Computably Continuous Reinforcement-Learning Objectives Are PAC-Learnable

Cambridge Yang[1], **Michael Littman**[2], **Michael Carbin**[1]

[1] MIT
[2] Brown University
camyang@csail.mit.edu, mlittman@cs.brown.edu, mcarbin@csail.mit.edu

Abstract

In reinforcement learning, the classic objectives of maximizing discounted and finite-horizon cumulative rewards are PAC-learnable: There are algorithms that learn a near-optimal policy with high probability using a finite amount of samples and computation. In recent years, researchers have introduced objectives and corresponding reinforcement-learning algorithms beyond the classic cumulative rewards, such as objectives specified as linear temporal logic formulas. However, questions about the PAC-learnability of these new objectives have remained open.

This work demonstrates the PAC-learnability of general reinforcement-learning objectives through sufficient conditions for PAC-learnability in two analysis settings. In particular, for the analysis that considers only sample complexity, we prove that if an objective given as an oracle is uniformly continuous, then it is PAC-learnable. Further, for the analysis that considers computational complexity, we prove that if an objective is computable, then it is PAC-learnable. In other words, if a procedure computes successive approximations of the objective's value, then the objective is PAC-learnable.

We give three applications of our condition on objectives from the literature with previously unknown PAC-learnability and prove that these objectives are PAC-learnable. Overall, our result helps verify existing objectives' PAC-learnability. Also, as some studied objectives that are not uniformly continuous have been shown to be not PAC-learnable, our results could guide the design of new PAC-learnable objectives.

1 Introduction

In reinforcement learning, we situate an agent in an environment with unknown dynamics. The agent acts in the environment by executing its current policy. Executing a policy in an environment induces an infinite-length path of states and actions. We specify an *objective*, a function that maps each possible infinite-length path to a real number—a score—for that path. Moreover, we request the agent learn a good policy that nearly maximizes the expected score over the distribution of paths induced by the environment and the policy.

PAC-learnability of Objectives. The classic reinforcement-learning objectives include infinite-horizon discounted cumulative rewards and finite-horizon cumulative rewards. These objectives are well-studied and have a desirable property: There are reinforcement-learning algorithms that learn a near-optimal policy with high probability with number of samples depending only on the parameters known by the algorithm. We call these algorithms probably approximately correct (PAC), and these objectives *PAC-learnable* under reinforcement learning.

PAC-learnability is essential. Specifically, we aim to specify an objective and let the agent learn a good policy on its own. Thus, we necessarily need some form of assurance of how close to optimal the learned policy is. If an objective is not PAC-learnable, then the hope of ensuring learning a near-optimal policy is lost, and the objective is effectively intractable to learn under reinforcement learning.

General Objectives. In recent years, researchers have introduced various objectives beyond the two classic rewards objectives (Henriques et al. 2012; Sadigh et al. 2014; Littman et al. 2017; Hasanbeig et al. 2019; Hahn et al. 2019; Camacho et al. 2019; Giacomo et al. 2019; Jothimurugan, Alur, and Bastani 2019; Bozkurt et al. 2020; Ronca and De Giacomo 2021). For example:

- Camacho et al. (2019) introduced the reward machine objective. A reward machine augments classic rewards with an automaton that makes the rewards history dependent.
- Bozkurt et al. (2020) introduced an objective based on limit deterministic Buchi Automaton (LDBA).[1] The objective features history-dependent discount factors, history-dependent rewards, and an augmented action space.

Researchers introduced reinforcement-learning algorithms for these objectives and showed that they empirically learn well-behaving policies with finitely many samples.

PAC-learnability of General Objectives. Despite the advances on empirical algorithms for these objectives, not all objectives are PAC-learnable: Recent work (Yang, Littman, and Carbin 2022) proved that infinite-horizon linear temporal logic objectives, a class of general objectives, are not PAC-learnable. Therefore, to the end of having assurance on the outcomes of learning, we desire to understand the PAC-learnability of general objectives.

Some previous work (Ashok, Křetínský, and Weininger 2019; Henriques et al. 2012; Fu and Topcu 2014; Ronca

[1] We summarize works along the same line in Appendix A.

and De Giacomo 2021) address the PAC-learnability of particular objectives. However, these analyses give reinforcement-learning algorithms for particular objectives and do not generalize to other objectives. Previous work (Alur et al. 2021) gave a framework of reductions between objectives whose flavor of generality is most similar to our work; however, they did not give a condition for when an objective is PAC-learnable. To our knowledge, the PAC-learnability of the objectives in Sadigh et al. (2014); Littman et al. (2017); Hasanbeig et al. (2019); Hahn et al. (2019); Camacho et al. (2019); Jothimurugan, Alur, and Bastani (2019); Bozkurt et al. (2020) are not known.

Relevant to model-based reinforcement learning, Bazille et al. (2020) showed that it is impossible to learn the transitions of a Markov chain such that the learned and true models agree on all first-order behaviors. However, this result does not apply to the general reinforcement-learning setting.

We thus raise a research question: *When is a reinforcement-learning objective PAC-learnable?*

Our Approach. We address the question by giving sufficient conditions for PAC-learnability. Specifically, we analyze PAC-learnability in both the information-theoretic setting, that only considers sample complexity, and the computation-theoretic setting, that also considers computability. We prove that, in the information-theoretic setting (resp. computation-theoretic setting), an objective is PAC-learnable if it is uniformly continuous (resp. computable). These conditions simplify proving objectives' PAC-learnability. In particular, our conditions avoid constraints on environments, policies, or reinforcement-learning algorithms but only require reasoning about the objective itself.

We give example applications of our conditions to three objectives in the literature whose PAC-learnability was previously unknown and prove that they are PAC-learnable.

Contributions. We make the following contributions about reinforcement-learning objectives:

- In the information-theoretic setting, we prove that a uniformly continuous objective is PAC-learnable.
- In the computation-theoretic setting, we prove that a computable objective is PAC-learnable.
- We apply our theorem to three objectives (Camacho et al. 2019; Bozkurt et al. 2020; Littman et al. 2017) from the literature whose PAC-learnability was previously unknown and show that they are PAC-learnable.

Our result makes checking the PAC-learnability of existing objectives easier. It also potentially guides the design of new PAC-learnable objectives.

2 Reinforcement Learning and Objectives

This section reviews reinforcement learning and defines general objectives and their learnability.

2.1 Markov Processes

A *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, A, P, s_0)$, where S and A are finite sets of states and actions, $P\colon (S \times A) \to \Delta(S)$ is a transition probability function that maps a current state and an action to a distribution over next states, and $s_0 \in S$ is an initial state. The MDP is sometimes referred to as the *environment MDP* to distinguish it from any specific objective.

A *policy* for an MDP is a function $\pi\colon ((S \times A)^* \times S) \to \Delta(A)$ mapping a history of states and actions to a distribution over actions. [2] An MDP and a policy induce a *discrete-time Markov chain* (DTMC). A DTMC is a tuple $\mathcal{D} = (S, P, s_0)$, where S is the set of states, $P\colon S \to \Delta(S)$ is a transition-probability function mapping states to distributions over next states, and $s_0 \in S$ is an initial state. The DTMC induces a probability space over the infinite-length sequences $w \in S^\omega$.[2]

2.2 Objective

Environment-specific Objective. An *environment-specific objective* for an MDP (S, A, P, s_0) is a measurable function $\kappa\colon (S \times A)^\omega \to \mathbb{R}$. We say such an objective is environment-specific since it is associated with MDPs with a fixed set of states and actions.

The *value* of an environment-specific objective for an MDP $\mathcal{M}$ and a policy π is the expectation of the objective under the probability space of the DTMC $\mathcal{D}$ induced by $\mathcal{M}$ and π: $V^\pi_{\mathcal{M},\kappa} = \mathsf{E}_{w\sim\mathcal{D}}[\kappa(w)]$. We consider only bounded objectives to ensure that the expectation exists and is finite. The *optimal value* is the supremum of the values achievable by all policies: $V^*_{\mathcal{M},\kappa} = \sup_\pi V^\pi_{\mathcal{M},\kappa}$. A policy π is ϵ-optimal if its value is ϵ-close to the optimal value: $V^\pi_{\mathcal{M},\kappa} \geq V^*_{\mathcal{M},\kappa} - \epsilon$.

Environment-generic Objective. An objective defined above is environment-specific because it is associated with a fixed set of states and actions. However, we would also like to talk about objectives in a form decoupled from any MDP. For example, the discounted classical cumulative rewards objective is not bound to any particular reward function. Further, the objective of "reaching the goal state" in a grid world environment is not bound to the size of the grid or the allowed actions. Such decoupling is desirable as it allows one to specify objectives independent of environments.

To that end, we define *environment-generic* objectives. The idea of such objectives is that a *labeling function* interfaces between the environment and the environment-generic objective. The definition decouples an environment-generic objective from environments by requiring different labeling functions for different environments.

Formally, an *environment-generic objective* is a measurable function: $\xi\colon F^\omega \to \mathbb{R}$, where F is a set called *features*. A *labeling function* maps the MDP's (current) states and actions to the features: $\mathcal{L}\colon (S \times A) \to F$. Composing ξ and the element-wise application of $\mathcal{L}$ induces an environment-specific objective. For example, the discounted cumulative rewards objective $\xi\colon \mathbb{Q}^\omega \to \mathbb{R}$ is $\xi(w) = \sum_{i=0}^\infty \gamma^i \cdot w[i]$. For each MDP, the labeling function is a classical reward function $\mathcal{L}\colon (S \times A) \to \mathbb{Q}$. [3]

[2] X^* denotes all finite-length sequences of the elements of X. X^ω denotes all infinite-length sequences of the elements of X.

[3] For simplicity of analysis, we will let objective specifications use rationals instead of reals so that they admit a finite representa-

The value of ξ for an MDP $\mathcal{M}$, a policy π, and a labeling function $\mathcal{L}$ is the value of the environment-specific objective induced by ξ, $\mathcal{M}$, π, and $\mathcal{L}$.

2.3 Learning Model

A reinforcement learning agent has access to a sampler of the MDP's transitions but does know the underlying probability values. The agent learns in two phases: sampling and learning. In the sampling phase, the agent starts from the initial state and follows a sequence of decision steps to collect sampled environment transitions. At every step, the agent may (1) act from the current state to sample the next state or (2) reset to the initial state. In the learning phase, it learns a policy from the collected sampled transitions.

Formally, a reinforcement-learning algorithm is a tuple $(\mathcal{A}^S, \mathcal{A}^L)$, where $\mathcal{A}^S$ is a *sampling algorithm* that drives how the environment is sampled, and $\mathcal{A}^L$ is a learning algorithm that learns a policy from the samples obtained by the sampling algorithm. Let $A_{\texttt{reset}} = A \cup \{\texttt{reset}\}$ be the set of actions with an additional `reset` operation, the sampling algorithm $\mathcal{A}^S\colon ((S \times A_{\texttt{reset}})^* \times S) \to A_{\texttt{reset}}$ maps from sampled transitions to the next operation. The learning algorithm $\mathcal{A}^L\colon ((S \times A_{\texttt{reset}})^* \times S) \to (((S \times A)^* \times S) \to \Delta(A))$ maps from the sampled transitions to the learned policy.

2.4 Learnability of Objectives

A good reinforcement-learning algorithm should learn the optimal policy that maximizes the given objective. In particular, we let the algorithm seek a near-optimal policy with high probability.

Definition 1 (PAC Algorithm for Environment-specific Objective)**.** Given an objective κ, a reinforcement-learning algorithm $(\mathcal{A}^{\mathsf{S}}, \mathcal{A}^{\mathsf{L}})$ is κ-PAC (probably approximately correct for objective κ) in an environment MDP $\mathcal{M}$ with N samples if, with the sequence of transitions T of length N sampled using the sampling algorithm $\mathcal{A}^{\mathsf{S}}$, the learning algorithm $\mathcal{A}^{\mathsf{L}}$ outputs an ϵ-optimal policy with probability at least $1-\delta$ for any given $\epsilon > 0$ and $0 < \delta < 1$. That is:

$$\mathsf{P}_{\mathsf{T}\sim\langle\mathcal{M},\mathcal{A}^{\mathsf{S}}\rangle_N}\left(V^{\mathcal{A}^{\mathsf{L}}(T)}_{\mathcal{M},\kappa} \geq V^{*}_{\mathcal{M},\kappa} - \epsilon\right) \geq 1 - \delta.$$

We use $\mathsf{T}\sim\langle\mathcal{M},\mathcal{A}^{\mathsf{S}}\rangle_N$ to denote that the probability space is over the set of length-N transition sequences sampled from the environment $\mathcal{M}$ using the sampling algorithm $\mathcal{A}^{\mathsf{S}}$. We will simply write $\mathsf{P}_{\mathsf{T}}(.)$ when it is clear from the context.

We will consider two settings: the information-theoretic setting that considers only sample complexity and the computation-theoretic setting that considers computability.

Definition 2 (PAC-learnable Environment-specific Objective)**.** In the information-theoretic setting (resp. computation-theoretic setting), an environment-specific objective κ is *κ-PAC-learnable* if there exists a function $C\colon (\mathbb{R} \times \mathbb{R} \times \mathbb{N} \times \mathbb{N}) \to \mathbb{N}$ such that, for all consistent environment MDPs for κ (i.e., the domain of κ uses the same set of states and actions as the MDP), there exists a κ-PAC reinforcement-learning algorithm with less than $C(\frac{1}{\epsilon}, \frac{1}{\delta}, |S|, |A|)$ samples (resp. computation steps).

Our definition focuses on the core tractability issue. Failure to respect our definition implies that PAC-learning is not achievable with finitely many samples (in the information-theoretic setting) or not computable (in the computation-theoretic setting). To that end, we have set the parameters of C to be the only quantities available to an algorithm under the standard assumptions of reinforcement learning. Specifically, since the transition dynamics are unknown, they are not parameters of C. Moreover, while some variants of PAC-learnability require C to be a polynomial to capture the notion of learning efficiency, we have dropped this requirement to focus on the core tractability issue.

We also define the PAC-learnability of environment-generic objectives, for both information- and computation-theoretic settings:

Definition 3 (PAC-learnable Environment-generic Objective)**.** An environment-generic objective ξ is *ξ-PAC-learnable* if for all labeling functions $\mathcal{L}$, the objective κ induced by ξ and $\mathcal{L}$ is κ-PAC-learnable.

Note that in the information-theoretic setting, we assume that the objectives κ and ξ are given as oracles: they take infinite-length inputs and return infinite-precision output with no computation overhead.

2.5 Established PAC-Learnable Objectives

The standard discounted cumulative rewards objective $\sum_{i=0}^{\infty} \gamma^i w[i]$ and the finite-horizon cumulative rewards objective $\sum_{i=0}^{H} w[i]$ are known to be PAC-learnable. The folklore intuition is that these objectives "effectively terminate" in an expected finite-length horizon, and rewards farther out of the horizon diminish quickly. This paper formalizes this intuition by connecting it to the standard definition of the objective function's uniform continuity and computability. Later in Section 4, we will prove that uniformly continuous and computable objectives are PAC-learnable.

3 Example: The Reward Machine Objective

This section gives an example general objective: the simple reward machine objective (Camacho et al. 2019). We will later use this objective as one of the examples to apply our core theorem and prove its PAC-learnability.

The Simple Reward Machine Specification. Simple reward machines generalize from classic Markovian rewards to non-Markovian rewards. In particular, a simple reward machine is a kind of deterministic finite automaton. Each automaton transition has a reward value and a tuple of truth values for a set of propositions about the environment's state. The simple reward machine starts from an initial state. As the agent steps through the environment, a labeling function classifies the environment's current state to a tuple of truth values of a set of propositions. The simple reward machine then transits to the next states according to the tuple. During each transition, the agent collects a scalar reward

tion. Nonetheless, our analyses also generalize to objective specifications that contain reals.

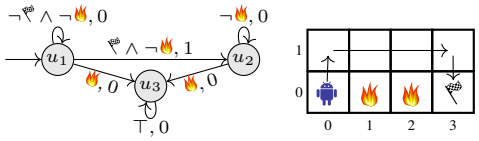

Figure 1: Left: simple reward machine. Right: environment.

along the transition of the simple reward machine. The overall objective is to maximize the γ-discounted sum of collected rewards. The formal definition of a simple reward machine given by Camacho et al. (2019) is:

Definition 4 (Simple Reward Machine)**.** Given a finite set Π called the propositions, a simple reward machine over Π is a tuple $(U, \delta_u, \delta_r, u_0, \gamma)$, where U is a finite set of states, $\delta_u \colon (U \times 2^\Pi) \to U$ is a deterministic state transition function, $\delta_r \colon (U \times U) \to \mathbb{Q}$ is a deterministic reward function, u_0 is an initial state, and $\gamma \in \mathbb{Q}$ is a discount factor.

Figure 1 shows an example simple reward machine and an accompanying grid environment. The states of the simple reward machine are $\{u_1, u_2, u_3\}$. The labeling function for this particular environment maps grid locations $(1, 0)$ and $(2, 0)$ to "fire" ( ) and $(3, 0)$ to "goal" ( ). Each transition of the simple reward machine is labeled by a tuple of truth values of the two propositions ("fire" and "goal"). For example, u_1 transits to u_2 and produces a reward of 1 if the environment's current state is labeled as "goal" but not "fire".

The Simple Reward Machine Objective. Formally, a simple reward machine R specifies an environment-generic objective $[\![R]\!] \colon \left(2^\Pi\right)^\omega \to \mathbb{R}$ given by:

$$[\![R]\!](w) = \sum_{k=1}^{\infty} \gamma^k \delta_r(u_k, u_{k+1}), \quad \forall k \geq 0.\, u_{k+1} = \delta_u(u_k, w[k]).$$

The set of features F corresponding to this environment-generic objective is the possible truth values of the propositions Π, that is $F = 2^\Pi$. The labeling function classifies each environment's current state and action to these features.

4 Condition for PAC-Learnability

This section presents our main result: sufficient conditions for an objective's learnability. The first two subsections analyze learnability in the information-theoretic setting. Specifically, we show that an objective given as an oracle is PAC-learnable if it is uniformly continuous. The next two subsections analyze learnability in the computation-theoretic setting. Specifically, using a standard result from computational analysis, we show that a computable objective is PAC-learnable. Appendix E complements our result by showing that our conditions are sufficient but not necessary.

4.1 Uniform Continuity

We first recall the following standard definition of a uniformly continuous function.

Definition 5 (Uniformly Continuous Function)**.** A function $f \colon X \to Y$ with metric spaces (X, d_X) and (Y, d_Y) is uniformly continuous if, for any $\epsilon > 0$, there exists $\delta > 0$ so that f maps δ-close elements in the domain to ϵ-close elements in the image [4] :

$$\forall \epsilon > 0. \exists \delta > 0. \forall x_1 \in X. \forall x_2 \in X :$$
$$d_X(x_1, x_2) \leq \delta \Rightarrow d_Y(f(x_1), f(x_2)) \leq \epsilon.$$

To specialize the above definition to an objective, we next note the metric space of the domain of an objective. An objective's domain is the set of infinite-length sequences X^ω, where $X = (S \times A)$ for an environment-specific objective and $X = F$ for an environment-generic objective. The domain forms a metric space by the standard distance function $d_{X^\omega}(w_1, w_2) = 2^{-L_{\text{prefix}}(w_1, w_2)}$, where $L_{\text{prefix}}(w_1, w_2)$ is the length of the longest common prefix of w_1 and w_2 (Manna and Pnueli 1987). We are now ready to specialize the definition of uniform continuity to objectives.

Definition 6 (Uniformly Continuous Objective)**.** An objective (environment-specific or environment-generic) $f \colon X^\omega \to \mathbb{R}$ is uniformly continuous if, for any $\epsilon > 0$, there exists a finite horizon H so that the objective maps all infinite-length sequences sharing the same prefix of length H to ϵ-close values:

$$\forall \epsilon > 0. \exists H \in \mathbb{N}. \forall w \in X^\omega. \forall w' \in X^\omega :$$
$$L_{\text{prefix}}(w, w') \geq H \Rightarrow |f(w) - f(w')| \leq \epsilon.$$

Note that since the domain of an objective is compact, the Heine–Cantor theorem guarantees that a continuous objective is also uniformly continuous. This paper only uses uniform continuity since it is more relevant to our theorem and proof. Nonetheless, theorems presented in the following section also hold for continuous objectives.

4.2 Continuity Implies PAC-learnability

Environment-specific Objectives. We give a sufficient condition for a learnable environment-specific objective:

Theorem 1. *An environment-specific objective κ is κ-PAC-learnable in the information-theoretic setting if it is uniformly continuous.*

We will prove the theorem by constructing a κ-PAC reinforcement-learning algorithm for any uniformly continuous κ. To that end, we reduce κ to a finite-horizon cumulative rewards problem; we then prove the theorem by invoking an existing PAC reinforcement-learning algorithm for finite-horizon cumulative rewards problems.

Proof of Theorem 1. For any $\epsilon' > 0$, since κ is uniformly continuous, there exists a bound H such that infinite-length sequences sharing a length-H prefix are all mapped to ϵ'-close values.

For concreteness, let us pick any $\dot{s} \in S$ and any $\dot{a} \in A$. For each length-H sequence $u \in (S \times A)^H$, we pick the representative infinite-length sequence $[u; (\dot{s}, \dot{a})^\omega]$ that starts with the prefix u and ends in an infinite repetition of $(\dot{s}, \dot{a})$. Using these representatives, we construct a finite-horizon rewards objective $\tilde{\kappa}_{\epsilon'}$ of horizon H. The construction assigns each infinite-length sequence with the value of the original

[4] Note that textbook definitions commonly use $<$ instead of $\leq$: our definition is equivalent. We use $\leq$ to match with the comparison operators in the PAC definitions.

κ at the chosen representative. That is, let $w[:H]$ denote the length-H prefix of w, we define $\tilde{\kappa}_{\epsilon'}$ as:

$$\tilde{\kappa}_{\epsilon'}(w) \triangleq \kappa([w[:H]; (\dot{s}, \dot{a})^\omega]),\ \forall w \in (S \times A)^\omega.$$

By construction, $\tilde{\kappa}_{\epsilon'}$ is ϵ'-close to κ, meaning that for any infinite-length input, their evaluations differ by at most ϵ':

$$|\tilde{\kappa}_{\epsilon'}(w) - \kappa(w)| \leq \epsilon',\ \forall w \in (S \times A)^\omega.$$

Thus, an ϵ'-optimal policy for $\tilde{\kappa}_{\epsilon'}$ is $2\epsilon'$-optimal for κ.

We then reduce the approximated objective $\tilde{\kappa}_\epsilon$, which assigns a history-dependent reward at the horizon H, into a finite-horizon cumulative rewards objective, which assigns a history-independent reward at each step. To that end, we lift the state space to $U = \bigcup_{t=1}^{H} (S \times A)^t$. Each state $u_t = (S \times A)^t$at step t in the lifted state space is the length-t history of states and actions encountered in the environment. For any state before step H, we assign a reward of zero. For any state $u_H = (S \times A)^H$ at step H, we assign a reward of $\tilde{\kappa}_\epsilon([u_H; (\dot{s}, \dot{a})^\omega])$. The lifted state space and the history-independent reward function above form the desired finite-horizon cumulative rewards problem.

Dann et al. (2019) introduced ORLC, a PAC reinforcement-learning algorithm for finite-horizon cumulative rewards problems.[5] Applying ORLC to the above finite-horizon cumulative rewards problem produces an $2\epsilon'$-optimal policy for κ. Finally, for any ϵ, choosing $\epsilon' = \frac{\epsilon}{2}$ gives a κ-PAC reinforcement-learning algorithm for κ. □

Environment-generic Objectives. Theorem 1 states a sufficient condition for when an environment-specific objective is PAC-learnable. The following corollary generalizes the condition to environment-generic objectives.

Corollary 2. *An environment-generic objective ξ is ξ-PAC-learnable in the information-theoretic setting if ξ is uniformly continuous.*

To the end of proving Corollary 2, we first observe the following lemma, which we prove in Appendix B.

Lemma 3. *If an environment-generic objective is uniformly continuous, then, for all labeling functions, the induced environment-specific objective is also uniformly continuous.*

With Lemma 3, Corollary 2 is straightforward. Since each induced environment-specific objective κ is uniformly continuous, each κ is κ-PAC-learnable by Theorem 1. Thus, the objective ξ is ξ-PAC-learnable by definition.

4.3 Computability

We now define the computability of an objective $f\colon X^\omega \to \mathbb{R}$. The standard definition of computability of such functions depends on Type-2 Turing machines (Weihrauch 2000, Chapter 2, Definition 2) and a representation of the reals by an infinite sequence of rational approximations, called the Cauchy-representation (Weihrauch 2000, Chapter 3, Definition 3). Informally, a Type-2 Turing machine is a Turing machine with an infinite-length input tape and a one-way infinite-length output tape. The machine reads the input tape and computes forever writing to the output tape.

[5]ORLC provides an individual policy certificates (IPOC) guarantee. Dann et al. showed that IPOC implies our PAC definition, which they called "supervised-learning style PAC".

Definition 7 (Computable Objective)**.** An objective f is computable if a Type-2 Turing machine reads w from the input tape and writes to the output tape a fast-converging Cauchy sequence $[q_0, q_1, \dots] \in \mathbb{Q}^\omega$ of rational approximations to $f(w)$, that is: $\forall n \in \mathbb{N}, |f(w) - q_n| \leq 2^{-n}$.

When proving computability, this definition is tedious to work with since it requires implementing the function on a Turing machine. Instead, we will use pseudocode to formulate an algorithm that takes in an infinite-stream input w and a natural number n and outputs the n-th rational approximation q_n. Repeatedly invoking the algorithm by enumerating n produces the Cauchy sequence of rational approximations.

A classic result in computable analysis is that computable functions are continuous (Weihrauch 2000, Theorem 2.5 and 4.3). Since an objective's domain is compact, by the Heine-Cantor theorem, this result also holds for uniform continuity. Even stronger, the following theorem, modified from Weihrauch (2000, Theorem 6.4) for our context, guarantees that for a computable objective, for any rational $\epsilon > 0$, we can compute a horizon H that satisfies the definition of uniform continuity. Define *the modulus of continuity* of an objective as a function $m\colon \mathbb{Q} \to \mathbb{N}$ that satisfies $\forall \epsilon \in \mathbb{Q}, \forall w_1 \in X^\omega, \forall w_2 \in X^\omega : L_{\text{prefix}}(w_1, w_2) \geq m(\epsilon) \Rightarrow |f(w_1) - f(w_2)| \leq \epsilon$. Then, we have:

Theorem 4. *A computable objective is uniformly continuous. Further, its modulus of continuity m is computable.*

For completeness, Appendix C gives pseudocode that computes the modulus of continuity for any computable objective specified by the interface described above.

4.4 Computability Implies PAC-learnability

We now extend our result in Section 4.2 from the information-theoretic to the computation-theoretic setting.

Theorem 5. *An (environment-generic or environment-specific) objective f is f-PAC-learnable in the computation-theoretic setting if f is computable.*

Proof. Combining theorems in Section 4.2 and Theorem 4, a computable objective f is uniformly continuous, therefore f-PAC-learnable in the information-theoretic setting. In the computational-theoretic setting, we need to further construct a computable reinforcement-learning algorithm. Note that our proof of Corollary 2 is already constructive of an algorithm. However, we need to:

- compute the bound H from the given ϵ' and
- computably evaluate the approximated objective $\tilde{\kappa}_{\epsilon'}$.

A computable objective resolves both points:

- By Theorem 4, the bound H is computable for any ϵ.
- Evaluating the approximate objective is computable, since the approximated objective only depends on the length-H prefix of the input. □

Appendix D provides pseudocode for an f-PAC reinforcement-learning for any computable objective f.

5 Theorem Applications

This section applies the core theorem and corollary to two objectives in the existing literature and proves each objective's PAC-learnability. Due to space, we give the third objective from Littman et al. (2017) and prove its PAC-learnability in Appendix G.

5.1 Reward Machine

Proof of PAC-learnability. We prove that the reward-machine objective reviewed in Section 3 is PAC-learnable.

Proposition 6. *The objective $[\![R]\!]$ of a simple reward machine R is $[\![R]\!]$-PAC-learnable.*

Proof. By Theorem 5, it is sufficient to show that a simple reward-machine objective is computable. Consider the pseudocode with Python-like syntax in Listing 1.

Listing 1 defines an algorithm for computing the simple reward-machine objective. It first initializes the state variable `u` to the initial state u_0. It then computes a horizon $H = (\lfloor \log_2(1-\gamma) \rfloor - n - \lceil \log_2 r_{\max} \rceil) / \lceil \log_2 \gamma \rceil$, where $r_{\max} = \max(|\delta_r(\cdot)|)$ is the maximum magnitude of all possible rewards. It iterates through the first H indices of the input and transits the reward machine's state according to the transitions δ_u. For each input w and n, the algorithm accumulates the discounted cumulative rewards truncated to the first H-terms: $\sum_{k=0}^{H-1} \gamma^k \delta_r(u_k, u_{k+1})$.

By definition of a computable objective, we need to show that the returned values for all n form a fast-converging Cauchy sequence: $\forall n \in \mathbb{N}, |\texttt{SRM}(w,n) - [\![R]\!](w)| \leq 2^{-n}$. To see this, let $\Delta \triangleq |\texttt{SRM}(w,n) - [\![R]\!](w)| = \left|\sum_{k=H}^{\infty} \gamma^k \delta_r(u_k, u_{k+1})\right|$. Then, we have $\Delta \leq {r_{\max} \cdot \gamma^H}/{1-\gamma}$ by upper bounding the rewards by $r_{\max}$, then simplifying the infinite sum into a closed form. By plugging in the value of H and simplifying the inequality, we have $\Delta \leq 2^{-n}$. Thus, the objective is computable and $[\![R]\!]$-PAC-learnable. □

Listing 1: Computation of the simple reward objective

```
# Given reward machine (U, δ_u, δ_r, u_0, γ)
def SRM(w: (2^Π)^ω, n: ℕ) → ℚ:
 u: U, value: ℚ = u_0, 0
 rmax: ℚ = max(abs(δ_r(u1, u2))
                   for u1, u2 in U^2))
 H: ℕ = (log2floor(1 - γ) - n
         - log2ceil(rmax)) / log2ceil(γ)
 for k in range(H):
    u' = δ_u(u, w[k])
    value += γ**k * δ_r(u, u')
    u = u'
 return value
```

5.2 LTL-in-the-limit Objectives

Linear temporal logic (LTL) objectives are measurable Boolean objectives that live in the first two-and-half levels of the Borel hierarchy (Manna and Pnueli 1987). Various works (Hasanbeig et al. 2019; Hahn et al. 2019; Bozkurt et al. 2020) considered LTL objectives for reinforcement learning and empirical algorithms for learning. A common pattern of these algorithms is that they convert a given LTL formula to an intermediate specification that takes in additional hyper-parameters. They show that in an unreachable limit of these hyper-parameters, the optimal policy for this intermediate specification becomes the optimal policy for the given LTL formula. We call such intermediate specifications *LTL-in-the-limit specifications*. Due to space, we will focus on Bozkurt et al. (2020) and give the objective specified by their LTL-in-the-limit-specification. We will show that this objective is PAC-learnable. The same process, namely writing down the LTL-in-the-limit specification and then proving that the specified objective is PAC-learnable, also applies to the approaches in Sadigh et al. (2014); Hasanbeig et al. (2019); Hahn et al. (2019).

Bozkurt et al.'s LTL-in-the-limit Specification. Given an LTL formula, Bozkurt et al. first convert the formula into a *limit deterministic Buchi Automaton (LDBA)* by a standard conversion algorithm (Sickert et al. 2016) with two additional discount factor parameters. An LDBA is a nondeterministic finite automaton. It is bipartite by two sets of states, those in an *initial component* and those in an *accepting component*. Transitions in the automaton can only go from the initial component to the accepting component, but not the reverse. An LDBA is "deterministic in the limit": it only has non-deterministic ϵ-transitions in the initial component, but it is deterministic in the accepting component. The formal definition of LDBA is:

Definition 8 (LDBA)**.** For an LTL formula over propositions Π, an LDBA converted from the formula is a tuple $(U, \mathcal{E}, \delta_u, u_0, B)$, where U is a finite set of states, $\delta_u \colon \left(U \times (2^\Pi \cup \{\epsilon\})\right) \to 2^U$ is a non-deterministic transition function, u_0 is an initial state, and $B \subseteq U$ is a set of accepting states. Additionally, U has a bi-partition of an initial component with states U_I and an accepting component with states U_B. An LDBA satisfies the conditions: (1) $\delta_u(u, \epsilon) = \emptyset$ for all $u \in U_B$, (2) $\delta_u(u, 2^\Pi) \subseteq U_B$ for all $u \in U_B$, and (3) $B \subseteq U_B$.

The agent and environment models are similar to a simple reward machine: At each step, the agent chooses an environment's action and steps in the environment. A labeling function classifies the current state of the environment to a tuple of truth values of the set of propositions Π.

At each step, an LDBA takes either a non-deterministic ϵ-transitions (if such transition is available) or the transition along the tuple of the truth values of the propositions. Each time the LDBA enters an accepting state, the agent receives a reward of $1-\gamma_1$, and discounts all future rewards by γ_1. Each time the LDBA enters a non-accepting state, the agent receives no reward and discounts all future rewards by γ_2. An oracle controls the ϵ-transitions. In words, the objective is to maximize the (state-dependent) discounted cumulative rewards, assuming the oracle always makes the optimal choice that helps to maximize the cumulative rewards.

Bozkurt et al.'s LTL-in-the-limit Objective. Bozkurt et al.'s LTL-in-the-limit specification is a tuple (L, γ_1, γ_2): the LDBA L and the two hyper-parameters $\gamma_1, \gamma_2 \in \mathbb{Q}$. It specifies an environment-generic objective $[\![(L, \gamma_1, \gamma_2)]\!] \colon (2^\Pi)^\omega \to \mathbb{R}$. Let $\mathcal{E}^+ = \mathcal{E} \cup \{\bot\}$, where $\mathcal{E}$

is the set of ϵ-transitions and $\perp$ is a non-ϵ-transition (i.e., following a transition with a tuple classified by the labeling function), the objective is:

$$
\begin{aligned}
&[\![(L,\gamma_1,\gamma_2)]\!](w) = \max_{w_{\mathcal{E}} \in \mathcal{E}^\omega} g(w_{\mathcal{E}}, w) \quad \text{where} \\
&g(w_{\mathcal{E}}, w) = \sum_{i=1}^{\infty} R(u_i) \prod_{j=1}^{i-1} \Gamma(u_j), \\
&R(u) = (1-\gamma_1)\mathbb{1}\{u \in B\}, \\
&\Gamma(u) = \gamma_1 \mathbb{1}\{u \in B\} + \gamma_2 \mathbb{1}\{u \notin B\}, \\
&\forall k \geq 0 \colon u_{k+1} = \delta_u(u_k, w_k^+), \\
&\quad t_k = \sum_{i=1}^{k} \mathbb{1}\{w_{\mathcal{E}}[i] = \perp \text{ or } (u_k, w_{\mathcal{E}}[i]) \notin \delta_u\}, \\
&\quad w_k^+ = \begin{cases} w[t_k] & \text{if } w_{\mathcal{E}}[k] = \perp \text{ or } (u_k, w_{\mathcal{E}}[k]) \notin \delta_u \\ w_{\mathcal{E}}[k] & \text{otherwise} \end{cases}
\end{aligned}
\quad . \tag{1}
$$

Here, t_k is the step count of the environment when the LDBA takes its k-th step. Note that $t_k \leq k$, since the environment does not step when LDBA takes an ϵ-transition. The value $w[t_k]$ is the tuple of truth values of the input infinite-length sequence w at t_k. We define w_k^+ as the transition label taken by the LDBA at the k-th step: It is either (1) a tuple of truth values $w[k]$, if $w_{\mathcal{E}}[k]$ is a non-ϵ-transition or an ϵ-transition that is not available from the current LDBA state u_k, or (2) the ϵ-transition $w_{\mathcal{E}}[k]$. By its definition, w_k^+ is always a valid transition of the LDBA, and it always leads to a deterministic next state. Therefore, we write $u_{k+1} = \delta_u(u_k, w_k^+)$ to denote that the LDBA state u_{k+1} follows this deterministic transition to the next state.

Proof of PAC-learnability. We now prove that the objective specified by an LTL-in-the-limit specification in Bozkurt et al. (2020) is PAC-learnable. Although this section aims to show an example, as we mentioned, the proof strategy here also applies to the approaches in Sadigh et al. (2014); Hasanbeig et al. (2019); Hahn et al. (2019).

Proposition 7. *Bozkurt et al.'s LTL-in-the-limit objective* $[\![(L,\gamma_1,\gamma_2)]\!]$ *is* $[\![(L,\gamma_1,\gamma_2)]\!]$*-PAC-learnable.*

Proof. By Theorem 5, it is sufficient to show that Bozkurt et al.'s objective is computable. Consider the pseudocode with Python-like syntax in Listing 2.

Listing 2 gives pseudocode for computing Bozkurt et al.'s objective. The pseudocode contains two procedures. The procedure `bozkurt_helper` computes g but truncates the sum to the first $H = (\lfloor \log_2(1 - \max(\gamma_1,\gamma_2) \rfloor - n) \,/\, \lceil \log_2 \max(\gamma_1,\gamma_2) \rceil$ terms. The procedure `bozkurt_objective` then computes the n-th rational approximation of the objective's value. It invokes the helper function for all $\hat{w}_\epsilon \in \mathcal{E}^n$ and calculates the value of $\max_{\hat{w}_\epsilon \in \mathcal{E}^n} \texttt{bozkurt\_helper}(\hat{w}_\epsilon, w, n)$.

Appendix F proves that the return values of `bozkurt_objective` for all $n \in \mathbb{N}$ form a fast-converging Cauchy sequence:

$$|\texttt{bozkurt\_objective}(w,n) - [\![(L,\gamma_1,\gamma_2)]\!](w)| \leq 2^{-n}.$$

Therefore, the objective is computable and consequently $[\![(L,\gamma_1,\gamma_2)]\!]$-PAC-learnable. □

Listing 2: Computation of Bozkurt et al.'s objective

```
# Given LDBA (U, E, δ_u, u_0, B) and γ_1, γ_2
def bozkurt_objective(w: (2^Π)^ω, n: ℕ) → ℚ:
  gamma_max: ℚ  = max(γ_1, γ_1)
  H: ℕ  = (log2floor(1 - gamma_max) - n)
            / log2ceil(gamma_max)
  v: ℚ = 0
  for w_e in E^H:
    v = max(v, bozkurt_helper(H, w_e, w))
  return v

def bozkurt_helper(H: ℕ, w_e: E^H, w: S^ω)→ ℚ:
  v: ℚ, u: U, discount: ℚ  = 0, u_0, 1
  for k in range(H):
    if u in B:
      reward, gamma = 1, γ_1
    else:
      reward, gamma = 0, γ_2
    v += reward * discount
    discount *= gamma
    if w_e[k] == ⊥ or (u, w_e[k]) not in δ_u:
      w_k_plus = w[k]
    else:
      w_k_plus = we[k]
    u = δ_u(u, w_k_plus)
  return v
```

6 Conclusion

This work studies the PAC-learnability of general reinforcement-learning objectives and gives the first sufficient condition of PAC-learnability of an objective. We use examples to show the applicability of our condition on various existing objectives whose learnability were previously unknown.

Applications to Existing Objectives. Although we only demonstrated three examples, our theorem also applies to other objectives in the literature. Some examples are (1) modifications to the simple reward machine such as the (standard) reward machine (Camacho et al. 2019) (where rewards depend on not only the reward machine's state but also the environment's state) and the stochastic reward machine (Corazza, Gavran, and Neider 2022), (2) other LTL-in-the-limit objectives (Sadigh et al. 2014; Hahn et al. 2019; Hasanbeig et al. 2019), and (3) various finite-horizon objectives (Henriques et al. 2012; Jothimurugan, Alur, and Bastani 2019; Giacomo et al. 2019).

Moreover, we gave an example objective in Appendix E showing our condition is sufficient but not necessary. However, to our knowledge, no previous objective has a similar pattern to our example. Therefore, we conjecture that our condition applies to most, if not all, existing PAC-learnable objectives in the literature. Nonetheless, verifying each objective's PAC-learnability is out of scope of this work.

Guiding The Design of New Objectives. Our main result could also help the design of new objectives. With our sufficient condition, researchers can create continuous and computable objectives by design, and our condition will ensure the PAC-learnability of such objectives.

Acknowledgments

We thank Eric Atkinson, Ellie Cheng, Tian Jin, Dongdong Li, Jesse Michel, Alex Renda, and the anonymous reviewers for their helpful comments and suggestions. This research is partly supported by Naval Research (ONR N00014-17-1-2699) and National Science Foundation (CCF-1918839).

References

Alur, R.; Bansal, S.; Bastani, O.; and Jothimurugan, K. 2021. A Framework for Transforming Specifications in Reinforcement Learning. *arXiv preprint: 2111.00272*.

Ashok, P.; Křetínský, J.; and Weininger, M. 2019. PAC Statistical Model Checking for Markov Decision Processes and Stochastic Games. In *Computer Aided Verification*.

Bazille, H.; Genest, B.; Jegourel, C.; and Sun, J. 2020. Global PAC Bounds for Learning Discrete Time Markov Chains. In *Computer Aided Verification*.

Bozkurt, A.; Wang, Y.; Zavlanos, M.; and Pajic, M. 2020. Control Synthesis from Linear Temporal Logic Specifications using Model-Free Reinforcement Learning. In *International Conference on Robotics and Automation*.

Camacho, A.; Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *The International Joint Conference on Artificial Intelligence*.

Corazza, J.; Gavran, I.; and Neider, D. 2022. Reinforcement Learning with Stochastic Reward Machines. In *aaai*.

Dann, C.; Li, L.; Wei, W.; and Brunskill, E. 2019. Policy Certificates: Towards Accountable Reinforcement Learning. In *International Conference on Machine Learning*.

Fu, J.; and Topcu, U. 2014. Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints. In *Robotics: Science and Systems X*.

Giacomo, G. D.; Iocchi, L.; Favorito, M.; and Patrizi, F. 2019. Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications. In *International Conference on Automated Planning and Scheduling*.

Hahn, E. M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2019. Omega-Regular Objectives in Model-Free Reinforcement Learning. In *Tools and Algorithms for the Construction and Analysis of Systems*.

Hasanbeig, M.; Kantaros, Y.; Abate, A.; Kroening, D.; Pappas, G.; and Lee, I. 2019. Reinforcement Learning for Temporal Logic Control Synthesis with Probabilistic Satisfaction Guarantees. In *Conference on Decision and Control*.

Henriques, D.; Martins, J. G.; Zuliani, P.; Platzer, A.; and Clarke, E. M. 2012. Statistical Model Checking for Markov Decision Processes. In *International Conference on Quantitative Evaluation of Systems*.

Jothimurugan, K.; Alur, R.; and Bastani, O. 2019. A Composable Specification Language for Reinforcement Learning Tasks. In *Neural Information Processing Systems*.

Littman, M. L.; Topcu, U.; Fu, J.; Isbell, C.; Wen, M.; and MacGlashan, J. 2017. Environment-Independent Task Specifications via GLTL. *arXiv preprint: 1704.04341*.

Manna, Z.; and Pnueli, A. 1987. A Hierarchy of Temporal Properties. In *Symposium on Principles of Distributed Computing*.

Ronca, A.; and De Giacomo, G. 2021. Efficient PAC Reinforcement Learning in Regular Decision Processes. In *The International Joint Conference on Artificial Intelligence*.

Sadigh, D.; Kim, E. S.; Coogan, S.; Sastry, S. S.; and Seshia, S. A. 2014. A Learning Based Approach to Control Synthesis of Markov Decision Processes for Linear Temporal Logic Specifications. In *Conference on Decision and Control*.

Sickert, S.; Esparza, J.; Jaax, S.; and Křetínský, J. 2016. Limit-Deterministic Büchi Automata for Linear Temporal Logic. In *Computer Aided Verification*.

Weihrauch, K. 2000. *Computable Analysis: An Introduction*. Springer Science & Business Media.

Yang, C.; Littman, M.; and Carbin, M. 2022. On the (In)Tractability of LTL Objectives for Reinforcement Learning. In *The International Joint Conference on Artificial Intelligence*.