similar issue arises. In this case, it has been shown that the extra  $H$  factor that shows up in the error accumulation cannot be removed. Approximate backwards induction is not the only method that incurs a linear-in-the-horizon error bound.

Other ways of removing the extra  $H$  factor are performing “soft” policy updates, which however means that more iterations may be needed [3, 5]. A simpler, more direct alternative is “confident approximate policy iteration”, where the policy is not updated unless the action value for the new policy is plausible larger than that of the previous policy [54].

- The best known lower bound in the realisable setting of this section is that the sample complexity to find an  $\varepsilon$ -optimal policy is at least

$$\Omega\left(\frac{d^2 H^3}{\varepsilon^2}\right).$$

This lower bound is shown for the narrower class of linear MDPs [54, Theorem H.3]. Most likely the lower bound is tight and the upper bound can be improved by using a more careful concentration analysis.

- The implementation of approximate backwards induction relied on a global simulator. Polynomial sample complexity is possible even in the online RL model using a more sophisticated procedure [51]. It is unknown whether or not this is possible with a polynomial time algorithm. Simpler, and efficient methods work when a local access simulator interface is available [54, 56].

## 7. ELUDER DIMENSION

The *eluder dimension* is a complexity measure of a real-valued function class that was proposed by Russo and Van Roy [39] for the analysis of *optimistic algorithms* with function approximation, usually in the online interaction protocol. Optimistic algorithms start with a set of possible models of the environment, gradually shrinking this set based on the information collected. In each step they choose the model from the remaining set that is the most rewarding and plays the policy that is optimal for this optimistic model.

### 7.1 Bandit environments

To keep things simple we consider *bandit problems*, which are essentially MDPs with a single state. In the

online interface the learner selects actions  $a_1, \dots, a_T$  in an action set  $A$  and observes rewards  $r_1, \dots, r_T$  where  $r_t \sim \mathbf{R}(\cdot|a_t)$  with  $\mathbf{R}$  a probability kernel from the action set  $A$  to  $[0, 1]$ . The mean reward of action  $a$  is  $f^*(a) = \int_0^1 \mathbf{R}(dx|a)$ . The learner's objective is to select actions with the highest immediate reward.

**REMARK.** Bandit problems look *much* simpler than RL. There is no state and the reward of an action is immediately observed. And yet, almost all of the techniques used to study bandit problems have been generalised to RL. This is not so surprising. An RL problem can be cast as a bandit by viewing policies as actions. In RL the learner also observes the sequence of states at the end of the episode. This information should be used to update statistical evidence but otherwise bandits and (episodic) RL are almost equivalent.

In the classical literature on finite-armed bandits the learner has no information about the unknown mean reward function  $r$  and the action set is not large. For our purposes, however, we are interested in the case where  $A$  is potentially enormous but the learner has prior information about the space of possible mean reward functions. To model this, let  $\mathcal{F}$  be the set of possible mean reward functions, which are functions from  $A$  to  $[0, 1]$ . This is the class of models we start with. The learner knows  $\mathcal{F}$  and the goal is to design an algorithm whose regret is small whenever the true (but initially unknown) mean reward function belongs to  $\mathcal{F}$ . Our standing assumption for the rest of the section is as follows:

**ASSUMPTION 7.1.** *The following hold:*

- (i) *Realisability:*  $f^* \in \mathcal{F}$ .
- (ii) *Finiteness:*  $\mathcal{F}$  and  $A$  are both finite.

The finiteness assumption is for convenience only, to ensure the existence of certain maximisations and to permit us to take union bounds over  $\mathcal{F}$  rather than relying on covering arguments. For  $\mathcal{C} \subset \mathcal{F}$ , we will find it useful to introduce  $\mathcal{W}_{\mathcal{C}} \triangleq \text{sym}(\mathcal{C}) = \{f - g : g \in \mathcal{C}\}$ . For reasons that will become clear soon, one should think of  $\mathcal{W}_{\mathcal{C}}$  as the set of possible regret functions.

## 7.2 Optimism and motivation for eluder dimension

After interacting with the bandit environment for some time, the learner somehow constructs a confidence set of plausible reward functions  $\mathcal{C} \subset \mathcal{F}$ . More on this later, but ideally  $\mathcal{C}$  should be as small as possible (in some sense) and contain the true reward function:  $f^* \in \mathcal{C}$ . The optimistic algorithm plays the action associated with the most rewarding function in  $\mathcal{C}$ , which is

$$a_{\mathcal{C}} = \arg \max_{a \in A} \max_{f \in \mathcal{C}} f(a),$$

which we assume exists. The following elementary proposition bounds the regret of the optimistic action:

**PROPOSITION 7.1.** *Take any  $\mathcal{C} \subset \mathcal{F}$  with  $f^* \in \mathcal{C}$  and assume that  $(f_{\mathcal{C}}, a_{\mathcal{C}}) = \arg \max_{(h,a) \in \mathcal{C} \times A} h(a)$  is well-defined. Then*

$$\sup_{a \in A} f^*(a) - f^*(a_{\mathcal{C}}) \leq \sup_{w \in \mathcal{W}_{\mathcal{C}}} w(a_{\mathcal{C}}) \triangleq \|a_{\mathcal{C}}\|_{\mathcal{W}_{\mathcal{C}}}.$$

**REMARK.** The notation  $\|a\|_{\mathcal{W}} = \sup_{w \in \mathcal{W}} w(a)$  is a little questionable. The set  $A$  may not even be a vector space. At least when  $\mathcal{W} = \text{sym}(\mathcal{C})$  with non-empty  $\mathcal{C}$ , then  $\|\cdot\|_{\mathcal{W}}$  is non-negative.

In words, given a subset  $\mathcal{C}$  of all possible models, the regret of the optimistic choice against any model that is in  $\mathcal{C}$  can be upper bounded by the value that some regret function takes when evaluated at the optimistic choice.

**PROOF.** Thanks to  $\sup_{a \in A} f^*(a) \leq f_{\mathcal{C}}(a_{\mathcal{C}})$ , which follows from the definitions, the desired inequality holds with  $w = f_{\mathcal{C}} - f^* \in \text{sym}(\mathcal{C}) = \mathcal{W}_{\mathcal{C}}$ .  $\square$

To motivate the definition of the eluder dimension we start by considering deterministic bandit environments where  $r_t = f^*(a)$  for some  $f^* \in \mathcal{F}$ . To keep things simple we assume that  $A$  and  $\mathcal{F}$  are finite. Since  $A$  is finite, the learner can play every action in  $A$  once and subsequently play the action leading to the maximum reward. Clearly this leads to at most  $|A| - 1$  suboptimal actions, which may be reasonable for small action sets. When  $A$  is large, and  $\mathcal{F}$  is restricted, one can exploit the prior knowledge  $f^*$  belongs to  $\mathcal{F}$  to do significantly better. The bandit algorithm that is based on optimism is as follows:

```

1 for  $t = 1$  to  $\infty$ :
2   let  $\mathcal{C}_t = \{f \in \mathcal{F} : f(a_s) = r_s \text{ for all } s < t\}$ 
3   let  $f_t = \arg \max_{f \in \mathcal{C}_t} \max_{a \in A} f(a)$ 
4   play  $a_t = \arg \max_{a \in A} f_t(a)$ 
5   observe  $r_t = f^*(a_t)$ 

```

**ALGORITHM 2:** Optimistic algorithm for structured deterministic bandits

Let us think about how many times Algorithm 2 can make a mistake. Let  $\mathcal{W}_t = \text{sym}(\mathcal{C}_t)$ . By Proposition 7.1,

$$\max_{a \in A} f^*(a) - f^*(a_t) \leq \|a_t\|_{\mathcal{W}_t},$$

Since  $r_s = f^*(a_s)$ , by definition of  $\mathcal{C}_t$

$$\begin{aligned}
\mathcal{W}_t &\subset \left\{ f - g : f, g \in \mathcal{F}, \sum_{s=1}^{t-1} (f(a_s) - g(a_s))^2 \leq 0 \right\} \\
(7) \quad &= \left\{ w \in \mathcal{W} : \sum_{s=1}^{t-1} w(a_s)^2 \leq 0 \right\} \triangleq \tilde{\mathcal{W}}_t,
\end{aligned}$$

where  $\mathcal{W} = \text{sym}(\mathcal{F})$ . Hence, Algorithm 2 can only make a mistake in round  $t$  if

$$\|a_t\|_{\mathcal{W}_t} > 0.$$

The eluder dimension (for deterministic problems) is defined exactly as the maximum number of times this can happen.

REMARK. Finiteness of  $A$  and  $\mathcal{F}$  were only assumed to ensure the existence of the maximisers  $f_t$  and  $a_t$ . One can easily adapt the arguments above to arbitrary action and function classes by allowing an arbitrarily small amount of error (even an optimal action may not exist in this case).

### 7.3 Eluder dimension

Because the eluder dimension is not always applied to bandit problems, we now make the notation more abstract. Let  $\mathcal{W}$  be a set of real-valued functions with domain  $X$ .

DEFINITION 7.2. A sequence  $x_1, \dots, x_k$  in  $X$  is called an *eluder sequence* for  $\mathcal{W}$  at level  $(\varepsilon, \beta) \in \mathbb{R}_+^2$  if for all  $t \leq k$ ,

$$\sup_{w \in \mathcal{W}} \left\{ w(x_t) : \sum_{s=1}^{t-1} w(x_s)^2 \leq \beta^2 \right\} > \varepsilon.$$

We say that  $x_1, \dots, x_k$  is an *eluder sequence* at level  $\varepsilon$  if it is an eluder sequence at level  $(\varepsilon, \varepsilon)$ .

DEFINITION 7.3. Given  $\varepsilon > 0$ , the *exact eluder dimension*  $\text{Edim}_=(\mathcal{W}, \varepsilon, \beta)$  of class  $\mathcal{W}$  at level  $(\varepsilon, \beta)$  is the length of the longest eluder sequence at level  $(\varepsilon, \beta)$ . We abbreviate  $\text{Edim}_=(\mathcal{W}, \varepsilon) = \text{Edim}_=(\mathcal{W}, \varepsilon, \varepsilon)$ , which is usually called the *exact eluder dimension* at level  $\varepsilon$ . Finally, the *eluder dimension* at level  $\varepsilon_0 > 0$  is  $\text{Edim}(\mathcal{W}, \varepsilon_0) = \sup_{\varepsilon \geq \varepsilon_0} \text{Edim}_=(\mathcal{W}, \varepsilon)$ .

REMARK. By construction,  $\varepsilon_0 \mapsto \text{Edim}(\mathcal{W}, \varepsilon_0)$  is non-increasing, which may not be true of the exact eluder dimension.

The definition of the exact eluder dimension almost immediately yields a bound on the number of mistakes made by Algorithm 2:

PROPOSITION 7.2. Let  $(a_t)_{t=1}^\infty$  be the sequence of actions generated by Algorithm 2. Then

$$\sum_{t=1}^\infty \mathbf{1}(f^*(a_t) \neq f^*(a_\star)) \leq \text{Edim}_=(\text{sym}(\mathcal{F}), 0).$$

PROOF. Let  $T$  be the set of rounds where  $\max_{a \in A} f^*(a) - f^*(a_t) > 0$ . By the definition of  $\mathcal{C}_t$  it holds that  $f^* \in \mathcal{C}_t$  for all rounds  $t$ . Hence, Proposition 7.1 gives that  $\max_{a \in A} f^*(a) - f^*(a_t) \leq \|a_t\|_{\mathcal{W}_t}$ , where  $\mathcal{W}_t = \mathcal{W}_{\mathcal{C}_t}$ . By Equation (7),  $\mathcal{W}_t \subset \mathcal{W}_t = \{w \in \mathcal{W} : \sum_{s=1}^{t-1} w(a_s)^2 = 0\}$ . Hence, the sequence  $(a_t)_{t \in T}$  is an Eluder sequence for  $\mathcal{W} = \text{sym}(\mathcal{F})$  at level  $\varepsilon = 0$  (for  $t \in T$ , the witness is  $f_t - f^* \in \mathcal{W}$ : Indeed for  $s < t$ ,  $f_t(a_s) = f^*(a_s)$ , while  $f_t(a_t) > f^*(a_t)$ ) and therefore  $|T| \leq \text{Edim}_=(\mathcal{F}, 0)$ .  $\square$

EXAMPLE (Linear bandits). Let  $X \subset \mathbb{B}_1 \subset \mathbb{R}^d$  and  $r > 0$  and  $\mathcal{F}_r = \{x \mapsto \langle x, \theta \rangle : x \in X, \theta \in \mathbb{B}_r\}$ . Then,

$$\text{Edim}_=(\text{sym}(\mathcal{F}_r), 0) \leq d$$

(in fact, equality holds). Hence, the number of mistakes made by Algorithm 2 is at most  $d$ , independently of the size of the action set.

When there is noise, the confidence set used by Algorithm 2 is no longer reasonable. The quantity  $\beta$  in the eluder dimension above is introduced to accommodate the presence of noise, while  $\varepsilon$  now appears because in general it is no longer reasonable to expect any algorithm to play suboptimally only finitely often. Rather, one should hope that “the number of large errors”,  $\text{Edim}(\mathcal{W}, \varepsilon, \beta)$ , is not too big. As it turns out, to guarantee this, it suffices to guarantee that  $\text{Edim}(\mathcal{W}, \varepsilon)$  is under control:

PROPOSITION 7.3. The following holds:

$$\text{Edim}(\mathcal{W}, \varepsilon, \beta) \leq \left( \frac{\beta^2}{\varepsilon^2} + 1 \right) \text{Edim}(\mathcal{W}, \varepsilon).$$

By this proposition,  $\text{Edim}(\mathcal{W}, \varepsilon, \beta)$  is redundant; while the literature does not define this quantity, we define it because we think it helps to understand the intuition underlying the eluder dimension.

PROOF. Let  $x_1, \dots, x_k$  be an eluder sequence at level  $(\varepsilon, \beta)$  and  $\|w\|_t^2 = \sum_{s=1}^t w(x_s)^2$  and  $\mathcal{W}_t = \{w \in \mathcal{W} : \|w\|_{t-1}^2 \leq \beta^2\}$ . By definition there exists a  $w_t \in \mathcal{W}_t$  such that  $w_t(x_t) > \varepsilon$ . Starting at  $t = 1$  we will iteratively assign the time indices  $\{1, 2, \dots, n\}$  to one of

$$m = \left\lceil \frac{\beta^2}{\varepsilon^2} \right\rceil \leq \frac{\beta^2}{\varepsilon^2} + 1$$

buckets. The buckets are initially empty and time index  $t$  is added to a bucket  $\mathcal{B}$  for which

$$\sum_{s \in \mathcal{B}} w_t^2(x_s) \leq \varepsilon^2$$

Such a bucket exists because there are at most  $m$  buckets and  $\|w_t\|_{t-1}^2 \leq \beta^2$  by assumption and the buckets contain indices only up to  $t - 1$ . Now, note that for any bucket  $\mathcal{B}$ , the sequence  $(x_t)_{t \in \mathcal{B}}$  is an eluder sequence at level  $(\varepsilon, \varepsilon)$ . Therefore  $k$  is at most  $m \text{Edim}(\mathcal{F}, \varepsilon)$ .  $\square$

Before continuing with the propositions justifying the choices in the definition of eluder sequences, let us bound the eluder dimension for linear function classes.

**PROPOSITION 7.4 ([39]).** *Let  $\mathcal{F}_r$ ,  $X$  be as in Example 7.3 and let  $\mathcal{W}_r = \text{sym}(\mathcal{F}_r)$ . Then, for any  $\varepsilon_0 > 0$ ,*

$$\text{Edim}(\mathcal{W}_r, \varepsilon_0) = O\left(d \log\left(1 + \frac{r}{\varepsilon_0}\right)\right).$$

**PROOF.** By definition, any eluder sequence  $x_1, \dots, x_k$  at level  $\varepsilon = 0$  must be pairwise orthogonal and non-zero. Hence  $\text{Edim}_=(\mathcal{W}_r, 0) = d$  with the lower bound witnessed by the standard basis vectors. Moving to the second part. By definition,  $\mathcal{W}_r = \text{sym}(\mathcal{F}_r) = \mathcal{F}_{2r}$ . Let  $\varepsilon_0 > 0$  and for  $\theta \in \mathbb{R}^d$  and  $x \in X$ , let  $w_\theta(x) = \langle x, \theta \rangle$ . Suppose that  $x_1, \dots, x_k \in X$  is an eluder sequence at level  $\varepsilon \geq \varepsilon_0$ . For  $1 \leq i \leq k$ , let  $G_i = \sum_{j=1}^i x_j x_j^\top$ , which is chosen so that

$$\sum_{j=1}^{i-1} w_\theta(x_j)^2 = \|\theta\|_{G_{i-1}}^2.$$

Using that  $x_1, \dots, x_k$  is an eluder sequence, for any  $1 \leq i \leq k$ , we have

$$\begin{aligned} \varepsilon &\leq \sup_{w \in \mathcal{W}_r} \left\{ w(x_i) : \sum_{j=1}^{i-1} w(x_j)^2 \leq \varepsilon^2 \right\} \\ &= \max\{w_\theta(x_i) : \theta \in \mathbb{B}_{2r}, \|\theta\|_{G_{i-1}}^2 \leq \varepsilon^2\} \\ &\leq \max\{\langle x_i, \theta \rangle : \theta \in \mathbb{R}^d, \|\theta\|_{G_{i-1} + \lambda I}^2 \leq \varepsilon^2 + 4r^2\lambda\} \\ &= 2\varepsilon \|x_i\|_{(G_{i-1} + \lambda I)^{-1}}, \end{aligned}$$

where in the final equality we analytically solved the quadratic program and chose  $\lambda = \frac{3\varepsilon^2}{4r^2}$  to get a simple expression. Therefore, by the elliptical potential lemma [30, Lemma 19.4] or [1, Lemma 11],

$$\begin{aligned} \frac{k}{4} &\leq \sum_{i=1}^k \min\left(1, \|x_i\|_{(G_{i-1} + \lambda I)^{-1}}^2\right) \\ &\leq 2d \log\left(1 + \frac{4kr^2}{3d\varepsilon^2}\right) \leq 2d \log\left(1 + \frac{4kr^2}{3d\varepsilon_0^2}\right). \end{aligned}$$

□

Finally, the proposition that justify the choices in the definition of eluder sequences is as follows:

**PROPOSITION 7.5.** *Fix  $n \geq 1$ ,  $\beta > 0$ ,  $x_1, \dots, x_n \in X$  and  $\mathcal{W}_1, \dots, \mathcal{W}_n \subset \mathcal{W}$ . Assume that for any  $1 \leq t \leq n$ ,  $w \in \mathcal{W}_t$ ,  $\sum_{s=1}^{t-1} w(x_s)^2 \leq \beta^2$ . Then, for any  $\varepsilon > 0$ ,*

$$(8) \quad \sum_{t=1}^n \mathbf{1}(\|x_t\|_{\mathcal{W}_t} > \varepsilon) \leq \left(\frac{\beta^2}{\varepsilon^2} + 1\right) \text{Edim}(\mathcal{W}, \varepsilon).$$

You should think of  $\mathcal{W}_t$  as the symmetrisation of some confidence set. Since all functions in a confidence set should approximately agree on the data  $(x_s)_{s=1}^{t-1}$ , functions in  $\mathcal{W}_t$  should nearly vanish (cumulatively) on  $(x_s)_{s=1}^{t-1}$ , which is the condition in Proposition 7.5. The proposition shows that as long as this is the case, then the total number of times functions in the confidence set disagree is controlled in terms of the eluder dimension.

**PROOF.** Let  $T = \{1 \leq t \leq n : \|x_t\|_{\mathcal{W}_t} > \varepsilon\}$  and apply Proposition 7.3 to the sequence  $(x_t)_{t \in T}$ . □

**COROLLARY 7.1.** *Assume the conditions in Proposition 7.5 hold. Let  $d = \text{Edim}(\mathcal{W}, \frac{1}{n})$ ,  $B = \sup_{x \in X} \|x\|_{\mathcal{W}}$ . Then,*

$$(9) \quad \sum_{t=1}^n \|x_t\|_{\mathcal{W}_t} = O\left(Bd + \sqrt{nd\beta^2}\right).$$

The significance of this result is that it allows the control of regret for any case when the eluder dimension of the set  $\mathcal{W}$  is finite. Note that the argument in this corollary is completely general and in fact shows how the uniform PAC-style error bound (Equation 8) gives rise to a regret bound that scales with  $O(\sqrt{n})$ .

**PROOF.** Let

$$k_{\min} = \lceil \log_2(1/n) \rceil \text{ and } k_{\max} = \lfloor \log_2(B) \rfloor,$$

and  $c_t = \|x_t\|_{\mathcal{W}_t}$ . Then, by Proposition 7.5,

$$\begin{aligned} \sum_{t=1}^n c_t &= \sum_{t=1}^n \sum_{k=-\infty}^{\infty} c_t \mathbf{1}_{[2^k, 2^{k+1})}(c_t) \\ &\leq n2^{k_{\min}} + \sum_{k=k_{\min}}^{k_{\max}} 2^{k+1} \left(\frac{4\beta^2}{2^{2k}} + 1\right) d \\ &= O\left(Bd + \sqrt{d\beta^2 n}\right), \end{aligned}$$

where the first equality is immediate. The inequality follows from Proposition 7.5, the definitions of  $B$  and  $k_{\max}$  and that  $\varepsilon \mapsto \text{Edim}(\mathcal{W}, \varepsilon)$  is non-increasing. The Big-O equality follows from the definitions of  $k_{\min}$  and  $k_{\max}$  and by evaluating the geometric sum. □

## 7.4 Stochastic setting

When the reward is noisy, the eluder dimension with a positive level is needed as in this case, to prevent eliminating the true reward function from the confidence set  $\mathcal{C}_t$  we need to allow functions that disagree with the observed (noisy) rewards. Algorithm 3 shows the necessary adjustment to the previous algorithm.

In line 2 the algorithm computes the best reward model given the data so far. The confidence set will then contain

```

1  args:  $\mathcal{F}$  and constants  $(\beta_t)_{t=1}^\infty$ 
2  for  $t = 1$  to  $\infty$ :
3    let  $\hat{f}_t = \arg \min_{f \in \mathcal{F}} \sum_{s=1}^{t-1} (f(a_s) - r_s)^2$ 
4    let  $\mathcal{C}_t = \{f \in \mathcal{F} : \sum_{s=1}^{t-1} (\hat{f}_t(a_s) - f(a_s))^2 \leq \beta_t^2\}$ 
5    let  $f_t = \arg \max_{f \in \mathcal{C}_t} \max_{a \in A} f(a)$ 
6    play  $a_t = \arg \max_{a \in A} f_t(a)$ 
7    observe  $r_t \sim \mathbf{R}(\cdot|a_t)$ 

```

ALGORITHM 3: Optimistic algorithm for structured stochastic bandits

all other functions from  $\mathcal{F}$  that are “within noise level” relative to what values this function returns at the actions for which the algorithm has data. We now show the following result, which, for simplicity, is stated for the case when  $\mathcal{F}$  is finite. We give a hint on how to extend the result to the general case later.

**THEOREM 7.4.** *Suppose that  $\delta \in (0, 1)$  and set*

$$\beta_t^2 = 8 \log \left( \frac{t(t+1)|\mathcal{F}|}{\delta} \right).$$

*Then, with probability at least  $1 - \delta$ , for any  $\varepsilon > 0$  and any  $n \geq 1$ ,*

$$\sum_{t=1}^n \mathbf{1}(f^*(a_*) - f^*(a_t) > \varepsilon) \leq \left( \frac{4\beta_n^2}{\varepsilon^2} + 1 \right) \text{Edim}(\mathcal{F}, \varepsilon).$$

**PROOF.** Suppose for a moment that  $f^* \in \mathcal{C}_t$  for all  $t \leq n$ . Let  $\mathcal{W}_t = \text{sym}(\mathcal{C}_t)$ . By Proposition 7.1, it suffices to bound

$$\sum_{t=1}^n \mathbf{1}(\|a_t\|_{\mathcal{W}_t} > \varepsilon).$$

Our goal now is to show that Proposition 7.5 can be applied with  $X = A$ ,  $\mathcal{W} = \text{sym}(\mathcal{F})$ ,  $\beta^2 = 4\beta_n^2$ . Indeed, if this true then

$$\sum_{t=1}^n \mathbf{1}(\|a_t\|_{\mathcal{W}_t} > \varepsilon) \leq \left( \frac{4\beta_n^2}{\varepsilon^2} + 1 \right) \text{Edim}(\mathcal{F}, \varepsilon).$$

Fix  $1 \leq t \leq n$ , since  $\mathcal{W}_t \subset \mathcal{W}$  trivially follows from  $\mathcal{C}_t \subset \mathcal{F}$ , it remains to verify that for any  $w \in \mathcal{W}_t$ ,  $\sum_{s=1}^{t-1} w^2(a_s) \leq \beta^2$ . To see that this holds pick  $w \in \mathcal{W}_t$  so that  $w = f - g$ ,  $f, g \in \mathcal{C}_t$ . Then,  $\sum_{s < t} w^2(a_s) = \sum_{s < t} (f(a_s) - \hat{f}_t(a_s) - (g(a_s) - \hat{f}_t(a_s)))^2 \leq 2 \sum_{s < t} (f - \hat{f}_t)^2(a_s) + 2 \sum_{s < t} (g - \hat{f}_t)^2(a_s) \leq 4\beta_t^2 \leq 4\beta_n^2 = \beta^2$ , where we used that  $\beta_t$  is increasing. All that remains is to prove that  $f^* \in \cap_{1 \leq t \leq n} \mathcal{C}_t$  with probability at least  $1 - \delta$ . This follows from standard concentration of measure arguments, which you can find in the original paper [39].  $\square$

## 7.5 Remarks and references

- The eluder dimension was introduced by Russo and Van Roy [39]. Its relation to other complexity measures is a subject of current research [13, 20, 33].
- In the applications above we applied the eluder dimension to what would be the class of action-value functions in the RL setting (these are  $A \mapsto \mathbb{R}$  mappings in the bandit setting). Later we see more subtle applications with different choices of function classes.
- Theorem 7.4 is vacuous when  $\mathcal{F}$  is infinite. To get a result for this case, one can replace  $|\mathcal{F}|$  with a suitable covering number of  $\mathcal{F}$ , while also replacing  $\text{Edim}(\mathcal{F}, \varepsilon)$  with  $\text{Edim}(\mathcal{F}, \frac{\varepsilon}{2})$ .
- The name ‘dimension’ suggests we might have lower bounds in terms of the eluder dimension, as we do for (say) the Vapnik-Chervonenkis dimension [42, Theorem 6.7]. But this does not work out because the eluder dimension captures only how optimistic algorithms behave and in some structured bandit examples, optimistic algorithms are highly suboptimal. As an example, consider the deterministic case when  $A \subset [0, 1]$ ,  $A$  is finite and  $0 \in A$ . Define

$$\mathcal{F} = \left\{ f : \|f\|_\infty \leq 1 \text{ and } f(0) = \arg \max_{a \in A} f(a) \right\}.$$

The set  $\mathcal{F}$  is chosen so that for any  $f \in \mathcal{F}$ , the value of  $f(0)$  gives away the identity of the optimal action. Since there is no other structure, for any  $f^* \in \mathcal{F}$ , an optimistic algorithm will exhaustively try all the actions even though playing first  $a_1 = 0$  and subsequently playing  $a_t = r_1$  gives the optimal reward for all rounds except for the first one. Moreover, any enumeration of the elements of  $A$  that ends with zero is an eluder sequence, hence  $\text{Edim}(\mathcal{F}, 0) = |A|$ .

## 8. GOLF AND THE BELLMAN ELUDER DIMENSION

We now explain an application of the eluder dimension to reinforcement learning. At the same time we give an example of learning in the online interaction protocol using a (potentially) non-linear function approximation class. The algorithm is based on the optimism principle and is conceptually very simple. As we discussed already, at least information-theoretically, the only ingenuity in designing an optimistic algorithm is the construction of the confidence set. Of course, the analysis is also important and for this the fundamental challenge is showing that suboptimality is only possible when the algorithm is gaining information.

### 8.1 Modelling assumptions

We start by explaining the modelling assumptions. Define the map (operator)  $\mathcal{T}$  whose domain and range are

real-valued function over  $S \times A$ , and which, given a function  $f : S \times A \rightarrow \mathbb{R}$ , returns a function  $f' = \mathcal{T}f$  such that

$$f'(s, a) = r(s, a) + \sum_{s' \in S} \mathbf{P}(s'|s, a) \max_{a' \in A} f(s', a').$$

By giving the application of an operator higher priority than to function evaluation, the left-hand side will also be written in the more compact form  $\mathcal{T}f(s, a)$ . We call  $\mathcal{T}$  the *Bellman optimality operator*. Let  $\mathcal{F}$  be a set of functions from  $S \times A$  to  $\mathbb{R}$ . We make the following assumption.

ASSUMPTION 8.1. *The following hold:*

- (a) *Vanish on horizon:*  $f(\perp, a) = 0$  for all  $a \in A$ .
- (b) *Realisability:*  $q^* \in \mathcal{F}$ .
- (c) *Closure:*  $\mathcal{T}f \in \mathcal{F}$  for all  $f \in \mathcal{F}$ .
- (d) *Finiteness:*  $\mathcal{F}$  is finite.
- (e) *Boundedness:*  $\max_{f \in \mathcal{F}} \|f\|_\infty \leq B$  with  $B \geq 1$  known.

The GOLF algorithm of [23] is designed to have small regret when  $\mathcal{F}$  and the MDP are in harmony in the sense that they satisfy Assumption 8.1. The closure assumption is quite restrictive when  $\mathcal{F}$  is finite. But the restriction to finite  $\mathcal{F}$  is for convenience only. The analysis holds for infinite  $\mathcal{F}$  with  $|\mathcal{F}|$  replaced by a suitable covering number and some extra steps. GOLF is a straightforward implementation of the optimism principle.

## 8.2 Confidence set construction

Remember that the optimal action-value function  $q^*$  satisfies the Bellman optimality equation:

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

The confidence set is designed to rule out value functions  $f$  that do not approximately satisfy the Bellman optimality equation. In the online interaction protocol the algorithm interacts with the environment and observes data  $(s_0^t, a_0^t, r_0^t, s_1^t, \dots, s_{H-1}^t, a_{H-1}^t, r_{H-1}^t, s_H^t)_t$ . This notation is not the most convenient for our purposes. Let us introduce a reindexing of the data used by the algorithm. By the end of episode  $t$  the learner has taken  $m(t) = tH$  actions. We let

$$\mathcal{D}_t = (s_k, a_k, r_k, s'_k)_{k=1}^{m(t)}$$

be the sequence of state/action/reward/next-state tuples observed by the algorithm. Let

$$\mathbb{E}_{k-1}[\cdot] = \mathbb{E}[\cdot | s_1, a_1, r_1, s'_1, \dots, r_{k-1}, s'_{k-1}, s_k, a_k],$$

which is the expectation operator conditioned on the first  $k-1$  observed transitions. Given  $f, g \in \mathcal{F}$ , abbreviate  $f_k = f(s_k, a_k)$  and  $g_k = g(s_k, a_k)$  and  $f_k^+ = \max_{a \in A} f(s_k, a)$  and  $Tf_k = \mathcal{T}f(s_k, a_k)$  and

$$\Delta_k(g, f) = r_k + f_k^+ - g_k.$$

The confidence set for episode  $t$  is

$$(10) \quad \mathcal{C}_t = \left\{ f \in \mathcal{F} : \max_{g \in \mathcal{F}} D_{t-1}(g, f) \leq \beta^2 \right\},$$

where  $\beta^2$  is a parameter to be tuned later and

$$L_t(g, f) = \sum_{k=1}^{m(t)} \Delta_k(g, f)^2 \quad \text{and}$$

$$D_t(g, f) = L_t(f, f) - L_t(g, f).$$

We now explain the idea underlying the construction of  $\mathcal{C}_t$ . Remember, we need  $\mathcal{F}$  to contain  $q$ -value functions that satisfy the Bellman optimality equation. Let  $\varepsilon_k(f) = r_k + f_k^+ - Tf_k$  and

$$N_t(g, f) = 2 \sum_{k=1}^{m(t)} \varepsilon_k(f)(g_k - f_k) \quad \text{and}$$

$$V_t(g, f) = \sum_{k=1}^{m(t)} (g_k - f_k)^2.$$

Note that  $\mathbb{E}_{k-1}[\varepsilon_k(f)] = 0$  for any  $f \in \mathcal{F}$ , so  $N_t(g, f)$  is a martingale. The following lemma provides the Doob decomposition of the process  $D_t(g, f)$  when  $g$  and  $f$  are fixed.

LEMMA 8.2. *For any  $f, g \in \mathcal{F}$ ,*

$$D_t(g, f) = N_t(g, f) + \sum_{k=1}^{m(t)} (f_k - g_k)(f_k + g_k - 2Tf_k).$$

PROOF. Substitute the definitions and simplify.  $\square$

Lemma 8.2 starts to reveal what is happening. Suppose that  $f$  satisfies the Bellman optimality equation:  $\mathcal{T}f = f$ . Then by Lemma 8.2,

$$(11) \quad D_{t-1}(g, f) = N_{t-1}(g, f) - V_{t-1}(g, f),$$

which means the process  $D_t(g, f)$  has negative drift. Moreover, the predictable variation of the noise can be bounded in terms of the drift to show that when  $f$  satisfies the Bellman optimality equation, then  $D_{t-1}$  grows slowly with high probability as we show in the proof of Theorem 8.3 below. The second observation that arises from Lemma 8.2 is that if  $g = \mathcal{T}f$  and  $f \in \mathcal{C}_t$ , then

$$\beta^2 \geq D_{t-1}(\mathcal{T}f, f) = N_{t-1}(\mathcal{T}f, f) + V_{t-1}(\mathcal{T}f, f),$$

which means that the confidence set  $\mathcal{C}_t$  contains those functions  $f$  that approximately satisfy the Bellman equation on the observed sequence of state/action pairs.

THEOREM 8.3. *Suppose that  $f \in \mathcal{F}$  and  $f = \mathcal{T}f$ . Let  $\beta^2 = CB^2 \log(2TB|\mathcal{F}|/\delta)$  with  $C$  a suitably large universal constant. Then  $\mathbb{P}(f \in \cap_{t=1}^T \mathcal{C}_t) \geq 1 - \delta$ .*

Note that this result holds regardless of how the actions are chosen.

PROOF SKETCH. Since  $f$  satisfies the Bellman optimality equation, by Lemma 8.2,

$$D_{t-1}(g, f) = N_{t-1}(g, f) - V_{t-1}(g, f).$$

As discussed,  $\mathbb{E}_{k-1}[\varepsilon_k(f)(g_k - q_k^*)] = 0$ . By Assumptions 2.1 and 8.1,  $|\varepsilon_k(f)| \leq 1 + B \leq 2B$  and therefore

$$\begin{aligned} \sum_{k=1}^{m(t)} \mathbb{E}_{k-1}[(2\varepsilon_k(f)(g_k - q_k^*))^2] &\leq 16B^2 \sum_{k=1}^{m(t-1)} (g_k - q_k^*)^2 \\ &= 16B^2 V_{t-1}(g, f), \end{aligned}$$

Hence, an application of a Freedman-like concentration inequality [62, Theorem 9] and a union bound over all  $g \in \mathcal{F}$  shows that with probability at least  $1 - \delta$  for all  $1 \leq t \leq T$  and  $g \in \mathcal{F}$ ,

$$D_{t-1}(g, f) \leq -V_t(g, f) + B\sqrt{CV_t(g, f)L} + CB^2L,$$

where  $C > 0$  is a universal constant and  $L = \log(2|\mathcal{F}|TB/\delta)$  is logarithmic factor. Hence, since  $\max_{v \geq 0}(a\sqrt{v} - v) = a^2/4$  for any  $a \geq 0$  it follows that

$$\max_{g \in \mathcal{F}} \max_{1 \leq t \leq T} D_{t-1}(g, f) \leq \frac{5CLB^2}{4} \triangleq \beta^2.$$

And on this event  $f \in \cap_{t=1}^T \mathcal{C}_t$  by definition.  $\square$

The next theorem is proven in a similar fashion. For any function  $w : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  and occupancy measure  $\nu$  let  $w(\nu) = \langle \nu, w \rangle$ .

**THEOREM 8.4.** *With probability at least  $1 - \delta$ ,*

$$\max_{1 \leq t \leq T} \max_{f \in \mathcal{C}_t} \sum_{s=1}^{t-1} \langle \mathbb{P}_s, (\mathcal{T}f - f)^2 \rangle = O(\beta^2),$$

where  $\mathbb{P}_s = \mathbb{P}_{SA}^{\pi_s}$  is the occupancy measure of the policy  $\pi_s$  used by GOLF in the  $s$ th episode.

PROOF SKETCH. We start by arguing that with high probability

$$(12) \quad \max_{1 \leq t \leq T} \max_{f \in \mathcal{C}_t} \sum_{k=1}^{m(t-1)} (\mathcal{T}f - f)_k^2 = O(\beta^2).$$

By definition, for  $f \in \mathcal{C}_t$ ,

$$\beta^2 \geq D_{t-1}(\mathcal{T}f, f) = N_{t-1}(\mathcal{T}f, f) + V_{t-1}(\mathcal{T}f, f).$$

Repeating the concentration analysis in the proof of Theorem 8.3 shows that with probability at least  $1 - \delta$ , for all

$f \in \mathcal{F}_t$ ,

$$\begin{aligned} \beta^2 &\geq V_{t-1}(\mathcal{T}f, f) + N_{t-1}(\mathcal{T}f, f) \\ &\geq V_{t-1}(\mathcal{T}f, f) - B\sqrt{CV_{t-1}(f, \mathcal{T}f)L} - CB^2L \\ &\geq \frac{V_{t-1}(f, \mathcal{T}f)}{2} - \frac{3B^2L}{2}. \end{aligned}$$

where  $C > 0$  is a universal constant. Moving now to the main claim, let  $T_s$  be the set of  $k$  such that the tuple  $(s_k, a_k, r_k, s'_k)$  was collected in episode  $s$ . Let  $X_s = \sum_{k \in T_s} (\mathcal{T}f - f)_k^2$ . By definition,

$$\langle \mathbb{P}_s, (\mathcal{T}f - f)^2 \rangle = \mathbb{E}_{s-1} \left[ \sum_{k \in T_s} (\mathcal{T}f - f)_k^2 \right] = \mathbb{E}_{s-1}[X_s].$$

Hence  $(X_s)$  is a sequence of non-negative random variables and by (12),  $\sum_{s=1}^t X_s = O(\beta^2)$  with high probability. For such sequences it also holds that  $\sum_{s=1}^t \mathbb{E}_{s-1}[X_s] = O(\beta^2)$  with high probability, which follows by carefully applying [30, Exercise 5.19].  $\square$

### 8.3 Algorithm

The greedy policy with respect to an action-value function  $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  is  $\pi_f$  defined by

$$\pi_f(s) = \arg \max_{a \in \mathcal{A}} f(s, a).$$

The GOLF algorithm implements the optimism in the face of uncertainty principle with the confidence set defined in Equation 10. At the beginning of the  $t$ th episode the algorithm computes

$$f_t = \arg \max_{f \in \mathcal{C}_t} \max_{a \in \mathcal{A}} f(s_1, a).$$

The algorithm then plays the greedy policy with respect to  $f_t$  for an entire episode, which is the policy  $\pi_t = \pi_{f_t}$ . The trajectory is stored and used to compute the new confidence set  $\mathcal{C}_{t+1}$ .

```

1  for  $t = 1$  to  $T$ :
2     $\mathcal{C}_t = \left\{ f \in \mathcal{F} : L_{t-1}(f, f) \leq \min_{g \in \mathcal{F}} L_{t-1}(g, f) + \beta^2 \right\}$ 
3    find candidate  $f_t = \arg \max_{f \in \mathcal{C}_t} \max_{a \in \mathcal{A}} f(s_{\text{init}}, a)$ 
4    for  $h = 0$  to  $H - 1$ :
5      play  $a_h^t = \arg \max_{a \in \mathcal{A}} f_t(s_h^t, a)$ 
6      observe and store  $r_{h+1}^t$  and  $s_{h+1}^t$ 
```

ALGORITHM 4: GOLF

### 8.4 Bellman eluder dimension

The Bellman-eluder dimension (BEDim) is a complexity measure designed for classes of MDPs based on action-value functions as used by GOLF. In Section 7 we explained that the Eluder dimension is not always applied to value functions, an example of which we see here. Let  $\mathcal{G}_g = \{\mathbb{P}_{SA}^{\pi_f} : f \in \mathcal{F}\}$  and let  $\mathcal{G}_b$  be the set of all Dirac measures on  $S \times A$ . For  $f \in \mathcal{F}$  define  $w_f : S \times A \rightarrow \mathbb{R}$  by

$$(13) \quad w_f = f - \mathcal{T}f.$$

By slightly abusing notation, we also view any function  $w : S \times A \rightarrow \mathbb{R}$  as a function mapping the distributions on trajectories of length  $H$  to  $\mathbb{R}$ . In particular, for  $\nu$  such a distribution, we let

$$(14) \quad w(\nu) = \mathbb{E}_\nu \left[ \sum_{h=0}^{H-1} w(s_h, a_h) \right].$$

The Bellman-Eluder dimension of  $\mathcal{F}$  in MDP  $M$  at level  $\varepsilon > 0$  is defined by

$$\text{BEDim}(\mathcal{F}, M, \varepsilon) = \min \left( \text{Edim}(\mathcal{W}(\mathcal{G}_g), \varepsilon), H^2 \text{Edim}(\mathcal{W}(\mathcal{G}_b), \varepsilon) \right),$$

where  $\mathcal{W}(\mathcal{G}) = \{(w_f : \mathcal{G} \rightarrow \mathbb{R}) : f \in \mathcal{F}\}$ .

**REMARK.** The Bellman-eluder dimension depends on the tolerance level  $\varepsilon$ , the function class  $\mathcal{F}$  and also the unknown MDP, which together with  $\mathcal{F}$  defines the classes of measures  $\mathcal{G}_g$  and  $\mathcal{G}_b$ .

There are two main ideas behind this definition that are worth describing as they show a pattern. First, just like in the bandit case, the arguments of the functions in our class  $\mathcal{W}$  represent possible choices. Since we are in an MDP, the choices are policies. However, we only consider policies that are greedy with respect to some function  $f \in \mathcal{F}$  because we know that  $q^* \in \mathcal{F}$  and a greedy policy with respect to  $q^*$  is optimal. Also, for mathematical convenience, policies are identified with the distribution that they induce when interacting with the MDP. Next, given a choice (policy, or its underlying distribution), the functions in our class  $\mathcal{W}$  give feedback about the chance that  $f = q^*$  holds. Furthermore, when  $w_f = 0$  then  $\pi_f$  is optimal, which follows from the fact that for such policies  $q^{\pi_f} = f$  and the performance difference identity (Proposition 8.2) with  $\pi = \pi_*$ . On the other hand, if  $w_f \neq 0$ , then  $\pi_f$  is not optimal at some state. Finally, for  $\nu \in \mathcal{G}$  and  $f \in \mathcal{F}$ , the value of  $w_f(\nu)$  can be estimated by interacting with the MDP using the policy corresponding to  $\nu$ .

In the next subsection we prove a regret bound for Algorithm 4 in terms of the Bellman eluder dimension. Before that, one may ask how the Bellman eluder dimension

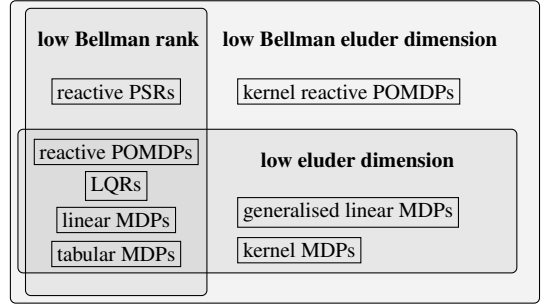


FIG 3: The Bellman eluder dimension captures a wide range of linear and non-linear function approximation classes. The figure is reproduced from [23]. Note that the bilinear dimension introduced by Du et al. [19] is not comparable to the Bellman eluder dimension.

behaves for various subclasses of MDPs. A summary of what is known about this is shown on Figure 3. The interpretation of the figure is that for a subclass of MDPs which are characterized by some complexity parameter, the Bellman eluder dimension is polynomial in the associated complexity parameter. For example, for tabular MDPs (MDPs with no structure at all, but which are finite), the complexity parameter is the number of state-action pairs. As another example consider linear quadratic regulators (LQRs). These are models with  $S = \mathbb{R}^d$  and  $A = \mathbb{R}^m$  for some  $d, m > 0$  integers where the dynamics are linear and the reward is quadratic. Here, the complexity parameters are  $d$  and  $m$ .

**REMARK.** The state and action spaces for LQRs are not finite, but these conditions only exist in our analysis to avoid the need for measure theory and the fact that optimal policies may need to be replaced with near-optimal policies.

Since low Bellman rank itself encompasses a large class of models and is thus of independent interest, we give the definition and the result connecting the Bellman eluder dimension to the Bellman rank.

**DEFINITION 8.5.** The Bellman rank  $\text{Brank}(\mathcal{F}, M, r)$  of  $\mathcal{F}$  in MDP  $M$  and with normalisation  $r > 0$  is the smallest integer  $d$  such that there exist maps  $\phi : \mathcal{G}_g \rightarrow \mathbb{B}_1^d$  and  $\theta : \mathcal{F} \rightarrow \mathbb{B}_r^d$  such that for all  $f \in \mathcal{F}$  and  $\nu \in \mathcal{G}_g$ ,

$$\mathbb{E}_\nu \left[ \sum_{h=1}^H (f - \mathcal{T}f)(s_h, a_h) \right] = \langle \phi(\nu), \theta(f) \rangle.$$

**PROPOSITION 8.1.** Let  $M$  be a finite MDP and  $\mathcal{F}$  be a class of functions  $f : S \times A \rightarrow \mathbb{R}$ . Then, for all  $\varepsilon > 0$ ,

$$\text{BEDim}(\mathcal{F}, M, \varepsilon) = O \left( \text{Brank}(\mathcal{F}, M, r) \left( 1 + \frac{r}{\varepsilon} \right) \right).$$

PROOF. Let  $\nu_1, \dots, \nu_k$  be an eluder sequence at level  $\varepsilon$ . Then there exists a sequence  $f_1, \dots, f_k$  such that for all  $t \leq k$ , (i)  $w_{f_t}(\nu_t) > \varepsilon$ . (ii)  $\sum_{s < t} w_{f_t}(\nu_s)^2 \leq \varepsilon$ . By the definition of the Bellman rank, for  $\nu \in \mathcal{G}_\varepsilon$ ,

$$w_f(\nu) = \langle \phi(\nu), \theta(f) \rangle.$$

Let  $\phi_t = \phi(\nu_t)$  and  $\theta_t = \theta(f_t)$ . Then by (i) and (ii), for all  $t \leq k$ ,

$$\langle \phi_t, \theta_t \rangle > \varepsilon \quad \text{and} \quad \sum_{s=1}^{t-1} \langle \phi_s, \theta_t \rangle^2 \leq \varepsilon^2.$$

The result follows from Proposition 7.4.  $\square$

### 8.5 Regret analysis

THEOREM 8.6. *Let  $M$  be a finite MDP with state-action space  $S \times A$  such that Assumption 8.1 holds. With probability at least  $1 - \delta$  the regret of GOLF is bounded by*

$$\text{Reg}_T = O\left(\sqrt{TH \text{BEdim}(\mathcal{F}, M, 1/T) \beta^2}\right).$$

PROOF SKETCH. By Theorem 8.3, with high probability  $q^* \subset \mathcal{C}_t$  for all episodes  $t$ . We assume for the remainder that this high probability event occurs. Then, by optimism,  $v^*(s_0) - v^{\pi_t}(s_0) \leq f_t^*(s_0) - v^{\pi_t}(s_0)$  and

$$(15) \quad \text{Reg}_T \leq \sum_{t=1}^T (f_t^*(s_0) - v^{\pi_t}(s_0)).$$

We now make use of the performance difference lemma, which decomposes the value difference on the right-hand side above in terms of the Bellman operator:

PROPOSITION 8.2 (Lemma 1, 22). *Let  $f : S \times A \rightarrow \mathbb{R}$  such that  $f(\perp, \cdot) \equiv 0$  and  $\pi = \pi_f$  be the greedy policy. Then,*

$$\begin{aligned} \max_{a \in A} f(s_0, a) - v^\pi(s_0) &= \mathbb{E}_\pi \left[ \sum_{h=0}^{H-1} (f - \mathcal{T}f)(s_h, a_h) \right] \\ &= w_f(\mathbb{P}_{SA}^\pi). \end{aligned}$$

We give the proof of Proposition 8.2 at the end of the subsection. Combining Equation (15) with the result of Proposition 8.2, we get

$$(16) \quad \text{Reg}_T \leq \sum_{t=1}^T w_{f_t}(\mathbb{P}_t) \leq \sum_{t=1}^T \|\mathbb{P}_t\|_{\mathcal{W}_t},$$

where we defined  $\mathbb{P}_t = \mathbb{P}_{SA}^{\pi_t}$  and  $\mathcal{W}_t = \{w_f : f \in \mathcal{C}_t\}$ . The next step is to bound the above using the definition of the Bellman eluder dimension. We will use Corollary 7.1 for this. To apply Corollary 7.1, we need to show that the

condition from Proposition 7.5 holds. That is, for any  $1 \leq t \leq T$ ,  $w \in \mathcal{W}_t$ ,

$$(17) \quad \sum_{s=1}^{t-1} (w(\mathbb{P}_s))^2 \leq C$$

for some  $C > 0$ . Fix  $1 \leq t \leq T$  and  $w \in \mathcal{W}_t$ . By the definition of  $\mathcal{W}_t$ ,  $w = w_f$  for some  $f \in \mathcal{C}_t$ . Therefore,

$$\begin{aligned} \sum_{s=1}^{t-1} (w(\mathbb{P}_s))^2 &= \sum_{s=1}^{t-1} (w_f(\mathbb{P}_s))^2 = \sum_{s=1}^{t-1} \langle \mathbb{P}_s, w_f \rangle^2 \\ (18) \quad &\leq H \sum_{s=1}^{t-1} \langle \mathbb{P}_s, w_f^2 \rangle, \end{aligned}$$

where in the final inequality we used Jensen's inequality along with the fact that  $\mathbb{P}_s$  is an occupancy measure so that  $\|\mathbb{P}_s\|_1 = H$ . Recalling that  $w_f = f - \mathcal{T}f$ , from Theorem 8.4 we get that Equation (17) holds with  $C = O(H\beta^2)$ . Then, from Corollary 7.1, combined with Equation (16) we conclude that

$$\text{Reg}_T = O\left(\sqrt{TH\beta^2 \text{Edim}\left(\mathcal{W}_\varepsilon, \frac{1}{T}\right)}\right).$$

We also need to show that

$$\text{Reg}_T = O\left(\sqrt{TH\beta^2 \text{Edim}\left(\mathcal{W}_\nu, \frac{1}{T}\right)}\right).$$

We omit the details, which can be found in the original article [23]. The main idea is to show that the state/occupancy measures that appear in (16) can be replaced by the empirical (Dirac) measures concentrating on the state/action pairs actually visited by the algorithm. This is done by a concentration of measure argument. The second step is to show an equivalent of (18), which actually follows almost immediately from (12).  $\square$

PROOF OF PROPOSITION 8.2. Clearly, the result holds if we can show that for any  $0 \leq h \leq H-1$ ,

$$\begin{aligned} \max_{a \in A} f(s_h, a) - v^{\pi_f}(s_h) &= \mathbb{E}_{\pi_f}[(f - \mathcal{T}f)(s_h, a_h) | s_h] \\ &\quad + \mathbb{E}_{\pi_f} \left[ \max_{a \in A} f(s_{h+1}, a) - v^{\pi_f}(s_{h+1}) \middle| s_h \right], \end{aligned}$$

because  $f(\perp, a) = 0 = v^{\pi_f}(\perp)$  for all  $a \in A$ . Fix  $0 \leq h \leq H-1$ . Since the value function of  $\pi_f$  satisfies the Bellman equation,  $v^{\pi_f}(s_h) = \mathbb{E}_{\pi_f}[r(s_h, a_h) + v^{\pi_f}(s_{h+1}) | s_h]$ . Plugging this in, using also that  $\max_{a \in A} f(s_h, a) = \mathbb{E}_{\pi_f}[f(s_h, a_h) | s_h]$ , adding and subtracting  $\max_{a \in A} f(s_{h+1}, a)$  and reordering gives

$$\begin{aligned} \max_{a \in A} f(s_h, a) - v^{\pi_f}(s_h) &= \mathbb{E}_{\pi_f} [f(s_h, a_h) - (r(s_h, a_h) + \max_{a \in A} f(s_{h+1}, a)) | s_h] \\ &\quad + \mathbb{E}_{\pi_f} [\max_{a \in A} f(s_{h+1}, a) - v^{\pi_f}(s_{h+1}) | s_h]. \end{aligned}$$

Now, notice from the definition of  $\mathcal{T}$  that  $\mathbb{E}_{\pi_f}[r(s_h, a_h) + \max_{a \in \mathcal{A}} f(s_{h+1}, a) | s_h] = \mathbb{E}_{\pi_f}[\mathcal{T}f(s_h, a_h) | s_h]$ . Plugging this in gives the desired result.  $\square$

## 8.6 Remarks and references

- Using  $\arg \min_f \max_{g \in \mathcal{F}} D_t(g, f)$  to find an approximation to  $q^*$  is similar to the approach of Antos, Szepesvári and Munos [7] who used a similar minimax approach for policy evaluation in the context of batch RL. The closure assumption, in a “relaxed form” so that misspecification is allowed, was introduced by Munos and Szepesvári [37] who studied a version of approximate value iteration, also known as “fitted  $Q$ -iteration”. This early paper did not use optimism, or any other form of deliberate strategy to discover critical information about the MDP to allow for the effective control of extrapolation errors.
- The first work to use optimism together with approximate value iteration is due to Wen and Van Roy [55] who used it in the context of MDPs with deterministic transitions and rewards, linear function approximation under the assumption that  $q^*$  lies in the span of the basis functions, in the online access model. When the system is stochastic, the assumption that the MDP is linear (see Section 4) implies that *any* function from the state-action space will be mapped to  $\mathcal{F}$  by the Bellman-optimality operator. Thus, one can add a bonus term to the value functions obtained in approximate value iteration, keeping all the errors under control [24].
- The GOLF algorithm was introduced by Jin, Liu and Miryoosefi [23]. There are some minor differences between our presentation and theirs. Most notably, they assume layer-wise function approximation classes  $(\mathcal{F})_{h=1}^H$  so that whenever  $s \in \mathcal{S}_h$  and  $a \in \mathcal{A}$ ,  $q^*(s, a) = f(s, a)$  for some  $f \in \mathcal{F}_h$ . The other key difference is a slightly different definition of the Bellman Eluder dimension. Our definition is that the Bellman eluder dimension is the eluder dimension of a carefully chosen function class, while they modify the definition in a minor way, effectively pushing the inequality in (18) into the definition. Our definition is never larger than the original and probably never much smaller either. Specialising GOLF to the case when  $\mathcal{F}$  is a set of linear functions we arrive at ELEANOR due to Zanette et al. [59].
- We mentioned other applications of the eluder dimension to reinforcement learning. This can be done in several ways. For example, Osband and Van Roy [38] bound the Bayesian regret of a posterior sampling algorithm in terms of the eluder dimension of a class of reward functions and transition kernels, which is a complexity of a class of models. Alternatively, Wang, Salakhutdinov and Yang [48] bound the regret of an algorithm in terms of the eluder dimension of a class

of action-value functions using an optimistic algorithm. The Bellman eluder dimension explained above is a strict improvement on this [23].

- Theorem 8.6 bounds the regret of GOLF in terms of the number episodes, the horizon and the Bellman eluder dimension, but to achieve this we also needed to assume completeness. That is, that  $\mathcal{T}f \in \mathcal{F}$  for all  $f \in \mathcal{F}$ . This assumption can be avoided if one is satisfied with a guarantee on the sample complexity instead. Jin, Liu and Miryoosefi [23] prove that OLIVE can find an  $\varepsilon$ -optimal policy in  $\text{poly}(\text{BEDim}(\mathcal{F}, M, \varepsilon), H, 1/\varepsilon)$  while only assuming that  $q^* \in \mathcal{F}$ . Note that unlike the eluder dimension,  $\text{BEDim}(\mathcal{F}, M, \varepsilon)$  may be exponentially large in  $d$  even when  $\mathcal{F}$  is the set of functions that are obtained as a linear combination of  $d$  basis functions. Even assuming that  $q^* \in \mathcal{F}$  is insufficient in this case, as follows from a result of Weisz, Amortila and Szepesvári [50]. Yet, if completeness also holds, the Bellman eluder dimension is at most  $\tilde{O}(d)$  [23].
- The definition of the Bellman eluder dimension as a minimum of two eluder dimensions looks a bit mysterious. Interestingly, it is not true that one of the terms in the minimum always dominates the other [23].

## 9. TENSORPLAN

In Section 6 we analysed approximate policy iteration under the assumption that the action-values for all policies are (approximately) linear in the given features and where the learner has access to a generative model. The assumption that the action-values of *all* policies can be represented linearly using the features is very strong. As also noted earlier, assuming only that  $q^*$  lies in the span of the features, all planning algorithms that can return  $O(1)$ -optimal policies, in some MDPs, need to query exponentially many times the underlying MDP [50, 52]. Remarkably, this ceases to be the case if the features depend only on the state and  $v^*$  is (nearly) linearly realisable. Our goal in this section is to explain why this is the case. The main assumption that we will work is as follows:

**ASSUMPTION 9.1.** *The learner has access to a state-based feature map  $\phi : \mathcal{S} \rightarrow \mathbb{R}^d$  and there exists a  $\theta_* \in \mathbb{R}^d$  such that:*

- (i) *Bounded features:*  $\|\phi(s)\| \leq B$  for all  $s \in \mathcal{S}$ .
- (ii) *Realisability:*  $v^*(s) = \langle \theta_*, \phi(s) \rangle$  for all  $s \in \mathcal{S}$ .
- (iii) *Bounded parameters:*  $(\theta_*)_d = 1$  and  $\|\theta_*\| \leq 2$ .

The assumption that  $(\theta_*)_d = 1$  is for convenience only and can be enforced by introducing an extra feature dimension with  $\phi_{d+1}(s) = 0$ . Thus, the modest price to guarantee that this assumption holds is the increase of the dimension by one. Like GOLF, TENSORPLAN is an optimistic algorithm, but operates in the local access interface

rather than the online setting. The aim is to find a near-optimal policy for the initial state with as few queries to the simulator as possible. The remainder of this section is devoted to proving the following theorem:

**THEOREM 9.2.** *Under Assumptions 9.1, there exists a learner using the local access interface that makes*

$$\tilde{O} \left( \text{poly} \left( |A| \left( \frac{dBH}{\varepsilon} \right)^{|A|} \right) \right),$$

*queries to the model and with probability at least  $1 - \delta$  outputs an action  $\hat{a}$  of a stochastic policy  $\hat{\pi}$  evaluated at  $s_{\text{init}}$  such that*

$$\sum_{a \in A} \hat{\pi}(a|s) \text{saReg}(s_{\text{init}}, a) \leq \varepsilon.$$

While the scaling with the number of actions is concerning, when there are only a few actions, the result promises that a clever learner exist that will relatively quickly discover a good policy.

### 9.1 High-level idea

Given  $\theta \in \mathbb{R}^d$  let  $v_\theta(s) = \langle \phi(s), \theta \rangle$  and define the Bellman residual associated with  $\theta$  for state/action pair  $s, a$  by

$$\Delta_\theta(s, a) = r(s, a) + \sum_{s' \in S} \mathbf{P}(s'|s, a) v_\theta(s') - v_\theta(s).$$

Clearly, the Bellman residual with the optimal parameter  $\theta^*$  satisfies that

$$(19) \quad \min_{a \in A} |\Delta_{\theta^*}(s, a)| = 0, \quad \text{for all } s \in S.$$

Indeed, for state  $a$ , choosing any optimal action  $a \in A^*(s)$  at the state will be so that  $\Delta_{\theta^*}(s, a) = 0$ .

The Bellman residual also satisfies the following decomposition comparing  $v_\theta$  to  $v^\pi$ :

**PROPOSITION 9.1.** *For all  $s \in S$  and policies  $\pi$*

$$v^\pi(s_{\text{init}}) - v_\theta(s_{\text{init}}) = \langle \mathbb{P}_{SA}^\pi, \Delta_\theta \rangle,$$

*where  $\mathbb{P}_{SA}^\pi$  is the state/action occupancy measure (Subsection 2.3).*

The proof is based on a telescoping argument and is deferred until the end of Subsection 9.3. Note that  $\Delta_\theta$  depends on the unknown reward and dynamics of the MDP, but can be estimated using the local access model for states  $s$  that have been visited.

TENSORPLAN operates in iterations  $k = 1, 2, \dots, k_{\text{max}}$  and maintains confidence sets  $\mathcal{C}_1, \dots, \mathcal{C}_{k_{\text{max}}} \subset \mathbb{B}_1$  such that with high probability  $\theta_\star \in \mathcal{C}_k$  for all  $k$ . At the start

of each iteration the learner finds an optimistic parameter in the confidence set, which is

$$\theta_k = \arg \max_{\theta \in \mathcal{C}_k} v_\theta(s_{\text{init}}),$$

Once the optimistic parameter has been found the algorithm uses the local access model to sample many rollouts from a policy  $\pi_k$  for which

$$\pi_k(s) = \arg \min_{a \in A} |\hat{\Delta}_{\theta_k}(s, a)|,$$

where  $\hat{\Delta}_{\theta_k}(s, a)$  is an estimate of  $\Delta_{\theta_k}(s, a)$ , which can be obtained using the local access model for any visited state  $s$  and any action  $a$ . You should think of  $\pi_k$  as the policy that plays the action that (approximately) minimises the Bellman residual. That is, it is (approximately) the most consistent policy. The next step is to argue that either  $\pi_k$  is near optimal or the learner gains information. By Proposition 9.1 and some calculation,

$$(20) \quad \begin{aligned} v^{\pi_k}(s_{\text{init}}) &= v_{\theta_k}(s_{\text{init}}) + \langle \mathbb{P}_{SA}^{\pi_k}, \Delta_{\theta_k} \rangle \\ &\geq v^\star(s_{\text{init}}) + \langle \mathbb{P}_{SA}^{\pi_k}, \Delta_{\theta_k} \rangle, \end{aligned}$$

where the inequality follows when  $\theta_\star \in \mathcal{C}_k$  because

$$v_{\theta_k}(s_{\text{init}}) = \max_{\theta \in \mathcal{C}_k} \langle \phi(s_{\text{init}}), \theta \rangle \geq \langle \phi(s_{\text{init}}), \theta_\star \rangle = v^\star(s_{\text{init}}).$$

When  $|\langle \mathbb{P}_{SA}^{\pi_k}, \Delta_{\theta_k} \rangle| \leq \varepsilon$ , Equation (20) implies that  $\pi_k$  is  $\varepsilon$ -optimal. On the other hand, if  $|\langle \mathbb{P}_{SA}^{\pi_k}, \Delta_{\theta_k} \rangle| > \varepsilon$ , then with reasonable probability, in the rollouts that it was doing, the algorithm will encounter a state  $s$  such that

$$(21) \quad \min_{a \in A} |\Delta_{\theta_k}(s, a)| > \varepsilon/H,$$

for if for  $s_h \in S_h$ ,  $\max_{0 \leq h \leq H-1} \min_{a \in A} |\Delta_{\theta_k}(s_h, a)| \leq \varepsilon/H$  then

$$\begin{aligned} \sum_{h=0}^{H-1} \Delta_{\theta_k}(s_h, \pi_k(s_h)) &\leq \sum_{h=0}^{H-1} |\Delta_{\theta_k}(s_h, \pi_k(s_h))| \\ &\approx \sum_{h=0}^{H-1} |\hat{\Delta}_{\theta_k}(s_h, \pi_k(s_h))| \\ &\leq \sum_{h=0}^{H-1} |\hat{\Delta}_{\theta_k}(s_h, \pi'_k(s_h))| \\ &\approx \sum_{h=0}^{H-1} |\Delta_{\theta_k}(s_h, \pi'_k(s_h))| \\ &= \sum_{h=0}^{H-1} \min_{a \in A} |\Delta_{\theta_k}(s_h, a)| \leq \varepsilon. \end{aligned}$$

where we set  $\pi'_k(s) = \arg \min_{a \in A} |\Delta_{\theta_k}(s, a)|$ . For the actual argument, one needs to replace  $\varepsilon/H$  with smaller quantity to make room for the errors induced by using the approximations  $\hat{\Delta}(s_h, a)$  to the Bellman errors.

Now, when (21) holds, the algorithm will conclude that its policy is not near-optimal and refine its confidence interval using the collected data. The main question is how to update the confidence set and prove that the number of iterations is well-controlled.

## 9.2 Learning

To get to the heart of the matter, we conduct a thought experiment by analysing a ‘cheating’ version of TENSORPLAN that can exactly compute quantities that later need to be estimated. The difference to what was described above is that now we take  $\pi_k = \pi'_k$ . That is,

$$\pi_k(s) = \arg \min_{a \in A} |\Delta_{\theta_k}(s, a)|$$

with ties broken arbitrarily. Iteration  $k$  ends once the learner encounters a state  $s_k$  such that  $\min_{a \in A} |\Delta_{\theta_k}(s_k, a)|$  is non-zero, or when a certain number of rollouts have been generated, whichever happens earlier. If the iteration ends not because of a timeout, the confidence set is refined as follows:

$$\mathcal{C}_{k+1} = \{\theta \in \mathcal{C}_0 : \min_{a \in A} |\Delta_{\theta}(s_i, a)| = 0 \text{ for all } 1 \leq i \leq k\},$$

where  $\mathcal{C}_0 = \{\theta \in \mathbb{B}_2 : \theta_d = 1\}$ . Thanks to Assumption 9.1,  $\theta_\star \in \mathcal{C}_0$ . Notice that

$$(\min_{a \in A} |\Delta_{\theta}(s, a)| = 0) \Leftrightarrow \left( \prod_{a \in A} \Delta_{\theta}(s, a) = 0 \right).$$

The idea is to linearise  $\theta \mapsto \prod_{a \in A} \Delta_{\theta}(s, a)$ . By definition,

$$\Delta_{\theta}(s, a) = r(s, a) + \sum_{s' \in S} \mathbf{P}(s'|s, a) \langle \phi(s'), \theta \rangle - \langle \phi(s), \theta \rangle,$$

which we will now make linear in  $\theta$  by introducing an appropriate feature map over the state-action space. Concretely, for all  $s \in S$  and  $a \in A$  and  $\theta \in \mathbb{R}^d$  such that  $\theta \in \mathcal{C}_0$ ,

$$\Delta_{\theta}(s, a) = \langle \phi(s, a), \theta \rangle,$$

where,

$$\phi(s, a) = r(s, a)e_d + \sum_{s' \in S} \mathbf{P}(s'|s, a)\phi(s') - \phi(s).$$

The function  $\theta \mapsto \prod_{a \in A} \Delta_{\theta}(s, a)$  is still not linear, but can be linearised by letting  $\theta^\otimes = \otimes_{a \in A} \theta$  be the  $|A|$ -times tensor product of  $\theta$  with itself and  $\phi^\otimes(s) = \otimes_{a \in A} \phi(s, a)$  and noting that

$$\prod_{a \in A} \Delta_{\theta}(s, a) = \langle \phi^\otimes(s), \theta^\otimes \rangle,$$

where the inner product between tensors is defined coordinate-wise so that tensors  $\phi^\otimes(s)$  and  $\theta^\otimes$  are identified with vectors of dimension  $m = d^{|A|}$ . The point is that the function  $\mathbb{R}^m \ni \xi \mapsto \langle \phi^\otimes(s), \xi \rangle$  is linear. At last we

are making some progress. Let  $\theta_k^\otimes = \otimes_{a \in A} \theta_k$ . Iteration  $k$  ends when the learner finds the state  $s_k$  such that with  $\phi_k^\otimes = \phi^\otimes(s_k)$ ,  $\langle \phi_k^\otimes, \theta_k^\otimes \rangle \neq 0$ . Furthermore, by the definition of  $\theta_k$  and the confidence set  $\mathcal{C}_k$ , for all  $i < k$ ,

$$\langle \phi_i^\otimes, \theta_k^\otimes \rangle = 0.$$

From this, one can conclude that  $\phi_k^\otimes$  is linearly independent of  $\phi_1^\otimes, \dots, \phi_{k-1}^\otimes$ . Since  $\phi_k^\otimes \in \mathbb{R}^m$  this can happen at most  $m$  times and hence the number of iterations is at most  $m = d^{|A|}$ . The analysis of TENSORPLAN follows this basic argument, but we need to make several modifications:

- The function  $\Delta_{\theta_k}(s, a)$  is not directly observed and must be estimated using the local access model.
- The state/action feature  $\phi(s, a)$  is also not observed but can also be estimated for visited states using the local access model.
- We need to quantify more precisely the fact that if  $v_{\theta_k}(s_{\text{init}}) - v^{\pi_k}(s_{\text{init}})$  is large, then the learner will likely encounter states  $s$  such that  $|\Delta_{\theta_k}(s, a)|$  is large for all  $a \in A$ .
- The linear independence argument above relied heavily on the definition of the confidence set. In the real algorithm, however, the confidence set needs to be defined with a degree of slack  $|\Delta_{\theta}(s, a)| \leq \varepsilon_E$  with  $\varepsilon_E \neq 0$ , which arises from the fact that the left-hand side can only be estimated. This means we need a new argument to bound the number of iterations. Here, the eluder dimension will come in handy again.

## 9.3 TensorPlan

We are now in a position to give the gory details of TENSORPLAN algorithm and its analysis. The algorithm is divided into two parts (Algorithm 5 and Algorithm 6), with the former computing the policy to be implemented by the learner and the latter driving the learning process. The algorithm uses parameters  $\varepsilon_E$  defined in (24) as well as

$$(22) \quad \begin{aligned} n_r &= \tilde{\Theta} \left( \frac{BH}{\varepsilon} \right) \\ n_E &= \tilde{\Theta} \left( \frac{B^2 H^2}{\varepsilon^2} \right) \\ n_F &= \tilde{\Theta} \left( \frac{|A|^2 d^{|A|+1} B^{2|A|+2}}{\varepsilon_F^2} \right). \end{aligned}$$

where  $\varepsilon_F$  is defined in (24). The quantity  $n_r$  is the number of rollouts used in each iteration to detect inconsistencies while  $n_E$  and  $n_F$  are the number of samples used to estimate the Bellman residuals and features, respectively.

```

1 def rollout( $\theta$ )
2    $s = s_{\text{init}}$ 
3   for  $h=1$  to  $H$ :
4      $a = \text{policy}(s, \theta)$ 
5     play action  $a$  and observe next-state  $s'$ 
6      $s = s'$ 
7   def policy( $s, \theta$ )
8     for  $a$  in  $A$ :
9        $\hat{\Delta}_\theta(s, a) = \text{estimate}(s, a, \theta)$ 
10    if  $\min_{a \in A} |\hat{\Delta}_\theta(s, a)| \leq \varepsilon_E$ : return  $\arg \min_{a \in A} |\hat{\Delta}_\theta(s, a)|$ 
11    else: throw ValidityFailure( $s$ )
12  def estimate( $s, a, \theta$ ):
13    query model  $n_E$  times at  $s, a$ 
14    observe rewards  $r_1, \dots, r_{n_E}$ 
15    observe next states  $s_1, \dots, s_{n_E}$ 
16    return  $\frac{1}{n_E} \sum_{i=1}^{n_E} (r_i + \langle \phi(s_i), \theta \rangle - \langle \phi(s), \theta \rangle)$ 

```

ALGORITHM 5: TENSORPLAN POLICY

```

1 args:  $\varepsilon, \delta \in (0, 1)$ 
2 set  $n_E, n_F, n_R$  using (22) and  $\varepsilon_E, \varepsilon_F, \varepsilon_D$  using (24)
3 for  $k=1$  to  $k_{\text{max}}$ 
4   let  $C_k = \left\{ \theta \in \mathbb{B}_1 : \max_{i < k} |\langle \hat{\phi}_i^\otimes, \theta^\otimes \rangle| \leq \varepsilon_F \right\}$ 
5   let  $\theta_k = \arg \max_{\theta \in C_k} v_\theta(s_{\text{init}})$ 
6   try:
7     for  $n=1$  to  $n_R$ :
8       rollout( $\theta_k$ )
9     return  $\hat{a} = \text{policy}(s_{\text{init}}, \theta_k)$ 
10  catch ValidityFailure( $s$ ):
11     $\hat{\phi}_k^\otimes = \text{estimateFeature}(s)$ 
12  def estimateFeature( $s$ ):
13    for  $a \in A$ :
14      query model  $n_F$  times with  $s, a$ 
15      observe rewards  $r_1, \dots, r_{n_F}$ 
16      observe next-states  $s_1, \dots, s_{n_F}$ 
17      compute  $\hat{\phi}(s, a) = \frac{1}{n_F} \left[ \sum_{i=1}^{n_F} r_i e_d + \phi(s_i) - \phi(s) \right]$ 
18    return  $\otimes_{a \in A} \hat{\phi}(s, a)$ 

```

ALGORITHM 6: TENSORPLAN

**PROOF SKETCH OF THEOREM 9.2.** We start with a little notation and setup. Recall that  $m = d^{|A|}$  is the dimension of the tensorised features. Let  $\mathcal{F}$  be the linear class in  $\mathbb{R}^m$  defined by

$$(23) \quad \mathcal{F} = \{f : f(x) = \langle \vartheta, x \rangle, \vartheta \in \mathbb{R}^m, \|\vartheta\| \leq 2^d\}.$$

The tuning parameters are chosen to satisfy the following constraints:

$$(24) \quad \varepsilon_E = \frac{3\varepsilon}{8H} \quad \varepsilon_D = \frac{1}{2}\varepsilon_E^{|A|} \quad \varepsilon_F = \frac{\varepsilon_D}{\sqrt{1 + \text{Edim}(\mathcal{F}, \varepsilon_D)}},$$

where  $\text{Edim}$  is the eluder dimension from Section 7.

*Step 1: Setup* Let  $\theta_k$  be the optimistic parameter selected at the start of episode  $k$  and  $\pi_k$  be the corresponding policy. We start with two lemmas, which we prove at the end of the section.

LEMMA 9.3. *Suppose that*

$$(25) \quad |\langle \mathbb{P}_{SA}^{\pi_k}, \Delta_{\theta_k} \rangle| \geq \varepsilon.$$

*With high probability at the end of the iteration the algorithm will have returned a state  $s$  such that*

$$(26) \quad \min_{a \in A} |\Delta_{\theta_k}(s, a)| \geq \varepsilon_E.$$

LEMMA 9.4. *With high probability the algorithm returns a tensorised feature estimate  $\hat{\phi}_k^\otimes \in \mathbb{R}^m$  such that*

$$|\langle \hat{\phi}_k^\otimes - \phi_k^\otimes, \theta_k^\otimes \rangle| \leq \varepsilon_F \text{ and } |\langle \hat{\phi}_k^\otimes - \phi_k^\otimes, \theta_\star^\otimes \rangle| \leq \varepsilon_F.$$

Recall that  $\theta_k^\otimes = \otimes_{a \in A} \theta_k$ . Combining (26) with the above shows that

$$(27) \quad |\langle \hat{\phi}_k^\otimes, \theta_k^\otimes \rangle| \geq |\langle \phi_k^\otimes, \theta_k^\otimes \rangle| - \varepsilon_F \geq \varepsilon_E^{|A|} - \varepsilon_F > \varepsilon_D.$$

By the assumption that  $v^\star$  is linearly realisable, for any  $s \in S$ ,

$$\langle \phi^\otimes(s), \theta_\star^\otimes \rangle = \prod_{a \in A} \langle \phi(s, a), \theta_\star \rangle = 0,$$

which means that

$$\begin{aligned} |\langle \hat{\phi}_k^\otimes, \theta_\star^\otimes \rangle| &\leq |\langle \hat{\phi}_k^\otimes - \phi_k^\otimes, \theta_\star^\otimes \rangle| + |\langle \phi_k^\otimes, \theta_\star^\otimes \rangle| \\ &= |\langle \hat{\phi}_k^\otimes - \phi_k^\otimes, \theta_\star^\otimes \rangle| \\ &\leq \varepsilon_F. \end{aligned}$$

Therefore  $\theta_\star \in C_k$  for all iterations  $k$ .

*Step 2: Correctness* Suppose now that the algorithm returns action  $\hat{a}$  in iteration  $k$ , which means no exception was thrown. Recall that  $\pi_k$  is the (stochastic) policy underlying parameter  $\theta_k$ . Since an exception is thrown with high probability when (25) holds, this means that with high probability

$$(28) \quad |\langle \mathbb{P}_{SA}^{\pi_k}, \Delta_{\theta_k} \rangle| < \varepsilon.$$

Let us introduce the notation that if  $q : S \times A \rightarrow \mathbb{R}$  and  $\pi$  is a stochastic memoryless policy, then  $q(s, \pi) = \sum_{a \in A} \pi(a|s)q(s, a)$ . With this, we have

$$\begin{aligned} \text{saReg}(s_{\text{init}}, \pi_k) &= v^\star(s_{\text{init}}) - q^\star(s_{\text{init}}, \pi_k) \\ &\leq v^\star(s_{\text{init}}) - q^{\pi_k}(s_{\text{init}}, \pi_k) \\ &= v^\star(s_{\text{init}}) - v^{\pi_k}(s_{\text{init}}) \\ &\stackrel{(a)}{\leq} |\langle \mathbb{P}_{SA}^{\pi_k}, \Delta_{\theta_k} \rangle| \\ &\stackrel{(b)}{<} \varepsilon, \end{aligned}$$

where (a) follows from (20) and (b) from (28).

*Step 3: Eluder dimension argument* In the previous step we argued that if the algorithm returns an action in the first  $k_{\max}$  iterations, then that action is near-optimal. We now prove by contradiction that the algorithm does return an action within the first  $k_{\max}$  iterations. Suppose the algorithm has not returned an action in any of the  $k_{\max}$  iterations. By the arguments in step 1 and the definition of  $\mathcal{C}_k$ , with high probability the following hold simultaneously for all  $k \leq k_{\max}$ :

- (i)  $\theta_* \in \mathcal{C}_k$
- (ii)  $|\langle \hat{\phi}_k^\otimes, \theta_k^\otimes \rangle| > \varepsilon_d$
- (iii)  $|\langle \hat{\phi}_i^\otimes, \theta_k^\otimes \rangle| \leq \varepsilon_f$  for all  $i < k$ .

We proceed by contradiction. By definition

$$k_{\max} = 1 + \text{Edim}(\mathcal{F}, \varepsilon_d),$$

where recall that  $\mathcal{F}$  is the set of linear functions over  $\mathbb{R}^m$  defined by Equation (23). We will now show that  $(\hat{\phi}_k^\otimes)_{k=1}^{k_{\max}}$  is an eluder sequence at level  $\varepsilon_d$  for  $\mathcal{F}$ , which is a contradiction. To see why, let  $k \leq k_{\max}$ . Then,

$$\sum_{i=1}^{k-1} \langle \hat{\phi}_i^\otimes, \theta_k^\otimes \rangle^2 \leq (k-1)\varepsilon_f^2 \leq \text{Edim}(\mathcal{F}, \varepsilon_d)\varepsilon_f^2 \leq \varepsilon_d.$$

On the other hand,  $\langle \hat{\phi}_k^\otimes, \theta_k^\otimes \rangle > \varepsilon_d$ . Therefore

$$\text{Edim}(\mathcal{F}, \varepsilon_d) \geq k_{\max} = \text{Edim}(\mathcal{F}, \varepsilon_d) + 1,$$

which is a contradiction.

*Step 4: Bounding the sample complexity* In every iteration the number of rollouts is  $n_r$  and during each rollout the learner visits  $H$  states and uses  $n_e|A|$  queries to the model to estimate the consistency. Hence the number of queries made by calls to **rollout** is

$$k_{\max} n_r n_e H |A|.$$

The expensive part is the estimation of the tensorised features, which potentially happens at the end of every iteration and results in  $k_{\max} n_f$  queries to the local access model. Hence, the total number of queries to the local access model is

$$k_{\max} (n_f + n_r n_e H |A|).$$

The result follows by substituting the definitions.  $\square$

**PROOF OF PROPOSITION 9.1.** Let  $s \in S$  and  $\mathbf{T}(s'|s, \pi) = \sum_{a \in A} \mathbf{T}(s'|s, a) \pi(a|s)$  where  $\pi(a|s)$  is the probability that policy  $\pi$  assigns to action  $a$  in state  $s$ . By the Bellman equation for  $v^\pi$ ,

$$\begin{aligned} v^\pi(s) - v_\theta(s) &= r_\pi(s) + \langle \mathbf{T}(\cdot|s, \pi), v^\pi \rangle - v_\theta(s) \\ &= \Delta_\theta(s, \pi(s)) + \langle \mathbf{T}(\cdot|s, \pi), v^\pi - v_\theta \rangle. \end{aligned}$$

Therefore  $v^\pi - v_\theta$  satisfies the Bellman equation with surrogate reward  $s \mapsto \Delta_\theta(s, \pi(s))$  and claim follows.  $\square$

**PROOF OF LEMMA 9.3.** We need to show three things hold with high probability:

- (i) With high probability the algorithm encounters are state/action pair such that  $|\Delta_{\theta_k}(s, a)|$  is large.
- (ii) Whenever this occurs, the algorithm throws an exception.
- (iii) Whenever the algorithm throws an exception,  $|\Delta_{\theta_k}(s, a)|$  is large.

We show (i) in the first step below and (ii), (iii) in the next step.

*Step 1: High probability of encountering inconsistency* Let  $\theta \in \mathcal{C}_0$  and  $\pi$  be a stochastic policy such that  $|\langle \mathbb{P}_{SA}^\pi, \Delta_\theta \rangle| \geq \varepsilon$ . By Assumption 9.1(iii) and the global assumption that the rewards are bounded in  $[0, 1]$ , it follows that  $|\Delta_\theta(s, a)| \leq 4B + 1$ . We claim that there exists  $0 \leq h \leq H - 1$  such that

$$\mathbb{P}_\pi \left( |\Delta_\theta(s_h, a_h)| \geq \frac{\varepsilon}{2H} \right) \geq \frac{\varepsilon}{(4B + 1)H} \triangleq P,$$

where the random variables are  $(s_0, a_0, \dots, s_{H-1}, a_{H-1})$  and have law  $\mathbb{P}^\pi$ . Indeed, this holds because our condition implies that for some  $0 \leq h \leq H - 1$ ,  $\mathbb{E}_\pi[|\Delta_\theta(s_h, a_h)|] \geq \varepsilon/H$ . Then, the desired inequality follows from letting  $p = \mathbb{P}_\pi(|\Delta_\theta(s_h, a_h)| \geq \varepsilon/(2H))$  and then using that  $\varepsilon/H \leq \mathbb{E}_\pi[|\Delta_\theta(s_h, a_h)|] \leq (4B + 1)p + (1 - p)\varepsilon/(2H) \leq (4B + 1)p + \varepsilon/(2H)$  and solving for  $p$ . By this, if (25) holds, we get that with  $n_r$  rollouts, the probability of encountering a pair  $(s, a) \in S \times A$  with  $|\Delta_{\theta_k}(s, a)| \geq \varepsilon/(2H)$  is at least

$$1 - (1 - P)^{n_r} = 1 - O(\delta),$$

where we used the definition of  $n_r = \tilde{\Theta}(\varepsilon/H(B + 1))$ .

*Step 2: High probability of detecting inconsistency* We need to show that if the encounters a state/action pair  $(s, a) \in S \times A$  with  $|\Delta_{\theta_k}(s, a)| \geq \varepsilon/(2H)$ , the algorithm will detect it. And that it does not spuriously return an inconsistency where there is none. For this, we show that the learner's estimates of  $\Delta_{\theta_k}(s, a)$  are accurate for all  $s, a$  pairs encountered with high probability, which corresponds to showing that the **estimate** function returns an accurate estimate with high probability. Let  $s, a$  be an arbitrary state/action pair and let  $(s_1, r_1), \dots, (s_k, r_k)$  be a sequence of states-reward pairs sampled independently from  $\mathbf{T}(s, a)$ . Let

$$\hat{\Delta}_\theta(s, a) = \frac{1}{n_e} \sum_{i=1}^{n_e} (r_i + \langle \phi'(s_i), \theta \rangle - \langle \phi(s), \theta \rangle).$$

By definition,

$$\mathbb{E}[r_i + \langle \phi(s_i), \theta \rangle - \langle \phi(s), \theta \rangle] = \Delta_\theta(s, a).$$

Under Assumption 9.1,  $|\langle \phi(s_i), \theta \rangle| \leq \|\phi(s_i)\| \|\theta\| \leq 2B$  and  $|r_i| \leq 1$ . It follows from Hoeffding's bound [11] that with probability at least  $1 - \delta$

$$|\widehat{\Delta}_\theta(s, a) - \Delta_\theta(s, a)| = \tilde{O}\left(\frac{B}{\sqrt{n_E}}\right) \leq \frac{\varepsilon}{8H},$$

where we used the definition of  $n_E$  in Line 22 of Algorithm 5. From this we learn two things. First, whenever the learner encounters a state  $s$  with  $\Delta_{\theta_k}(s, a) \geq \frac{\varepsilon}{2H}$ , then

$$\widehat{\Delta}_{\theta_k}(s, a) \geq \Delta_{\theta_k}(s, a) - \frac{\varepsilon}{8H} \geq \frac{3\varepsilon}{8H} = \varepsilon_E,$$

which means the algorithm throws an exception. Furthermore, if the algorithm throws an exception, then

$$\frac{3\varepsilon}{8H} = \varepsilon_E \leq \min_{a \in A} \widehat{\Delta}_{\theta_k}(s, a) \leq \min_{a \in A} \Delta_{\theta_k}(s, a) + \frac{\varepsilon}{4H},$$

hence,  $\min_{a \in A} \Delta_{\theta_k}(s, a) \geq \frac{\varepsilon}{4H}$ .  $\square$

**PROOF OF LEMMA 9.4.** Our goal now is to show that estimate of  $\phi_k^\circ$  is accurate. For each action  $a \in A$  let  $(r_1, s_1), \dots, (r_{n_F}, s_{n_F})$  be in i.i.d. sample from  $\mathbf{T}(s, a)$  and

$$\widehat{\phi}(s, a) = \frac{1}{n_F} \sum_{i=1}^{n_F} [\phi(s) - r_i e_d - \phi(s_i)].$$

By assumption  $\|\phi(s)\|_\infty \leq \|\phi(s)\|_2 \leq B$  and hence by Hoeffding's bound

$$\|\widehat{\phi}(s, a) - \phi(s, a)\|_\infty = \tilde{O}\left(B\sqrt{\frac{d}{n_F}}\right).$$

Next, let  $\widehat{\phi}^\circ = \otimes_{a \in A} \widehat{\phi}(s, a)$  and remember that  $\phi^\circ(s, a) = \otimes_{a \in A} \phi(s, a)$ . Then, by Lemma 9.5, for any  $\theta \in \mathbb{B}_1$ ,

$$\begin{aligned} \left| \langle \widehat{\phi}^\circ(s, a) - \phi^\circ(s, a), \theta \rangle \right| &\leq \|\widehat{\phi}^\circ(s, a) - \phi^\circ(s, a)\| \|\theta\| \\ &= \tilde{O}\left(B^{|A|+1} |A| \sqrt{\frac{d^{|A|+1}}{n_F}}\right) \\ &\leq \varepsilon_F. \end{aligned}$$

$\square$

**LEMMA 9.5.** Suppose that  $u, \hat{u} : A \rightarrow \mathbb{R}^d$  and  $\|u(a)\| \leq B$  for all  $a \in A$  and  $\Delta = \max_{a \in A} \|u(a) - \hat{u}(a)\|_\infty$ .

$$\|\otimes_{a \in A} u(a) - \otimes_{a \in A} \hat{u}(a)\| \leq \Delta |A| d^{|A|/2} (B + \Delta)^{|A|-1}.$$

**PROOF.** Apply Lemma 9.6 below.  $\square$

**LEMMA 9.6.** Let  $x, y \in \mathbb{R}^{|A|}$  Then

$$\left| \prod_{a=1}^{|A|} x_a - \prod_{a=1}^{|A|} (x_a + y_a) \right| \leq |A| \|y\|_\infty [\|x\|_\infty + \|y\|_\infty]^{|A|-1}.$$

## 9.4 Remarks and references

- Theorem 9.2 was proven under Assumption 9.1 that  $v^*$  is linearly realisable. Suppose instead we assume there exists a  $\pi^\dagger$  and  $\theta \in \mathcal{C}$  such that  $v^{\pi^\dagger}(s) = \langle \phi(s), \theta \rangle$ . Then the same analysis shows that TENSORPLAN finds a policy  $\pi$  such that  $v^\pi(s_1) \geq v^{\pi^\dagger}(s_1) - \varepsilon$ . That is, TENSORPLAN finds a policy that is nearly as good as the best policy for which the value function is linearly realisable.
- TENSORPLAN has been generalised to the discounted setting [50].
- The exponential dependence on  $|A|$  seems like an unfortunate consequence of the linearisation argument. The reader should wonder whether or not this essential. After all, we did not have to do this linearisation and intuitively it may feel quite wasteful.

## 10. SUMMARY AND OUTLOOK

This tutorial has covered only a miniscule fraction of the literature. We have focussed entirely on our primary expertise, which is the *theory* of sample-based planning and learning. Most research on this topic is about discovering the minimum set of conditions under which sample and/or computationally efficient planning/learning is possible. Unsurprisingly there are tradeoffs between the desiderata. Weaker assumptions on the class of possible MDPs makes learning more challenging. The environment class is not the only factor. The interaction protocol also plays an essential role. And to a lesser extent, so too does the performance metric.

A large part of the literature that we have not touched on is the study of special environment classes such as factored MDPs [12], block MDPs [61], linear quadratic regulators [2], queueing systems [34] and many more. Some of these can be addressed (at least theoretically) by the general results described above. And indeed, they serve as good benchmarks for complexity measures like the Bellman eluder dimension. On the other hand, specific environment classes often have structure that can be exploited for computational or statistical gains relative to the generic algorithms.

There are still a huge number of open theoretical questions. The minimax regret or PAC bound is only known with any degree of precision for a few simple classes (tabular and linear MDPs, for example). The relationship between the many complexity measures is still murky and we do not quite have the analogue of VC dimension for RL. We imagine there will be considerable progress in this direction in the coming years.

Maybe the biggest outstanding problem is the chasm between theory and practice. Many of the theoretically motivated algorithms cannot be reasonably implemented in practice. One of the obstructions is that theoretically we

understand the need for sophisticated exploration techniques that rely on high quality uncertainty quantification. Practically speaking uncertainty quantification is generally quite challenging. Moreover, the exploration techniques themselves, including optimism, introduce computational issues (how to find the optimistic policy?). So far there has been disappointingly little progress on this issue. Maybe one way to make progress is to understand under what circumstances sophisticated exploration is not needed. This kind of result has been explored in contextual bandits [9].

## ACKNOWLEDGMENTS

Csaba Szepesvári gratefully acknowledges funding from the Canada CIFAR AI Chairs Program, Amii and NSERC.

## REFERENCES

- [1] ABBASI-YADKORI, Y., PÁL, D. and SZEPESVÁRI, C. (2011). Improved Algorithms for Linear Stochastic Bandits. In *Advances in Neural Information Processing Systems* 2312–2320. Curran Associates, Inc.
- [2] ABBASI-YADKORI, Y. and SZEPESVÁRI, C. (2011). Regret bounds for the adaptive control of linear quadratic systems. In *Proceedings of the 24th Annual Conference on Learning Theory* 1–26. JMLR Workshop and Conference Proceedings.
- [3] ABBASI-YADKORI, Y., BARTLETT, P., BHATIA, K., LAZIC, N., SZEPESVARI, C. and WEISZ, G. (2019). Polite: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning* 3692–3702. PMLR.
- [4] AGARWAL, A., HENAFF, M., KAKADE, S. and SUN, W. (2020). Pc-pg: Policy cover directed exploration for provable policy gradient learning. *Advances in neural information processing systems* **33** 13399–13412.
- [5] AGARWAL, A., KAKADE, S. M., LEE, J. D. and MAHAJAN, G. (2021). On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research* **22** 1–76.
- [6] AGARWAL, A., JIANG, N., KAKADE, S. M. and SUN, W. (2022). *Reinforcement Learning: Theory and Algorithms*.
- [7] ANTOS, A., SZEPESVÁRI, C. and MUNOS, R. (2006). Learning Near-optimal Policies with Bellman-residual Minimization based Fitted Policy Iteration and a Single Sample Path. In *COLT* 574–588.
- [8] BAGNELL, J., KAKADE, S. M., SCHNEIDER, J. and NG, A. (2003). Policy Search by Dynamic Programming. In *Advances in Neural Information Processing Systems* (S. THRUN, L. SAUL and B. SCHÖLKOPF, eds.) **16**. MIT Press.
- [9] BASTANI, H., BAYATI, M. and KHOSRAVI, K. (2021). Mostly exploration-free algorithms for contextual bandits. *Management Science* **67** 1329–1349.
- [10] BITTANTI, S. and CAMPI, M. C. (2006). Adaptive Control of Linear Time Invariant Systems: The “Bet on the Best” Principle. *Communications in Information & Systems* **6** 299–320.
- [11] BOUCHERON, S., LUGOSI, G. and MASSART, P. (2013). *Concentration inequalities: A nonasymptotic theory of independence*. OUP Oxford.
- [12] BOUTILIER, C., DEARDEN, R., GOLDSZMIDT, M. et al. (1995). Exploiting structure in policy construction. In *IJCAI* **14** 1104–1113.
- [13] BRUKHIM, N., DUDIK, M., PACCHIANO, A. and SCHAPIRE, R. E. (2023). A unified model and dimension for interactive estimation. *Advances in Neural Information Processing Systems* **36** 64589–64617.
- [14] CAMPI, M. C. (1997). Achieving optimality in adaptive control: the “bet on the best” approach. In *Proceedings of the 36th IEEE Conference on Decision and Control* **5** 4671–4676.
- [15] CHOW, C. S. and TSITSIKLIS, J. N. (1989). The complexity of dynamic programming. *Journal of Complexity* **5** 466–488.
- [16] DANN, C., LATTIMORE, T. and BRUNSKILL, E. (2017). Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning. *Advances in Neural Information Processing Systems* **30**.
- [17] DANN, C., MOHRI, M., ZHANG, T. and ZIMMERT, J. (2021). A provably efficient model-free posterior sampling method for episodic reinforcement learning. *Advances in Neural Information Processing Systems* **34** 12040–12051.
- [18] DU, S. S., KAKADE, S. M., WANG, R. and YANG, L. F. (2019). Is a Good Representation Sufficient for Sample Efficient Reinforcement Learning? In *International Conference on Learning Representations*.
- [19] DU, S., KAKADE, S., LEE, J., LOVETT, S., MAHAJAN, G., SUN, W. and WANG, R. (2021). Bilinear classes: A structural framework for provable generalization in rl. In *International Conference on Machine Learning* 2826–2836. PMLR.
- [20] FOSTER, D. J., RAKHLIN, A., SIMCHI-LEVI, D. and XU, Y. (2020). Instance-dependent complexity of contextual bandits and reinforcement learning: A disagreement-based perspective. *arXiv preprint arXiv:2010.03104*.
- [21] FOSTER, D. J., KAKADE, S., QIAN, J. and RAKHLIN, A. (2021). The statistical complexity of interactive decision making. *arXiv preprint arXiv:2112.13487*.
- [22] JIANG, N., KRISHNAMURTHY, A., AGARWAL, A., LANGFORD, J. and SCHAPIRE, R. E. (2017). Contextual decision processes with low bellman rank are PAC-learnable. In *International Conference on Machine Learning* 1704–1713. PMLR.
- [23] JIN, C., LIU, Q. and MIRYOSEFI, S. (2021). Bellman eluder dimension: New rich classes of RL problems, and sample-efficient algorithms. *Advances in neural information processing systems* **34** 13406–13418.
- [24] JIN, C., YANG, Z., WANG, Z. and JORDAN, M. I. (2020). Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory* 2137–2143. PMLR.
- [25] KEARNS, M. and SINGH, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine learning* **49** 209–232.
- [26] KIEFER, J. and WOLFOWITZ, J. (1960). The equivalence of two extremum problems. *Canadian Journal of Mathematics* **12** 363–365.
- [27] KUMAR, P. and BECKER, A. (1982). A new family of optimal adaptive controllers for Markov chains. *IEEE Transactions on Automatic Control* **27** 137–146.
- [28] LAI, T. L. and ROBBINS, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics* **6** 4–22.
- [29] LATTIMORE, T. and SZEPESVÁRI, C. (2019). Learning with Good Feature Representations in Bandits and in RL with a Generative Model. *arXiv:1911.07676*.
- [30] LATTIMORE, T. and SZEPESVÁRI, C. (2020). *Bandit algorithms*. Cambridge University Press.
- [31] LI, Y., WANG, R. and YANG, L. F. (2022). Settling the horizon-dependence of sample complexity in reinforcement learning. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)* 965–976. IEEE.

- [32] LI, S. and YANG, L. (2023). Horizon-free learning for Markov decision processes and games: stochastically bounded rewards and improved bounds. In *International Conference on Machine Learning 2021–20252*. PMLR.
- [33] LI, G., KAMATH, P., FOSTER, D. J. and SREBRO, N. (2022). Understanding the eluder dimension. In *Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22*.
- [34] LIU, B., XIE, Q. and MODIANO, E. (2019). Reinforcement learning for optimal control of queueing systems. In *2019 57th annual allerton conference on communication, control, and computing (allerton)* 663–670. IEEE.
- [35] LU, X. (2020). *Information-directed sampling for reinforcement learning*. Stanford University.
- [36] LU, X., VAN ROY, B., DWARACHERLA, V., IBRAHIMI, M., OSBAND, I., WEN, Z. et al. (2023). Reinforcement learning, bit by bit. *Foundations and Trends® in Machine Learning* **16** 733–865.
- [37] MUNOS, R. and SZEPEŠVÁRI, C. (2008). Finite Time Bounds for Fitted Value Iteration. **9** 815–857.
- [38] OSBAND, I. and VAN ROY, B. (2014). Model-based reinforcement learning and the eluder dimension. *Advances in Neural Information Processing Systems* **27**.
- [39] RUSSO, D. and VAN ROY, B. (2013). Eluder Dimension and the Sample Complexity of Optimistic Exploration. In *Advances in Neural Information Processing Systems* 2256–2264. Curran Associates, Inc.
- [40] RUST, J. (1996). Using randomization to break the curse of dimensionality. *Econometrica* **65** 487–516.
- [41] SCHERRER, B. (2013). On the Performance Bounds of some Policy Search Dynamic Programming Algorithms. *CoRR*.
- [42] SHALEV-SHWARTZ, S. and BEN-DAVID, S. (2009). *Understanding Machine Learning. From Theory to Algorithms*. Cambridge University Press, Cambridge.
- [43] SHARIFF, R. and SZEPEŠVÁRI, C. (2020). Efficient planning in large MDPs with weak linear function approximation. *Advances in Neural Information Processing Systems* **33** 19163–19174.
- [44] SUTTON, R. and BARTO, A. (2018). *Reinforcement Learning: An Introduction*, Second ed. MIT Press.
- [45] SZEPEŠVÁRI, C. (2001). Efficient Approximate Planning in Continuous Space Markovian Decision Problems. *AI Communications* **13** 163–176.
- [46] SZEPEŠVÁRI, C. (2010). *Algorithms for Reinforcement Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers.
- [47] TODD, M. J. (2016). *Minimum-volume ellipsoids: Theory and algorithms*. SIAM.
- [48] WANG, R., SALAKHUTDINOV, R. and YANG, L. F. (2020). Provably efficient reinforcement learning with general value function approximation. *arXiv preprint arXiv:2005.10804*.
- [49] WANG, R., DU, S. S., YANG, L. F. and KAKADE, S. M. (2020). Is long horizon reinforcement learning more difficult than short horizon reinforcement learning? *arXiv preprint arXiv:2005.00527*.
- [50] WEISZ, G., AMORTILA, P. and SZEPEŠVÁRI, C. (2021). Exponential lower bounds for planning in MDPs with linearly-realizable optimal action-value functions. In *Algorithmic Learning Theory* 1237–1264. PMLR.
- [51] WEISZ, G., GYÖRGY, A. and SZEPEŠVÁRI, C. (2023). On-line RL in Linearly  $q^\pi$ -Realizable MDPs Is as Easy as in Linear MDPs If You Learn What to Ignore. In *Advances in Neural Information Processing Systems* (A. OH, T. NEUMANN, A. GLOBERSON, K. SAENKO, M. HARDT and S. LEVINE, eds.) **36** 59172–59205. Curran Associates, Inc.
- [52] WEISZ, G., SZEPEŠVÁRI, C. and GYÖRGY, A. (2022). Tensorplan and the few actions lower bound for planning in MDPs under linear realizability of optimal value functions. In *International Conference on Algorithmic Learning Theory* 1097–1137. PMLR.
- [53] WEISZ, G., AMORTILA, P., JANZER, B., ABBASI-YADKORI, Y., JIANG, N. and SZEPEŠVÁRI, C. (2021). On query-efficient planning in MDPs under linear realizability of the optimal state-value function. In *Conference on Learning Theory* 4355–4385. PMLR.
- [54] WEISZ, G., GYÖRGY, A., KOZUNO, T. and SZEPEŠVÁRI, C. (2022). Confident Approximate Policy Iteration for Efficient Local Planning in  $q^\pi$ -realizable MDPs. *Advances in Neural Information Processing Systems* **35** 25547–25559.
- [55] WEN, Z. and VAN ROY, B. (2017). Efficient reinforcement learning in deterministic systems with value function generalization. *Mathematics of Operations Research* **42** 762–782.
- [56] YIN, D., HAO, B., ABBASI-YADKORI, Y., LAZIĆ, N. and SZEPEŠVÁRI, C. (2022). Efficient local planning with linear function approximation. In *International Conference on Algorithmic Learning Theory* 1165–1192. PMLR.
- [57] ZANETTE, A. and BRUNSKILL, E. (2019). Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *International Conference on Machine Learning* 7304–7312. PMLR.
- [58] ZANETTE, A., LAZARIC, A., KOCHENDERFER, M. J. and BRUNSKILL, E. (2019). Limiting extrapolation in linear approximate value iteration. *Advances in Neural Information Processing Systems* **32**.
- [59] ZANETTE, A., LAZARIC, A., KOCHENDERFER, M. and BRUNSKILL, E. (2020). Learning near optimal policies with low inherent Bellman error. In *International Conference on Machine Learning* 10978–10989. PMLR.
- [60] ZHANG, Z., JI, X. and DU, S. (2022). Horizon-free reinforcement learning in polynomial time: the power of stationary policies. In *Conference on Learning Theory* 3858–3904. PMLR.
- [61] ZHANG, X., SONG, Y., UEHARA, M., WANG, M., AGARWAL, A. and SUN, W. (2022). Efficient reinforcement learning in block mdps: A model-free representation learning approach. In *International Conference on Machine Learning* 26517–26547. PMLR.
- [62] ZIMMERT, J. and LATTIMORE, T. (2022). Return of the bias: Almost minimax optimal high probability bounds for adversarial linear bandits. In *Proceedings of Thirty Fifth Conference on Learning Theory* (P.-L. LOH and M. RAGINSKY, eds.). *Proceedings of Machine Learning Research* **178** 3285–3312. PMLR.