Bayesian Multiagent Inverse Reinforcement Learning for Policy Recommendation

Carlos Martin,[1] **Tuomas Sandholm**[1,2,3,4]
[1] Carnegie Mellon University
[2] Strategy Robot, Inc.
[3] Optimized Markets, Inc.
[4] Strategic Machine, Inc.
cgmartin@cs.cmu.edu, sandholm@cs.cmu.edu

Abstract

We study the following problem, which to our knowledge has been addressed only partially in the literature and not in full generality. An agent observes two players play a zero-sum game that is known to the players but not the agent. The agent observes the actions and state transitions of their game play, but not rewards. The players may play either optimally (according to some Nash equilibrium) or according to any other solution concept, such as a quantal response equilibrium. Following these observations, the agent must recommend a policy for one player, say Player 1. The goal is to recommend a policy that is minimally exploitable under the true, but unknown, game. We take a Bayesian approach. We establish a likelihood function based on observations and the specified solution concept. We then propose an approach based on Markov chain Monte Carlo (MCMC), which allows us to approximately sample games from the agent's posterior belief distribution. Once we have a batch of independent samples from the posterior, we use linear programming and backward induction to compute a policy for Player 1 that minimizes the sum of exploitabilities over these games. This approximates the policy that minimizes the expected exploitability under the full distribution. Our approach is also capable of handling counterfactuals, where known modifications are applied to the unknown game. We show that our Bayesian MCMC-based technique outperforms two other techniques—one based on the equilibrium policy of the maximum-probability game and the other based on imitation of observed behavior—on all the tested stochastic game environments.

Introduction

Multiagent reinforcement learning (MRL) extends reinforcement learning to multiple agents, and its environments are typically formulated as repeated games (Sandholm and Crites 1996) or more generally as stochastic games (Shapley 1953), also known as Markov games. For stochastic games, Littman (1994) studies the two-player zero-sum case. Hu and Wellman (2003) extend this to the general-sum case, adopting the game-theoretic solution concept of the Nash equilibrium (Nash 1950, 1951), in which each agent's strategy is a best response to the other agents' strategies.

Inverse reinforcement learning (IRL) aims to recover the reward (a.k.a. payoff) function of an agent given observations of its behavior. IRL was introduced by Russell (1998) and formalized by Ng and Russell (2000). IRL may be useful for apprenticeship learning to acquire skilled behaviour, and for ascertaining the reward function being optimized by a natural system. As Ng and Russell (2000) point out, a major advantage of IRL is that, in many applications, the reward function provides a parsimonious description of behavior that is succinct, robust, and *transferable with respect to changes in the environment*. It can also yield insights into the value systems driving agent behavior.

Most of the IRL literature assumes a single-agent setting. Yet many real-world applications involve multiple agents. The presence of these other agents makes the environment, from the perspective of any one agent, potentially non-stationary because the other agents might be learning and thus changing their strategies (e.g., Sandholm and Crites (1996)). So, different techniques are needed that take into account the decision-making processes of other agents.

Multiagent inverse reinforcement learning (MIRL) extends IRL to multiple agents. The canonical MIRL problem is estimating the payoffs of a stochastic game given observations of the actions taken by the players and their state transitions. This brings several new challenges. For one, the concept of single-agent optimality must be replaced with a multiagent notion of optimal behavior, such as a Nash equilibrium (Hu and Wellman 2003) or quantal response equilibrium (McKelvey and Palfrey 1995; Mckelvey and Palfrey 1998).

Reddy et al. (2012) study MIRL to learn the reward function in a setting where the agents can either cooperate or be strictly non-cooperative. They assume that the policies of the agents are known and that the agents are rational and follow an optimal policy in the sense of Nash equilibrium. Under those assumptions, they reduce the problem to a distributed solution where the reward function for each agent can be solved independently using a similar formulation as in the single-agent case.

Ling, Fang, and Kolter (2018) tackle the problem of learning the parameters of an unknown game, such as payoffs or chance node probabilities, from observed actions. Their goal is to maximize the likelihood of realizing the observed sequence from the player, assuming they act according to a quantal response equilibrium. To do this, they consider a regularized version of the game that is equivalent to the quantal response equilibrium and develop a primal-dual Newton method for finding the solution. They also develop a backpropagation method that analytically computes gradients of all relevant game parameters through the solution itself. This lets them learn the game by incorporating the solver into the loop of larger deep network architectures and training in an end-to-end fashion.

Wang and Klabjan (2018) study MIRL in zero-sum stochastic games when expert demonstrations are known to be suboptimal. They present an algorithm for estimating (using deep learning) payoffs so that the players' observed play is close to what Nash equilibrium policies would be under those payoffs. Their approach is not Bayesian. Lin, Adams, and Beling (2019) study MIRL in two-player general-sum stochastic games. They consider five variants of MIRL, distinguished by solution concept used. That work assumes that the game observer either knows, or is able to accurately estimate, the policies and solution concepts of the players. In a very different direction, Zhang et al. (2019) study IRL in two-player zero-sum setting where only one of the agents knows the utility function. By interacting with the informed player, the uninformed player attempts to both infer and optimize its objective.

In this paper, we extend MIRL beyond learning the game to making policy (strategy) recommendations, using a Bayesian approach. Specifically, we study the problem of recommending a policy for a zero-sum stochastic game with unknown parameters based only on observations of game play. We observe the actions and state transitions (but not rewards) incurred during game play by two players playing the game. These players might be playing according to a Nash equilibrium or according to any other game-theoretic solution concept, such as a quantal response equilibrium. Our objective is to minimize the expected exploitability of our recommended policy under the true, but unknown, game.

To define the posterior distribution over the unknown game parameters, we require a likelihood function that tells us the probability of our observations given a candidate game. One of our recommendation strategies also requires sampling from the posterior distribution, for which we use *Markov chain Monte Carlo (MCMC)*. Once we have a batch of independent samples from the posterior, we use linear programming and backward induction to compute a policy for Player 1 that minimizes the sum of exploitability over these games. This approximates the policy that minimizes the expected exploitability under the full distribution.

We show that our Bayesian MCMC-based technique outperforms two other techniques—one based on the equilibrium policy of the maximum-probability game and the other based on imitation of observed behavior—on all the tested stochastic game environments. Our approach can also handle the case where we want to recommend a strategy for a known modification of the unknown game.

In this work, we take an emphatically *instrumental* view of IRL. The reason we are interested in the true parameters (e.g., rewards) of the game is because we are interested in *doing* something with this knowledge. We would like to recommend a minimally-exploitable policy for Player 1 under the same unknown game or a known modification thereof.

In terms of goals, the closest prior work is that of Lin, Beling, and Cogill (2018). They propose a Bayesian approach to MIRL and establish a theoretical foundation for two-agent zero-sum MIRL problems. Their generative model is based on an assumption that the two agents follow a minimax policy profile; our approach works with a broad range of game-theoretic behavioral models. Like us, they work in the context of stochastic games. However, their aim is to estimate what the true reward function of the stochastic game is. That problem was previously studied in the one-stage setting by Waugh, Ziebart, and Bagnell (2011) and in the setting of non-competing agents by Natarajan et al. (2010). Our goal is different and more end to end: making a good policy recommendation. Another difference is that Lin, Beling, and Cogill (2018) measure the quality of learned rewards using distance metrics in reward and probability space, as well as the game play performance of agents using those learned rewards as the basis for an equilibrium policy. Specifically, they use the average reward distance (the average Euclidean distance from the true rewards) and a domain-specific evaluation metric. A further difference is that their model assumes that the *complete bi-policy* of the two players is observed. We only observe the players' actions. Their approach also requires knowing the state transition probabilities, whereas in our work these must also be inferred.

Again, we emphasize that the recommender is *not* either of the players who played the game. They are a third-party observer, one who does not know the true game and does not know what rewards the players received. We are also dealing with an *offline* setting. That is, the recommender cannot interact with the game. They only have empirical observations of gameplay. Therefore, they cannot use standard reinforcement learning to learn Player 1's optimal policy, because they have no ability to interact with the environment at all.

Zero-sum stochastic games

Let $\triangle\mathcal{X}$ denote the set of probability distributions on a set $\mathcal{X}$. Let $[n] = \{0, \ldots, n-1\}$ for $n : \mathbb{N}$.

A zero-sum finite-horizon stochastic game is a tuple $g = (\mathcal{S}, s, \mathcal{A}_1, \mathcal{A}_2, R, T, H) : \mathcal{G}$ where $\mathcal{S}$ is a set of states, $s : \mathcal{S}$ is the initial state, $\mathcal{A}_i$ is the set of actions available to Player i, $R : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \to \mathbb{R}$ is the reward function (which yields a reward to Player 1 for every state and action profile), $T : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \to \triangle\mathcal{S}$ is the state transition function (which yields a distribution of next states for every state and action profile), and $H : \mathbb{N}$ is the game's time horizon (which is the duration of each episode in timesteps).

A Player i policy is a function $\pi_i : \Pi_i \overset{\text{def}}{=} [H] \times \mathcal{S} \to \triangle\mathcal{A}_i$ that yields a distribution of actions for every time horizon (remaining number of timesteps) and state. A policy profile is a policy for each player. The expected return of policy profile (π_1, π_2) in game g is

$$u(g, \pi_1, \pi_2) = \mathop{\mathbb{E}}_{\substack{(a_t)_i \sim \pi_i(H-1-t, s_t) \\ s_{t+1} \sim T(s_t, a_t)}} \sum_{t=0}^{H-1} R(s_t, a_t) \quad (1)$$

for Player 1 and $-u(g, \pi_1, \pi_2)$ for Player 2. Under the assumption that Player 2 plays optimally, the utility to Player 1 of policy π_1 is $u(g, \pi_1) = \min_{\pi_2 : \Pi_2} u(g, \pi_1, \pi_2)$. The optimal policy is $\pi_1^{\mathrm{O}} = \operatorname{argmax}_{\pi_1 : \Pi_1} u(g, \pi_1)$. The *regret* incurred by a policy π_1 is $R(g, \pi_1) = u(g, \pi_1^{\mathrm{O}}) - u(g, \pi_1)$.

In the special case $|\mathcal{S}| = 1$ we have a repeated game (Sandholm and Crites 1996). If in addition $H = 1$, we have a normal-form game. In the special case $|\mathcal{A}_2| = 1$ we have a single-player Markov decision process. If both of the above conditions hold, we have a multi-armed bandit. In the special case where the state transition graph induced by T is a tree, we have a perfect-information extensive-form game.

Policy recommendation under uncertainty

In this section we present three ways of recommending a policy in the end given our final pre-recommendation belief distribution over games. Later we show how the belief distribution is constructed from observations.

Bayesian recommendation

Suppose we are uncertain about some aspects of the game, such as its rewards or state transition probabilities. Our beliefs can be modelled by a belief distribution $D : \triangle\mathcal{G}$ over games. Given this belief distribution, what policy should we recommend for Player 1? We want to maximize expected utility, so we should recommend

$$\pi_1^{\mathrm{B}} = \operatorname*{argmax}_{\pi_1 : \Pi_1} \mathop{\mathbb{E}}_{g \sim D} \min_{\pi_2 : \Pi_2} u(g, \pi_1). \tag{2}$$

We call this the *Bayesian* recommendation.

Since we lack a closed-form solution for π_1^{B} under general distributions D, we replace it with the approximation that is obtained by replacing the expectation with a Monte Carlo estimator (an empirical average):

$$\pi_1^{\mathrm{MCB}} = \operatorname*{argmax}_{\pi_1 : \Pi_1} \sum_{(j,g) : B} \min_{\pi_2 : \Pi_2} u(g, \pi_1, \pi_2) \tag{3}$$

where B is a batch of independent samples from D.

We can compute π_1^{MCB} as follows. Let $R(j)$ and $T(j)$ be the reward and transition functions of $B(j)$. The V function yields the expected utility for Player 1 in a given game when starting from a given horizon and state: $V : \operatorname{dom} B \times [H] \times \mathcal{S} \to \mathbb{R}$. The Q function yields the expected utility for Player 1 in a given game when starting from a given horizon, state, *and* action profile: $Q : \operatorname{dom} B \times [H] \times \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \to \mathbb{R}$. Player 1's recommended max-sum-min policy is $\pi_1 : [H] \times \mathcal{S} \to \triangle\mathcal{A}_1$ Player 2's best-response policy in each game in the game batch B is $\pi_2 : \operatorname{dom} B \times [H] \times \mathcal{S} \to \triangle\mathcal{A}_2$.

We compute π_1 using backward induction as follows. We initialize $V(j, 0, s) = 0$ and repeat

$$\begin{aligned} Q(j, h, s, a_1, a_2) = R(j, s, a_1, a_2) + \\ \sum_{s' : \mathcal{S}} T(j, s, a_1, a_2, s') V(j, h, s') \end{aligned} \tag{4}$$

$$\begin{aligned} \pi_1(h, s) &= \operatorname*{argmax}_{\sigma_1 : \triangle\mathcal{A}_1} \sum_{j : \operatorname{dom} B} \min_{\sigma_2 : \triangle\mathcal{A}_2} Q(j, h, s, \sigma_1, \sigma_2) \\ \pi_2(j, h, s) &= \operatorname*{argmin}_{\sigma_2 : \triangle\mathcal{A}_2} Q(j, h, s, \pi_1(h, s), \sigma_2) \\ V(j, h+1, s) &= Q(j, h, s, \pi_1(h, s), \pi_2(j, h, s)) \end{aligned} \tag{5}$$

from $h = 0$ to $H - 1$ (inclusive), where

$$Q(\ldots, \sigma_1, \sigma_2) \stackrel{\text{def}}{=} \sum_{a_1 : \mathcal{A}_1} \sum_{a_2 : \mathcal{A}_2} \sigma_1(a_1) \sigma_2(a_2) Q(\ldots, a_1, a_2) \tag{6}$$

for $\sigma_i : \triangle\mathcal{A}_i$. To compute $\pi_1(h, s)$, we solve the following linear program over variables $\sigma_1 : \mathbb{R}^{\mathcal{A}_1}$ and $\mathbf{v} : \mathbb{R}^{\operatorname{dom} B}$.

$$\begin{array}{ll} \text{maximize} & \mathbf{1} \cdot \mathbf{v} \\ \text{subject to} & \mathbf{1} \cdot \sigma_1 = 1 \\ & \sigma_1 \geq \mathbf{0} \\ & \mathbf{v}(j) \leq Q(j, \ldots, \sigma_1, a_2) \quad \forall j : \operatorname{dom} B, a_2 : \mathcal{A}_2 \end{array} \tag{7}$$

Then π_1^{MCB} is the obtained π_1. This algorithm also lets us compute π_1^{O} by letting B contain just the true game g.

Maximum probability recommendation

The Bayesian recommendation is very different from maximizing utility under the *most likely* game, which is

$$\pi_1^{\mathrm{MP}} = \operatorname*{argmax}_{\pi_1 : \Pi_2} \min_{\pi_2 : \Pi_2} u(g^{\mathrm{MP}}, \pi_1, \pi_2) \tag{8}$$

where $g^{\mathrm{MP}} = \operatorname{argmax}_{g : \mathcal{G}} p(g)$ is the most likely game. The latter is the objective sought by Lin, Beling, and Cogill (2018), where selected rewards maximize the posterior of the observed state-action pairs. We call this the *maximum probability* recommendation. For our problem, it is suboptimal, since it does not maximize expected utility.

For a concrete example, suppose that Player 1 faces a multi-armed bandit with two actions. We believe its rewards are $(1, 0)$ with 60% probability and $(0, 2)$ with 40% probability. The maximum probability recommendation is to play the first action, which yields an expected payoff of 0.6, while the Bayesian recommendation is to play the second action, which yields a higher expected payoff of 0.8.

Computing π_1^{MP} requires finding the global maximum of D. To do this, we use the *simplicial homology global optimisation (SHGO)* algorithm (Endres, Sandrock, and Focke 2018) as implemented in SciPy 1.5.2 (Virtanen et al. 2020), an open-source Python library for scientific computing. SHGO is a general-purpose, derivative-free global optimisation algorithm based on simplicial integral homology and combinatorial topology.

Imitation recommendation

This recommendation simply tries to imitate Player 1's policy based on the empirical frequencies of its actions:

$$\pi_1^{\mathrm{I}}(h, s, a_1) = \frac{\alpha + n(h, s, a_1)}{\sum_{a_1' : \mathcal{A}_1} (\alpha + n(h, s, a_1'))} \tag{9}$$

where $n(h, s, a_1)$ is the number of times Player 1 has played a_1 at time horizon h and state s. The pseudocount $\alpha > 0$ is an additive smoothing parameter. From a Bayesian perspective, this can be interpreted as maintaining separate and independent strategy distributions for each time horizon and state. Each such distribution starts as a symmetric Dirichlet distribution with concentration parameter α and is updated according to Player 1's actions. We use $\alpha = 1$, which is the uniform Dirichlet distribution.

Unlike the other two approaches, which are *model-based* (i.e., they explicitly model the game or a distribution thereof), this approach cannot handle counterfactuals. The other two approaches can handle the scenario where a *known modification* or transformation $f : \mathcal{G} \to \mathcal{G}$ is applied to the unknown game. A real-world example of such a known modification might be the introduction of an obstacle, elimination of a pathway, or other change in environmental conditions. Since the imitation recommendation simply tries to imitate Player 1's policy under the *original* game, it can become useless under the *modified* game.

Belief distribution: Concept and computation

We now describe how the belief distribution D is determined and computed in our setting after we have observed the two players play the unknown game.

Before observing the players' game play, we start with some initial distribution over games—reflecting our prior beliefs. This prior can be as informative or uninformative as desired, depending on our *a priori* knowledge of the game environment. For example, we might place a Gaussian prior on the rewards for a particular state, or a Dirichlet prior on the transition probabilities for a different state. Our Bayesian framework is flexible in this regard, since it allows us to incorporate any useful information into the prior.

Bayes' theorem tells us that our *posterior* distribution—that is, our distribution after making observations of the two players' game play—is proportional (as a function of the game) to the product of the prior and the *likelihood*.

$$\underbrace{p(\text{game} \mid \text{observations})}_{\text{posterior}} \propto \underbrace{p(\text{observations} \mid \text{game})}_{\text{likelihood}} \underbrace{p(\text{game})}_{\text{prior}} \tag{10}$$

The likelihood of a game g tells us the probability that we would have observed the behavior we did observe *if this had been the true game*.

Our observations of the two players' game play constitute a sequence of *observation tuples*. Each observation tuple (h, s, a_1, a_2, s') consists of the current horizon (number of remaining time steps) h, the current state s, Player 1's action a_1, Player 2's action a_2, and the next state s'.

Using the chain rule for probability and the Markov property of the environment (state transition probabilities depend only on the current state and action profile, not the number of remaining time steps), we have

$$\begin{aligned} &p(s', a_1, a_2 \mid h, s) \\ &\qquad = p(s' \mid h, s, a_1, a_2) p(a_1, a_2 \mid h, s) \\ &\qquad = p(s' \mid s, a_1, a_2) p(a_1 \mid h, s) p(a_2 \mid h, s) \end{aligned} \tag{11}$$

To get the likelihood, we take the product of this expression over all observation tuples. As this expression shows, there are three components to the likelihood. The first component is the probabilities of the observed state transitions *given* current states and action profiles. This component is purely a function of the environment itself (more precisely, its state transition function T) and does not depend on the players' policies: $p(s' \mid s, a_1, a_2) = T(s, a_1, a_2)(s')$.

The second and third components are the probabilities of the observed actions *given* current states and time horizons. These depend on the players' policies: $p(a_i \mid h, s) = \pi_i(h, s)(a_i)$ So, we must derive the policies for both players under this game. This is a function of the players' *behavior model*. For example, we may assume they are playing rationally according to a Nash equilibrium, or according to a more relaxed game-theoretic solution concept such as a quantal response equilibrium. We cover these in detail later.

A more concise representation of observations is in terms of *transition counts*. Let $n(h, s, a_1, a_2, s')$ denote the number of times (h, s, a_1, a_2, s') is observed. Missing entries imply summation over those entries, for example

$$n(s, a_1, a_2) = \sum_{h:[H]} \sum_{s':\mathcal{S}} n(h, s, a_1, a_2, s') \tag{12}$$

If the true transition function were T, the counts for the next state s' would follow a multinomial distribution whose probabilities are $T(s, a_1, a_2)$:

$$n(s, a_1, a_2, s') \sim \operatorname{multinomial}(T(s, a_1, a_2), n(s, a_1, a_2)) \tag{13}$$

and the counts for Player i's action a_i would follow a multinomial distribution whose probabilities are $\pi_i(h, s)$:

$$n(h, s, a_i) \sim \operatorname{multinomial}(\pi_i(h, s), n(h, s)). \tag{14}$$

In general, if $x \sim \operatorname{multinomial}\left(\theta, \sum_{i:[k]} x_i\right)$ where $x : \mathbb{N}^k$ and $\theta : \triangle[k]$, then the probability mass function is

$$p(x) = \frac{(\sum_{i:[k]} x_i)!}{\prod_{i:[k]} x_i!} \prod_{i:[k]} \theta_i^{x_i} \tag{15}$$

In computations, we work with the *logarithms* of probabilities to avoid numerical issues with underflow and overflow. The likelihood tends to become more sharply peaked around the true game as the number of observations increases. Figure 4 illustrates an example of how the likelihood evolves when Nash equilibrium play is observed for a normal-form game with two unknown parameters.

Nash equilibrium policies

We compute π_1 and π_2 using backward induction as follows. Let $V : [H] \times \mathcal{S} \to \mathbb{R}$, $Q : [H] \times \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \to \mathbb{R}$, and $[H] \times \mathcal{S} \to \triangle\mathcal{A}_i$, as before. Initialize $V(0, s) = 0$ and repeat

$$\begin{aligned} Q(h, s, a_1, a_2) &= R(s, a_1, a_2) + \\ &\qquad \sum_{s':\mathcal{S}} T(s, a_1, a_2, s') V(h, s') \\ (\pi_1(h, s), \pi_2(h, s)) &= \operatorname*{argNash}_{(\sigma_1,\sigma_2):\triangle\mathcal{A}_1 \times \triangle\mathcal{A}_2} Q(h, s, \cdot, \cdot) \\ V(h+1, s) &= Q(h, s, \pi_1(h, s), \pi_2(h, s)) \end{aligned} \tag{16}$$

from $h = 0$ to $H - 1$ (inclusive), where argNash denotes the Nash equilibrium strategies of the specified normal-form game. These strategies can be computed by solving the linear program in Equation 7 with $|\operatorname{dom} B| = 1$. Player 2's strategy $\sigma_2 : \triangle\mathcal{A}_2$ is then contained in the the dual variables of the solution that correspond to the last inequality.

Quantal response equilibrium policies

The *quantal response equilibrium (QRE)* is a solution concept in game theory, like Nash equilibrium. It applies quantal choice analysis (McFadden 1976) to the game-theoretic setting. It was first defined for normal-form games in McKelvey and Palfrey (1995) and extensive-form games in Mckelvey and Palfrey (1998).

QRE can model situations where payoff matrices are injected with noise, or where players are boundedly rational. Its smoothness makes gradient-based approaches feasible (Amin, Singh, and Wellman 2016). The most common type of QRE is a *logit equilibrium (LQRE)*, where we have the fixpoint equations

$$\sigma_i(a_i) = \frac{\exp \lambda u_i(a_i, \sigma_{-i})}{\sum_{a_i'} \exp \lambda u_i(a_i', \sigma_{-i})} \tag{17}$$

over all players i, where σ_i is Player i's strategy and $u_i(a_i, \sigma_{-i})$ is their expected utility under action a_i and the other players' strategy profile σ_{-i}.

The number $\lambda \geq 0$ acts a rationality parameter. As $\lambda \to 0$, the players become completely non-rational and play each action with equal probability. As $\lambda \to \infty$, they become rational and approach a Nash equilibrium.

For a zero-sum normal-form game with payoff matrix $P : \mathbb{R}^{n\times m}$, the LQRE (σ_1, σ_2) satisfies

$$\sigma_1 = \operatorname{softmax}(P \cdot \sigma_2) \text{ and } \sigma_2 = \operatorname{softmax}(-P^{\mathrm{T}} \cdot \sigma_1) \tag{18}$$

$$\text{where} \quad \operatorname{softmax}(x)_i = \frac{\exp x_i}{\sum_j \exp x_j}. \tag{19}$$

This is equivalent to solving the regularized max-min game

$$\begin{aligned} &\max_{x:\mathbb{R}^n} \min_{y:\mathbb{R}^m} \quad x^{\mathrm{T}} P y + H(x) - H(y) \\ &\text{subject to} \quad 1^{\mathrm{T}} x = 1, \quad 1^{\mathrm{T}} y = 1, \quad x \geq 0, \quad y \geq 0 \end{aligned} \tag{20}$$

where $H(x)$ is the Gibbs entropy $\sum_i x_i \log x_i$. Entropy regularization encourages players to play more randomly and no action has zero probability. Furthermore, since the objective is strictly a convex-concave problem, it has a *unique* saddle point (x, y), which is the LQRE.

Ling, Fang, and Kolter (2018) compute this saddle point using a primal-dual Newton method. They also present the gradient with respect to P in terms of the obtained solution and gradients with respect to x and y, making the whole procedure end-to-end differentiable. This means it can be integrated into differentiable learning procedures. It also opens the door to the use of gradient-based MCMC approaches.

In our stochastic game setting, we define the LQRE as the policies derived by the backward induction procedure we used to find the Nash equilibrium policies, except we replace the strategies yielded by $\operatorname{argNash}$ with the strategies yielded by the normal-form game LQRE on $Q(h, s, \cdot, \cdot)$.

Sampling from the belief distribution

To compute π_1^{MCB}, we must sample from D, the posterior belief distribution $p(\text{game} \mid \text{observations})$. One way to do this is to sample from the prior $p(\text{game})$ and then reweigh the sample's contribution to the expectation according to the likelihood $p(\text{observations} \mid \text{game})$. The problem is that the likelihood becomes very sharply peaked as the number of observations grows (that is, fewer and fewer hypotheses are able to explain the data well), so the likelihood is effectively zero for the vast majority of samples from the prior (Figure 4), rendering the Monte Carlo estimate useless.

We could try using importance sampling to bias the distribution we sample from (and rescale the weights of the expectation accordingly) towards regions of higher posterior probability. The problem with importance sampling is that, in high-dimensional problems, it requires very careful tuning of the proposal distribution. Importance weights tend to blow up exponentially with dimensionality and it is easy for the variance of the expectation estimator to diverge.

A different approach to the problem is *Markov chain Monte Carlo (MCMC)*. MCMC methods are a class of algorithms for sampling from a probability distribution by constructing a Markov chain over the sample space whose limit distribution is the desired distribution f. That is, $\lim_{t\to\infty} p(x_t = x) = f(x)$. To do this, we only need the ability to query a function *proportional* to the desired distribution. In our case, this means we only need the prior and likelihood, and not the evidence $p(\text{observations})$, which would require computing an intractable integral.

The more MCMC steps are included, the more closely the distribution of samples matches the actual desired distribution. One MCMC method is the Metropolis-Hastings algorithm (Metropolis et al. 1953; Hastings 1970). Figure 5 shows an ensemble of walkers evolving according to that algorithm. The proposal distribution used was a Gaussian distribution with variance 0.01. After many steps, the walkers are approximately distributed according to the target distribution. Therefore, to approximate a desired expectation, one can average over the points where the walkers are located.

Metropolis-Hastings requires choosing a proposal distribution. A bad proposal distribution may cause the chain to take a long time to converge. For example, suppose the target distribution is a very elongated Gaussian but the proposal distribution is circular. If the standard deviation of the latter is small, it will take a long time to traverse the space. If the standard deviation is large, the walker will frequently move perpendicularly to the elongation into regions of very low probability, resulting in high rejection rates.

Many other MCMC techniques and variants have been developed, such as the Metropolis-adjusted Langevin algorithm (Roberts and Tweedie 1996), parallel tempering (Swendsen and Wang 1986; Geyer 1991), Hamiltonian Monte Carlo (Duane et al. 1987; Neal 2012), No-U-Turn Sampling (Hoffman and Gelman 2014), etc. Another example is the Affine-Invariant Ensemble Sampler (AIES) proposed by Goodman and Weare (2010). We use a well-tested Python implementation of this algorithm called `emcee` (Foreman-Mackey et al. 2013, 2019) in our experiments.

Experiments

We compare the performance of our recommendation strategies on various stochastic games, evaluating the *regret* of the recommended policy.

One class of games we use as a benchmark are randomly-generated stochastic games. For each state and action profile, transition probabilities are sampled from the uniform Dirichlet distribution and rewards are sampled from the standard uniform distribution. The games have 3 states, 3 actions per player at each state, and 10 time steps. 10 episodes of game play were observed under an LQRE rationality parameter of 10 for both players. We used 100 walkers and did 10 trials. We let the unknown parameters be the rewards, using the standard uniform distribution as their prior. We let the modified game be the same game with the rewards negated, effectively changing Player 1's goal from reward maximization to reward minimization. Figure 1 shows the performance of each recommendation strategy on two such games with different random seeds. Lines indicate the median and bands indicate the 25th and 75th percentiles.

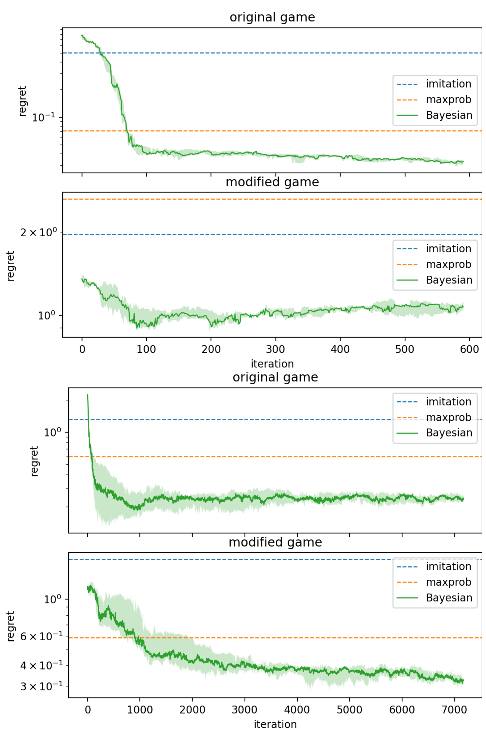

Figure 1: Performance on randomly-generated games.

The MCB recommendation outperforms both the imitation and maximum probability recommendations in both the original game and the modified game, after enough MCMC iterations are performed for sufficient mixing.

We also created a stochastic game, *bombardment game* (Figure 3), as a more structured benchmark. It is a gridworld-like environment in which Player 1 controls an entity that starts in the top left corner and moves around a maze for H time steps. In each step, Player 1 can choose to stay put or move in one of four cardinal directions. Player 2 (the crosshair) simultaneously chooses to target either Player 1's current position or one of its 4 neighboring positions. Player 1 receives a reward of -1 whenever Player 2's crosshair coincides with Player 1's next position. Therefore, in order to minimize damage, Player 1 should move with some degree of unpredictability.

Each grid tile has an associated reward that is sampled from the standard uniform distribution when the game is created. We let the unknown parameters be these rewards and use the same distribution as their priors. Again, the Bayesian recommendation performed the best of the three.

We observed that Player 1 tends to seek areas with more room for maneuverability. A corridor, for example, would restrict Player 1's next position to three possible locations, making it an easier target. Player 2 knows this preference as well and adjusts its bombardment strategy accordingly. This interplay results in complex emergent behavior.

Conclusions and future research

We studied the problem of recommending a policy for an unknown zero-sum stochastic game, given only observations of the actions and state transitions incurred during play. The players might play according to Nash equilibrium, quantal response equilibrium, or any other behavioral assumption.

This work begets several future directions. First, the work could be extended to general-sum stochastic games involving more than two players. In that setting, depending on the game-theoretic solution concept used to model the players' observed behavior, one might have to deal with the problem of selecting among equilibria with different payoffs. For instance, in the case of multiple Nash equilibria, one might choose payoff-dominant or risk-dominant equilibria.

Second, there are many gradient-based MCMC techniques (such as Hamiltonian Monte Carlo) that make use of the *gradient* of the posterior density to speed up convergence. Ling, Fang, and Kolter (2018) show how to backpropagate gradients through a quantal response equilibrium, while Amos and Kolter (2017) show how to backpropagate gradients through a linear program (and therefore, in our case, a Nash equilibrium). By using these in our algorithms, one could find the gradient of the posterior density with respect to the unknown game parameters.

Third, one could relax the assumption that one knows the players' behavior model. For example, if they play according to a quantal response equilibrium, one might have a belief distribution over the rationality parameter of each player.

Fourth, one could generalize this work in the direction of imperfect-information extensive-form games. In that setting, algorithms such as counterfactual regret minimization (Zinkevich et al. 2007), the excessive gap technique (Hoda et al. 2010; Kroer et al. 2020), or full-width fictitious play (Heinrich, Lanctot, and Silver 2015) can be used to converge to a Nash equilibrium. Furthermore, Farina, Kroer, and Sandholm (2018) present a regret-minimization algorithm for computing reduced normal-form quantal response equilibria by minimizing local regrets, allowing one to compute quantal response equilibria in extremely large games. To make a Bayes-optimal recommendation under uncertainty, one could sample multiple games from the belief distribution and place them under a root chance node, tagging the information sets belonging to Player 2 with the corresponding subtree so that Player 2, but not Player 1, knows which game is being played.

Acknowledgements

This material is based on work supported by the National Science Foundation under grants IIS-1718457, IIS-1901403, and CCF-1733556, and the ARO under award W911NF2010081.

References

Amin, K.; Singh, S.; and Wellman, M. 2016. Gradient Methods for Stackelberg Security Games. In *UAI*.

Amos, B.; and Kolter, Z. 2017. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *PMLR*.

Duane, S.; Kennedy, A.; Pendleton, B.; and Roweth, D. 1987. Hybrid Monte Carlo. *Physics Letters B* .

Endres, S.; Sandrock, C.; and Focke, W. 2018. A simplicial homology algorithm for Lipschitz optimisation. *Journal of Global Optimization* .

Farina, G.; Kroer, C.; and Sandholm, T. 2018. Online Convex Optimization for Sequential Decision Processes and Extensive-Form Games. In *arXiv*.

Foreman-Mackey, D.; Farr, W.; Sinha, M.; Archibald, A.; Hogg, D.; Sanders, J.; Zuntz, J.; Williams, P.; Nelson, A.; de Val-Borro, M.; and et al. 2019. emcee v3: A Python ensemble sampling toolkit for affine-invariant MCMC. *JOSS* .

Foreman-Mackey, D.; Hogg, D. W.; Lang, D.; and Goodman, J. 2013. emcee: The MCMC Hammer. *PASP* .

Geyer, C. 1991. Markov Chain Monte Carlo Maximum Likelihood. *Computing Science and Statistics* .

Goodman, J.; and Weare, J. 2010. Ensemble samplers with affine invariance. *CAMCoS* .

Hastings, W. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* .

Heinrich, J.; Lanctot, M.; and Silver, D. 2015. Fictitious Self-Play in Extensive-Form Games. In *ICML*.

Hoda, S.; Gilpin, A.; Peña, J.; and Sandholm, T. 2010. Smoothing Techniques for Computing Nash Equilibria of Sequential Games. *Mathematics of Operations Research* .

Hoffman, M.; and Gelman, A. 2014. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *JMLR* .

Hu, J.; and Wellman, M. 2003. Nash Q-Learning for General-Sum Stochastic Games. *JMLR* .

Kroer, C.; Waugh, K.; Kılınç-Karzan, F.; and Sandholm, T. 2020. Faster algorithms for extensive-form game solving via improved smoothing functions. *Mathematical Programming* .

Lin, X.; Adams, S.; and Beling, P. 2019. Multi-agent Inverse Reinforcement Learning for Certain General-sum Stochastic Games. *JAIR* .

Lin, X.; Beling, P. A.; and Cogill, R. 2018. Multiagent Inverse Reinforcement Learning for Two-Person Zero-Sum Games. *IEEE Transactions on Games* .

Ling, C. K.; Fang, F.; and Kolter, J. Z. 2018. What game are we playing? End-to-end learning in normal and extensive form games. In *IJCAI*.

Littman, M. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *ICML*.

McFadden, D. 1976. Quantal Choice Analysis: A Survey. *Annals of Economic and Social Measurement* .

McKelvey, R.; and Palfrey, T. 1995. Quantal Response Equilibria for Normal Form Games. *Games and Economic Behavior* .

Mckelvey, R.; and Palfrey, T. 1998. Quantal Response Equilibria for Extensive Form Games. *Experimental Economics* .

Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, A.; and Teller, E. 1953. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* .

Nash, J. 1951. Non-Cooperative Games. *Annals of Mathematics* .

Nash, J. F. 1950. Equilibrium points in n-person games. *PNAS* .

Natarajan, S.; Kunapuli, G.; Judah, K.; Tadepalli, P.; Kersting, K.; and Shavlik, J. W. 2010. Multi-Agent Inverse Reinforcement Learning. In *ICMLA*.

Neal, R. 2012. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC.

Ng, A.; and Russell, S. 2000. Algorithms for Inverse Reinforcement Learning. In *ICML*.

Reddy, T. S.; Gopikrishna, V.; Zaruba, G.; and Huber, M. 2012. Inverse reinforcement learning for decentralized non-cooperative multiagent systems. In *SMC*.

Roberts, G.; and Tweedie, R. 1996. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli* .

Russell, S. 1998. Learning Agents for Uncertain Environments. In *COLT*.

Sandholm, T.; and Crites, R. 1996. Multiagent Reinforcement Learning in the Iterated Prisoner's Dilemma. *Biosystems* .

Shapley, L. 1953. Stochastic Games. *Proceedings of the National Academy of Sciences* .

Swendsen, R.; and Wang, J.-S. 1986. Replica Monte Carlo Simulation of Spin-Glasses. *Physical Review Letters* .

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Jarrod Millman, K.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C.; Polat, İ.; Feng, Y.; Moore, E. W.; Vand erPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and Contributors, S. . . 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* .

Wang, X.; and Klabjan, D. 2018. Competitive Multi-agent Inverse Reinforcement Learning with Sub-optimal Demonstrations. In *PMLR*.

Waugh, K.; Ziebart, B. D.; and Bagnell, J. A. 2011. Computational Rationalization: The Inverse Equilibrium Problem. In *ICML*.

Zhang, X.; Zhang, K.; Miehling, E.; and Basar, T. 2019. Non-Cooperative Inverse Reinforcement Learning. In *NeurIPS*.

Zinkevich, M.; Bowling, M.; Johanson, M.; and Piccione, C. 2007. Regret Minimization in Games with Incomplete Information. In *NIPS*.

Appendix

In this appendix we present additional technical material that did not fit in the body of the paper.

Illustration of suboptimality of maximum probability recommendation

For an intuitive visual illustration of this, suppose our belief distribution over a one-dimensional continuous parameter is as shown in Figure 2. The mode, which is the peak on the right, is atypical of the vast majority of the distribution, which lies on the left. Thus the maximum probability recommendation ignores the bulk of the distribution completely, even though most of the probability mass lies there.

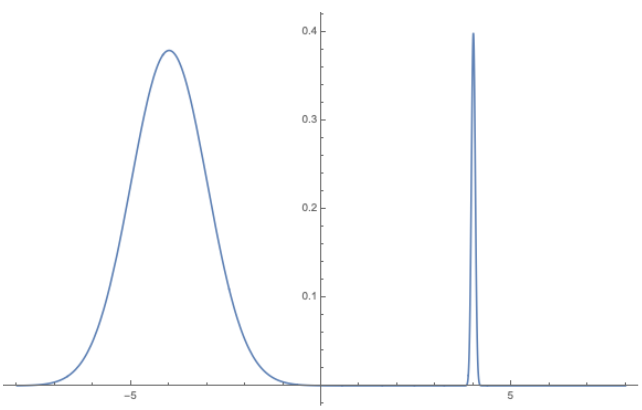

Figure 2: The bimodal mixture distribution $0.95\mathcal{N}(-4, 1) + 0.05\mathcal{N}(4, 0.05)$, where $\mathcal{N}(\mu, \sigma)$ is the normal distribution with mean μ and standard deviation σ.

Additional figures

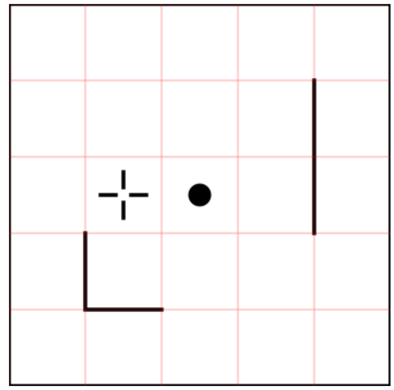

Figure 3: An example layout of a bombardment game.

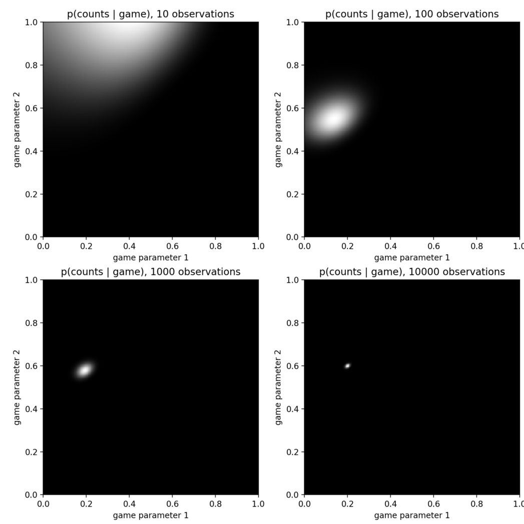

Figure 4: Likelihood function under Nash equilibrium play for a normal-form game with two unknown parameters, with a growing number of observations.

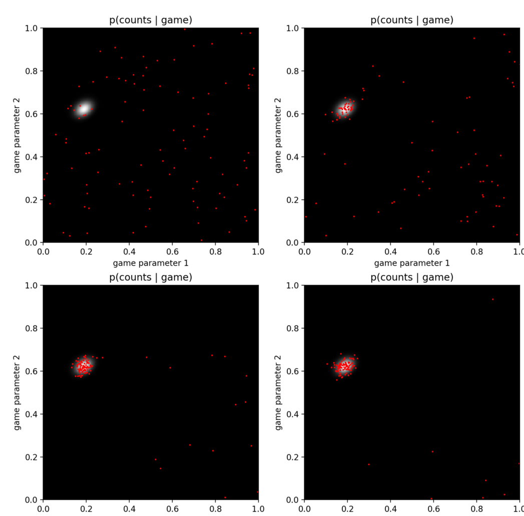

Figure 5: Evolution of the walker ensemble.