On Reinforcement Learning for Turn-based Zero-sum Markov Games

Devavrat Shah
devavrat@mit.edu
LIDS, MIT

Varun Somani
vs472@cornell.edu
ORIE, Cornell University

Qiaomin Xie
qiaomin.xie@cornell.edu
ORIE, Cornell University

Zhi Xu
zhixu@mit.edu
LIDS, MIT

ABSTRACT

We consider the problem of finding Nash equilibrium for two-player turn-based zero-sum games. Inspired by the AlphaGo Zero (AGZ) algorithm [26], we develop a Reinforcement Learning based approach. Specifically, we propose Explore-Improve-Supervise (EIS) method that combines "exploration", "policy improvement" and "supervised learning" to find the value function and policy associated with Nash equilibrium. We identify sufficient conditions for convergence and correctness for such an approach. For a concrete instance of EIS where random policy is used for "exploration", Monte-Carlo Tree Search is used for "policy improvement" and Nearest Neighbors is used for "supervised learning", we establish that this method finds an ε-approximate value function of Nash equilibrium in $\widetilde{O}(\varepsilon^{-(d+4)})$ steps when the underlying state-space of the game is continuous and d-dimensional. This is nearly optimal as we establish a lower bound of $\widetilde{\Omega}(\varepsilon^{-(d+2)})$ for any policy.

KEYWORDS

Reinforcement Learning, Markov Games, Sample Complexity

ACM Reference Format:
Devavrat Shah, Varun Somani, Qiaomin Xie, and Zhi Xu. 2020. On Reinforcement Learning for Turn-based Zero-sum Markov Games. In *Proceedings of the 2020 ACM-IMS Foundations of Data Science Conference (FODS '20), October 19–20, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3412815.3416888

1 INTRODUCTION

In 2016, AlphaGo [24] became the first program to defeat the world champion in the game of Go. Soon after, another program, AlphaGo Zero (AGZ) [26], achieved even stronger performance despite learning the game from scratch given only the rules. Starting *tabula rasa*, AGZ mastered the game of Go entirely through self-play using a new reinforcement learning algorithm. The same algorithm was shown to achieve superhuman performance in Chess and Shogi [25].

One key innovation of AGZ is to learn a policy and value function using supervised learning from samples generated via Monte-Carlo Tree Search. Motivated by the remarkable success of this method, in this work we study the problem of finding Nash Equilibrium for two-player turn-based zero-sum games and in particular consider a reinforcement learning based approach.

Our Contributions. The central contribution of this work is the Explore-Improve-Supervise (EIS) method for finding Nash Equilibrium for two-player turn-based zero-sum games with *continuous* state space, modeled through the framework of Markov game. It is an iterative method where in each iteration three components are intertwined carefully: "explore" that allows for measured exploration of the state space, "improve" which allows for improving the current value and policy for the state being explored, and "supervise" which learns the improved value and policy over the explored states so as to generalize over the entire state space.

Importantly, we identify sufficient conditions, in terms of each of the "explore", "improve" and "supervise" modules, under which convergence to the value function of the Nash equilibrium is guaranteed. In particular, we establish a finite sample complexity bound for such a generic method to find the ε-approximate value function of Nash equilibrium. See Theorem 2 and Proposition 3 for the precise statements.

We establish that when random sampling is used for "explore", Monte-Carlo-Tree-Search (MCTS) is used for "policy improvement" and Nearest Neighbor is used for "supervised learning", the theoretical conditions identified for convergence of EIS are satisfied. Using our finite sample bound for EIS, and quantification of conditions as stated above, we conclude that such an instance of EIS method find an ε-approximate value function of Nash equilibrium in $\tilde{O}\big(\varepsilon^{-(d+4)}\big)$ steps, where d is the dimension of the state space of the game (cf. Theorem 8). We also establish a mini-max lower bound on the number of steps required for learning an ε-approximate value function of Nash equilibrium as $\tilde{\Omega}\big(\varepsilon^{-(d+2)}\big)$ for any method (cf. Theorem 4). This establishes near-optimality of an instance of EIS.

Related Work. The Markov Decision Processes (MDP) provide a canonical framework to study the single-agent setting. Its natural extension, the Markov Games, provide a canonical framework to study multi-agent settings [12]. In this work, we consider an instance of it—turn-based two players or agents with zero-sum rewards. Analogous to learning the optimal policy in MDP, here we consider finding the Nash Equilibrium in the setting of Markov Games. There has been a rich literature on existence, uniqueness as well as algorithms for finding the Nash Equilibrium. In what follows, we describe the most relevant literature in that regard.

To start with, in [22], for finite state and action spaces where game would terminate in a finite number of stages with positive probability, the existence of optimal stationary strategies and uniqueness of the optimal value function are established. For generic state space, the existence of Nash Equilibrium has been established for Markov Games with discounted rewards. Particularly, when the state space is a compact metric space, [15, 16] and [17] show the uniqueness of value function and existence of optimal stationary policy. The same result has been established by [10] when the state space is complete, separable metric space. For two-player zero-sum discounted Markov games, the Bellman operator corresponding to the Nash equilibrium is a contraction and hence, the value function is unique and there exists a deterministic stationary optimal policy [3, 28]. We also note that existence of Nash equilibrium for a general class of games (stochastic shortest path) is established by [18]. It argues that the optimal value function is unique and can be achieved by mixed stationary strategies.

For computing or finding optimal value function and policy associated with the Nash equilibrium, there are two settings considered in the literature: (i) when system model is entirely known, and (ii) when model is not known but one can sample from the underlying model. In the first setting, classical approaches from the setting of MDPs such as value/policy iteration are adapted to find the optimal value function or policy associated with the Nash equilibrium [3, 18]. In the second setting which is considered here, various approximate dynamic programming algorithms have been proposed [1, 4, 5, 11, 13, 14, 19, 20, 28]. More recent work approximates the value function/policy by deep neural networks [24–26].

In terms of theoretical results, there has been work establishing asymptotic convergence to the optimal value function when the state space is finite. For example, Q-learning for MDP adapted to the setting of two-player zero-sum games converges asymptotically [28]. Non-asymptotic results are available for model-based algorithms developed for Markov games with finite states, including R-max algorithm [2] and an algorithm that extends upper confidence reinforcement learning algorithm [31]. Recent work in [23] provides an algorithm that computes an ε-optimal strategy with near-optimal sample complexity for Markov games with finite states. For Markov games where the transition function can be embedded in a given feature space, the work by [6] analyzes the sample complexity of a Q-learning algorithm. However, non-asymptotic or finite sample analysis for continuous state space without a special structure, such as that considered in this work, receives less attention in the literature.

Comparison with Prior Work. In this work, we develop Explore-Improve-Supervise (EIS) policy when the model is unknown, but one is able to sample from the underlying model. We study the convergence and sample complexity of our approach. Our goal is to provide a *formal* study on the general framework of EIS. The overall framework is inspired by AlphaGo Zero and inherits similar components. However, we take an important step towards bridging the gap between sound intuitions and theoretical guarantees, which is valuable for a better understanding on applying or extending this framework with different instantiations. We note that EIS bears certain similarities with another AlphaGo-inspired study [21]. Both works follow the main idea of coupling improvements with supervised learning. However, there are major differences. Shah et al. [21] focus on approximating value function in deterministic MDPs and only studies a particular instance of the modules. In contrast, we focus on a broader class of algorithms, formulating general principles and studying the guarantees. This poses different challenges and requires generic formulations on properties of the modules that are technically precise and practically implementable.

Finally, as mentioned previously, non-asymptotic analysis for continuous state space, considered in this work, is scarce for Markov games. While there are some results for finite states, the bounds are not directly comparable. For example, the complexity in [20] depends on some oracle complexities for linear programming and regression.

For the setting with continuous state space, the sample complexity results in [6] for Q-learning rely on the assumption of linear structure of the transition kernel. The recent work [32] studies the finite-sample performance of minimax deep Q-learning for two-player zero-sum games, where the convergence rate depends on the family of neural networks. We remark that these belong to a different class of algorithms. We also derive a fundamental mini-max lower bound on sample-complexity for any method (cf. Theorem 4). The lower bound is interesting on its own. Moreover, it shows near optimal dependence on dimension for an instance of our EIS framework.

Organization. The remainder of the paper is organized as follows. We formally introduce the framework of Markov Games and Nash equilibrium in Section 2. Section 3 describes a generic Explore-Improve-Supervise (EIS) algorithm. The precise technical properties for the modules of EIS are then stated in Section 4, under which we establish our main results, convergence and sample complexity of EIS, as stated in Section 5. Finally, a concrete instantiation is provided in Section 6, demonstrating the applicability of the generic EIS algorithm. The proofs of our main results are presented in Sections 7 and 8.

2 TWO-PLAYER MARKOV GAMES AND NASH EQUILIBRIUM

We introduce the framework of Markov Games (MGs) (also called Stochastic Games [22]) with two players and zero-sum rewards. The goal in this setting is to learn the Nash equilibrium.

2.1 Two-player Zero-sum Markov Game

We consider two-player *turn-based* Markov games like Go and Chess, where players take turns to make decisions. We denote the two players as P1 and P2. Formally, a Markov game can be expressed as a tuple $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{A}_1, \mathcal{A}_2, r, P, \gamma)$, where $\mathcal{S}_1$ and $\mathcal{S}_2$ are the set of states controlled by P1 and P2 respectively, $\mathcal{A}_1$ and $\mathcal{A}_2$ are the set of actions P1 and P2 can take, r represents reward function, P represents transition kernel and $\gamma \in [0, 1)$ is the discount factor. Specifically, for $i = 1, 2$, let $\mathcal{A}_i(s)$ be the set of feasible actions for player i in a given state $s \in \mathcal{S}_i$. We assume that $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$[1].

[1]This assumption is typical for turn-based games. In particular, one can incorporate the "turn" of the player as part of the state, and thus P1's state space (i.e. when it is player 1's turn) is disjoint from that of P2's turn.

Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$. For each state $s \in \mathcal{S}$, let $I(s) \in \{1, 2\}$ indicate the current player to play. At state s, upon taking action a by the corresponding player $I(s)$, player $i \in \{1, 2\}$ receives a reward $r^i(s, a)$. In zero-sum games, $r^1(s, a) = -r^2(s, a)$. Without loss of generality, we let P1 be our reference and use the notation $r(s, a) \triangleq r^1(s, a)$ for the definitions of value functions.

Let $P(s, a)$ be the distribution of the new state after playing action a, in state s, by player $I(s)$. In this paper, we focus on the setting where the state transitions are deterministic. This means that $P(s, a)$ is supported on a single state, $s \circ a$, where $s \circ a$ denotes the state after taking action a at state s.

For each $i \in \{1, 2\}$, let π_i be the policy for player i, where $\pi_i(\cdot|s)$ is a probability distribution over $\mathcal{A}_i(s)$. Denote by Π_i the set of all stationary policies of player i, and let $\Pi = \Pi_1 \times \Pi_2$ be the set of all polices for the game. A two-player zero-sum game can be seen as player P1 aiming to maximize the accumulated discounted reward while P2 attempting to minimize it. The value function and Q function for a zero-sum Markov game can be defined in a manner analogous to the MDP setting:

$$V_{\pi_1,\pi_2}(s) = \mathbb{E}_{a_t, s_{t+1}, a_{t+1}, \dots}\Big[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_t = s\Big],$$

$$Q_{\pi_1,\pi_2}(s, a) = \mathbb{E}_{s_{t+1}, a_{t+1}, \dots}\Big[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_t = s, a_t = a\Big],$$

where $a_l \sim \pi_{I(s_l)}(\cdot|s_l)$ and $s_{l+1} \sim P(s_l, a_l)$. That is, $V_{\pi_1,\pi_2}(s)$ is the expected total discounted reward for P1 if the game starts from state s, players P1 and P2 use the policies π_1 and π_2 respectively. The interpretation for Q-value is similar.

To simplify the notation, we assume that $\mathcal{A}_1 = \mathcal{A}_2 \triangleq \mathcal{A}$, where $\mathcal{A}$ is a finite set. We consider $\mathcal{S}$ to be a compact subset of $\mathbb{R}^d$, where $d \geq 1$. The rewards $r(s, a)$ are independent random variables taking value in $[-R_{\max}, R_{\max}]$ for some $R_{\max} > 0$. Define $V_{\max} \triangleq R_{\max}/(1 - \gamma)$. It follows that absolute value of value function and Q function for any policy is bounded by $V_{\max}$.

Regarding Deterministic Transitions Assumption. In fact, our approach and main results of EIS framework (i.e., Sections 4 and 5) apply to general non-deterministic cases as well, though the example we consider in Section 6 is for deterministic cases. In particular, the improvement module is instantiated by a variant of Monte Carlo Tree Search, where a non-asymptotical analysis has been only established for the deterministic case [21]. To facilitate a coherent exposition, we focus on deterministic cases here. Indeed, many games, such as Go and Chess, are deterministic. Additionally, note that one could instantiate our EIS framework with other methods for the non-deterministic cases—for instance, by adapting the sparse sampling oracle [8] as the improvement module—to obtain a similar analysis.

2.2 Nash Equilibrium

Definition 1 (Optimal Counter Policy). *Given a policy $\pi_2 \in \Pi_2$, policy $\pi_1 \in \Pi_1$ for* P1 *is said to be an optimal counter-policy against π_2, if and only if for every $s \in \mathcal{S}$, we have $V_{\pi_1,\pi_2}(s) \geq V_{\pi_1',\pi_2}(s), \forall \pi_1' \in \Pi_1$. Similarly, a policy $\pi_2 \in \Pi_2$ for* P2 *is said to be an optimal counter-policy against a given policy $\pi_1 \in \Pi_1$ for* P1*, if and only if for every $s \in \mathcal{S}$ $V_{\pi_1,\pi_2} \leq V_{\pi_1,\pi_2'}, \forall \pi_2' \in \Pi_2$.*

In a two-player zero-sum game, it has been shown that the pairs of optimal policies coincides with the Nash equilibrium of this game [10, 15, 17]. In particular, a pair of policies (π_1^*, π_2^*) is called an equilibrium solution of the game, if π_1^* is an optimal counter policy against π_2^* and π_2^* is an optimal counter policy against π_1^*. The value function of the optimal policy, $V_{\pi_1^*,\pi_2^*}$, is the *unique* fixed point of a γ-contraction operator. In the sequel, we will simply refer to the strategy $\pi^* = (\pi_1^*, \pi_2^*)$ as the optimal policy. Finally, we use the concise notation V^* and Q^* to denote the value function and the Q function under the optimal policy, respectively, i.e., $V^*(s) = V_{\pi_1^*,\pi_2^*}(s)$ and $Q^*(s, a) = Q_{\pi_1^*,\pi_2^*}(s, a)$.

Algorithm 1 The Generic EIS Algorithm

```
1: Initialization: a supervised learning model $f_0(s) = (V_0(s), \pi_0(\cdot|s))$ for every $s \in \mathcal{S}$.
2: for $l = 1, 2, \dots, L$ do
3:     receives an initial state $s_1$.
4:     /* data generation via improvement & exploration */
5:     for $i = 1, 2, \dots, n_l$ do
6:         query the improvement module, which takes as inputs the current model $f_{l-1}$, and outputs estimates $\hat{V}(s_i)$ for the optimal value $V^*(s_i)$ and $\hat{\pi}(\cdot|s_i)$ for the optimal policy $\pi^*(\cdot|s_i)$:
               $\big(\hat{V}(s_i),\ \hat{\pi}(\cdot|s_i)\big) = \text{Improvement Module}(f_{l-1}, s_i)$    (1)
7:         query the exploration module for the next state $s_{i+1}$:
               $s_{i+1} = \text{Exploration Module}(s_i)$    (2)
8:     end for
9:     /* model update via supervised learning */
10:    query the supervised learning module with the collected data $\mathcal{D}^{(l)} = \{(s_i, \hat{V}(s_i), \hat{\pi}(\cdot|s_i)\}_{i=1}^{n_l}$ and obtained an updated model $f_l(s)$ for every $s \in \mathcal{S}$.
               $f_l = \text{Supervised Learning Module}\big(\mathcal{D}^{(l)}\big)$    (3)
11: end for
12: Output: final model $f_L$.
```

3 EIS: EXPLORE-IMPROVE-SUPERVISE

We describe Explore-Improve-Supervise (EIS) algorithm for learning the optimal value function V^* and optimal policy π^*. The algorithm consists of three separate, but intertwined modules: exploration, improvement and supervised learning. Below is a brief summary of these modules. The precise, formal description of properties desired from these modules is stated in Section 4, which will lead to convergence and correctness of the EIS algorithm as stated in Theorem 2. Section 6 provides a concrete example of modules of EIS satisfying properties stated in Section 4.

Exploration Module. To extract meaningful information for the entire game, sufficient exploration is required so that enough *representative* states will be visited. This is commonly achieved by an appropriate exploration policy, such as ϵ-greedy policy and Boltzmann policy. The existence of an exploration module is required to guarantee sufficient exploration.

Improvement Module. For the overall learning to make any progress, the improvement module improves the existing estimates of the optimal solution. In particular, given the current estimates

$\hat{V}$ for V^* and $\hat{\pi}$ for π^*, for a state s, a query of the improvement module produces better estimates $\hat{V}'(s)$ and $\hat{\pi}'(\cdot|s)$ that are *closer* to the optimal $V^*(s)$ and $\pi^*(\cdot|s)$ in an appropriate sense.

Supervised Learning Module. The previous two modules can be collectively viewed as a data generation process: the exploration module samples sufficient representative states, while a query of the improvement module provides improved estimates for the optimal value and policy. With these as training data, supervised learning module would learn and generalize the improvement of the training data to the entire state space. Subsequently, the trained supervised learning module produces better estimates for V^* and π^*.

Combining together, the three modules naturally lead to the following iterative algorithm whose pseudo-code is provided in Algorithm 1. Initially, the algorithm starts with an arbitrary model for the value function and policy. In each iteration $l \geq 1$, it performs two steps:

Step 1. Data Generation. Given current model $f_{l-1} = (V_{l-1}, \pi_{l-1})$: for current state s, query the improvement module to obtain better estimates $\hat{V}(s)$ and $\hat{\pi}(\cdot|s)$ than the current estimates $f_{l-1}(s)$; and then query the exploration module to arrive at the next state s'; repeat the above process to obtain training data of n samples, $\{(s_i, \hat{V}(s_i), \hat{\pi}(\cdot|s_i))\}_{i=1}^n$.

Step 2. Supervised Learning. Given the improved estimates $\{(s_i, \hat{V}(s_i), \hat{\pi}(\cdot|s_i))\}_{i=1}^n$, use the supervised learning module to build a new model $f_l = (V_l, \pi_l)$.

Intuitively, the iterative algorithm keeps improving our estimation after each iteration, and eventually converges to optimal solutions. The focus of this paper is to *formally* understand under what conditions on each of the exploration, improvement and supervised learning module does the algorithm work. Of course, proof is in the puddling—we provide examples of existence of such modules in Section 6.

4 PROPERTIES OF MODULES

In this section we formally state the desired properties of each of the three modules of EIS. With these properties, we establish convergence and correctness of EIS algorithm in Section 5 to follow. We remark that the properties are not made for the ease of technical analysis. Examples satisfying these properties shall be provided in Section 6.

4.1 Improvement Module

This module improves both value function and policy. The value function is real-valued, whereas policy for each given state can be viewed as a probability distribution over all possible actions. This requires a careful choice of metric for quantifying improvement. Let $\hat{V}(s)$ and $\hat{\pi}(\cdot|s)$ be the estimates output by the improvement module in the l-th iteration of EIS. Improvement of value function means $|\hat{V}(s) - V^*(s)| < |V_l(s) - V^*(s)|$. Improvement for policy is measured by the KL divergence between $\hat{\pi}(\cdot|s)$ and $\pi^*(\cdot|s)$. Here some care is needed as KL divergence would become infinite if supports of the distributions mismatch.

Note that the optimal policy π^* only assigns positive probability to the optimal actions. On the other hand, there is no guarantee that $\hat{\pi}(\cdot|s)$ always has a full support on $\mathcal{A}$. To overcome these challenges, we instead measure the KL divergence with an alternative "optimal policy" that guarantees a full support on $\mathcal{A}$. This naturally leads to the optimal Boltzmann policy: given a temperature $\tau > 0$, the optimal Boltzmann policy is given by

$$P^*_\tau(a|s) = \frac{e^{Q^*(s,a)/\tau}}{\sum_{a' \in \mathcal{A}} e^{Q^*(s,a')/\tau}}, \quad \text{for } a \in \mathcal{A}. \tag{4}$$

If $I(s)$ is player P2, use $-Q^*(s,a)$ instead in the above equation to construct the Boltzmann policy (Recall that player P1 is set to be our reference in Section 2). By definition, $D_{\mathrm{KL}}\big(\hat{\pi}(\cdot|s)||P^*_\tau(\cdot|s)\big)$ is guaranteed to be finite for any estimate $\hat{\pi}(\cdot|s)$. Furthermore, P^*_τ converges to π^* as $\tau \to 0$. Therefore, we could use the KL divergence $D_{\mathrm{KL}}\big(\hat{\pi}(\cdot|s)||P^*_\tau(\cdot|s)\big)$ with a small enough τ to measure the improvement of the estimates.

Finally, the statistical efficiency of the module is measured by the number of samples (i.e., observed state transitions) required to improve the policy and value function. We now formally state the following desirable property for the improvement module.

Property 1. (Improvement Property) *Suppose the current model $f(s) = (V(s), \pi(\cdot|s))$ (potentially random) has estimation errors $\varepsilon_{0,v} > 0$ and $\varepsilon_{0,p} > 0$ for the value and policy estimates, respectively, i.e.,*

$$\mathbb{E}\Big[||V - V^*||_\infty\Big] \leq \varepsilon_{0,v},$$
$$\mathbb{E}\Big[D_{\mathrm{KL}}\big(\pi(\cdot|s)||P^*_\tau(\cdot|s)\big)\Big] \leq \varepsilon_{0,p}, \ \forall\, s \in \mathcal{S},$$

where the expectations are taken with respect to the randomness of the model $f = (V, \pi)$.

Fix any state $s \in \mathcal{S}$, and query the improvement module on s via $\big(\hat{V}(s), \hat{\pi}(\cdot|s)\big) =$ Improvement Module(f, s). Let the temperature be $\tau > 0$, and improvement factors be $0 < \zeta_v < 1$ and $0 < \zeta_p < 1$. Then, there exists a function $\kappa(\tau, \varepsilon_{0,v}, \varepsilon_{0,p}, \zeta_v, \zeta_p)$ such that if $\kappa(\tau, \varepsilon_{0,v}, \varepsilon_{0,p}, \zeta_v, \zeta_p)$ number of samples are used, the new estimates satisfy that

$$\mathbb{E}\Big[\big|\hat{V}(s) - V^*(s)\big|\Big] \leq \zeta_v \cdot \varepsilon_{0,v},$$
$$\mathbb{E}\Big[D_{\mathrm{KL}}\big(\hat{\pi}(\cdot|s)||P^*_\tau(\cdot|s)\big)\Big] \leq \zeta_p \cdot \varepsilon_{0,p},$$

where the expectations are with respect to the randomness in the model f and the improvement module.

Property 1 allows for a randomized improvement module, but requires that on average, the errors for the value and policy estimates should strictly shrink.

4.2 Supervised Learning Module

To direct the model update in an improving manner, the supervised learning step (line 10 of Algorithm 1) should be able to learn from the training data, $\hat{V}$ and $\hat{\pi}$, and generalize to unseen states by preserving the same order of error as the training data. Generically speaking, generalization would require two conditions: (1) sufficiently many training data that are "representative" of the underlying state space; (2) the model itself is expressive enough to capture the characteristics of the function that is desired to be learned.

Before specifying the generalization property, let us provide a few remarks on the above conditions. Condition (1) is typically ensured by using an effective exploration module. Recall that the state space is continuous. The exploration module should be capable of navigating the space until sufficiently many different states are visited. Intuitively, these *finite* states should properly cover the entire space, i.e., they are representative of the entire space so that learning from these states provide enough information for other states. Formally, this means that given the current estimation errors $\varepsilon_{1,v}$ and $\varepsilon_{1,p}$ for the optimal value and policy, there exists a sufficiently large set of $N(\varepsilon_{1,v}, \varepsilon_{1,p})$ training states, such that supervised learning applied to those training data would generalize to the entire state space with the same order of accuracy. The precise definition of representative states may depend on the particular supervised learning algorithm.

Regarding condition (2), generalization performance of traditional models has been well studied in classical statistical learning theory. More recently, deep neural networks exhibit superior empirical generalization ability, although a complete rigorous proof seems beyond the reach of existing techniques. Our goal is to seek general principle underlying the supervised learning step and as such, we do not limit ourselves to specific models—the learning model could be a parametric model that learns via minimizing empirical squared loss and cross-entropy loss, or it could be a non-parametric model such as nearest neighbors regression. With the above conditions in mind, we state the following general property for the supervised learning module:

Property 2. (Generalization Property) *Let temperature $\tau > 0$, estimation errors $\varepsilon_{1,v} > 0$ and $\varepsilon_{1,p} > 0$ be given. There exists at least one set of finite states, denoted by $\mathcal{S}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$, with size $N_{\mathcal{S}}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$, so that the following* **generalization bound** *holds: Suppose that a training dataset $\left\{\left(s_i, \hat{V}(s_i), \hat{\pi}(\cdot|s_i)\right)\right\}_{i=1}^n$ satisfies $\mathcal{S}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) \subset \{s_i\}_{i=1}^n$ and the following error guarantees:*

$$\mathbb{E}\Big[|\hat{V}(s_i) - V^*(s_i)|\Big] \leq \varepsilon_{1,v},$$

$$\mathbb{E}\Big[D_{\mathrm{KL}}\big(\hat{\pi}(\cdot|s_i) \| P_\tau^*(\cdot|s_i)\big)\Big] \leq \varepsilon_{1,p}, \ \forall\, i \in [n],$$

where the expectation is taken with respect to the randomness of the value $\hat{V}(s_i)$ and $\hat{\pi}(\cdot|s_i)$. Then, there exist non-negative universal constants c_p and c_v such that after querying the supervised learning module, i.e., $(V, \pi) =$ Supervised Learning Module$(\left\{\left(s_i, \hat{V}(s_i), \hat{\pi}(\cdot|s_i)\right)\right\}_{i=1}^n)$, (V, π) satisfy

$$\mathbb{E}\Big[\big\|V - V^*\big\|_\infty\Big] \leq c_v \cdot \varepsilon_{1,v};$$

$$\mathbb{E}\Big[D_{\mathrm{KL}}\big(\pi(\cdot|s) \| P_\tau^*(\cdot|s)\big)\Big] \leq c_p \cdot \varepsilon_{1,p}, \quad \forall s \in \mathcal{S}.$$

4.3 Exploration Module

With the above development, it is now straightforward to identify the desired property of the exploration module. In particular, as part of the data generation step, it should be capable of exploring the space so that a set of representative states $\mathcal{S}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$ are visited. Consequently, the supervised learning module can then leverage the training data to generalize. Formally, let $\mathcal{E}$ be the set of all possible representative sets that satisfy the Generalization Property:

$$\mathcal{E}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) = \Big\{\mathcal{S}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) \subset \mathcal{S} \ : \ \text{Property 2 is satisfied with } \mathcal{S}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}).\Big\}.$$

Denote by $\mathcal{T}(t) \triangleq \{s_i\}_{i=1}^t$ the set of states explored by querying the exploration module up to time t, with s_1 being the initial state and $s_{i+1} =$ Exploration Module(s_i) (cf. line 7 of Algorithm 1). We now state the exploration property, which stipulates that starting at an arbitrary state s, the explored states should contain one of the representative sets in $\mathcal{E}$, within a *finite* number of steps.

Property 3. (Exploration Property) *Given the temperature $\tau > 0$, and estimation errors $\varepsilon_{1,v} > 0$ and $\varepsilon_{1,p} > 0$ for the value and policy, define*

$$T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}, s) \triangleq \min\Big\{ t \geq 1 \ : \ s_1 = s; \exists\, \hat{\mathcal{S}} \in \mathcal{E}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) \text{ such that } \hat{\mathcal{S}} \subset \mathcal{T}(t)\Big\}.$$

Then, the exploration module satisfies that $\forall s \in \mathcal{S}$,

$$\mathbb{E}\Big[T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}, s)\Big] < B(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}),$$

for some $B(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) < \infty$ independent of s. The above expectation is taken with respect to the randomness in the exploration module and the environment (i.e., state transitions).

In the sequel, when the context is clear or the initial state does not matter, we usually drop the dependence in s to simplify the notation, i.e., $T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$.

5 MAIN RESULTS: CONVERGENCE GUARANTEES AND SAMPLE COMPLEXITY

5.1 Convergence Guarantees

As the main result of this paper, we establish convergence of the EIS algorithm under the three desired properties given in Section 4, and quantify the corresponding finite sample complexity. We also provide an algorithm-independent minimax lower bound; in Section 6 we introduce an instance of EIS that essentially matches this lower bound.

Theorem 2. *Given a small enough $\tau > 0$, let Properties 1, 2 and 3 hold. Let $C_{0,v} = \|V_0 - V^*\|_\infty$ and $C_{0,p} = \sup_{s \in \mathcal{S}} D_{\mathrm{KL}}\big(\pi_0(\cdot|s) \| P_\tau^*(\cdot|s)\big)$ be initialization errors. Then for a given $\rho \in (0, 1)$, with appropriate parameters for Algorithm 1, the output $f_L = (V_L, \pi_L)$ after L-th iteration satisfies*

$$\mathbb{E}\Big[\big\|V_L - V^*\big\|_\infty\Big] \leq C_{0,v}\rho^L, \tag{5}$$

$$\mathbb{E}\Big[D_{\mathrm{KL}}\big(\pi_L(\cdot|s) \,\|\, P_\tau^*(\cdot|s)\big)\Big] \leq C_{0,p}\rho^L, \quad \forall s \in \mathcal{S}. \tag{6}$$

The proof of Theorem 2 is provided in Section 7. Theorem 2 implies that the model (V_L, π_L) learned by the generic EIS algorithm converges to the optimal value function V^* and the optimal Boltzmann policy P_τ^* exponentially with respect to the number of iterations. In particular, after

$$L = \left\lceil \frac{\log \frac{\varepsilon}{\max\{C_{0,v}, C_{0,p}\}}}{\log \rho} \right\rceil = \Theta\Big(\log \frac{1}{\varepsilon}\Big)$$

iterations, we can obtain estimates for both V^* and P_τ^* that are within ε estimation errors. We note that with a sufficiently small temperature, P_τ^* is close to the optimal policy π^*. Therefore, the model $f_L = (V_L, \pi_L)$ can be close to (V^*, π^*) for a large L.

5.2 Sample Complexity

We can also characterize the sample complexity of the EIS algorithm. Recall that the sample complexity is defined as the total number of state transitions required for the algorithm to learn ϵ-approximate value/policy function. The sample complexity of Algorithm 1 comes from two parts: the improvement module and the exploration module. Recall that the improvement module requires $\kappa(\tau, \varepsilon_{0,v}, \varepsilon_{0,p}, \zeta_v, \zeta_p)$ samples for each call (cf. Property 1). The sample complexity of exploration module is proportional to $T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$, which satisfies $\mathbb{E}[T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})] \leq B(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$ (cf. Property 3). The following proposition, whose proof is given in Section 8, bounds the sample complexity in terms of the above relevant quantities.

Proposition 3. *Consider the setting of Theorem 2. Then, with probability at least* $1 - \delta$*, the convergence result (i.e., Eqs (5) and (6)) is achieved with sample complexity* M *such that*

$$M \leq \sum_{l=1}^{L} \Bigg[\kappa\Big(\tau, C_{0,v}\rho^{l-1}, C_{0,p}\rho^{l-1}, \frac{\rho}{c_v}, \frac{\rho}{c_p}\Big) \times B\Big(\tau, \frac{C_{0,v}\rho^{l}}{c_v}, \frac{C_{0,p}\rho^{l}}{c_p}\Big) \cdot e \cdot \log \frac{L}{\delta} \Bigg].$$

In Section 6, we provide a concrete instance of EIS that finds ε-approximate value function and policy of Nash equilibrium with $\widetilde{O}(\varepsilon^{-(d+4)})$ transitions.

5.3 A Generic Lower Bound

To understand how good the above sample complexity upper bound is, we establish a *lower bound* for any algorithm under any sampling policy. In particular, we leverage the the minimax lower bound for the problem of non-parametric regression [27, 29] to establish the lower bound, as stated in the following theorem.

Theorem 4. *Given an algorithm, let* V_T *be the estimate of* V^* *after* T *samples of transitions for the given Markov game. Then, for each* $\delta \in (0, 1)$*, there exists a two-player zero-sum Markov game such that in order to achieve* $\mathbb{P}\Big[\big\|\hat{V}_T - V^*\big\|_\infty < \varepsilon\Big] \geq 1 - \delta$*, it must be that*

$$T \geq C'd \cdot \varepsilon^{-(d+2)} \cdot \log(\varepsilon^{-1}),$$

where $C' > 0$ *is an algorithm-independent constant.*

6 IMPLEMENTATION: A CONCRETE INSTANTIATION OF THE KEY MODULES

In this section, we demonstrate the applicability of the generic EIS algorithm by giving a concrete instantiation. Specifically, we will use a variant of Monte Carlo Tree Search (MCTS) as the improvement module, nearest neighbor regression as the supervised learning module, and random sampling as the exploration module. We prove that all properties in Section 4 are satisfied. This shows that these properties are reasonable and hence gives a strong support for the generic recipe developed in this paper. Due to space limit, we provide discussions here with technical results. The complete proof is deferred to the technical report.

6.1 Improvement Module: MCTS

Recall that the improvement module should be capable of providing improved estimates for both the value and policy functions, at the queried state s. Since both the value and the policy are closely related to the Q function, one simple approach to simultaneously produce improved estimates is to obtain better estimates of Q^* first and then construct the improved estimates of value and policy from $\hat{Q}$. We will take this approach in this example and use MCTS to obtain the Q estimates. Throughout this section, we assume the existence of a generative model (i.e., a simulator).

MCTS is a class of popular search algorithms for sequential decision-makings, by building search trees and randomly sampling the state space. It is also one of the key components underlying the success of AlphaGo Zero. Most variants of MCTS in literature uses some forms of upper confidence bound (UCB) algorithm to select actions at each depth of the search tree. Since our focus is to demonstrate the improvement property, we employ a fixed H-depth MCTS, which takes the current model of the value function V_l as inputs and outputs a value estimate $\hat{V}(s)$ of the root node s. The current model V_l of the value function is used for evaluating the value of the leaf nodes at depth H during the Monte Carlo simulation. This fixed depth MCTS has been rigorously analyzed in [21] with non-asymptotic error bound for the root node, when the state transition is deterministic. We refer readers to [21] (precisely, Algorithm 1) for the details of the pseudo code. We remark that in principle, many other variants of MCTS that has a precise error guarantee could be used instead; we choose the fixed H-depth variant here to provide a concrete example.

We now lay down the overall algorithm of the improvement module in Algorithm 2 below. Recall that the state transition is deterministic and the reward $r(s, a)$ could be random (cf. Section 2). Given the queried state s, note that the Q-value estimate $\hat{Q}(s, a)$ for each $a \in \mathcal{A}$ is given by the sum of two components: (1) empirical average of the reward $r(s, a)$; (2) the estimated value $\hat{V}(s \circ a)$ for the next state, returned from calling the fixed depth MCTS algorithm with $s \circ a$ being the root node. Further recall that we use player P1 as the reference (i.e., $r(s, a) \triangleq r^1(s, a)$). The module then obtains improved value estimate $\hat{V}(s)$ by taking proper max or min of the Q-value estimates $\hat{Q}(s, a)$—depending on whether $I(s)$ is player P1 (maximizer) or player P2 (minimizer)—and improved policy estimate $\hat{\pi}(\cdot|s)$ as the Boltzmann policy based on $\hat{Q}(s, a)$. It is worth mentioning that the fixed depth MCTS algorithm was applied for discounted MDP in [21], but extending to game setting is straightforward as in literature [7, 9], i.e., by alternating between max nodes (i.e., P1 plays) and min nodes (i.e., P2 plays) for each depth in the tree and properly adjusting the sign of the upper confidence term.

The following theorem states the property of this specific improvement module (Algorithm 2). It is not hard to see that it directly implies the desired improvement property, i.e., Property 1.

Proposition 5. *Suppose that the state transitions are deterministic. Given the current model* $f = (V, \pi)$*, a small temperature* $\tau < 1$*, and the improvement factors* $0 < \zeta_v < 1$ *and* $0 < \zeta_p < 1$*. Suppose that the current value model,* V*, satisfies that*

$$\mathbb{E}\Big[||V - V^*||_\infty\Big] \leq \varepsilon_{0,v}.$$

Algorithm 2 Improvement Module

```
1: Input: (1) Current model $f = (V, \pi)$; (2) root node $s$.
2: /* Q-value estimates*/
3: for each $a \in \mathcal{A}$ do
4:     call the fixed depth MCTS (Algorithm 1 of [21]) with: (1) depth $H$; (2) root node $s \circ a$; (3) current model $V$ for evaluating value of the leaf nodes at depth $H$; (4) number of MCTS simulation $m$. That is,
           $\hat{V}(s \circ a) = \text{Fixed Depth MCTS}(H, s \circ a, V, m)$
5:     simulate action $a$ for $m$ times to obtain an empirical average, $\hat{r}(s, a)$ , of the reward $r(s, a)$.
6:     form Q-value estimate for $Q^*(s, a)$:
           $\hat{Q}(s, a) = \hat{r}(s, a) + \gamma \hat{V}(s \circ a)$
7: end for
8: /* Improved value and policy estimates*/
9: form value and policy estimates for $V^*(s)$ and $P^*_\tau(s)$ based on the Q-value estimates, i.e.,
       $\hat{V}(s) = \max_{a \in \mathcal{A}} \hat{Q}(s, a)$
       $\hat{\pi}(a|s) = \frac{e^{\hat{Q}(s,a)/\tau}}{\sum_{a' \in \mathcal{A}} e^{\hat{Q}(s,a')/\tau}}$ for every $a \in \mathcal{A}$.
   if $I(s)$ is player P2, then replace max with min in the value estimate and replace $\hat{Q}$ with $-\hat{Q}$ in the policy estimate (recall that the maximizer P1 is the reference).
10: Output: improved estimates $\hat{V}(s)$ and $\hat{\pi}(\cdot|s)$.
```

Then, with appropriately chosen parameters for Algorithm 2, for each query state $s_0 \in \mathcal{S}$, $\big(\hat{V}(s_0), \hat{\pi}(\cdot|s_0)\big) =$ Improvement Module(f, s_0), we have:

$$\mathbb{E}\Big[\big|\hat{V}(s_0) - V^*(s_0)|\big|\Big] \le \zeta_v \cdot \varepsilon_{0,v}$$
$$\mathbb{E}\Big[D_{\text{KL}}\big(\hat{\pi}(\cdot|s_0)||P^*_\tau(\cdot|s_0)\big)\Big] \le \zeta_p \cdot \varepsilon_{0,v}.$$

The above is achieved with a sample complexity of

$$O\Big(\big(\tau \cdot \min\{\zeta_v, \zeta_p\} \cdot \varepsilon_{0,v}\big)^{-2} \cdot \frac{\log(\tau \cdot \min\{\zeta_v, \zeta_p\})}{\log \gamma}\Big).$$

6.2 Supervised Learning Module: Nearest Neighbor Regression.

We employ a nearest neighbor algorithm to learn the optimal value function and policy. Intuitively, suppose that the optimal value function and the Boltzmann policy is smooth in the state space, then this algorithm will generalize well if there are sufficiently many training data points around each state in the state space $\mathcal{S}$. Formally, to establish the generalization property of nearest neighbor supervised learning algorithm, we make the following smoothness assumption about the Markov game.

Assumption 1. *(A1.) The state space $\mathcal{S}$ is a compact subset of $\mathbb{R}^d$. The chosen distance metric $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_+$ associated with the state space $\mathcal{S}$ satisfies that $(\mathcal{S}, d)$ forms a compact metric space. (A2.) The optimal value function $V^* : \mathcal{S} \to \mathbb{R}$ satisfies Lipschitz continuity with parameter L_v, i.e., $\forall s, s' \in \mathcal{S}$, $|V^*(s) - V^*(s')| \le L_v d(s, s')$. (A3.) The optimal Boltzmann policy P^*_τ defined in Eq. (4) is Lipschitz continuous with parameter $L_p(\tau)$, i.e., $\forall s, s' \in \mathcal{S}$, $\forall a \in A$, $|P^*_\tau(a|s) - P^*_\tau(a|s')| \le L_p(\tau) d(s, s')$.*

For each $h > 0$, the compact $\mathcal{S}$ has a finite $h/2$-covering number $N(h)$. There exists a *partition* of $\mathcal{S}$, $\{B_j, j \in [N(h)]\}$, such that each B_j has a diameter at most h, that is, $\sup_{x,y \in B_j} d(x, y) \le h$. We assume that states in the training set, $T := \{s_i, i \in [n]\}$, are sufficiently representative in the following sense: for any given h and K, the sample size n can be chosen large enough to ensure that $|B_j \cap T| \ge K$ for all $j \in [N(h)]$, i.e., each B_j having at least K training data. If T satisfies this condition, we call it *(h, K)-representative*.

Given the training data, we fit a value function $V_{\text{NN}} : \mathcal{S} \to \mathbb{R}$ using the following Nearest Neighbor type algorithm: set

$$V_{\text{NN}}(s) = \frac{1}{|B_j \cap T|} \sum_{s_i \in B_j \cap T} \hat{V}(s_i), \quad \forall s \in B_j.$$

For each $a \in \mathcal{A}$, a similar algorithm can be used to fit the action probability $\pi_{\text{NN}}(a|\cdot) : \mathcal{S} \to [0, 1]$. The proposition below shows that this algorithm has the desired generalization property. To simplify the notation, we use ε_v and ε_p to represent the estimation errors of the value function and the policy, respectively, for the training data. That is, $\varepsilon_v \triangleq \varepsilon_{1,v}$ and $\varepsilon_p \triangleq \varepsilon_{1,p}$ in Property 2.

Proposition 6. *Suppose Assumption 1 holds. If the training data is representative with respect to appropriate $h > 0$ and $K > 0$, the above regression algorithm satisfies Property 2. In particular, if*

$$h = \min\Big\{\frac{\varepsilon_v}{L_v}, \frac{\sqrt{\varepsilon_p/2}}{L_p}\Big\}, \tag{7}$$

$$K = \max\Big\{\frac{V^2_{\max}}{2\varepsilon^2_v} \log\Big(\frac{4V_{\max}N(h)}{\varepsilon_v}\Big), \frac{1}{\varepsilon_p} \log\Big(\frac{4N(h)}{\varepsilon_p}\Big)\Big\}, \tag{8}$$

we have

$$\mathbb{E}\big[\,\|V_{\text{NN}} - V^*\|_\infty\,\big] \le 4 \cdot \varepsilon_v,$$
$$\mathbb{E}\big[D_{\text{KL}}\big(\pi_{\text{NN}}(\cdot|s) \,\|\, P^*_\tau(\cdot|s)\big)\big] \le c \cdot \varepsilon_p, \quad \forall s \in \mathcal{S},$$

where the constant $c > 0$ is independent of n, the size of the training data.

The size of a representative data set should at least scale as $KN(h)$. Consider a simple setting where the state space is a unit volume hypercube in $\mathbb{R}^d$ with l_∞ metric. By [30, Lemma 5.7], the covering number $N(h)$ of $\mathcal{S}$ scales as $\Theta\big((1/h)^d\big)$. Let $\varepsilon = \min\{\varepsilon_v, \sqrt{\varepsilon_p}\}$. Note that $h = \Theta(\varepsilon)$. Therefore, to achieve the desired generalization property, the size of the training dataset should satisfy

$$n \ge KN(h) = O\Big(\frac{K}{h^d}\Big) = O\Big(\frac{d}{\varepsilon^{d+2}} \log \frac{1}{\varepsilon}\Big).$$

6.3 Exploration Module: Random Sampling Policy

As stated in Proposition 6, the sampled states for nearest neighbor regression should be (h, K)-representative, where h, K are given by Eq. (7)-(8). We will show that a random sampling policy—uniformly sampling the state space—is able to visit a set of (h, K)-representative states within a finite expected number of steps. We need to assume that the state space $\mathcal{S}$ is sufficiently regular near the boundary. In particular, we impose the following assumption which is naturally satisfied by convex compact sets in $\mathbb{R}^d$, for example.

Assumption 2. *The partition $\{B_j, j \in [N(h)]\}$ of $\mathcal{S}$ satisfies*

$$\lambda(B_j) \geq c_0 \frac{\lambda(\mathcal{S})}{N(h)}, \quad \forall j \in [N(h)],$$

for some constant $c_0 > 0$, where λ is the Lesbegue measure in $\mathbb{R}^d$.

Proposition 7. *Suppose that the state space $\mathcal{S}$ is a compact subset of $\mathbb{R}^d$ satisfying Assumption 2. Given temperature $\tau > 0$, and estimation errors $\varepsilon_v > 0$ and $\varepsilon_p > 0$ for the value and policy respectively, define h, K as in Eq. (7)-(8). Then the expected number of steps to obtain a set of (h, K)-representative states under the random sampling policy is upper bounded by*

$$B(\tau, \varepsilon_v, \varepsilon_p) = O\left(\frac{KN(h)}{c_0} \log N(h)\right).$$

6.4 Convergence Guarantees of the Instance

For the instance of EIS algorithm with MCTS, random sampling policy and nearest neighbor supervised learning, we have shown that each module satisfies the desired properties (cf. Propositions 5-7). Therefore, the convergence result stated in Theorem 2 holds for the specific instance we consider here. Below we make this result explicit, providing concrete bounds on the estimation errors and sample complexity.

Theorem 8. *Suppose that Assumptions 1 and 2 hold. For a given $\rho \in (0, 1)$, and a small $\tau < 1$, there exist appropriately chosen parameters for the instance of Algorithm 1 with MCTS, random sampling and nearest neighbor supervised learning, such that:*

(1) *The output $f_L = (V_L, \pi_L)$ at the end of L-th iteration satisfies*

$$\mathbb{E}\left[\left\|V_L - V^*\right\|_\infty\right] \leq V_{\max}\rho^L,$$

$$\mathbb{E}\left[D_{\mathrm{KL}}\left(\pi_L(\cdot|s) \,\|\, P^*_\tau(\cdot|s)\right)\right] \leq \frac{cV_{\max}}{4}\rho^L, \quad \forall s \in \mathcal{S},$$

where c is the generalization constant for policy in Proposition 6.

(2) *With probability at least $1 - \delta$, the above result is achieved with sample complexity of*

$$\sum_{l=1}^{L} c' \log \frac{1}{\tau\rho} \cdot \frac{1}{\tau^2\rho^{4l}} \cdot \log \frac{N(c_1\rho^l)}{\rho^l} \cdot N(c_1\rho^l) \cdot \log N(c_1\rho^l) \cdot \log \frac{L}{\delta},$$

where $c' > 0$ and $c_1 > 0$ are constants independent of ρ and l.

(3) *In particular, if $\mathcal{S}$ is a unit volume hypercube in $\mathbb{R}^d$, then the total sample complexity to achieve ε-error value function and policy is given by*

$$O\left(\log \frac{1}{\tau\rho} \cdot \frac{1}{\tau^2\varepsilon^{d+4}} \cdot \log\left(\frac{1}{\varepsilon}\right)^4 \cdot \log \frac{1}{\delta}\right).$$

Theorem 8 states that for a unit hypercube, the sample complexity of the instance of EIS scales as $\widetilde{O}\left(\varepsilon^{-(4+d)}\right)$ (omitting the logarithmic factor). Note that the minimax lower bound in Theorem 4 scales as $\widetilde{\Omega}\left(\varepsilon^{-(2+d)}\right)$. Hence, in terms of the dependence on the dimension, the instance we consider here is nearly optimal. We note that the $\widetilde{O}\left(\varepsilon^{-(4+d)}\right)$ sample complexity results from two parts: the MCTS contributes a sample complexity scaling as ε^{-2} due to simulating the search tree, while nearest neighbor requires $\varepsilon^{-(2+d)}$ samples due to the need of sufficiently many good neighbors. Obtaining tighter bound with potentially more powerful improvement module or supervised learning module such as neural networks is an interesting future avenue.

7 PROOF OF THEOREM 2

Proof. With the three detailed properties, the proof is conceptually straightforward. At each iteration, the improvement module would produce better estimates for the explored states, by factors of ζ_v and ζ_p. The exploration continues until one of the desired representative sets in $\mathcal{E}$ has been visited, and the exploration property guarantees that the exploration time will be finite. The current iteration then ends by calling the supervised learning module to generalize the improvement to the entire state space. In what follows, we make these statements formal.

Let us first introduce some notion. We will use the term iteration to refer to a complete round of improvement, exploration and supervised learning (cf. Line 2 of Algorithm 1). In general, at each iteration, we use a superscript (l) to denote quantities relevant to the l-th iteration, except that for the supervised learning module, we follow the convention in the paper and use a subscript l, i.e., $f_l = (V_l, \pi_l)$. We denote by $Z^{(l)}$ all the information during the l-th iteration. Let $\{\mathcal{F}^{(l)}\}$ be the sigma-algebra generated by the stochastic process $\{Z^{(l)}\}$, where the randomness comes from the environment and any randomness that may be used in the three modules. Let $\omega_v^{(l)}$ and $\omega_p^{(l)}$ be the estimation errors of the model at the beginning of the iteration, i.e.,

$$\omega_v^{(l)} \triangleq \mathbb{E}\left[\left\|V_{l-1} - V^*\right\|_\infty\right],$$

$$\omega_p^{(l)} \triangleq \sup_{s \in \mathcal{S}} \mathbb{E}\left[D_{\mathrm{KL}}\left(\pi_{l-1}(\cdot|s) \| P^*_\tau(\cdot|s)\right)\right].$$

We use $\mathcal{D}^{(l)} = \left\{\left(s_i, \hat{V}^{(l)}(s_i), \hat{\pi}^{(l}(\cdot|s_i)\right)\right\}_{i=1}^{n_l}$ to denote the set of training data generated by the exploration module and querying the improvement module during the l-th iteration. Let $S^{(l)} = \{s_i\}_{i=1}^{n_l}$ be the set of states visited by the exploration module. Correspondingly, the estimation errors for the value function and the optimal policy after querying the improvement module are denoted by $\varepsilon_v^{(l)}$ and $\varepsilon_p^{(l)}$, respectively:

$$\varepsilon_v^{(l)} = \sup_{s \in S^{(l)}} \mathbb{E}\left[\left|\hat{V}^{(l)}(s) - V^*(s)\right|\right],$$

$$\varepsilon_p^{(l)} = \sup_{s \in S^{(l)}} \mathbb{E}\left[D_{\mathrm{KL}}\left(\hat{\pi}^{(l)}(\cdot|s) \| P^*_\tau(\cdot|s)\right)\right].$$

At the l-th iteration, the supervised learning modules takes the outputs of the improvement module, $\mathcal{D}^{(l)}$, as the training data. Note that the estimation errors $\omega_v^{(l+1)}$ and $\omega_p^{(l+1)}$ for the new model $f_l = (V_l, \pi_l)$, after querying the supervised learning module, is given by:

$$\omega_v^{(l+1)} = \mathbb{E}\left[\left\|V_l - V^*\right\|_\infty\right],$$

$$\omega_p^{(l+1)} = \sup_{s \in S} \mathbb{E}\left[D_{\mathrm{KL}}\left(\pi_l(\cdot|s) \| P^*_\tau(\cdot|s)\right)\right].$$

First, the improvement property of the improvement module (cf. Property 1) implies that

$$\varepsilon_v^{(l)} \le \zeta_v \omega_v^{(l)}, \tag{9}$$

$$\varepsilon_p^{(l)} \le \zeta_p \omega_p^{(l)}. \tag{10}$$

For the supervised learning module, according to the generalization property (cf. Property 2), when the size of training set n_l is sufficiently large and the sampled states $S^{(l)} = \{s_i\}_{i=1}^{n_l}$ are representative of the state space, the same order of accuracy of the training data will be generalized to the entire state space. For now, let us assume that this is the case; we will come back to verify the generalization bound in Property 2 can indeed be satisfied by $S^{(l)}$. Then, the following bounds hold:

$$\omega_v^{(l+1)} \le c_v \varepsilon_v^{(l)} \le c_v \zeta_v \omega_v^{(l)},$$

$$\omega_p^{(l+1)} \le c_p \varepsilon_p^{(l)} \le c_p \zeta_p \omega_p^{(l)}.$$

Therefore, when querying the improvement module, if we select the improvement factors to be

$$\zeta_v = \rho / c_v \text{ and } \zeta_p = \rho / c_p, \tag{11}$$

then we have

$$\omega_v^{(l+1)} \le \rho \omega_v^{(l)}, \tag{12}$$

$$\omega_p^{(l+1)} \le \rho \omega_p^{(l)}. \tag{13}$$

It is worth taking note of the fact that c_v and c_p would be larger than 1 (cf. Property 2): a reasonable supervised learning model may generalize the same order of accuracy as training data, but unlikely for it be smaller; hence, ζ_v and ζ_p are required to be smaller than 1 in Property 1 so that $\rho < 1$.

By definition, $\omega_v^{(1)} = C_{0,v}$ and $\omega_p^{(1)} = C_{0,p}$. Therefore, we have the desired inequalities:

$$\mathbb{E}\Big[\|V_L - V^*\|_\infty\Big] \le C_{0,v}\rho^L$$

$$\mathbb{E}\Big[D_{\mathrm{KL}}\big(\pi_L(\cdot|s) \| P_\tau^*(\cdot|s)\big)\Big] \le C_{0,p}\rho^L, \quad \forall s \in \mathcal{S}.$$

Finally, to complete the proof, as we mentioned before, we need to verify that we could sample enough representative states at each iteration in finite time steps. This is indeed guaranteed by the exploration property. In particular, note that at the l-th iteration, we would like to sample enough representative states that are of errors $\zeta_v \omega_v^{(l)}$ and $\zeta_p \omega_p^{(l)}$ for the value and policy functions (cf. Eqs. (9) and (10)). By a recursive argument (cf. Eqs. (12) and (13)), it is not hard to see that at the l-th iteration, we need to query the exploration module until the sampled states, $S^{(l)} = \{s_i\}_{i=1}^{n_l}$, contain one of the representative sets in $\mathcal{E}(\tau, \zeta_v C_{0,v}\rho^{l-1}, \zeta_p C_{0,p}\rho^{l-1})$, i.e., we immediately stop querying the exploration module at time n_l when the following holds:

$$\exists\, \hat{S} \in \mathcal{E}(\tau, \zeta_v C_{0,v}\rho^{l-1}, \zeta_p C_{0,p}\rho^{l-1}) \text{ such that } \hat{S} \subset \{s_i\}_{i=1}^{n_l},$$

where ζ_v and ζ_p are given by Eq. (11). From the exploration property, we know that $\mathbb{E}\big[T\big(\tau, \zeta_v C_{0,v}\rho^{l-1}, \zeta_p C_{0,p}\rho^{l-1}\big)\big]$ is finite, which implies that n_l is also finite with high probability. Therefore, we are guaranteed that the training data $D^{(l)}$ contains one of the representative sets, and hence the supervised learning module generalizes at each iteration. This completes the proof of Theorem 2. □

8 PROOF OF PROPOSITION 3

To prove Proposition 3, we first establish the following lemma.

Lemma 9. *Consider the exploration module and suppose that* $\mathbb{E}[T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})] \le B(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$. *Then, with probability at least* $1 - \delta$,

$$T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) \le e \cdot B(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) \cdot \log \frac{1}{\delta}.$$

Proof of Lemma 9. Consider a total time steps of $n = eB(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) \log \frac{1}{\delta}$. All the states, $\{s_i\}_{i=1}^{n}$, are sampled via querying the exploration module. Let us divide the total time steps n into $M \triangleq \log(1/\delta)$ segments, each consisting of $J \triangleq eB(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$ states. Denote by $\mathcal{S}(m)$ the set of states in the m-th segment, i.e., $\mathcal{S}(m) = \{s_i\}_{i=(m-1)J}^{mJ-1}$. The key idea of the proof is to argue that with high probability, at least one of the sets $\mathcal{S}(m)$, $m = 1, 2, \ldots, M$ will contain a representative set in $\mathcal{E}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$.

Denote by E_m the event that the m-th segment does not contain any the representative sets, i.e.,

$$E_m = \{\, \nexists\, \hat{S} \in \mathcal{E}(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) \text{ such that } \hat{S} \in \mathcal{S}(m)\}.$$

Let $\mathcal{F}_m$ be the filtration containing information untill the end of segment m. Since $\mathbb{E}[T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})] \le B(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})$, by Markov inequality, we have,

$$\mathbb{P}\big(T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) \ge J + 1\big) \le \frac{\mathbb{E}[T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})]}{J+1} \le \frac{B(\tau, \varepsilon_{1,v}, \varepsilon_{1,p})}{J} = \frac{1}{e}.$$

This then implies that

$$\mathbb{P}(E_m | \mathcal{F}_{m-1}) \le \frac{1}{e}, \quad m \in [M].$$

Therefore,

$$\begin{aligned} &\mathbb{P}\Big(T(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) > e \cdot B(\tau, \varepsilon_{1,v}, \varepsilon_{1,p}) \cdot \log \frac{1}{\delta}\Big) \\ &\le \mathbb{P}(\,\forall\, m \in [M],\ E_m \text{ occurs}) \\ &\le (\frac{1}{e})^m \\ &= \delta, \end{aligned}$$

which completes the proof of Lemma 9. □

Proof of Proposition 3. With Lemma 9, we are now ready to prove Proposition 3. This is achieved by simply counting the sample complexity for each of the L iterations. As discussed in the convergence proof of Theorem 2, at the l-th iteration, we need to query the exploration module until the sampled states, $S^{(l)} = \{s_i\}_{i=1}^{n_l}$, contains one of the representative sets in $\mathcal{E}(\tau, C_{0,v}\rho^l / c_v, C_{0,p}\rho^l / c_p)$. For each of the explored states, a query of the improvement module incurs a deterministic sample complexity of

$$\kappa\Big(\tau, C_{0,v}\rho^{l-1}, C_{0,p}\rho^{l-1}, \frac{\rho}{c_v}, \frac{\rho}{c_p}\Big),$$

for the required improvement factors $\zeta_v = \rho / c_v$ and $\zeta_p = \rho / c_p$. Let us now apply Lemma 9. We have

$$\mathbb{P}\Big(n_l \le e \cdot B\Big(\tau, \frac{C_{0,v}\rho^l}{c_v}, \frac{C_{0,p}\rho^l}{c_p}\Big) \cdot \log \frac{L}{\delta}\Big) \ge 1 - \frac{\delta}{L}.$$

That is, with probability at most δ/L, the sample complexity of the l-th iteration is larger than

$$\kappa\Big(\tau, C_{0,v}\rho^{l-1}, C_{0,p}\rho^{l-1}, \frac{\rho}{c_v}, \frac{\rho}{c_p}\Big) \cdot B\Big(\tau, \frac{C_{0,v}\rho^l}{c_v}, \frac{C_{0,p}\rho^l}{c_p}\Big) \cdot e \cdot \log \frac{L}{\delta}.$$

Finally, applying union bound over the L iterations, we have

$$\begin{aligned}
&\mathbb{P}\Big(\ \exists\, l \in [L] \text{ such that } n_l > e \cdot B\Big(\tau, \frac{C_{0,v}\rho^l}{c_v}, \frac{C_{0,p}\rho^l}{c_p}\Big) \cdot \log \frac{L}{\delta}\Big) \\
\le &\sum_{l=1}^{L} \mathbb{P}\Big(n_l > e \cdot B\Big(\tau, \frac{C_{0,v}\rho^l}{c_v}, \frac{C_{0,p}\rho^l}{c_p}\Big) \cdot \log \frac{L}{\delta}\Big) \\
\le &L \cdot \frac{\delta}{L} \\
= &\delta.
\end{aligned}$$

Therefore, with probability at least $1-\delta$, for every $l \in [L]$,

$$n_l \le e \cdot B\Big(\tau, \frac{C_{0,v}\rho^l}{c_v}, \frac{C_{0,p}\rho^l}{c_p}\Big) \cdot \log \frac{L}{\delta}.$$

Equivalently, with probability at least $1-\delta$, the total sample complexity is upper bounded by

$$\sum_{l=1}^{L} \kappa\Big(\tau, C_{0,v}\rho^{l-1}, C_{0,p}\rho^{l-1}, \frac{\rho}{c_v}, \frac{\rho}{c_p}\Big) \cdot B\Big(\tau, \frac{C_{0,v}\rho^l}{c_v}, \frac{C_{0,p}\rho^l}{c_p}\Big) \cdot e \cdot \log \frac{L}{\delta}.$$

□

9 CONCLUSION

In this paper, we take theoretical steps towards understanding reinforcement learning for zero-sum turn-based Markov games. We develop the Explore-Improve-Supervise (EIS) method with three intuitive modules intertwined carefully. Such an abstraction of three key modules allows us to isolate the fundamental principles from the implementation details. Importantly, we identify conditions for successfully finding the optimal solutions, backed by a concrete instance satisfying those conditions. Overall, the abstraction and the generic properties developed in this paper could serve as some guidelines, with the potential of finding broader applications with different instantiations. Finally, it would be interesting to extend this framework to general Markov games with simultaneous moves. We believe the generic modeling techniques in Section 4 could be applied, but a key challenge is to develop an improvement module with rigorous non-asymptotic guarantees that satisfies the desired property. We believe that addressing this challenge and formally establishing the framework is a fruitful future direction.

REFERENCES

[1] Michael Bowling and Manuela Veloso. 2001. Rational and convergent learning in stochastic games. In *International joint conference on artificial intelligence*, Vol. 17. Lawrence Erlbaum Associates Ltd, 1021–1026.

[2] Ronen I Brafman and Moshe Tennenholtz. 2002. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3, Oct (2002), 213–231.

[3] Thomas Dueholm Hansen, Peter Bro Miltersen, and Uri Zwick. 2013. Strategy Iteration Is Strongly Polynomial for 2-Player Turn-Based Stochastic Games with a Constant Discount Factor. *J. ACM* 60, 1, Article 1 (Feb. 2013), 16 pages.

[4] Junling Hu and Michael P Wellman. 2003. Nash Q-learning for general-sum stochastic games. *Journal of machine learning research* 4, Nov (2003), 1039–1069.

[5] Junling Hu, Michael P Wellman, et al. 1998. Multiagent reinforcement learning: theoretical framework and an algorithm.. In *ICML*, Vol. 98. Citeseer, 242–250.

[6] Zeyu Jia, Lin F. Yang, and Mengdi Wang. 2019. Feature-Based Q-Learning for Two-Player Stochastic Games. arXiv:1906.00423 [cs.LG]

[7] Emilie Kaufmann and Wouter M. Koolen. 2017. Monte-carlo tree search by best arm identification. In *Advances in Neural Information Processing Systems*. 4897–4906.

[8] Michael Kearns, Yishay Mansour, and Andrew Y Ng. 2002. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine learning* 49, 2-3 (2002), 193–208.

[9] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. 2006. Improved Monte-Carlo search. *Univ. Tartu, Estonia, Tech. Rep* (2006).

[10] Panganamala R. Kumar and Tzong-Huei Shiau. 1981. Existence of value and randomized strategies in zero-sum discrete-time stochastic dynamic games. *SIAM Journal on Control and Optimization* 19, 5 (1981), 617–634.

[11] Michail G Lagoudakis and Ronald Parr. 2002. Value function approximation in zero-sum markov games. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 283–292.

[12] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*. Elsevier, 157–163.

[13] Michael L Littman. 2001. Friend-or-foe Q-learning in general-sum games. In *ICML*, Vol. 1. 322–328.

[14] Michael L Littman. 2001. Value-function reinforcement learning in Markov games. *Cognitive Systems Research* 2, 1 (2001), 55–66.

[15] A. Maitra and T. Parthasarathy. 1970. On stochastic games. *Journal of Optimization Theory and Applications* 5, 4 (1970), 289–300.

[16] A. Maitra and T. Parthasarathy. 1971. On stochastic games, II. *Journal of Optimization Theory and Applications* 8, 2 (1971), 154–160.

[17] T Parthasarathy. 1973. Discounted, positive, and noncooperative stochastic games. *International Journal of Game Theory* 2, 1 (1973), 25–37.

[18] Stephen D. Patek. 1997. *Stochastic Shortest Path Games: Theory and Algorithms*. PhD dissertation. Massachusetts Institute of Technology.

[19] Julien Pérolat, Bilal Piot, Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. 2016. Softened approximate policy iteration for Markov games. In *ICML 2016-33rd International Conference on Machine Learning*.

[20] Julien Perolat, Bruno Scherrer, Bilal Piot, and Olivier Pietquin. 2015. Approximate Dynamic Programming for Two-Player Zero-Sum Markov Games. In *Proceedings of the 32nd International Conference on Machine Learning*. 1321–1329.

[21] Devavrat Shah, Qiaomin Xie, and Zhi Xu. 2020. Non-Asymptotic Analysis of Monte Carlo Tree Search. *ACM SIGMETRICS 2020, arXiv:1902.05213* (2020).

[22] Lloyd S. Shapley. 1953. Stochastic games. *Proceedings of the National Academy of Sciences* 39, 10 (1953), 1095–1100.

[23] Aaron Sidford, Mengdi Wang, Lin F. Yang, and Yinyu Ye. 2019. Solving Discounted Stochastic Two-Player Games with Near-Optimal Time and Sample Complexity. arXiv:1908.11071 [cs.LG]

[24] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[25] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[26] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (2017), 354.

[27] Charles J. Stone. 1982. Optimal Global Rates of Convergence for Nonparametric Regression. *The Annals of Statistics* (1982), 1040–1053.

[28] Csaba Szepesvári and Michael L. Littman. 1996. Generalized Markov decision processes: Dynamic-programming and reinforcement-learning algorithms. In *Proceedings of International Conference of Machine Learning*, Vol. 96.

[29] Alexandre B. Tsybakov. 2009. *Introduction to Nonparametric Estimation*. Springer.

[30] Martin J. Wainwright. 2019. *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge University Press.

[31] Chen-Yu Wei, Yi-Te Hong, and Chi-Jen Lu. 2017. Online reinforcement learning in stochastic games. In *Advances in Neural Information Processing Systems*. 4987–4997.

[32] Zhuora Yang, Yuchen Xie, and Zhaoran Wang. 2019. A Theoretical Analysis of Deep Q-Learning. arXiv:1901.00137 [cs.LG]