Exponential Lower Bounds for Batch Reinforcement Learning: Batch RL can be Exponentially Harder than Online RL

Andrea Zanette [1]

Abstract

Several practical applications of reinforcement learning involve an agent learning from past data without the possibility of further exploration. Often these applications require us to 1) identify a near optimal policy or to 2) estimate the value of a target policy. For both tasks we derive *exponential* information-theoretic lower bounds in discounted infinite horizon MDPs with a linear function representation for the action value function even if 1) *realizability* holds, 2) the batch algorithm observes the exact reward and transition *functions*, and 3) the batch algorithm is given the *best* a priori data distribution for the problem class. Our work introduces a new 'oracle + batch algorithm' framework to prove lower bounds that hold for every distribution. The work shows an exponential separation between batch and online reinforcement learning.

1. Introduction

While the grand goal of reinforcement learning (RL) is to design fully autonomous agents capable of improving their performance over time by learning from past mistakes, oftentimes a *batch* approach — which predicts some quantity of interest using past observations only — is preferable. For example, past data may be available in large quantities and should not be disregarded by adopting a purely online method. In other applications, safety concerns require that the dataset be collected by a carefully monitored procedure, involving a human or a safe controller. In addition, batch algorithms are key tools to build complex online procedures.

These considerations motivate us to investigate whether there exists any *fundamental limitation* to using a batch approach for RL compared to online learning. Concretely, we consider two classical batch RL problems: 1) the *off-policy evaluation* (OPE) problem, where the batch algorithm needs to predict the performance of a *target policy* and 2) the *best policy identification* (BPI) problem, where the batch algorithm needs to identify a near optimal policy.

Linear function approximation As many applications of RL require very large state and action spaces, the hope is that by leveraging strong prior knowledge, for example on the form of the optimal solution, we can learn the critical features of a Markov decision process (MDP) to solve the task at hand without probing the full state-action space. For the OPE problem, we assume that the action-value function Q^π of the target policy π can be written at any state action pair (s, a) as an inner product $\phi(s, a)^\top \theta$ between a known feature extractor ϕ and an unknown parameter θ that the learner seeks to identify. For the BPI problem, this representation condition applies to the action-value function $Q^\star$ of an optimal policy.

Quality of the dataset and assumptions Batch algorithms are limited by the quality and quantity of the available data: for example, in a tabular MDP, without adequate coverage of the area of the MDP that the optimal policy tends to visit, there is little hope that we can identify such policy or even predict its performance. However, given enough samples in each state-action pairs, it becomes easy to identify the optimal policy (Azar et al., 2012) or to evaluate the value of another (Yin et al., 2020). In addition, without any prior knowledge about the problem or information about the target policy, a uniform distribution over the state-action space is a priori optimal.

Very recently, some authors have derived exponential lower bounds for the off-policy evaluation problem with linear function approximations (Wang et al., 2020a; Amortila et al., 2020). They discover that even if a sampling distribution induces the best-conditioned covariance matrix, at least exponentially many samples are needed to estimate the value of a target policy reasonably well. As their hard instances are tabular MDPs (i.e., with small state-action spaces), certainly *there exists a better batch distribution* for their setting: the OPE problem in tabular MDPs is easily solvable by using a uniform distribution over the state-

[1]Institute for Computational and Mathematical Engineering, Stanford University, Stanford, USA. Correspondence to: Andrea Zanette <zanette@stanford.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

actions coupled with the policy evaluation algorithm on the empirical model. Therefore, their works show that the assumption on the condition number of the covariance matrix is insufficient in batch RL. However, their works leave open the question of whether the OPE problem with linear value functions can *always* be solved by using a better sampling distribution that generates a dataset of polynomial size (in the horizon and feature dimension), leading to the following, more fundamental question in the context of batch RL with function approximation:

What can a batch algorithm learn using the best a priori distribution for the problem class at hand?

In particular, we wonder whether there is any penalty in using an entirely batch approach in place of an online (and adaptive) method if a good dataset is provided to the batch algorithm. To answer this question for both the OPE and BPI problems in the strongest possible way, we consider an auxiliary process or *oracle* that provides the *batch algorithm* with the best data distribution for the task; together they form a *learning algorithm*. However, in contrast to fully online / adaptive algorithms, the oracle is not allowed to change its data acquisition strategy while information is being acquired. As we explain in Appendix A, this framework allows us to derive strong batch RL lower bounds, because they *will hold for every batch distribution*. Thanks to this framework, we can recover the concurrent lower bound by (Wang et al., 2020a) — which holds for one specific choice of the sampling distribution — in infinite horizon RL as a corollary.

Learning process and assumptions We consider oracles that can decide on a set of strategic policies to gather data. Alternatively, the oracles can directly specify the state-actions where they want to observe the rewards and transitions (for example, with a simulator). In either case, the oracle selects a sampling strategy and the batch algorithm later receives a *dataset* of states, actions, rewards and transitions. Using the dataset, the batch algorithm makes a prediction, i.e., it estimates the value of a target policy (OPE problem) or returns a near optimal policy (BPI problem). We make two other assumptions that favor the learner:

- *Realizability*, i.e., there is no misspecification.
- *Exact feedback*: the exact reward and transition *function* is observed for each point in the dataset. This is equivalent to observing *infinite data*.

1.1. Contributions

With such strong assumptions and the freedom of selecting any state-action, identifying a near optimal policy or predicting the value of another should be easy tasks for the learner. For example, in tabular MDPs the oracle could prescribe one query in each state-action pair. This way, the reward and transition functions would become known everywhere, giving the batch algorithm complete knowledge of the MDP. For linear bandits with feature vectors spanning $\mathbb{R}^d$, we know that d queries along a basis suffice to identify the full bandit instance. For H horizon MDPs with linear representations, backward induction with dH queries and exact feedback precisely identifies the MDP at hand. In all these cases the oracle can find a good sampling distribution for the batch learner, and there is little or no advantage to using an oracle that adapts the query selection strategy as feedback is received. However, in infinite horizon problems the situation is quite different; we show that

1. there exists OPE and BPI problems where any batch algorithm must receive an *exponential* $\approx (\frac{1}{1-\gamma})^d$ dataset to return a good answer,
2. if the dataset does not originate from policy rollouts then the lower bounds hold even if the action-value function of *every policy* admits a linear representation,
3. there exist exponentially hard batch BPI problems (even under the best data distribution) which are easy to solve with online / adaptive algorithms, showing *an exponential separation between batch and online RL*,
4. there exist exponentially hard problems for infinite horizon batch RL which cannot arise in finite horizon problems, showing *exponential separation between finite and infinite horizon batch RL*.

As a corollary, our work recovers (Wang et al., 2020a)'s lower bound in infinite horizon RL and also shows that the classical globally optimal experimental design yields *provably bad* distributions for infinite horizon RL, making learning impossible even in the limit of infinite data; we discuss this in Appendix A.

We make the following *technical* contributions:

1. we introduce a new 'oracle + batch algorithm' framework to derive lower bounds for *every a priori distribution*; this automatically yields fixed distribution lower bounds (e.g., (Wang et al., 2020a)) as a special case,
2. we help formalize the hardness induced by the *deadly triad* (Sutton & Barto, 2018), i.e., the combination of off-policy learning, bootstrapping and function approximation; in particular, we explain that the *bootstrapping* problem is fundamentally different and potentially more severe than the *extrapolation* problem,

3. we present new classes of hard MDPs where critical MDP information is 'deferred — through bootstrapping — and hidden in an unknown and exponentially small region *in feature space*, too small to be covered by a batch dataset but easy to locate using an online algorithm.

1.2. Literature

Polynomial lower bounds are often derived to certify that a certain algorithm is sample efficient (Jiang & Li, 2016; Duan & Wang, 2020; Hao et al., 2020); in this work we are interested in exponential lower bounds. Divergence of dynamic programing algorithms with function approximation is well known (Baird, 1995; Tsitsiklis & Van Roy, 1996) and has prompted researchers to look for information-theoretic lower bounds for generic predictors (Chen & Jiang, 2019) and in presence of misspecification (Du et al., 2019). Concurrently, Weisz et al. (2020); Wang et al. (2020a) also show information-theoretic lower bounds highlighting the danger of *extrapolation*; for additional literature, please see app. B.

2. Preliminaries

Discounted infinite horizon MDPs (Puterman, 1994) are defined by a tuple $M = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}_s$ is the action space in state $s \in \mathcal{S}$ and $\mathcal{A} = \cup_{s \in \mathcal{S}} \{\mathcal{A}_s\}$ is the set of all state-dependent action spaces. We assume that for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}_s$ there exists a measure $p(\cdot \mid s, a)$ representing the transition dynamics and a scalar reward function $r(s, a) \in [-1, 1]$. The discount factor γ is assumed to be in $(0, 1)$. We consider the discounted return of any policy to be $\in [-1, +1]$ (without loss of generality upon rescaling the reward function). A deterministic policy π maps every state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}_s$. The value function of policy π in state $s \in \mathcal{S}$ is defined as $V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \, \mathbb{E}_{x_t \sim \pi | x_0 = s} \, r(x_t, \pi(x_t))$; the limit always exists and it is finite under our assumptions as long as the expectation is well defined. The action-value function Q^π of policy π in (s, a) also exists and is defined as $Q^\pi(s, a) = r(s, a) + \sum_{t=1}^{\infty} \gamma^t \, \mathbb{E}_{x_t \sim \pi | (s,a)} \, r(x_t, \pi(x_t))$. An optimal policy $\pi^\star$, when it exists, is defined in every state as $\pi^\star(s) = \arg\max_\pi V^\pi(s)$ and the corresponding value function and action-value function are denoted with $V^\star = V^{\pi^\star}$ and $Q^\star = Q^{\pi^\star}$. We sometime add the subscript M to indicate that a certain quantity depends on the MDP M under consideration (e.g., we write $V^\star_M$). The Euclidean ball in $\mathbb{R}^d$ is defined as $\mathcal{B} = \{x \in \mathbb{R}^d \mid \|x\|_2 \leq 1\}$. The *Bellman optimality operator* $\mathcal{T}$ and the *Bellman evaluation operator* $\mathcal{T}^\pi$ are mappings between action value functions:

$$(\mathcal{T}Q)(s, a) = r(s, a) + \gamma \, \mathbb{E}_{s' \sim p(s,a)} \sup_{a' \in \mathcal{A}_{s'}} Q(s', a')$$

$$(\mathcal{T}^\pi Q)(s, a) = r(s, a) + \gamma \, \mathbb{E}_{s' \sim p(s,a)} \, Q(s', \pi(s')).$$

3. Batch Reinforcement Learning

We first formally define the two *learning problems*, namely the task of returning a near optimal policy, also known as *best policy identification* (BPI) problem, and the task of predicting the value of a *target policy*, also known as *off-policy evaluation* (OPE) problem. Then, in the next four sub-sections we describe the learning process in more detail. The lower bounds that we later derive will hold for all algorithms of the form described in this section.

A *BPI problem* $(s^\star, \{M \in \mathcal{M}\})$ is defined by a starting state $s^\star$ and a class $\mathcal{M}$ of MDPs sharing the same state space, action space and discount factor. An *OPE problem* $(s^\star, \{(M, \pi_M) \mid M \in \mathcal{M}\})$ additionally requires us to identify one or more target policies π_M for each $M \in \mathcal{M}$.

Depending on the problem, the oracle selects a sampling strategy such that for every *problem instance* $(s^\star, M, \pi_M)$ (of an OPE problem) or $(s^\star, M)$ (of a BPI problem) the batch algorithm experiences a dataset $\mathcal{D}$ from M and uses it to return an accurate estimate $\widehat{Q}_\mathcal{D}$ of the action-value function of π_M at $s^\star$ (for the OPE problem) or a near optimal policy $\widehat{\pi}^\star_\mathcal{D}$ on M from $s^\star$ (for the BPI problem).

Protocol 1 BATCH RL WITH A STRATEGIC ORACLE

1: *(Input)* Oracle receives a batch problem and budget n
2: *(Query Selection)* Oracle chooses μ (def. 1) or T (def. 2).
3: *(Data Collection)* Batch algorithm receives the dataset $\mathcal{D}$
4: *(Output)* Batch algorithm returns $\widehat{Q}_\mathcal{D}$ or $\widehat{\pi}^\star_\mathcal{D}$

3.1. Step I: Input

The oracle receives *either* an OPE or a BPI problem together with a *query budget* $n \in \mathbb{N}$. The oracle has access to each instance of the problem it is given, and in particular it knows the state and action spaces $\mathcal{S}, \mathcal{A}$, the discount factor γ, and the full set of transition functions and reward functions for each MDP $M \in \mathcal{M}$, but it does not know which of these MDPs in $\mathcal{M}$ it will interact with or what the target policy will be. In addition, the oracle has full knowledge of the batch algorithm.

3.2. Step II: Query Selection

The purpose of the oracle is to help the batch algorithm by providing it with the best dataset for the specific problem at hand. To capture different mechanisms of data acquisition, we consider two methods to specify the *query set*, i.e., the set of state-action pairs where the oracle wants the batch

algorithm to observe the rewards and the transitions. In the first mechanism the oracle directly selects the state-actions; we place no restriction on the mechanism to obtain these queries as long as the sampling distribution is fixed for all MDPs in the class.

Definition 1 (Policy-Free Queries)**.** *A set μ of state-action pairs is said to be policy-free for an OPE or BPI problem if μ does not depend on the specific MDP instance $M \in \mathcal{M}$.*

Since the batch algorithm observes the exact reward and transition function at the selected query points, the number of policy-free queries is *the size of the support of the distribution μ.*

The second mechanism to collect data is by selecting deterministic[1] policies. The policies will generate random trajectories that the batch algorithm later observes. Since we are interested in what the agent can learn in the limit of infinite data (i.e., under exact feedback) we let the batch algorithm observe *all possible realizations* of such trajectories. With this aim, we define the state-action space reachable in c or less timesteps from s_0 using policy π as $\mathrm{Reach}(s_0, \pi, c) =$

$$\{(s,a) \mid \exists t < c, \text{ s.t. } \mathbf{P}\left((s_t, a_t) = (s,a) \mid \pi, s_0\right) > 0\}$$

where (s_t, a_t) is the random state-action encountered at timestep t upon following π from s_0. We let the oracle select the best combination of trajectory lengths and number of distinct policies.

Definition 2 (Policy-Induced Queries)**.** *Consider an OPE or BPI problem and fix a set $T = \{(s_{0i}, \pi_i, c_i)\}_{i=1}^{\kappa}$ of triplets, each containing a starting state s_{0i}, a deterministic policy π_i and a trajectory length c_i such that $\sum_{i=1}^{\kappa} c_i \leq n$. Then the query set μ induced by T is defined as*

$$\mu \overset{def}{=} \bigcup_{(s_0,\pi,c)\in T} \mathrm{Reach}(s_0, \pi, c).$$

The condition $\sum_{i=1}^{\kappa} c_i \leq n$ attempts to make the amount of information acquired using policy-free queries comparable to the policy-induced query method, although the latter generates $|\mu| \geq n$ if the dynamics are stochastic. In other words, the batch learner always observes at least n state-actions if these are induced by policies.

Remark 1. *A policy-induced query set is also policy-free whenever the dynamics of each MDP $M \in \mathcal{M}$ are the same.*

Remark 2. *When prescribing a set T the oracle has knowledge of the induced dataset μ for each choice of T (e.g., by having access to a connectivity graph of the MDP).*

[1]This is not a restriction: for any given stochastic policy the agent can sample an action from the distribution of actions in every state before deploying the policy.

While a policy-free query set may seem less constrained than a policy-induced query set, the latter may implicitly reveal additional information about the dynamics of the MDP whenever the induced trajectories are all different across different MDPs.

3.3. Step III: Data Collection

After the oracle has submitted the sampling strategy, the batch learner receives a dataset $\mathcal{D}$ which contains the *exact* reward and transition *function*, $r(s,a)$ and $p(s,a)$, from the MDP $M \in \mathcal{M}$ for each (s,a) in the query set μ.

3.4. Step IV: Output

The batch algorithm is finally required to make a prediction using the acquired dataset $\mathcal{D}$. For the OPE problem, the batch algorithm also receives the target policy π_M and it is required to output an estimator $\widehat{Q}_{\mathcal{D}}$ for the action-value function of the target policy $Q_M^{\pi_M}(s^\star, \cdot)$; for the BPI problem, it must output a near-optimal policy $\widehat{\pi}^\star_{\mathcal{D}}$ at $s^\star$.

3.5. Evaluation Criterion

The *oracle* and the *batch algorithm* together form a *learning algorithm*. This framework allows us to derive batch lower bounds in a strong form as they will hold for any data distribution. We say that the learning algorithm is (ϵ, δ)-*sound* for an OPE problem if for every instance $(s^\star, M, \pi_M)$ of the problem the returned estimator $\widehat{Q}_{\mathcal{D}}$ is accurate w.h.p.:

$$\mathbf{P}\left(\sup_{a\in\mathcal{A}_{s^\star}} |(Q_M^{\pi_M} - \widehat{Q}_{\mathcal{D}})(s^\star, \cdot)| < \epsilon\right) > 1 - \delta.$$

Similarly, we say that that the learning algorithm is (ϵ, δ)-*sound* for a BPI problem if for every instance $(s^\star, M)$ it holds that the returned policy $\widehat{\pi}^\star_{\mathcal{D}}$ is near optimal w.h.p.:

$$\mathbf{P}\left((V_M^\star - V_M^{\widehat{\pi}^\star_{\mathcal{D}}})(s^\star) < \epsilon\right) > 1 - \delta.$$

As the query set is always non-random and the batch algorithm experiences the exact reward and transition function, the only randomness lies in the possible randomization internal to the batch algorithm when it returns an answer.

3.6. Adaptive and Online Algorithms

Consider a policy-free mechanism. We say that a learning algorithm is *adaptive* if every time the oracle submits a state-action it receives the feedback from the environment and can use it to select the next state-action to query the MDP.

Likewise, consider a policy-induced mechanism. We say that a learning algorithm is acting *online* if every time the

oracle selects a policy and trajectory length, it can use the acquired feedback to select the next combination of policy and trajectory length (its position is reset in every episode).

4. Intuition

The mechanism that induces hardness for infinite horizon problems must be different than that in finite horizon: the constructions from Weisz et al. (2020); Wang et al. (2020b) rely on the *extrapolation* issue that compounds the errors multiplicatively. In our case, the reward and transition functions are observed exactly, so there is no error to extrapolate in the first place. Instead, *bootstrapping*, i.e., the fact that the value function in one state depends on the same value function at successor states (Sutton & Barto, 2018), is the root cause of hardness; here we provide some intuition.

While each of our theorems need a different construction, they all build on the intuition presented in this section; at a high level, bootstrapping can "erase" the information gained along certain directions in feature space.

Suppose that the oracle is trying to find a good policy-free query set for the off-policy problem on a class of MDPs with feature vectors $\phi(\cdot,\cdot)$ anywhere in the unit Euclidean ball $\mathcal{B}$. Denote with $(s_1,a_1),\dots,(s_n,a_n)$ the state-actions chosen by the oracle and with $(s_1^+,a_1^+),\dots,(s_n^+,a_n^+)$ the corresponding successor states and actions[2]. Intuitively, choosing a set $\phi(s_1,a_1),\dots,\phi(s_n,a_n)$ of orthogonal feature vectors of maximal length seems to be the best the oracle can do because it gives rise to a covariance matrix $\Phi^\top\Phi$ (where Φ is described below) that is a multiple of the identity. In Fig. 1 (left) we represent the feature space of an RL problem where the learner can choose any feature vector in the Euclidean ball in $\mathbb{R}^2$; the agent's choice of the feature vectors $\phi_i = \phi(s_i,a_i)$ arise from globally optimal experimental design (Pukelsheim, 2006). Define

$$\Phi = \begin{bmatrix} \phi(s_1,a_1)^\top \\ \dots \\ \phi(s_n,a_n)^\top \end{bmatrix}, r = \begin{bmatrix} r(s_1,a_1) \\ \dots \\ r(s_n,a_n) \end{bmatrix}, \Phi^+ = \begin{bmatrix} \phi(s_1^+,a_1^+)^\top \\ \dots \\ \phi(s_n^+,a_n^+)^\top \end{bmatrix}$$

After observing the reward and successor states, we can write down the local (i.e., at the state-actions chosen by the oracle) Bellman equations. If we leverage the known functional form of the action-value function, we can write the following linear system, whose solution (in terms of $\theta \in \mathbb{R}^d$ or action-value function $\Phi\theta$) the batch algorithm seeks to discover:

$$\Phi\theta = r + \gamma\Phi^+\theta \quad \longrightarrow \quad (\Phi - \gamma\Phi^+)\theta = r. \qquad (1)$$

Unlike machine learning or bandit problems where the agent directly observes a response from $\phi(s_i,a_i)$, bootstrapping the future returns makes the agent observe a response (i.e., the reward) corresponding not to $\phi(s_i,a_i)$ but to $\phi(s_i,a_i) - \gamma\phi(s_i^+,a_i^+)$, which we call "effective feature vector", see Eq. (1). Unfortunately, $\phi(s_i^+,a_i^+)$ can act adversarially. As an example, consider Fig. 1 (left) in $\mathbb{R}^2$ where the orthogonal feature vectors chosen by the agent are projected along the vertical direction by bootstrapping: the net effect is that the agent only learns along one axis. Mathematically, when this happens the final system of linear equations in Eq. (1) admits infinitely-many solutions since $\Phi - \gamma\Phi^+$ is rank deficient[3]. From a reinforcement learning perspective, the local Bellman equations in Eq. (1) admit multiple linear value functions as possible solutions. This happens because the rewards and transitions that the batch algorithm has received could have originated from any MDP whose action-value function is a solution to the local Bellman equations.

Unfortunately, choosing more feature vectors does not easily solve the problem (Fig. 1 (middle)). To acquire information in all directions (i.e., to ensure that $\Phi - \gamma\Phi^+$ in Eq. (1) is full rank) the oracle must probe the small spherical cap in Fig. 1 (right). Which spherical cap to probe is unknown ahead of sampling (as the target policy or the optimal policy are unknown), and thus the oracle needs to probe all of them if it wants the batch algorithm to confidently solve the problem. Using the fact that there are exponentially many spherical caps in an Euclidean ball in $\mathbb{R}^d$, we conclude.

Exponential separation with online learning An online algorithm can detect that the next-state feature matrix Φ^+ is acting adversarially. In addition, Φ^+ reveals the location of the spherical cap in Fig. 1 (right). The online algorithm can then probe the spherical cap to ensure that the linear system in Eq. (1) is full rank.

5. Exponential Lower Bounds

We define the *query complexity to* (ϵ,δ)*-soundness of a problem* (OPE or BPI) to be the minimum value for n (as in Definitions 1 and 2) such that there exists a (ϵ,δ)-sound learning algorithm for that problem. In particular, the query complexity depends on the MDP class $\mathcal{M}$ and can be different for the OPE and BPI tasks and for the policy-free and policy-induced query mechanism. Our lower bounds on the query complexity are *significantly stronger* than typical sample complexity lower bounds: they are really lower bounds on the *size of the support* of the distribution μ and automatically imply *infinite sample complexity* lower bounds since the batch algorithm already observes the ex-

[2] Assume deterministic successor states; the actions in the successor states are determined by the target policy (OPE problem) or by an $\arg\max$ function (BPI problem).

[3] This follows from the fundamental theorem of linear algebra. In particular, since the row space of $\Phi - \gamma\Phi^+$ does not span $\mathbb{R}^d$, the nullspace of $\Phi - \gamma\Phi^+$ must have dimension at least 1.

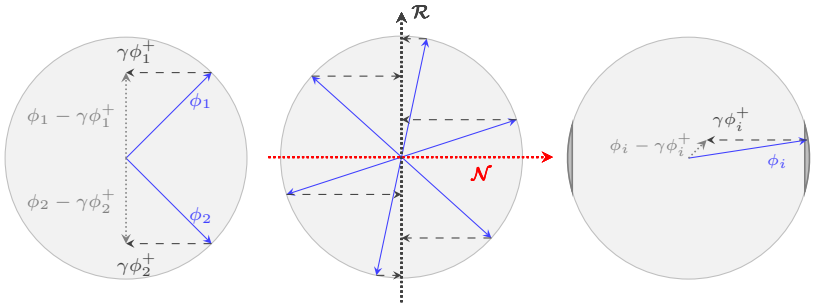

Figure 1. *Left*: the orthogonal blue vectors ϕ_1, ϕ_2 represent the choice of the algorithm, while ϕ_1^+, ϕ_2^+ are the corresponding successor features. The learner thus acquires information only along the vertical direction. *Middle*: in the figure $\mathcal{R} = \mathrm{Range}(\Phi^\top - \gamma(\Phi^+)^\top)$ and $\mathcal{N} = \mathrm{Null}(\Phi - \gamma\Phi^+)$. *Right*: in this case the next-state discounted feature $\gamma\phi_i^+$ cannot project ϕ_i onto the vertical plane because of the discount factor $\gamma < 1$.

act reward and transition functions where μ is supported.

The lower bounds are expressed in terms of the regularized incomplete beta function $I_x(a,b) = B(x,a,b)/B(1,a,b)$ where $B(x,a,b)$ is the incomplete beta function for some positive real numbers a, b and $x \in [0,1]$; for the precise definitions, please see Appendix D.1. For brevity, define $\mathcal{N}(\gamma, d) = I^{-1}_{1-\gamma^2}\left(\frac{d-1}{2}, \frac{1}{2}\right)$. Corollary 2 ensures $\mathcal{N}(\gamma,d)$ is exponential in the dimension d for $d \geq 5$ (here the $\approx$ symbol highlights an approximate dependence without a formal definition):

$$\mathcal{N}(\gamma,d) > 2^{-d}\mathcal{N}(\gamma,d) \geq \gamma\sqrt{d}\left(\frac{1}{2^{2.5}(1-\gamma)}\right)^{\frac{d-1}{2}} \approx \left(\frac{1}{1-\gamma}\right)^d.$$

5.1. Realizability Assumptions

Unless additional assumptions are made regarding the MDP class $\mathcal{M}$ that defines the OPE and BPI problems, the query complexity of a reasonably sound learner is exactly the size of the state and action space (we assume exact feedback). One hopes that by restricting the MDP class $\mathcal{M}$, the query complexity of the OPE and BPI problems can be brought down to a more manageable level, in particular, independent of the state-action spaces.

We make one of the following three assumptions (only one assumption will hold at any given time, depending on the theorem); two are known as realizability, and the third is strictly stronger than the first two. The first concerns the OPE problem; $\mathcal{B}$ is the unit Euclidean ball.

Assumption 1 (Q^π is Realizable)**.** *Given an OPE problem $(s^\star, \{(M, \pi_M), M \in \mathcal{M}\})$, there exists a d-dimensional map $\phi(\cdot,\cdot)$ such that $\|\phi(\cdot,\cdot)\|_2 \leq 1$ and for any $M \in \mathcal{M}$ the action-value function of the target policy π_M satisfies $\forall (s,a), Q_M^{\pi_M}(s,a) = \phi(s,a)^\top \theta_M$ for some $\theta_M \in \mathcal{B}$.*

For the BPI problem the representation condition applies to the action-value function of an optimal policy.

Assumption 2 ($Q^\star$ is Realizable)**.** *Given a BPI problem $(s^\star, \{M \in \mathcal{M}\})$, there exists a d-dimensional feature map $\phi(\cdot,\cdot)$ such that $\|\phi(\cdot,\cdot)\|_2 \leq 1$ and for any $M \in \mathcal{M}$ there exists $\theta_M^\star \in \mathcal{B}$ such that the optimal action-value function is linear: $\forall (s,a), Q_M^\star(s,a) = \phi(s,a)^\top \theta_M^\star$.*

A stronger assumption we make is that the action-value function of *every* policy has a linear representation.

Assumption 3 (Q^π is Realizable for every Policy)**.** *Given a BPI problem $(s^\star, \{M \in \mathcal{M}\})$ or an OPE problem $(s^\star, \{(M, \pi_M), M \in \mathcal{M}\})$, there exists a d-dimensional feature map $\phi(\cdot,\cdot)$ such that $\|\phi(\cdot,\cdot)\|_2 \leq 1$ and for any MDP $M \in \mathcal{M}$ the value of every policy π satisfies $\forall (s,a), Q_M^\pi(s,a) = \phi(s,a)^\top \theta_M^\pi$ for some $\theta_M^\pi \in \mathcal{B}$.*

The learners that we consider are aware of these assumptions because they can examine each MDP in the class $\mathcal{M}$ they receive (see Section 3.1).

5.2. Off-Policy Evaluation

The first result of this work is contained in the following lower bound for the OPE problem; since all MDPs in $\mathcal{M}$ share the same dynamics, the oracle has full knowledge of the actions it needs to take to visit any state-action it desires.

Theorem 1 (OPE Policy-Induced Lower Bound)**.** *There exists an OPE problem $(s^\star, \{(M, \pi_M), M \in \mathcal{M}\})$ satisfying Assumption 1 such that its policy-induced query complexity to $(1, 1/2)$-soundness is at least $\mathcal{N}(\gamma, d)$.*

It is useful to compare the *form* of the above lower bound with that of some concurrent results for the off-policy evaluation problem: for example, (Wang et al., 2020a; Amortila et al., 2020) show that *there exist* a sampling distribution μ

(that induces the best-conditioned covariance matrix), a target policy π and an MDP class $\widetilde{\mathcal{M}}$, each satisfying certain properties, such that the best estimator on the most difficult MDP in $\widetilde{\mathcal{M}}$ performs poorly if less than exponentially many samples are used. However, in their case there exists a better batch distribution of polynomial size that solves the problem. Our instances are instead much harder, as they command an exponential dataset even for the *best* distribution for the task: since the oracle can prescribe any data distribution, we can claim that *for all* distributions of polynomial size we can find an MDP subclass $\widetilde{\mathcal{M}} \subseteq \mathcal{M}$ and a target policy such that all MDPs in $\widetilde{\mathcal{M}}$ generate similar datasets but the value of the target policy is very different on these MDPs in $\widetilde{\mathcal{M}}$, giving lower bounds in a *stronger form*.

5.3. Best Policy Identification

As in Theorem 1, in the next lower bound the oracle knows the set μ induced by any choice of T (see Definition 2).

Theorem 2 (BPI Policy-Induced Lower Bound)**.** *There exists a BPI problem $(s^\star, \{M \in \mathcal{M}\})$ satisfying Assumption 2 with features in dimension $d+1$ such that its policy-induced query complexity to $(1/2, 1/2)$-soundness is at least $2^{-d}\mathcal{N}(\gamma, d)$.*

Hard BPI problems for adaptive and online algorithms are given by Weisz et al. (2020). Clearly, their construction can be embedded in an infinite horizon MDP, giving a lower bound under their assumptions. However, our framework allows the learner to observe the *exact* reward and transition function, which is equivalent to having infinite data at the selected state-actions. In such case, the construction of Weisz et al. (2020) no longer gives rise to hard instances. In particular, the BPI problem with our assumptions becomes straightforward in finite horizon, showing an exponential separation between finite and infinite horizon batch RL.

5.4. Lower Bounds with Stronger Representation

We wonder what is achievable if the oracle can specify the queries anywhere in the state-action space without the restriction imposed by following policies (i.e., using a policy-free query set). Under this assumption the situation gets surprisingly worse, as the lower bounds now hold even if the action-value function of *every* policy is linear.

Theorem 3 (Policy-Free Lower Bounds)**.** *There exist an OPE problem $(s^\star, \{(M, \pi_M), M \in \mathcal{M}\})$ and a BPI problem $(s^\star, \{M \in \mathcal{M}\})$, which satisfy Assumption 3 and share the same $s^\star$ and $\mathcal{M}$, such that their policy-free query complexity to $(1, 1/2)$-soundness is at least $\mathcal{N}(\gamma, d)$. In addition, an MDP class $\mathcal{M}$ that yields the lower bounds (but with $\frac{1}{\sqrt{d}}, 1/2$)-soundness) can be constucted with at most $|\mathcal{A}| = 2d$ actions.*

Contrasting Theorem 3 with Theorems 1 and 2 shows that there is a sharp distinction in what can be achieved depending on the *assumptions on the mechanism* that generates the batch dataset, a distinction which is absent in tabular RL.

6. Exponential Separation with Online Learning

Theorem 4 (Exponential Separation with Online Learning)**.** *Consider the same BPI problem as in Theorem 2. There exists an online algorithm that can identify an optimal policy with probability one by observing trajectories of length one with exact feedback from $d+1$ distinct policies from $s^\star$.*

This result shows *exponential separation with online learning* even when the best batch distribution is used. The key information is hidden in an exponentially small area of the feature space whose position is a priori unknown. This region is too small to be covered by a batch dataset. However, an online algorithm can learn where this information is hidden and then probe such region as we show in Section 7.

A similar exponential separation with online learning is available for the BPI problem in Theorem 3 (see the end of the proof in the appendix). In addition, assuming access to a generative model an even stronger result is readily available in the literature using the Least-Square Policy Iteration (LSPI) algorithm (Lagoudakis & Parr, 2003): with a generative model (Lattimore et al., 2020) show that with probability at least $1-\delta$, LSPI finds an ϵ optimal policy *for any BPI problem* $(s^\star, \{M \in \mathcal{M}\})$ that satisfies Assumption 3 using at most $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \ln \frac{1}{\delta})$ samples. However, LSPI is non-batch as it relies on Monte-Carlo rollouts at every iteration. Our result thus shows that it is not possible to start from a batch dataset obtained from a generative model and run LSPI (or any algorithm) successfully without acquiring further data even if a strong representation holds.

7. Proof Sketch of Theorem 2 and Theorem 4

We first describe the state and action space which are fixed across all MDPs in the class $\mathcal{M}$. Then we describe the instance-dependent reward and transition functions. Finally we prove the theorems.

At a high level, each MDP contains a two-armed bandit instance in $s^\star$ (the starting state). There, the learner has two choices: 1) take the special action $a^\star$ that gives a known return or 2) take any other action in the positive orthant $\mathcal{B}^+ = \{x \in \mathcal{B} \mid x_i \geq 0, i \in [d]\}$, see Fig. 2.

Crucially, on $\mathcal{B}^+$ the reward function is almost everywhere zero except inside the exponentially small spherical cap

$\mathcal{C}_\gamma(w) = \{\|x\|_2 \le 1 \mid \frac{x^\top w}{\|w\|_2} \ge \gamma\}$ (Fig. 2) for some $w \in \mathcal{B}$. Unless the oracle prescribes an action inside $\mathcal{C}_\gamma(w)$, *the batch algorithm only observes a zero reward function* and is unable to distinguish different MDPs using this information.

7.1. Setup: State-Action Space and Feature Extractor

State space The state space $\mathcal{S} = \{s^\star, \overline{s}, s^\dagger\}$ consists of a start state $s^\star$, an intermediate state $\overline{s}$ and a terminal state $s^\dagger$.

Action space In the starting state $s^\star$ the special action $a^\star$ is available in addition to any action $a \in \mathcal{B}^+$. In the intermediate state $\overline{s}$ any action in $\mathcal{B}^+$ is available but not $a^\star$. Finally, in the terminal state $s^\dagger$ only $\vec{0} \in \mathcal{B}^+$ is available. Mathematically $\mathcal{A}_{s^\star} = \mathcal{B}^+ \cup \{a^\star\}$; $\mathcal{A}_{\overline{s}} = \mathcal{B}^+$; $\mathcal{A}_{s^\dagger} = \{\vec{0}\}$.

Feature map The feature map only depends on the action:

$$\forall s \in \mathcal{S}: \ \phi(s,a) = \begin{cases} [\vec{0}, 1] & \text{if } a = a^\star \quad \text{(only available in } s^\star\text{)} \\ [a, 0] & \text{if } a \in \mathcal{B}^+. \end{cases}$$

7.2. Setup: MDP-specific Rewards and Transitions

Every MDP $M \in \mathcal{M}$ is identified by a vector w in the outer portion of the positive orthant $\partial\mathcal{B}^+ = \{x \in \mathcal{B}^+ \mid \|x\|_2 = 1\}$ and by a $\pm$ sign, and is denoted with $M_{w,+}$ or $M_{w,-}$.

Transition function The transition function p_w depends on the vector w that identifies each MDP in the class, *but not on the sign $+$ or $-$*. Fix the MDP by fixing $w \in \partial\mathcal{B}^+$ (two MDPs correspond to a given choice of w). If the agent plays the special action $a^\star$, which is only available in the starting state $s^\star$, it transitions with probability one to the terminal state $s^\dagger$. If the agent plays $a \neq a^\star$, the transition function only depends on the action a (and not on the current state s) and the successor state is $\overline{s}$ with some probability, and is otherwise the absorbing state $s^\dagger$.

Mathematically, if $a = a^\star$ then $p_w(s^\dagger \mid (s^\star, a^\star)) = 1$ and if conversely $a \in \mathcal{B}^+$:

$$\begin{cases} p_w(\overline{s} \mid s, a) & = \min\{(1/\gamma) a^\top w, 1\}, \\ p_w(s^\dagger \mid s, a) & = 1 - p_w(\overline{s} \mid s, a). \end{cases} \tag{2}$$

The definition implies that the successor state is always either $\overline{s}$ or the terminal state $s^\dagger$.

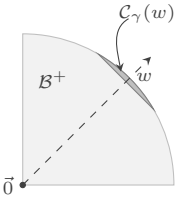

Figure 2. Action space $\mathcal{B}^+$. On $M_{w,-}$ the reward function is zero for any action $a \in \mathcal{B}^+$; on $M_{w,+}$ it is nonzero only inside $\mathcal{C}_\gamma(w)$.

Reward function The reward function $r_{w,\pm}$ depends on both the vector $w \in \partial\mathcal{B}^+$ and on the sign $+$ or $-$ that identifyies the MDP. It is always $\frac{1}{2}$ if the special action $a^\star$ is taken and otherwise it is everywhere 0 on $M_{w,-}$ or is positive only in the spherical cap on $M_{w,+}$. Mathematically:

$$\begin{aligned} &\text{on } M_{w,+}: \ r_{w,+}(s,a) \overset{def}{=} \begin{cases} \frac{1}{2} & \text{if } (s,a) = (s^\star, a^\star) \\ \max\{a^\top w - \gamma, 0\} & \text{otherwise,} \end{cases} \\ &\text{on } M_{w,-}: \ r_{w,-}(s,a) \overset{def}{=} \begin{cases} \frac{1}{2} & \text{if } (s,a) = (s^\star, a^\star) \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

7.3. Proof Sketch of Theorem 2 (Batch Lower Bound)

The steps for the proof are the following: we show that 1) $Q^\star$ is linear 2) policy-induced queries are also policy-free for this problem 3) using less than exponentially many queries ensures that at least one spherical cap $\mathcal{C}_\gamma(\widetilde{w})$ is not probed 4) the corresponding MDPs $M_{\widetilde{w},+}$ and $M_{\widetilde{w},-}$ look the same outside the spherical cap $\mathcal{C}_\gamma(\widetilde{w})$ 5) the agent does not have enough information to distinguish $M_{\widetilde{w},+}$ from $M_{\widetilde{w},-}$.

Realizability By inspection we can verify realizability.

Lemma 1 ($Q^\star$ is Realizable; Lemma 8 in appendix)**.** *For any $w \in \partial\mathcal{B}^+$ let $Q^\star_{w,+}$ and $Q^\star_{w,-}$ be the optimal $Q^\star$ values on $M_{w,+}$ and $M_{w,-}$, respectively. It holds that*

$$\forall (s,a), \quad \begin{cases} Q^\star_{w,+}(s,a) = \phi(s,a)^\top [w, \frac{1}{2}] & \text{on } M_{w,+}, \\ Q^\star_{w,-}(s,a) = \phi(s,a)^\top [\vec{0}, \frac{1}{2}] & \text{on } M_{w,-}. \end{cases}$$

Policy-free vs policy-induced queries Notice that although the dynamics are different for different w's (that identify the MDP), any set T (Definition 2) induces the same set μ of state-actions (possibly with the exception of $(s^\star, a^\star)$ and $(s^\dagger, \vec{0})$) regardless of the vector w and the sign $\pm$. We can therefore consider the case that the oracle has chosen a policy-free query set $\mu = \{(s_i, a_i)\}_{i=1}^n \cup \{(s^\star, a^\star), (s^\dagger, \vec{0})\}$.

Existence of exponentially many spherical caps

Lemma 2 (Follows from Lemma 5 in appendix)**.** *Assume that less than $2^{-d}\mathcal{N}(\gamma, d)$ actions $a_1, \ldots, a_n$ on $\mathcal{B}^+$ are selected. Then there exists a spherical cap $\mathcal{C}_\gamma(\widetilde{w})$ that no action has probed, i.e., $\exists \widetilde{w} \in \partial\mathcal{B}^+ s.t. \forall i \in [n], a_i \notin \mathcal{C}_\gamma(\widetilde{w})$.*

In this case, the agent does not have any information originating from inside the dark gray spherical cap in Fig. 2.

$M_{w,+}$, $M_{w,-}$ are identical outside of the spherical cap If $n < 2^{-d}\mathcal{N}(\gamma, d)$, consider $\widetilde{w}$ given by the above lemma and the two associated MDPs $M_{\widetilde{w},+}$ and $M_{\widetilde{w},-}$. Notice that the transition function (Eq. (2)) is by construction identical on $M_{\widetilde{w},+}$ and $M_{\widetilde{w},-}$ while the reward function (Eq. (3)) is zero at any action $a \notin \mathcal{C}_\gamma(\widetilde{w})$ selected by the oracle on both $M_{w,+}, M_{w,-}$. Then for any $(s,a) \in \{(s_i,a_i)\}_{i=1}^n$ that the batch algorithm receives it holds that

$$r_{\widetilde{w},+}(s,a) = r_{\widetilde{w},-}(s,a) \quad \text{and} \quad p_{\widetilde{w},+}(s,a) = p_{\widetilde{w},-}(s,a).$$

Batch algorithm does not have enough information The above equation implies that the transitions and the rewards in the dataset could have originated from either $M_{\widetilde{w},+}$ or $M_{\widetilde{w},-}$. In $s^\star$, the batch algorithm has two choices to determine the policy $\widehat{\pi}^\star_{\mathcal{D}}$ to return: choose action $a^\star$ and get a total return of $\frac{1}{2}$ or choose an action $a \neq a^\star$. The second choice is $\frac{1}{2}$-suboptimal on $M_{\widetilde{w},-}$, while the first is at least $\frac{1}{2}$-suboptimal on $M_{\widetilde{w},+}$. At best, the batch algorithm can randomize between the two, showing the result.

7.4. Proof Sketch of Theorem 4 (Online Upper Bound)

It remains to exhibit an online algorithm that can solve every problem instance from this MDP class using $d+1$ queries. The algorithm proceeds as follows: 1) it tries to locate the position of the spherical cap by learning the vector w and 2) it probes the spherical cap to learn the $\pm$ sign of the MDP, precisely identifying the MDP.

Identifying the position w of the spherical cap Consider the following adaptive algorithm that submits policy-induced queries. The algorithm first plays the actions $\gamma e_1, \ldots, \gamma e_d$ in $s^\star$ where e_i is the vector of all zeros and 1 in position i (these are d policies that generate trajectories of length one where γe_i is the only action).

Upon receiving the transition functions $p_w(\overline{s} \mid s^\star, \gamma e_i) = \min\{\frac{1}{\gamma}(\gamma e_i)^\top w, 1\} = e_i^\top w$ for all $i \in [d]$ (see Eq. (2)), the agent can determine each component of the vector w. By construction, w identifies the spherical cap $\mathcal{C}_\gamma(w)$.

Identifying the sign $\pm$ of the MDP Next, the algorithm plays the state-action $(s^\star, w)$ to probe the spherical cap and observe the reward ($1-\gamma$ on $M_{w,+}$ and 0 on $M_{w,-}$) which identifies the sign of the MDP. Since the MDP is now precisely identified, the agent can predict the value of any policy, and in particular, it can return the optimal policy.

8. Discussion

This work presents exponential lower bounds for batch RL. In general, models are never correct, observations are noisy, and a batch algorithm needs to return an answer using whatever dataset is available; clearly the lower bounds continue to hold in these more general settings as much as they do when using more general predictors (like neural networks) which contain the linear setting as a special case. As these hard instances can only arise in infinite horizon settings, there is an exponential separation between finite and infinite horizon batch RL.

The strength of our results arise from the 'oracle + batch algorithm' protocol which allows us to derive lower bounds for *every* a priori data distribution; as a special case, we recover the concurrent lower bound of (Wang et al., 2020a) for the infinite horizon setting. We highlight that our lower bounds always imply an infinite sample complexity.

Beyond the exponential lower bounds, an important result is that online exploration may be required to achieve polynomial sample efficiency on certain RL problems. This is surprising, because online RL has the additional exploration burden compared to batch RL with a good dataset.

Finally, this work helps formalize some of the dangers of the deadly triad, which has long been known to cause algorithmic instabilities and divergence of dynamic programming algorithms. In a sense, the bootstrapping problem is for infinite horizon what the extrapolation problem is for finite horizon MDPs (and finite-steps algorithms), but unlike extrapolation, it cannot be mitigated by adding more samples.

Acknowledgment

The author is grateful to Emma Brunskill, Mykel Kochenderfer and Martin Wainwright for providing useful feedback. The author also thanks the reviewers for their helpful and detailed comments. The work was done while the author was visiting the Simons Institute for the Theory of Computing.

References

Agarwal, A., Henaff, M., Kakade, S., and Sun, W. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *arXiv preprint arXiv:2007.08459*, 2020a.

Agarwal, A., Kakade, S., and Yang, L. F. Model-based reinforcement learning with a generative model is minimax optimal. In *Conference on Learning Theory*, pp. 67–83, 2020b.

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. Optimality and approximation with policy gradient methods in markov decision processes. In *Conference on Learning Theory*, pp. 64–66, 2020c.

Amortila, P., Jiang, N., and Xie, T. A variant of the wang-

foster-kakade lower bound for the discounted setting, 2020.

Antos, A., Szepesvári, C., and Munos, R. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

Ayoub, A., Jia, Z., Szepesvari, C., Wang, M., and Yang, L. F. Model-based reinforcement learning with value-targeted regression. *arXiv preprint arXiv:2006.01107*, 2020.

Azar, M., Munos, R., and Kappen, H. J. On the sample complexity of reinforcement learning with a generative model. In *International Conference on Machine Learning (ICML)*, 2012.

Baird, L. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning (ICML)*. 1995.

Bradtke, S. J. and Barto, A. G. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1-3):33–57, 1996.

Chen, J. and Jiang, N. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pp. 1042–1051, 2019.

Cui, Q. and Yang, L. F. Is plug-in solver sample-efficient for feature-based reinforcement learning? *arXiv preprint arXiv:2010.05673*, 2020.

Du, S. S., Kakade, S. M., Wang, R., and Yang, L. F. Is a good representation sufficient for sample efficient reinforcement learning? *arXiv preprint arXiv:1910.03016*, 2019.

Du, S. S., Lee, J. D., Mahajan, G., and Wang, R. Agnostic q-learning with function approximation in deterministic systems: Tight bounds on approximation error and sample complexity, 2020.

Duan, Y. and Wang, M. Minimax-optimal off-policy evaluation with linear function approximation. *arXiv preprint arXiv:2002.09516*, 2020.

Hao, B., Duan, Y., Lattimore, T., Szepesvári, C., and Wang, M. Sparse feature selection makes batch reinforcement learning more sample efficient, 2020.

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 652–661. PMLR, 2016.

Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R. E. Contextual decision processes with low Bellman rank are PAC-learnable. In Precup, D. and Teh, Y. W. (eds.), *International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1704–1713, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/jiang17c.html.

Jin, C., Yang, Z., Wang, Z., and Jordan, M. I. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, 2020.

Lagoudakis, M. G. and Parr, R. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec): 1107–1149, 2003.

Lattimore, T. and Szepesvári, C. *Bandit Algorithms*. Cambridge University Press, 2020.

Lattimore, T., Szepesvari, C., and Weisz, G. Learning with good feature representations in bandits and in RL with a generative model. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5662–5670. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/lattimore20a.html.

Lazaric, A., Ghavamzadeh, M., and Munos, R. Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research*, 13(Oct):3041–3074, 2012.

Li, G., Wei, Y., Chi, Y., Gu, Y., and Chen, Y. Breaking the sample size barrier in model-based reinforcement learning with a generative model. *arXiv preprint arXiv:2005.12900*, 2020.

Li, L., Munos, R., and Szepesvári, C. Toward minimax off-policy value estimation. 2015.

Li, S. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4(1):66–70, 2011.

Liu, Q., Li, L., Tang, Z., and Zhou, D. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pp. 5356–5366, 2018.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020.

Marjani, A. A. and Proutiere, A. Best policy identification in discounted mdps: Problem-specific sample complexity. *arXiv preprint arXiv:2009.13405*, 2020.

Munos, R. Error bounds for approximate policy iteration. In *ICML*, volume 3, pp. 560–567, 2003.

Munos, R. Error bounds for approximate value iteration. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2005.

Munos, R. and Szepesvári, C. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857, 2008.

Precup, D. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

Pukelsheim, F. *Optimal design of experiments*. SIAM, 2006.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994. ISBN 0471619779.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, 2018.

Thomas, P. and Brunskill, E. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 2139–2148, 2016.

Tsitsiklis, J. N. and Van Roy, B. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1-3):59–94, 1996.

Wang, R., Foster, D. P., and Kakade, S. M. What are the statistical limits of offline rl with linear function approximation? *arXiv preprint arXiv:2010.11895*, 2020a.

Wang, R., Salakhutdinov, R., and Yang, L. F. Provably efficient reinforcement learning with general value function approximation, 2020b.

Weisz, G., Amortila, P., and Szepesvári, C. Exponential lower bounds for planning in mdps with linearly-realizable optimal action-value functions. *arXiv preprint arXiv:2010.01374*, 2020.

Wen, Z. and Van Roy, B. Efficient exploration and value function generalization in deterministic systems. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Xie, T. and Jiang, N. Q* approximation schemes for batch reinforcement learning: A theoretical comparison. volume 124 of *Proceedings of Machine Learning Research*, pp. 550–559, Virtual, 03–06 Aug 2020a. PMLR. URL http://proceedings.mlr.press/v124/xie20a.html.

Xie, T. and Jiang, N. Batch value-function approximation with only realizability. *arXiv preprint arXiv:2008.04990*, 2020b.

Xie, T., Ma, Y., and Wang, Y.-X. Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. In *Advances in Neural Information Processing Systems*, pp. 9668–9678, 2019.

Yang, L. F. and Wang, M. Reinforcement leaning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning (ICML)*, 2020.

Yin, M., Bai, Y., and Wang, Y.-X. Near optimal provable uniform convergence in off-policy evaluation for reinforcement learning. *arXiv preprint arXiv:2007.03760*, 2020.

Zanette, A., Brunskill, E., and J. Kochenderfer, M. Almost horizon-free structure-aware best policy identification with a generative model. In *Advances in Neural Information Processing Systems*, 2019a.

Zanette, A., Lazaric, A., J. Kochenderfer, M., and Brunskill, E. Limiting extrapolation in linear approximate value iteration. In *Advances in Neural Information Processing Systems*, 2019b.

Zanette, A., Brandfonbrener, D., Pirotta, M., and Lazaric, A. Frequentist regret bounds for randomized least-squares value iteration. In *AISTATS*, 2020a.

Zanette, A., Lazaric, A., Kochenderfer, M., and Brunskill, E. Learning near optimal policies with low inherent bellman error. In *International Conference on Machine Learning (ICML)*, 2020b.

Zanette, A., Lazaric, A., Kochenderfer, M. J., and Brunskill, E. Provably efficient reward-agnostic navigation with linear value iteration. In *Advances in Neural Information Processing Systems*, 2020c.

Zhou, D., He, J., and Gu, Q. Provably efficient reinforcement learning for discounted mdps with feature mapping. *arXiv preprint arXiv:2006.13165*, 2020.