Convergence, Targeted Optimality, and Safety in Multiagent Learning

Abstract

This paper introduces a novel multiagent learning algorithm which achieves convergence, targeted optimality against memory-bounded adversaries, and safety, in arbitrary repeated games. Called *CMLeS*, its most novel aspect is the manner in which it guarantees (in a PAC sense) targeted optimality against memory-bounded adversaries, via efficient exploration and exploitation. *CMLeS* is fully implemented and we present empirical results demonstrating its effectiveness.

1. Introduction

In recent years, great strides have been made towards creating autonomous agents that can learn via interaction with their environment. When considering just an individual agent, it is often appropriate to model the world as being *stationary*, meaning that the same action from the same state will always yield the same (possibly stochastic) effects. However, in the presence of other independent agents, the environment is not stationary: an action's effects may depend on the actions of the other agents. This non-stationarity poses the primary challenge of *multiagent learning* (MAL) and comprises the main reason that it is best considered distinctly from single agent learning.

The simplest, and most often studied, MAL scenario is the stateless scenario in which agents repeatedly interact in the stylized setting of a matrix game (a.k.a. normal form game). In the multiagent literature, various criteria have been proposed to evaluate MAL algorithms, emphasizing what behavior they will converge to against various types of opponents,[1] in such settings. The contribution of this paper is that it proposes a novel MAL algorithm, *CMLeS*, that for a multi-player multi-action (arbitrary) repeated game, achieves the following three goals:

1. Convergence : converges to playing a Nash equilibrium in self-play (other agents are also *CMLeS*);
2. Targeted Optimality : for any arbitrary $\epsilon > 0$ and $\delta > 0$, with probability at least 1-δ, achieves at least within $\epsilon + L(\delta)$ of the expected value of the best response against any *memory-bounded*, or *adaptive*,[2] opponent of memory size K, in time polynomial in $\frac{1}{\epsilon}$, $ln(\frac{1}{\delta})$ and $\lambda^{-Size(K+1)}$. $L(\delta) \in [0, 1]$, is a decreasing function w.r.t. 1-δ and assumes a very small value for small values of δ. λ is the minimum non-zero probability that the opponent assigns to an action, in any history and $Size(K + 1)$ denotes number of feasible joint histories of size $K+1$. The same guarantee also holds for opponents which eventually become memory-bounded, with the time complexity claim now holding from the point that the opponent becomes memory-bounded. The main advance of *MLeS* lies in reducing the exponential dependence on $Size(K_{max})$ in time complexity, that is achieved by the current state of the art algorithm, to an exponential dependence on $Size(K + 1)$, where K_{max} is an upper bound on the opponent's memory size, K.
3. Safety : converges to playing the maximin strategy against any other opponent which cannot be approximated as a K_{max} memory-bounded opponent.

1.1. Related work

Bowling et al. (Bowling & Veloso, 2001) were the first to put forth a set of criterion for evaluating multiagent learning algorithms. In games with two players and two actions per player, their algorithm WoLF-IGA converges to playing best response against stationary, or *memoryless*, opponents (rationality), and converges to playing the Nash equilibrium in self-play (convergence). Subsequent approaches extended the rationality and convergence criteria to arbitrary (multi-player, multi-action) repeated games (Banerjee & Peng, 2004; Conitzer & Sandholm, 2006). Amongst them, AWESOME (Conitzer & Sandholm, 2006) achieves conver-

[1]Although we refer to other agents as opponents, we mean any agent (cooperative, adversarial, or neither)

[2]Consistent with the literature (Powers et al., 2005), we call memory-bounded opponents as *adaptive opponents*.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

gence and rationality in arbitrary repeated games without requiring agents to observe each others' mixed strategies. However, none of the above algorithms have any guarantee about the payoffs achieved when they face arbitrary non-stationary opponents. More recently, Powers et al. proposed a newer set of evaluation criteria that emphasizes compatibility, targeted optimality and safety (Powers & Shoham, 2005). Compatibility is a stricter criterion than convergence as it requires the learner to converge within ϵ of the payoff achieved by a Pareto optimal Nash equilibrium. Their proposed algorithm, PCM(A) (Powers et al., 2007) is, to the best of our knowledge, the only known MAL algorithm to date that achieves compatibility, safety and targeted optimality against adaptive opponents in arbitrary repeated games.

1.2. Contributions

CMLeS improves on Awesome by guaranteeing both safety and targeted optimality against adaptive opponents. It improves upon PCM(A) in five ways.

1. The only guarantees of optimality against adaptive opponents that PCM(A) provides are against the ones that are drawn from an initially chosen target set. In contrast, *CMLeS* can model every adaptive opponent whose memory is bounded by K_{max}. Thus it does not require a target set as input: its only input is K_{max}, an upper bound on the memory size of adaptive opponents that it is willing to model and exploit.
2. PCM(A) achieves targeted optimality against adaptive opponents by requiring all feasible joint histories of size K_{max} to be visited a sufficient number of times. K_{max} for PCM(A) is the maximum memory size of any opponent from its target set. *CMLeS* significantly improves this by requiring a sufficient number of visits to all feasible joint histories only of size $K+1$. Thus *CMLeS* promises targeted optimality in number of steps polynomial in $\lambda^{-Size(K+1)}$ in comparison to PCM(A) which provides similar guarantees, but in steps polynomial in $\lambda^{-Size(K_{max})}$. The above sample efficiency property makes *CMLeS* a good candidate for online learning.
3. Unlike PCM(A), *CMLeS* promises targeted optimality against opponents which eventually become memory-bounded with $K \leq K_{max}$.
4. PCM(A) can only guarantee convergence to a payoff within ϵ of the desired Nash equilibrium payoff with a probability δ. In contrast, *CMLeS* guarantees convergence in self-play with probability 1.
5. *CMLeS* is relatively simple in its design. It tackles the entire problem of targeted optimality and safety by running an algorithm that implicitly achieves either of the two, without having to reason separately about adaptive and arbitrary opponents.

The remainder of the paper is organized as follows. Section 2 presents background and definitions, Section 3 and 4 presents our algorithm, Section 5 presents empirical results and Section 6 concludes.

2. Background and Concepts

This section reviews the definitions and concepts necessary for fully specifying *CMLeS*.

A *matrix game* is defined as an interaction between n agents. Without loss of generality, we assume that the set of actions available to all the agents are same, i.e., $A_1 = \ldots = A_n = A$. The payoff received by agent i during each step of interaction is determined by a utility function over the agents' *joint action*, $u_i : A^n \mapsto \Re$. Without loss of generality, we assume that the payoffs are bounded in the range [0,1]. A *repeated game* is a setting in which the agents play the same matrix game repeatedly and infinitely often. A single stage *Nash equilibrium* is a stationary strategy profile $\{\pi_i^*, \ldots, \pi_n^*\}$ such that for every agent i and for every other possible stationary strategy π_i, the following inequality holds: $E_{(\pi_1^*,\ldots,\pi_i^*,\ldots,\pi_n^*)} u_i(\cdot) \geq E_{(\pi_1^*,\ldots,\pi_i,\ldots,\pi_n^*)} u_i(\cdot)$. It is a strategy profile in which no agent has an incentive to unilaterally deviate from its own share of the strategy. A *maximin* strategy for an agent is a strategy which maximizes its own minimum payoff. It is often called the safety strategy, because resorting to it guarantees the agent a minimum payoff.

An *adaptive* opponent strategy looks back at the most recent K joint actions played in the current *history* of play to determine its next stochastic action profile. K is referred to as the *memory size* of the opponent.[3] The strategy of such an opponent is then a mapping, $\pi : A^{nK} \mapsto \Delta A$. If we consider opponents whose future behavior depends on the entire history, we lose the ability to (provably) learn anything about them in a single repeated game, since we see a given history only once. The concept of memory-boundedness limits the opponent's ability to condition on history, thereby giving us a chance to learning its policy.

We now specify what we mean by playing optimally against adaptive opponents. For notational clarity, we denote the other agents as a single agent o. It has been shown previously (Chakraborty & Stone, 2008) that the dynamics of playing against such an o can be modeled as a Markov Decision Process (MDP) whose transition probability function and reward function are

[3]K is the minimum memory size that fully characterizes the opponent strategy.

determined by the opponents' (joint) strategy π. As the MDP is induced by an adversary, this setting is called an *A*dversary *I*nduced *M*DP, or AIM in short.

An AIM is characterized by the K of the opponent which induces it: the AIM's state space is the set of all feasible joint action sequences of length K. By way of example, consider the game of Roshambo or rock-paper-scissors (Figure 1) and assume that o is a single agent and has $K = 1$, meaning that it acts entirely based on the immediately previous joint action. Let the current state be (R, P), meaning that on the previous action, i selected R, and o selected P. Assume that from that state, o plays actions R, P and S with probability $0.25, 0.25$, and 0.5 respectively. When i chooses to take action S in state (R, P), the probabilities of transitioning to states $(S, R), (S, P)$ and (S, S) are then $0.25, 0.25$ and 0.5 respectively. Transitions to states that have a different action for i, such as (R, R), have probability 0. The reward obtained by i when it transitions to state (S, R) is -1 and so on.

The optimal policy of the MDP associated with the AIM is the optimal policy for playing against o. A policy that achieves an expected return within ϵ of the expected return achieved by the optimal policy is called an *ϵ-optimal policy* (the corresponding return is called ϵ-optimal return). If π is known, then we can have computed the optimal policy (and hence ϵ-optimal policy) by doing dynamic programming (Sutton & Barto, 1998). However, we do not assume that π or even K are known in advance: they need to be learned in online play. We use the discounted payoff criterion in our computation of an ϵ-optimal policy, with γ denoting the discount factor.

Finally, it is important to note that there exist opponents in the literature which do not allow convergence to the optimal policy once a certain set of moves have been played. For example, the *grim-trigger* opponent in the well-known *Prisoner's Dilemma* (*PD*) game, an opponent with memory size 1, plays *cooperate* at first, but then plays *defect* forever once the other agent has played *defect* once. Thus, there is no way of detecting its strategy without defecting, after which it is impossible to recover to the optimal strategy of mutual cooperation. In our analysis, we constrain the class of adaptive opponents to include only those which do not negate the possibility of convergence to optimal exploitation, given any arbitrary initial sequence of exploratory moves (Powers & Shoham, 2005).

Equipped with the required concepts, we are now ready to specify our algorithms. First, in Section 3, we present an algorithm that only guarantees safety and targeted optimality against adaptive opponents.

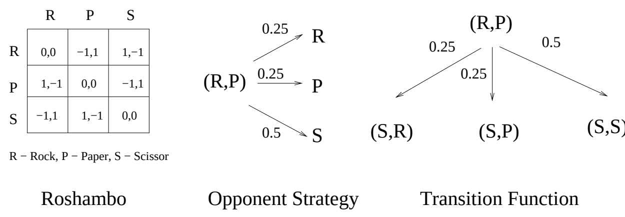

Figure 1. Example of AIM

Then, in Section 4, we introduce the full-blown *CMLeS* algorithm that incorporates convergence additionally.

3. Model learning with Safety (*MLeS*)

In this section, we introduce a novel algorithm, *M*odel *Le*arning with *S*afety (*MLeS*), that ensures safety and targeted optimality against adaptive opponents.

3.1. Overview

MLeS begins with the hypothesis that the opponent is an adaptive opponent (denoted as o) with an unknown memory size, K, that is bounded above by a known value, K_{max}. *MLeS* maintains a model for each possible value of o's memory size, from $k = 0$ to K_{max}, plus one additional model for memory size $K_{max}+1$. Each model $\hat{\pi}_k$ is a mapping $A^{nk} \mapsto \Delta A$ representing a possible o strategy. $\hat{\pi}_k$ is the maximum likelihood distribution based on the observed actions played by o for each joint history of size k encountered. Henceforth we will refer to a joint history of size k as s_k and the empirical distribution captured by $\hat{\pi}_k$ for s_k as $\hat{\pi}_k(s_k)$. $\hat{\pi}_k(s_k, a_o)$ will denote the probability assigned to action a_o, by $\hat{\pi}_k(s_k)$. When a particular s_k is encountered and the respective o's action in the next step is observed, the empirical distribution $\hat{\pi}_k(s_k)$ is updated. Such updates happen for every $\hat{\pi}_k$, on every step. For every s_k, *MLeS* maintains a count value $v(s_k)$, which is the number of times s_k has been encountered. We call an opponent model an ϵ approximation of π, when for any history of size K, it predicts the true opponent action distribution with error at most ϵ.

On each step, *MLeS* selects $\hat{\pi}_{best}$ (and correspondingly k_{best}) as the one from among the $K_{max}+1$ (from 0 to K_{max}) models that currently appears to best describe o's behavior. The mechanism for selecting $\hat{\pi}_{best}$ ensures that, with high probability, it is either $\hat{\pi}_K$ (the most compact representation of π) or a model with a smaller k which is a good approximation of π. Once such a $\hat{\pi}_{best}$ is picked, *MLeS* takes a step towards learning an ϵ-optimal policy for the underlying AIM induced by k_{best}. If it cannot determine such a $\hat{\pi}_{best}$, it defaults to playing the maximin strategy for safety.

Thus, the operations performed by *MLeS* on each step

can be summarized as follows:
1. Update all models based on the past step.
2. Determine $\hat{\pi}_{best}$ (and hence k_{best}). If a $\hat{\pi}_{best}$ cannot be determined, then return null.
3. If $\hat{\pi}_{best} \neq$ null, take a step towards solving the reinforcement learning (RL) problem for the AIM induced by k_{best}. Otherwise, play the maximin strategy.
Of these three steps, step 2 is by far the most complex. We present how *MLeS* addresses it next.

3.2. Model selection

The objective of *MLeS* is to find a k_{best} which is either K (the true memory size) or a suboptimal k s.t. $\hat{\pi}_k$ is a good approximation of π (o's true policy). It does so by comparing models of increasing size to determine at which point the larger models cease to become more predictive of o's behavior. We start by proposing a metric called Δ_k, which is an estimate of how much models $\hat{\pi}_k$ and $\hat{\pi}_{k+1}$ differ from each other. But, first, we introduce two notations that will be instrumental in explaining the metric. We denote $(a_i, a_o) \cdot s_k$ to be a joint history of size k+1, that has s_k as its last k joint actions and (a_i, a_o) as the last k+1'th joint action. For any s_k, we define a set $Aug(s_k) = \cup_{\forall a_i, a_o \in A^2}((a_i, a_o) \cdot s_k | v((a_i, a_o) \cdot s_k) > 0)$. In other words $Aug(s_k)$ contains all joint histories of size k+1 which have s_k as their last k joint actions and have been visited at least once. Δ_k is then defined as $max_{s_k, s_{k+1} \in Aug(s_k), a_o \in A} |\hat{\pi}_k(s_k, a_o) - \hat{\pi}_{k+1}(s_{k+1}, a_o)|$. We say that $\hat{\pi}_k$ and $\hat{\pi}_{k+1}$ are Δ_k distant from one another.

Based, on the concept of Δ_k, we make two observations that will come in handy for our theoretical claims made later in this subsection.

Observation 1. *For all,* $k \in [K, K_{max}] | k \in \mathbb{N}$*, and for any* k *sized joint history* s_k *and any* $s_{k+1} \in Aug(s_k)$*,* $E(\hat{\pi}_k(s_k)) = E(\hat{\pi}_{k+1}(s_{k+1}))$*. Hence* $E(\Delta_k) = 0$*.*

Let, s_K be the last K joint actions in s_k and s_{k+1}. $\hat{\pi}_k(s_k)$ and $\hat{\pi}_{k+1}(s_{k+1})$ represent draws from the same fixed distribution $\pi(s_K)$. So, their expectations will always be equal to $\pi(s_K)$. This is because o just looks at the most recent K joint actions in its history, to decide on its next step action.

Observation 2. *For* $k < K | k \in \mathbb{N}$*,* Δ_k *is a random variable with* $0 \leq E(\Delta_k) \leq 1$*.*

In this case, in the computation of $\hat{\pi}_k(s_k)$, the draws can come from different distributions. This is because, $k < K$ and there is no guarantee of stationarity of $\hat{\pi}_k(s_k)$. Thus, Δ_k can be any arbitrary random variable with an expected value within 0 and 1.

High-level idea: Alg. 1 presents how *MLeS* selects k_{best}. We denote the current values of $\hat{\pi}_k$ and Δ_k at time t, as $\hat{\pi}_k^t$ and Δ_k^t respectively.

Definition 1. $\{\sigma_k^t\}_{t \in 1,2,\ldots}$ *is a sequence of real numbers, unique to each* k*, s.t. it satisfies the following:*
1. *it is a positive monotonically decreasing sequence, tending to 0 as* $t \rightarrow \infty$*;*
2. *for a fixed high probability* $\rho > 0$ *and for* $k \in [K, K_{max}]$*,* $Pr(\Delta_k^t < \sigma_k^t) > \rho$*;*

The reason for choosing such a $\{\sigma_k^t\}_{t \in 1,2,\ldots}$ sequence for each k will be clear from the next two paragraphs. Later, we will show, how we compute the σ_k^t's. *MLeS* iterates over values of k starting from 0 to K_{max} and picks the minimum k s.t for all $k \leq k' \leq K_{max}$, the condition $\Delta_{k'}^t < \sigma_{k'}^t$ is satisfied (steps 3-11).

For $k < K$, there is no guarantee that Δ_k will tend to 0, as $t \rightarrow \infty$ (Observation 2). More often than not, Δ_k will tend to a positive value quickly. On the other hand, $\sigma_k^t \rightarrow 0$ as $t \rightarrow \infty$ (condition 1 of Definition 1). This leads to one of the following two cases:
1) σ_k^t becomes $\leq \Delta_k^t$ and step 6 of Alg. 1 holds, thus rejecting k as a possible candidate for selection.
2) k gets selected. However, then we are sure that $\hat{\pi}_k^t$ is no more than $\sum_{k \leq k' < K} \sigma_{k'}^t$ distant from $\hat{\pi}_K^t$ (the best model of π we have at present). With increasingly many time steps, $\hat{\pi}_k^t$ needs to be an increasingly better approximation of π_K^t, to keep getting selected.

For $k \geq K$, all Δ_k^t's $\rightarrow 0$, as $t \rightarrow \infty$ (Observation 1). Since for all $k \geq K : Pr(\Delta_k^t < \sigma_k^t) > \rho$ (condition 2 of Definition 1), K gets selected with a high probability $\rho^{K_{max}-K+1}$. A model with memory size more than K is selected with probability at most $(1 - \rho^{K_{max}-K+1})$, which is a small value.

We now address the final part of Alg. 1 that we have yet to specify: setting the σ_k^t's (step 2).
Choosing σ_k^t**:** In the computation of Δ_k^t, *MLeS* chooses a specific s_k^t from the set of all possible joint histories of size k, a specific s_{k+1}^t from $Aug(s_k^t)$ and an action a_o^t, for which the models $\hat{\pi}_k^t$ and $\hat{\pi}_{k+1}^t$ differ maximally on that particular time step. So,

$$\Delta_k^t < \sigma_k^t \equiv |\hat{\pi}_k^t(s_k^t, a_o^t) - \hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t)| < \sigma_k^t \tag{1}$$

The goal will be to select a value for σ_k^t s.t. condition 2 of Definition 1 is always satisfied. Condition 1 will implicitly follow from the above. For $k \in [K, K_{max}]$, we can rewrite Inequality 1 as,

$$\begin{aligned} \equiv \quad & |(|\hat{\pi}_k^t(s_k^t, a_o^t) - E(\hat{\pi}_k^t(s_k^t, a_o^t)|) - (|\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t) \\ & -E(\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t)|)| < \sigma_k^t \end{aligned} \tag{2}$$

The above step follows from using $E(\hat{\pi}_k^t(s_k^t, a_o^t)) =$

Algorithm 1: FIND-MODEL

```
output : $k_{best}$, $\hat{\pi}_{best}$,
1  $k_{best} \leftarrow -1$, $\hat{\pi}_{best} \leftarrow null$
2  for all $0 \leq k \leq K_{max}$, compute $\Delta_k^t$ and $\sigma_k^t$
3  for $0 \leq k \leq K_{max}$ do
4      $flag \leftarrow$ true
5      for $k \leq k' \leq K_{max}$ do
6          if $\Delta_{k'}^t \geq \sigma_{k'}^t$ then
7              $flag \leftarrow$ false
8              break
9      if $flag$ then
10         $k_{best} \leftarrow k$; $\hat{\pi}_{best} \leftarrow \hat{\pi}_k^t$
11         break
12 return $k_{best}$ and $\hat{\pi}_{best}$
```

$E(\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t)) \geq 0$ (Observation 1). One way to satisfy Inequality 2 is to have both $|\hat{\pi}_k^t(s_k^t, a_o^t) - E(\hat{\pi}_k^t(s_k^t, a_o^t))|$ and $|\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t) - E(\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t))|$ be $< \sigma_k^t$. Thus, to ensure $[k \in [K, K_{max}] : Pr(\Delta_k^t < \sigma_k^t) > \rho]$, we need a lower bound of $\sqrt{\rho}$, on the probabilities of the above 2 inequalities.

Also, we observe that the following holds :

$$Pr(|\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t) - E(\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t))| < \sigma_k^t) > \sqrt{\rho} \quad (3)$$
$$\implies Pr(|\hat{\pi}_k^t(s_k^t, a_o^t) - E(\hat{\pi}_k^t(s_k^t, a_o^t))| < \sigma_k^t) > \sqrt{\rho} \quad (4)$$

This can be derived by applying Hoeffding's inequality (Hoeffding, 1963) and using $v(s_k^t) \geq v(s_{k+1}^t)$. $v(s_k^t) \geq v(s_{k+1}^t)$ because the number of visits to a joint history s_k must be at least the number of visits to any member from $Aug(s_k)$. So,

$$Pr(|\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t) - E(\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t))| < \sigma_k^t) > \sqrt{\rho} \quad (5)$$
$$\implies Pr(\Delta_k^t < \sigma_k^t) > \rho$$

The problem now boils down to selecting a suitable σ_k^t s.t. Inequality 5 is satisfied. Hoeffding's inequality gives us an upper bound for σ_k^t in Inequality 5. Using that upper bound and solving for σ_k^t, we get, $\sigma_k^t = \sqrt{(\frac{1}{2v(s_{k+1}^t)} ln(\frac{2}{1-\sqrt{\rho}}))}$. So in general, for each $k \in [0, K_{max}]$, the σ_k^t value is set as above. Note that, $v(s_{k+1}^t)$ is the number of visits to the specific s_{k+1}^t chosen for the computation of Δ_k^t. Setting σ_k^t as above satisfies both the conditions specified in Definition 1. Condition 1 follows implicitly since in infinite play, the action selection mechanism ensures infinite visits to all joint histories of a finite length.

Theoretical underpinnings: Now, we state our main theoretical result regarding model selection.

Lemma 3.1. *After all feasible joint histories of size $K+1$ have been visited $\frac{(K+1)^2}{2\epsilon^2} ln(\frac{2}{1-\sqrt{\rho}})$ times, then with probability at least $\rho^{K_{max}+2}$, the $\hat{\pi}_{best}$ returned by Alg. 1 is an ϵ approximation of π. ρ is the fixed high probability value from Condition 2 of Definition 1.*

PROOF. When all $k < K$ have been rejected, Alg. 1 selects K with probability at least $\rho^{K_{max}-K+1}$. If p is the probability of selecting any $k < K$ as k_{best}, the probability of selecting any $k \leq K$ as k_{best}, is then at least $p + (1-p)\rho^{K_{max}-K+1} > \rho^{K_{max}-K+1} > \rho^{K_{max}+1}$. If $k_{best} = K$, then we know that $\Delta_K^t < \sigma_K^t$. So from Inequality 4,

$$Pr(|\hat{\pi}_K^t(s_K^t, a_o^t) - \pi(s_K^t, a_o^t))| < \sigma_K^t) > \sqrt{\rho}$$
$$\implies Pr(|\hat{\pi}_K^t(s_K^t, a_o^t) - \pi(s_K^t, a_o^t)| < \sigma_K^t) > \rho$$

s_K^t and a_o^t are the respective joint history of size K and action, for which models $\hat{\pi}_K^t$ and $\hat{\pi}_{K+1}^t$ maximally differ at t. So in this case, with probability ρ, $\hat{\pi}_{best}$ is a σ_K^t approximation of π. In similar fashion it can be shown that if $k_{best} < K$, then with probability ρ, $\hat{\pi}_{best}$ is a $\sum_{k \leq k' \leq K} \sigma_{k'}^t$ approximation of π.

If an ϵ approximation of π is desired, a sufficient condition is to ensure that for all $0 \leq k \leq K$, σ_k^t gets assigned a value $\leq \frac{\epsilon}{K+1}$. If all feasible joint histories of size $K+1$ are visited $\frac{(K+1)^2}{2\epsilon^2} ln(\frac{2}{1-\sqrt{\rho}})$ times, then σ_K^t must be less than $\frac{\epsilon}{K+1}$ (from Inequality 5 and Hoeffding's inequality). Also every feasible history of smaller sizes, must also have been visited at least $\frac{(K+1)^2}{2\epsilon^2} ln(\frac{2}{1-\sqrt{\rho}})$ times. Hence σ_k^t for all $k < K$ also must have values less than $\frac{\epsilon}{K+1}$. □

For Alg. 1 to return an ϵ approximation of π, *MLeS* does not need to know the number of visits (Lemma 3.1) required beforehand; it just needs to ensure that every feasible $K+1$ history gets visited so many times. Finally, what remains to be addressed is the action-selection mechanism (step 3, main algorithm).

3.3. Action selection

On each time step, the action selection mechanism decides on what action to take for the ensuing time step. If the $\hat{\pi}_{best}$ returned is null, it plays the maximin strategy. If $\hat{\pi}_{best} \neq$ null, the action selection strategy picks the AIM associated with opponent memory k_{best} and takes the next step in the reinforcement learning problem of computing a near-optimal policy for that AIM. In order to solve this RL problem, *MLeS* uses the variant of the R-Max algorithm that does not assume that the mixing time of the underlying MDP is known (Brafman & Tennenholtz, 2003). R-Max is a model based RL algorithm that converges to playing an ϵ-optimal policy for an MDP with probability

1-δ, in time complexity polynomial in $\frac{1}{\epsilon}$, $ln(\frac{1}{\delta})$ and certain parameters of the MDP. A separate instantiation of the R-Max algorithm is maintained for each of the possible K_{max}+1 AIMs pertaining to the possible memory sizes of o, i.e, $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_{K_{max}}$. On each step, based on the k_{best} returned, the R-Max instance for the AIM $\mathcal{M}_{k_{best}}$ is selected to take an action.

The steps that ensure targeted optimality against adaptive opponents are then as follows:
1. First, ensure that Alg. 1 keeps returning a $k_{best} \leq K$ with a high probability $\sqrt{1-\delta}$ s.t. $\hat{\pi}_{best}$ is an $\frac{\epsilon(1-\gamma)}{2Size(K)}$ approximation of π. The conditions for that to happen are given by Lemma 3.1. Playing optimally against such an approximation of π, guarantees an $\frac{\epsilon}{2}$-optimal payoff against o (Lemma 4 of (Brafman & Tennenholtz, 2003)). Thus an $\frac{\epsilon}{2}$-optimal policy for such a model will guarantee an ϵ-optimal payoff against o.
2. Once such a $k_{best} \leq K$ is selected by Alg. 1 with a high probability $\sqrt{1-\delta}$ on every step, then with a probability $\sqrt{1-\delta}$, converge to playing an $\frac{\epsilon}{2}$-optimal policy for $\mathcal{M}_{k_{best}}$. In order to achieve that, the R-Max instantiation for $\mathcal{M}_{k_{best}}$ will require a certain fixed number of visits to every joint history of size k_{best}. Since the k_{best} selected by Alg. 1 is at most K with a high probability , a sufficient number of visits to every joint history of size K will suffice convergence to an $\frac{\epsilon}{2}$-optimal policy.

It can be shown that our R-Max-based action selection strategy implicitly achieves both of above steps in number of time steps polynomial in $\frac{1}{\epsilon}$, $ln(\frac{1}{\delta})$ and $\lambda^{-Size(K+1)}$. Note, we do not have the ability to take samples at will from different histories, but may need to follow a chain of different histories to get a sample pertaining to one history. In the worst case, the chain can be the full set of all histories, with each transition occurring with λ. Hence the unavoidable dependence on $\lambda^{-Size(K+1)}$, in time complexity. The bounds we provide are extremely pessimistic and likely to be tractable against most opponents. For example against opponents which only condition on *MLeS*'s recent history of actions, $\lambda^{-Size(K+1)}$ dependency gets replaced by a dependency over just $|A|^{K+1}$.

So far what we have shown is that *MLeS*, with a high probability 1-δ on each step, converges to playing an ϵ-optimal policy. It is important to note that, acting in this fashion does not guarantee it a return that is 1-δ times the ϵ-optimal return. However, we can compute an upper bound on the loss and show that the loss is extremely small for small values of δ. Let r_t be the random variable that denotes the reward obtained on time step t by following the ϵ-optimal policy. The maximum loss incurred is : $|(1-\delta)\sum_{t=0}^{\infty}\gamma^t E(r_t) - \sum_{t=0}^{\infty}\gamma^t(1-\delta)^t E(r_t)| < |\sum_{t=0}^{\infty}\gamma^t E(r_t) - \sum_{t=0}^{\infty}\gamma^t(1-\delta)^t E(r_t)| \leq |\sum_{t=0}^{\infty}\gamma^t - \sum_{t=0}^{\infty}\gamma^t(1-\delta)^t| \leq \frac{\gamma\delta}{(1-\gamma)(1-\gamma(1-\delta))}$. In the above computation, we assume that whenever *MLeS* does not play the ϵ-optimal policy, it gets the minimum reward of 0. We denote this loss as $L(\delta)$, since it is a function of δ (γ being fixed). Note that $L(\delta)$ can be made extremely small by selecting a very small δ.

This brings us to our main theorem regarding *MLeS*.
Theorem 3.2. *For any arbitrary $\epsilon > 0$ and $\delta > 0$, MLeS with probability at least 1-δ, achieves at least within ϵ+$L(\delta)$ of the expected value of the best response against any adaptive opponent, in number of time steps polynomial in $\frac{1}{\epsilon}$, $ln(\frac{1}{\delta})$ and $\lambda^{-Size(K+1)}$.*
Against an arbitrary o, our claims rely on o not behaving as a K_{max} adaptive opponent in the limit. This means $\Delta_{K_{max}}$ tends to a positive value, as $t \to \infty$. Alg. 1 returns $\hat{\pi}_{best}$ as null in the limit, with probability 1. *MLeS* will then subsequently converge to playing the maximin strategy, thus ensuring safety.

4. Convergence and Model learning with Safety (*CMLeS*)

In this section we build on *MLeS* to introduce a novel MAL algorithm for an arbitrary repeated game which achieves safety, targeted optimality, and convergence, as defined in Section 1. We call our algorithm, *C*onvergence with *M*odel *L*earning and *S*afety: (*CMLeS*). *CMLeS* begins by testing the opponents to see if they are also running *CMLeS* (self-play); when not, it uses *MLeS* as a subroutine.

4.1. Overview

CMLeS (Alg. 2) can be tuned to converge to any Nash equilibrium of the repeated game in self-play. Here, for the sake of clarity, we present a variant which converges to the single stage Nash equilibrium. This equilibrium also has the advantage of being the easiest of all Nash equilibria to compute and hence has historically been the preferred solution concept in multiagent learning (Bowling & Veloso, 2001; Conitzer & Sandholm, 2006). The extension of *CMLeS* to allow for convergence to other Nash equilibria is straightforward, only requiring keeping track of the probability distribution for every conditional strategy present in the specification of the equilibrium.

Steps 1 - 2: Like AWESOME, we assume that all agents have access to a Nash equilibrium solver and they compute the same Nash equilibrium profile. If there are finitely many equilibria, then this assumption can be lifted with each agent choosing randomly an equilibrium profile, so that there is a non-zero prob-

Algorithm 2: *CMLeS*

```
input : $n, \tau = 0$
1  for $\forall j \in \{1, 2, \ldots, n\}$ do
2      $\pi_j^* \leftarrow$ ComputeNashEquilibriumStrategy()
3  $AAPE \leftarrow true$
4  while $AAPE$ do
5      for $N_\tau$ rounds do
6          Play $\pi_{self}^*$
7          for each agent $j$ update $\phi_j^\tau$
8      recompute $AAPE$ using the $\phi_j^\tau$'s and $\pi_j^*$'s
9      if $AAPE$ is false then
10         if $\tau = 0$ then
11             Play $a_o$, $K_{max}$+1 times
12         else if $\tau = 1$ then
13             Play $a_o$, $K_{max}$ times followed by a
14             random action other than $a_o$
15         else
16             Play $a_o$, $K_{max}$+1 times
17         if any other agent plays differently then
18             $AAPE \leftarrow false$
19         else
20             $AAPE \leftarrow true$
21     $\tau \leftarrow \tau + 1$
22 Play MLeS
```

ability that the computed equilibrium coincides.
Steps 3 - 4: The algorithm maintains a null hypothesis that all agents are playing equilibrium ($AAPE$). The hypothesis is not rejected unless the algorithm is certain with probability 1 that the other agents are not playing *CMLeS*. τ keeps count of the number of times the algorithm reaches step 4.
Steps 5 - 8 (Same as AWESOME): Whenever the algorithm reaches step 5, it plays the equilibrium strategy for a fixed number of episodes, N_τ. It keeps a running estimate of the empirical distribution of actions played by all agents, including itself, during this run. At step 8, if for any agent j, the empirical distribution ϕ_j^τ differs from π_j^* by at least ϵ_e^τ, $AAPE$ is set to false. The *CMLeS* agent has reason to believe that j may not be playing the same algorithm. The ϵ_e^τ and N_τ values for each τ are assigned in a similar fashion to AWESOME (Definition 4 of (Conitzer & Sandholm, 2006)).
Steps 10 - 20: Once $AAPE$ is set to false, the algorithm goes through a series of steps in which it checks whether the other agents are really *CMLeS* agents. The details are explained below when we describe the convergence properties of *CMLeS* (Theorem 4.1).
Step 22: When the algorithm reaches here, it is sure (probability 1) that the other agents are not *CMLeS* agents. Hence it switches to playing *MLeS*.

4.2. Theoretical underpinnings

We now state our main convergence theorems.

Theorem 4.1. *CMLeS satisfies both the criteria of targeted optimality and safety.*

PROOF. To prove the theorem, we need to prove:
1. For opponents not themselves playing *CMLeS*, *CMLeS* always reaches step 22 with some probability;
2. There exists a value of τ, for and above which, the above probability is at least δ.
Proof of 1. We utilize the property that a K adaptive opponent is also a K_{max} adaptive opponent (see Observation 1). The first time $AAPE$ is set to false, it selects a random action a_o and then plays it K_{max}+1 times in a row. The second time when $AAPE$ is set to false, it plays a_o, K_{max} times followed by a different action. If the other agents have behaved identically in both of the above situations, then *CMLeS* knows : 1) either the rest of the agents are playing *CMLeS*, or, 2) they are adaptive and plays stochastically for a K_{max} bounded memory where all agents play a_o. The latter observation comes in handy below. Henceforth, whenever $AAPE$ is set to false, *CMLeS* always plays a_o, K_{max}+1 times in a row. Since a non-*CMLeS* opponent must be stochastic (from the above observation), at some point of time, it will play a different action on the K_{max}+1'th step with a non-zero probability. *CMLeS* then rejects the null hypothesis that all other agents are *CMLeS* agents and jumps to step 22.
Proof of 2. This part of the proof follows from Hoeffding's inequality. *CMLeS* reaches step 22 with a probability at least δ in τ polynomial in $\frac{1}{\kappa}$ and $ln(\frac{1}{\delta})$, where κ is the maximum probability that any agent assigns to any action other than a_o for a recent K_{max} joint history of all agents playing a_o. □

Theorem 4.2. *In self-play, CMLeS converges to playing the Nash equilibrium of the repeated game, with probability 1.*

The proof follows from the corresponding proof for AWESOME (Theorem 3 of (Conitzer et al., 2006)).

5. Results

We now present empirical results that supplement the theoretical claims. We focus on how efficiently *CMLeS* models adaptive opponents in comparison to existing algorithms, PCM(A) and AWESOME. For *CMLeS*, we set $\epsilon = 0.1$, $\delta = 0.01$ and $K_{max} = 10$. To make the comparison fair with PCM(A), we use the same values of ϵ and δ and always include the respective opponent in the target set of PCM(A). We also add an adaptive strategy with $K = 10$ to the target set of PCM(A), so that it needs to explore joint histories of size 10.

We use the 3-player Prisoner's Dilemma (PD) game as our representative matrix game. The game is a 3 player version of the N-player PD present in GAMUT.[4] The adaptive opponent strategies we test against are :
1. Type 1: every other player plays *defect* if in the last 5 steps *CMLeS* played *defect* even once. Otherwise, they play *cooperate*. The opponents are thus deterministic adaptive strategies with $K = 5$.
2. Type 2: every other player behaves as type-1 with 0.5 probability, or else plays completely randomly. In this case, the opponents are stochastic with $K = 5$.
The total number of joint histories of size 10 in this case is 8^{10}, which makes PCM(A) highly inefficient. However, $CMLeS$ quickly figures out the true K and converges to optimal behavior in tractable number of steps. Figure 2 shows our results against these two types of opponents. The Y-axis shows the payoff of each algorithm as a fraction of the optimal payoff achievable against the respective opponent. Also plotted in the same graph, is the fraction of times *CMLeS* chooses the right memory size (denoted as *convg* in the plot). Each plot has been averaged over 30 runs to increase robustness. Against type-1 opponents (Figure 2(i)), *CMLeS* figures out the true memory size in about 2000 steps and converges to playing optimally by 16000 episodes. Against type-2 opponents (Figure 2(ii)), it takes a little longer to figure out the correct memory size (about 35000 episodes) because in this case, the number of feasible joint histories of size 6 are much more. Both AWESOME and PCM(A) perform much worse. PCM(A) plays a random exploration strategy until it has visited every possible joint history of size K_{max}, hence it keeps getting a constant payoff during this whole exploration phase.

Due to space constraints we skip the results for convergence and safety. However, it will be worthwhile to mention, that when K_{max} was set to 4, *MLeS* converged to playing the maximin strategy in about 10000 episodes, against both of the above opponents. The convergence part of *MLeS* uses the framework of AWESOME and the results are exactly similar to it.

6. Conclusion and Future Work

In this paper, we introduced a novel MAL algorithm, *CMLeS*, which in an arbitrary repeated game, achieves convergence, targeted-optimality against adaptive opponents, and safety. One key contribution of *CMLeS* is in the manner it handles adaptive opponents: it requires only a loose upper bound on the opponent's memory size. In contrast, the existing state of the art algorithm, PCM(A), requires a complete specification of the adaptive opponents at the beginning, which it calls a target set. Second, and more importantly, *CMLeS* improves on PCM(A), by promising targeted optimality against adaptive opponents in time steps polynomial in $\lambda^{-Size(K+1)}$ where $Size(K+1)$ is the number of feasible histories of size $K+1$, and λ is the minimum non-zero probability that the opponent assigns to an action, in any history. PCM(A) guarantees the same, but in steps polynomial in $\lambda^{-Size(K_{max})}$.

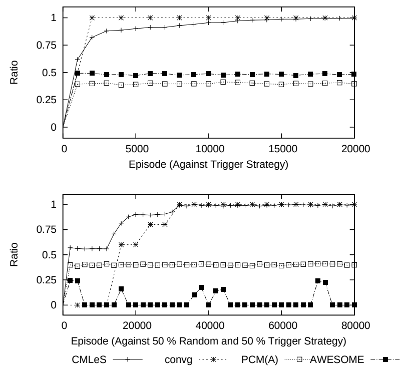

Figure 2. Against adaptive opponents

Right now, the guarantees of *CMLeS* are only in self-play or when all other agents are adaptive. Any other distribution of agents is considered arbitrary, and *MLeS* converges to playing the maximin strategy. Our ongoing research agenda includes improving *CMLeS* to have better performance guarantees against arbitrary mixes of agents, i.e., some adaptive, some self-play, and the rest arbitrary.

References

Banerjee, B and Peng, J. Performance bounded reinforcement learning in strategic interactions. In *AAAI'04*

Bowling, M and Veloso, M. Convergence of gradient dynamics with a variable learning rate. In *ICML'01*

Brafman, R and Tennenholtz, M. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. In *J. Mach. Learn. Res.'03*

Chakraborty, D and Stone, P. Online multiagent learning against memory bounded adversaries. In *ECML'08*

Conitzer, V and Sandholm, T. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *J. Mach. Learn. Res.'06*

Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association'1963*

Powers, R and Shoham, Y. Learning against opponents with bounded memory. In *IJCAI'05*

Powers, R, Shoham, Y, and Vu, T. A general criterion and an algorithmic framework for learning in multi-agent systems. *Mach. Learn.'07*

Sutton, R and Barto, A. *Reinforcement Learning: An Introduction' 98*

[4]`http://gamut.stanford.edu/userdoc.pdf`