Balancing Safety and Exploitability in Opponent Modeling

Zhikun Wang, Abdeslam Boularias, Katharina Mülling, Jan Peters
Max Planck Institute for Intelligent Systems
Spemannstr 38, 72076 Tübingen, Germany
{firstname.lastname}@tuebingen.mpg.de

Abstract

Opponent modeling is a critical mechanism in repeated games. It allows a player to adapt its strategy in order to better respond to the presumed preferences of his opponents. We introduce a new modeling technique that adaptively balances exploitability and risk reduction. An opponent's strategy is modeled with a set of possible strategies that contain the actual strategy with a high probability. The algorithm is safe as the expected payoff is above the minimax payoff with a high probability, and can exploit the opponents' preferences when sufficient observations have been obtained. We apply them to normal-form games and stochastic games with a finite number of stages. The performance of the proposed approach is first demonstrated on repeated rock-paper-scissors games. Subsequently, the approach is evaluated in a human-robot table-tennis setting where the robot player learns to prepare to return a served ball. By modeling the human players, the robot chooses a forehand, backhand or middle preparation pose before they serve. The learned strategies can exploit the opponent's preferences, leading to a higher rate of successful returns.

Introduction

Opponent modeling allows a player to exploit the opponents' preferences and weaknesses in repeated games. Approaches that discard opponent modeling usually need to make worst-case assumptions, e.g. following a minimax strategy. Such approaches are considered safe as their expected payoff is lower-bounded by the minimax payoff. However, they lack the ability to exploit non-competitive or imperfect opponents in general-sum games. In contrast, the idea of fictitious play (Brown 1951) has been extensively used for sixty years. Assuming the opponents are playing stationary strategies, fictitious play consists of modeling them by the empirical probabilities of the observed actions and then optimally responding according to the model. Using a sufficiently accurate model of a stationary opponent, fictitious play yields a higher expected payoff than any approach without modeling. However, even when the stationarity assumption holds, inaccurate models are inevitable for a limited number of observations, which exposes the player to adopting overly risky strategies.

Online learning algorithms, e.g. (Zinkevich 2003) and (Bowling 2005), were developed to guarantee zero average regret in the worst case. However, the opponents are usually suboptimal (Simon 1991) or not completely competitive (e.g. in general-sum games) in many real-world situations. A practical criterion (Powers, Shoham, and Vu 2007) is concerned with learning to play optimally against stationary opponents or opponents whose strategies converge to a stationary one. For example, WoLF-IGA (Bowling and Veloso 2002) and AWESOME (Conitzer and Sandholm 2007) learn the best-response against stationary opponents, and converge in self-plays. However, these algorithms may adapt to unsafe counter-strategies due to inaccurate estimates of the opponents' strategies. To address the safety issue in opponent modeling, Markovitch and Reger (2005) proposed to infer a weakness model instead of estimating the precise model. McCracken and Bowling (2004) proposed an ϵ-safe learning algorithm that chooses the best counter-strategy from a safe set of strategies. Strategies of the corresponding set do not lose more than ϵ in the worst case. In more recent work, Johanson (2009) proposed robust learners by considering restricted Nash responses and data-biased responses for imperfect information games.

In this paper, we propose a different approach to modeling an opponent's strategy with a set of possible strategies that contains the actual strategy with a high probability. For simplicity, we limit the discussion in this paper to the two-player games. Nevertheless, the modeling technique can be extended to multi-player games. Given a parameter δ that controls the safety-exploitability trade-off, a stationary opponent's strategy is modeled with a set of possible strategies that contains the actual one with probability no less than $1-\delta$. These possible strategies are chosen according to their consistency with the observations. We propose an algorithm that provides a counter-strategy with a lower-bound above the minimax payoff with probability no less than $1-\delta$. Such a strategy is said to be δ-safe, where δ is a parameter indicating the trade-off between the safety and the exploitability. The model of possible strategies shrinks along with the increased number of observations, and converges to the actual opponent's strategy. Therefore, the algorithm ensures the convergence of the counter-strategy to the actual best-response.

Unfortunately, the stationarity assumption is often unreal-

istic. A more reasonable assumption is local stationarity, i.e., the opponent's strategy is assumed to be stationary within a number of consecutive repetitions. A statistical hypothesis tests is used to detect significant changes in the opponent's strategy, so that the algorithm can efficiently adapt to them. Given a sequence of consecutive observations that are likely to be drawn from a locally stationary strategy, the proposed algorithm computes a locally δ-safe counter-strategy.

The proposed modeling technique allows a table-tennis playing robot to improve its response to balls served by human opponents. The used robot setting (Muelling, Kober, and Peters 2010) has three possible high-level actions, i.e. setting to either a forehand, backhand, or middle preparation pose while the opponent serves. Each action has a relatively high success rate when the ball is served to its corresponding region. However, the robot is limited in its acceleration, resulting in low success rate for incoming balls far away from the preparation pose. Assuming that the opponent uses a stationary strategy, this algorithm generates counter-strategies such that the robot is more likely to successfully return the served ball. We use the low-level planner to evaluate the expected success rate of the learned strategies.

Strategy Learning in Normal-Form Games

The two participants are indicated by player i and player j, and the algorithm plays on behalf of player i. A repeated game consists of several repetitions of a base game. Such a base game can either be a normal-form game or a stochastic game. A normal-form game is represented by reward matrices for both participants, among which the reward matrix for player i is denoted by $\mathbf{R}$. In each game, two players choose their own actions a_i, a_j independently from action spaces $\mathcal{A}_i, \mathcal{A}_j$ respectively. The reward for player i is given by $\mathbf{R}_{a_i,a_j}$ depending on their joint action.

For normal-form games, the strategies are probability distributions over all possible actions a_i and a_j, which we denote by $\boldsymbol{\pi}_i(a_i) \in \Delta^{|\mathcal{A}_i|}$ and $\boldsymbol{\pi}_j(a_j) \in \Delta^{|\mathcal{A}_j|}$, where $|\mathcal{A}_i|, |\mathcal{A}_j|$ are the size of $\mathcal{A}_i, \mathcal{A}_j$, and Δ^n is the n-simplex set $\{\boldsymbol{\pi} \in \mathbb{R}^n | \boldsymbol{\pi}^T \mathbf{1} = 1 \text{ and } \boldsymbol{\pi} \succeq \mathbf{0}\}$. Consider the case when player i has played the game N times and observed the opponent's actions $\{a_j^k | k = 1 \dots N\}$. Assuming the actual opponent's strategy $\boldsymbol{\pi}_j^*$ is stationary during these N repetitions, the goal of the player is to learn a best-response strategy $\boldsymbol{\pi}_i$ against the opponent's possible strategies.

Fictitious play uses the empirical distribution $\tilde{\boldsymbol{\pi}}_j(a_j) = N_{a_j}/N$ to model the opponent, where N_{a_j} is the number of times action a_j was observed. Given the opponent's model, player i wants to learn a best-response strategy that maximizes its expected payoff. In normal-form games, the expected payoff for strategies $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}_j$ is computed as $\boldsymbol{\pi}_i^T \mathbf{R} \boldsymbol{\pi}_j$. When the model is a single estimate $\tilde{\boldsymbol{\pi}}_j$, the best-response strategy is given by $\text{BR}(\tilde{\boldsymbol{\pi}}_j) = \arg\max_{\boldsymbol{\pi}_i} \boldsymbol{\pi}_i^T \mathbf{R} \tilde{\boldsymbol{\pi}}_j$. It tends to always take the same action, which can be unsafe when the opponent's model is not sufficiently precise.

To ensure the safety of the learned counter-strategy, we propose an δ-safe technique for modeling the opponent with a set of possible strategies instead of a single estimate. Furthermore, we compute the generalized best-response strategy against the proposed opponent's model, resulting in a δ-safe counter-strategy.

δ-Safe Opponent Modeling

Assume $\boldsymbol{\pi}_j^*$ is the true opponent's strategy, according to which the opponent's actions are drawn, and denote by $\tilde{\boldsymbol{\pi}}_j$ the empirical distribution. According to Theorem 5 in (Seldin and Tishby 2010), the Kullback-Leibler (KL) divergence between $\tilde{\boldsymbol{\pi}}_j$ and $\boldsymbol{\pi}_j^*$ is bounded by the following inequality with probability no less than $1 - \delta$,

$$\text{KL}(\tilde{\boldsymbol{\pi}}_j || \boldsymbol{\pi}_j^*) \leq \varepsilon(\delta) \triangleq \frac{(|\mathcal{A}_j| - 1)\ln(N+1) - \ln(\delta)}{N}.$$

The opponent's possible strategies are selected by their distance to the empirically observed behavior, where the KL divergence serves as the natural measure of distance between probability distributions. Hence, the model of the opponent is given by

$$\Omega(\delta) \triangleq \{\boldsymbol{\pi}_j \in \Delta^{|\mathcal{A}_j|} | \text{KL}(\tilde{\boldsymbol{\pi}}_j || \boldsymbol{\pi}_j) \leq \varepsilon(\delta)\}.$$

The true opponent's strategy $\boldsymbol{\pi}_j^*$ lies in the set $\Omega(\delta)$ with probability no less than $1 - \delta$.

In practice, the algorithms is not very sensitive to the choice of the parameter if $\delta \leq 0.5$. We set $\delta = C/(N+1)$, where $C \leq 1$ is a constant, so that the algorithm has an upper-bound on its regret.

δ-Safe Strategy Learning

With probability no less than $1-\delta$, the true opponent's strategy is inside $\Omega(\delta)$. However, the player has no further information to choose the real opponent's strategy among all possible strategies in this set. To learn a safe counter-strategy, we generalize the best-response strategy to be the counter-strategy that has the maximal expected payoff in the worst case. This problem can be formulated as:

$$\text{BR}(\Omega(\delta)) \triangleq \arg\max_{\boldsymbol{\pi}_i \in \Delta^{|\mathcal{A}_i|}} \min_{\boldsymbol{\pi}_j \in \Omega(\delta)} \boldsymbol{\pi}_i^T \mathbf{R} \boldsymbol{\pi}_j. \tag{1}$$

Finding the best-response solution in Problem (1) is a convex optimization problem that can be solved efficiently using sub-gradient methods. When the opponent's model contains the true strategy $\boldsymbol{\pi}_j^*$, the expected payoff of the best-response strategy has a lower-bound above the minimax payoff. Therefore, the learned strategy is safe with probability no less than $1 - \delta$.

The resulting counter-strategy eventually converges to $\text{BR}(\boldsymbol{\pi}_j^*)$ if the opponent's strategy converges to a stationary strategy $\boldsymbol{\pi}_j^*$. As the number of observations increases infinitely, the bound ε on the KL divergence converges to zero. For an infinite amount of data, the set of possible opponent's strategies shrinks to only one element which is the empirical estimate $\tilde{\boldsymbol{\pi}}_j$, and the empirical distribution $\tilde{\boldsymbol{\pi}}_j$ converges to $\boldsymbol{\pi}_j^*$. As a result, the counter-strategy will eventually converge to the best-response against $\boldsymbol{\pi}_j^*$.

The learned counter-strategy also converges sufficiently fast to the best-response against a stationary opponent. We now assume that the opponent uses a stationary strategy $\boldsymbol{\pi}_j^*$

and the game has been played t times. The *regret* is given by the difference of expected payoffs between the learned strategy and the best-response. As shown in the appendix, the expected regret for the next game is upper-bounded as $\mathbb{E}[r_t] \leq 4\beta\sqrt{2\varepsilon} + \delta\tau$, where β and τ are constants and $\varepsilon = (|\mathcal{A}_j| \ln(t+1) - \ln C)/t$. Thus, the expected accumulated regret in the first T games is upper-bounded as

$$\mathcal{R}_T = \sum_{t=1}^{T} \mathbb{E}[r_t] \leq \sum_{t=1}^{T} \left(\text{const}_1 \sqrt{\ln(t+1)/t} + \text{const}_2/t \right).$$

As the partial sums of harmonic series have logarithmic growth rate and the partial sums of the series $\sqrt{\ln t/t}$ have growth rate of $\mathcal{O}(\sqrt{T \ln T})$, the accumulated regret bound has a growth rate of $\mathcal{O}(\sqrt{T \ln T})$. Therefore, the algorithm has zero average regret and converges fast.

Adaptive Strategy Update

Given N observed actions from consecutive games, the learned strategy is δ-safe only if these actions are sampled from the same strategy $\boldsymbol{\pi}_j^*$. However, the opponent may also update its strategy during the games. Therefore, an adaptive learning algorithm is required to deal with the changes in the opponent's strategy and learn against locally stationary strategies.

We propose an algorithm that keeps accumulating observations and improves the counter-strategy when the opponent's strategy is locally stationary. It also detects changes in the strategy of the opponent and recomputes the counter-strategy accordingly. The proposed algorithm, as outlined in Algorithm 1, maintains two sets of samples: a set $\mathbf{X}$ that contains observed actions for learning a counter-strategy, and a set $\mathbf{Y}$ that serves as a validation set to test if the strategy has changed. The algorithm is online so that the strategy $\boldsymbol{\pi}_i$ can be updated while the opponent is also adjusting its strategy against $\boldsymbol{\pi}_i$.

In step 12 of the algorithm, we test the hypothesis that the probability of executing action a_j is the same in local strategies that generated the sample set $\mathbf{X}$ and the validation set $\mathbf{Y}$. The sampled actions are converted into binary variables indicating if they are equal to a_j. Then, the empirical mean of these binary variables is tested with the hypotheses: $H_0 : p_X(a_j) = p_Y(a_j)$, vs. $H_1 : p_X(a_j) \neq p_Y(a_j)$.

Let $\mu = (\tilde{p}_X(a_j) + \tilde{p}_Y(a_j)/(|\mathbf{X}| + |\mathbf{Y}|)$, and $\sigma^2 = \mu(1-\mu)$. With large sample sets $\mathbf{X}$ and $\mathbf{Y}$, the statistic

$$z_0 = \frac{\tilde{p}_X(a_j) - \tilde{p}_Y(a_j)}{\sigma\sqrt{1/|\mathbf{X}| + 1/|\mathbf{Y}|}}$$

is approximately normally distributed, where $\tilde{p}_X(a_j)$ and $\tilde{p}_Y(a_j)$ are empirical probabilities of action a_j in $\mathbf{X}$ and $\mathbf{Y}$. The test of H_0 fails if $|z_0| > z_{\alpha/2}$, where α is the significance level of the test.

Extension to Finite Stage Stochastic Games

For stochastic games, we only consider those with a finite number of stages, which can be represented by a tree structure of states $s \in \mathcal{S}$. The game starts from an initial state s_0 at stage 1. At each state s, the players choose their actions a_i^s and a_j^s from action spaces $\mathcal{A}_i^s$ and $\mathcal{A}_j^s$ respectively. Let Child(s) denote the set of states that possibly follow after the state s. The game transfers to a subsequent state $s_k \in$ Child(s) with probability $P_s(s_k|a_i^s, a_j^s)$ based on the joint action, and the player i receives an immediate reward $\mathbf{R}^s_{a_i^s, a_j^s}$. The game terminates when a leaf state s_l is reached, and the global reward for player i is the sum of all rewards that it obtained in all stages. For stochastic games, a strategy consists of local strategies at every state.

The strategy learning algorithm can be extended to repeated stochastic games with a finite number of stages. Here, the same opponent modeling technique is used at every state. We model the opponent's strategy at state s by the set $\Omega^s(\delta) \triangleq \{\boldsymbol{\pi}_j^s \in \Delta^{|\mathcal{A}_j^s|} | \text{KL}(\tilde{\boldsymbol{\pi}}_j^s || \boldsymbol{\pi}_j^s) \leq \varepsilon(\delta, N_s)\}$, where the function $\varepsilon(\delta, N_s) = ((|\mathcal{A}_j^s| - 1)\ln(N_s + 1) - \ln(\delta/|\mathcal{S}|))/N_s$ depends on N_s, i.e. the number of observations available at state s.

A different model of the opponent is used at every state. The counter-strategy is recursively computed, starting with the states of the last stage. At a state in the last stage, only the immediate reward is considered. Therefore, learning the counter-strategy is the same as in normal-form games, resulting in a δ-safe estimate of expected payoff at this state. The estimated payoff is back propagated to the previous stage as a lower-bound of the future expected reward. Hence, the reward at the previous stage is given by the sum of the immediate reward and the estimated future reward. Then, the counter-strategy at the previous stage is learned with the updated reward matrices. The counter-strategies at every state

1 $\mathbf{X} := \emptyset$.;
2 Initialize $\boldsymbol{\pi}_i$ to be the minimax strategy;
3 **repeat**
4 **while** $|\mathbf{X}| < N_{\min}$ **do**
5 Draw an action a_i from the current strategy $\boldsymbol{\pi}_i$;
6 Observe a new sample a_j;
7 $\mathbf{X} := \mathbf{X} \cup \{a_j\}$;
8 Update $\boldsymbol{\pi}_i$ by solving Problem (1);
9 **end**
10 **repeat**
11 Get $N_{\min}$ new samples as $\mathbf{Y}$;
12 Perform two-sample tests for each action a_j in $\mathbf{X}$ and $\mathbf{Y}$;
13 **if** *all the tests were passed* **then**
14 $\mathbf{X} := \mathbf{X} \cup \mathbf{Y}$;
15 Update $\boldsymbol{\pi}_i$ by solving Problem (1);
16 **end**
17 **until** *test failed or game is over*;
18 $\mathbf{X} := \emptyset$;
19 Reset $\boldsymbol{\pi}_i$ to be the minimax strategy;
20 **until** *the game is over.*;

Algorithm 1: Adaptive algorithm for learning counter-strategies in normal-form games.

(a) The payoff matrix in rock-paper-scissors games.

	Rock	Paper	Scissor
Rock	0	-1	1
Paper	1	0	-1
Scissor	-1	1	0

(b) The empirical reward matrix for the table-tennis playing robot.

	Right	Left	Middle
Forehand	0.6506	0.0648	0.5222
Backhand	0.0449	0.3889	0.1222
Middle	0.4103	0.5648	0.7444

Table 1: Reward matrices for the rock-paper-scissors game and the table-tennis playing robot setting.

of each stage are obtained by traversing the state tree in a bottom-up order.

An on-policy learning algorithm is used for updating the strategy. Unlike in normal-form games, stochastic games require the exploration of the entire state space in order to find the best-response counter-strategy. If the strategy $\boldsymbol{\pi}_i^s$ at the state s has zero probability for an action a_i^s, some subsequent states may not be reachable. Therefore, we propose to use smoothed strategies with an exploration parameter $\xi \in [0, 1]$ such that $\bar{\boldsymbol{\pi}}_i^s(a_j^s) = (1 - \xi)\boldsymbol{\pi}_i^s(a_j^s) + \xi/|\mathcal{A}_j^s|$, to ensure sufficient exploration.

Evaluation

We first evaluate our algorithms in rock-paper-scissors games. Then we apply it to a table-tennis playing robot in order to improve the performance of returning served balls.

Repeated Rock-Paper-Scissors

Rock-paper-scissors is a zero-sum normal-form game. The reward matrix for player i is defined in Table 1(a). We represent strategies $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}_j$ by vectors with elements corresponding to the probabilities of taking rock, paper and scissor respectively. The minimax strategy in the game is to draw actions uniformly, i.e. $\boldsymbol{\pi}_i = [1/3, 1/3, 1/3]$. If player i follows this uniform strategy, its expected reward is zero regardless of what the opponent's strategy is.

First, we evaluate the opponent modeling technique by playing against a stationary opponent's strategy $\boldsymbol{\pi}_j^* = [0.6, 0.2, 0.2]$. The performance of the algorithm is tested on samples ranging exponentially from 1 to 10^8 observations. For each sample size, the experiment is repeated 20 times. The plot in Figure 1 show the estimated expected reward and true expected reward. The true reward is the expected reward when the learned strategy plays against the true opponent's strategy. It is lower-bounded by the estimated reward with probability no less than $1 - \delta$, which is the expected reward when the learned strategy plays against the worst-case opponent's strategy in the model. The reward gradually converges to the optimal reward, showing that the algorithm can exploit the opponent with sufficient observed data.

Then we evaluate the change detection mechanism in Algorithm 1. The game has three thousand repetitions, where the opponent uses a stationary strategy $[0.8, 0.1, 0.1]$ for the first thousand actions and changes to another stationary strategy $[0.1, 0.1, 0.8]$ for the remaining two thousand actions. After the first thousand repetitions, the learned strategy $\boldsymbol{\pi}_i$ tends to take paper more frequently than other actions. Therefore, the sudden switch of the opponent's strategy causes a significant drop of the true expected reward. As

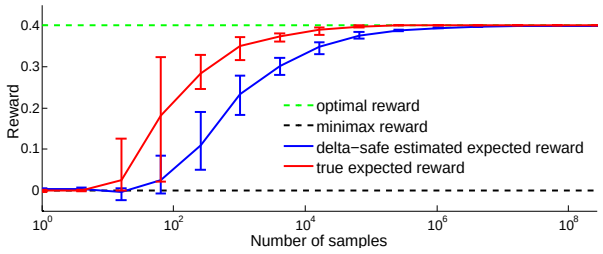

Figure 1: With a stationary opponent's strategy, the expected reward grows with the increasing number of samples, and converges to the optimal reward. The error bars show the maximal and minimal reward in 20 repeated tests.

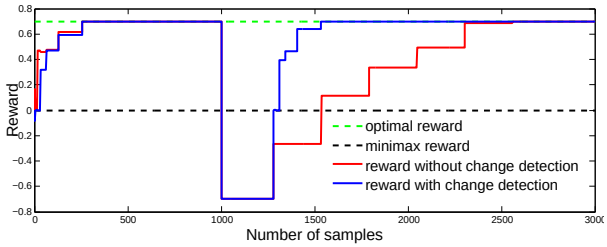

Figure 2: Comparison of true reward by algorithms with or without change detection. The algorithm with change detection can adapt to strategy switch more efficiently. The step curve is caused by the minimal sample size $N_{\min} = 256$.

s_i	$a_i^{s_0}$	$a_j^{s_0}$	$\boldsymbol{\pi}_j^{s_i}$	s_i	$a_i^{s_0}$	$a_j^{s_0}$	$\boldsymbol{\pi}_j^{s_i}$
s_1	R	R	1/3,1/3,1/3	s_2	R	P	1/5,3/5,1/5
s_3	R	S	2/5,2/5,1/5	s_4	P	R	1/5,2/5,2/5
s_5	P	P	1/3,1/3,1/3	s_6	P	S	1/5,1/5,3/5
s_7	S	R	3/5,1/5,1/5	s_8	S	P	2/5,1/5,2/5
s_9	S	S	1/3,1/3,1/3				

Table 2: The local-strategies for the designed opponent. (i) It tends to avoid taking the previous action if it lost the first stage. (ii) It prefers to take the previous action if it won the first stage. (iii) It draws actions uniformly if the first stage was a tie game.

displayed in Figure 2, the algorithm shows its efficiency in adjusting to such situations.

Two-Stage Rock-Paper-Scissors

Considering that human players change their strategies based on previous games in practice, we use a two-stage form of the regular rock-paper-scissors as an example of stochastic games with a finite number of stages, We assign a reward matrix in Stage 2 that is twice as much as it is in Stage 1. The game states can be represented by a two-level tree of states. The root state s_0 corresponds to a regular rock-paper-scissors game at Stage 1, and the game transfers to one of nine following states according to the executed joint action.

We design an opponent with a stationary strategy [0.6, 0.2, 0.2] in the first stage. Its preferences in Stage 2 are shown in detail in Table 2, as well as the conditions for transferring to a subsequent state s_i.

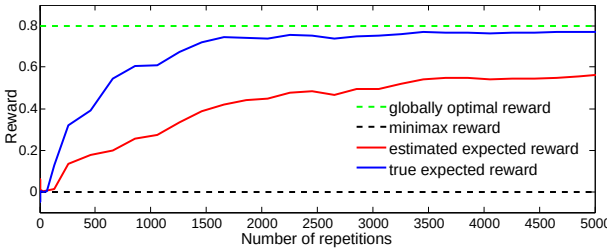

Figure 3: In two-stage rock-paper-scissors games, both the estimated expected reward and true expected reward converge to the globally optimal reward.

(a) forehand pose (b) middle pose (c) backhand pose

Figure 4: Three pre-defined preparation poses. They are optimized for hitting points in different regions.

The performance of the learning algorithm is evaluated by its expected reward. As shown in Figure 3, it converges to the globally optimal reward.

Returning Served Table-Tennis Balls

The proposed modeling technique is used to learn strategies which allow a table-tennis playing robot to improve its response to balls served by human opponents. The used table-tennis robot has three possible high-level actions, namely, choosing the forehand, backhand or middle preparation pose before the opponent serves. Each action has a relatively high success rate when the ball is served to its corresponding region (Figure 4). However, the robot is limited in its velocity and acceleration, hence, it has a low success rate if the served ball is far from the chosen preparation pose. This algorithm allows generating strategies such that the robot would be more likely to successfully return the balls. We could use the low-level planner to evaluate the learned strategies.

The robot chooses its preparation pose while the opponent serves. Therefore, they take actions independently. We can consider the problem as a repeated two-player game. The empirical reward matrix for the robot can be computed as the success rates for every joint action. The success rates are estimated by the low-level planner with 600 recorded served ball trajectories from different human players. As shown in Table 1(b), we roughly know how well a robot's action can return balls in those three regions. According to this reward matrix, the minimax play follows the strategy of $[0.2088, 0, 0.7912]$, whose elements correspond to choosing the forehand, backhand and middle pose respectively. It assumes a competitive opponent and prefers to choose the middle position. However, it is not the optimal strategy if an opponent tends to serve the ball to the right region more frequently.

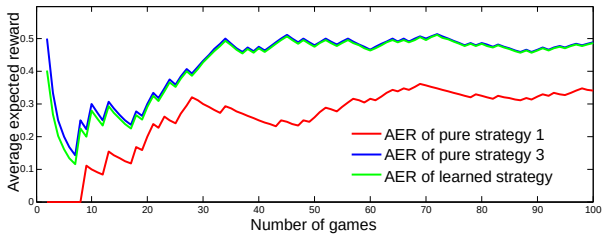

Figure 5: The curves show the average expected reward. The pure strategy 1 is to always prepare at the forehand position. The pure strategy 3 is to always rest at the middle position, which is the optimal strategy.

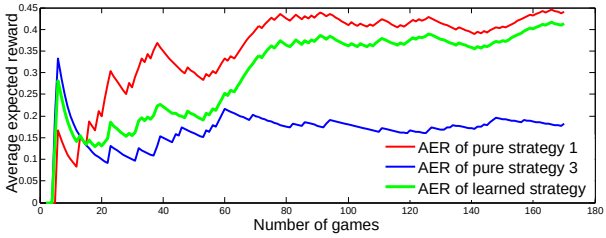

Figure 6: The curves show the average expected reward. The pure strategy 1 is the optimal counter-strategy.

We recruited three volunteers to repeatedly serve the ball. The experiment with each volunteer consists in over one hundred trials. For each trial, the low-level planer provides three binary outputs, indicating whether a high-level action can successfully return the served ball according to its trajectory. The learned stochastic strategy at each trial is evaluated by the expected reward, i.e., the success rate. To analyze the performance of the algorithms, we show the curve of its Average Expected Reward (AER), which is the sum of its expected reward divided by the number of trials.

The first volunteer served the balls with approximately a uniform distribution over the three regions. Therefore, *pure strategy* 3, which always takes action 3, leads to the maximal average expected reward. The results are shown in Figure 5. Our δ-safe strategies are slightly worse than the pure strategy 3 in the beginning as it starts from playing the minimax strategy, yet quickly converge to the optimal strategy.

The second volunteer had a significant preference to serve the balls to the right region. According to the results in Figure 6, the learned strategies follow the minimax strategy at first, as there are not enough observations. With the accumulation of observations, it moves gradually towards *pure strategy* 1 that only takes action 1, which is the optimal counter-strategy against this opponent.

The third volunteer started with a strategy that prefers to serve the ball to the middle/right side, and intentionally switched his strategy after around 85 trials to preferring the middle/left side. Pure strategies 1 and 3 are respectively the optimal counter-strategy before and after the switch. We compare the Algorithm 1 to its simplified version without change detection in this case. As shown in Figure 7, both al-

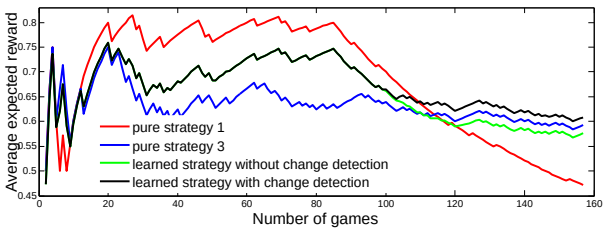

Figure 7: The algorithm with change detection outperforms other strategies against the volunteer 3.

gorithms have decreased performance right after the switch happens. As the two-sample test failed after playing one hundred trials, the proposed algorithm successfully detected the strategy switch and adapted to it by re-initializing to the minimax strategy. Therefore, it outperforms other strategies after the switch happens, and results in the best average expected reward for all trials.

The algorithm can run efficiently in real table-tennis games. We use the projected sub-gradient method to find the optimal counter-strategy in each repetition of the game, which is initialized with the counter-strategy used in the previous game. In the table-tennis problem, the algorithm takes less than 50ms on average to compute a response for each serve.

Conclusions

We introduced a new opponent modeling technique, which models the opponent by a set of strategies whose KL divergence from the empirical distribution is bounded. Based on it, we proposed δ-safe algorithms for both normal-form games and stochastic games with a finite number of stages. The algorithms learn strategies whose expected reward has a lower-bound of minimax payoff with probability no less than $1-\delta$. We showed that the learned strategies are converging to the best-response strategy and have zero average regret. We also employ two-sample tests to deal with locally stationary opponents. We evaluated the performance of our algorithms in rock-paper-scissors games and a table-tennis playing robot setting. The experimental results show that the proposed algorithms can balance safety and exploitability in opponent modeling, and adapt to changes in the opponent's strategy.

Acknowledgement

The authors would like to thank Yevgeny Seldin for valuable discussions.

Derivation of Expected Regret Bound

The expected regret for the game $t+1$ is $\mathbb{E}[r_t] = \boldsymbol{\pi}_i^{*T}\mathbf{R}\boldsymbol{\pi}_j^* - \boldsymbol{\pi}_i^{tT}\mathbf{R}\boldsymbol{\pi}_j^*$, where $\boldsymbol{\pi}_i^*$ is the best-response against $\boldsymbol{\pi}_j^*$. There are two situations.
(1): When $\boldsymbol{\pi}_j^* \notin \Omega(\delta)$,

$$r_t \le \tau \triangleq \max_{a_i,a_j} \mathbf{R}_{a_i,a_j} - \min_{a_i,a_j} \mathbf{R}_{a_i,a_j}.$$

(2): When $\boldsymbol{\pi}_j^* \in \Omega(\delta)$, $\|\boldsymbol{\pi}_j^* - \boldsymbol{\pi}_j^t\|_1 \le \|\boldsymbol{\pi}_j^t - \tilde{\boldsymbol{\pi}}_j^t\|_1 + \|\boldsymbol{\pi}_j^* - \tilde{\boldsymbol{\pi}}_j^t\|_1 \le \sqrt{2\mathrm{KL}(\tilde{\boldsymbol{\pi}}_j^t\|\boldsymbol{\pi}_j)} + \sqrt{2\mathrm{KL}(\tilde{\boldsymbol{\pi}}_j^t\|\boldsymbol{\pi}_j^*)} \le 2\sqrt{2\varepsilon}$, where $\tilde{\boldsymbol{\pi}}_j^t$ is the empirical distribution obtained during those t games. $\mathbb{E}[r_t] = \boldsymbol{\pi}_i^{*T}\mathbf{R}\boldsymbol{\pi}_j^* - \boldsymbol{\pi}_i^{tT}\mathbf{R}\boldsymbol{\pi}_j^*$. As $\boldsymbol{\pi}_i^t$ is a best-response strategy against $\boldsymbol{\pi}_j^t$, $\boldsymbol{\pi}_i^{tT}\mathbf{R}\boldsymbol{\pi}_j^t \ge \boldsymbol{\pi}_i^{*T}\mathbf{R}\boldsymbol{\pi}_j^t$.

$$\begin{aligned}
\mathbb{E}[r_t] &= \boldsymbol{\pi}_i^{*T}\mathbf{R}\boldsymbol{\pi}_j^* - \boldsymbol{\pi}_i^{tT}\mathbf{R}\boldsymbol{\pi}_j^t + \boldsymbol{\pi}_i^{tT}\mathbf{R}\boldsymbol{\pi}_j^t - \boldsymbol{\pi}_i^{tT}\mathbf{R}\boldsymbol{\pi}_j^* \\
&\le (\boldsymbol{\pi}_i^{*T}\mathbf{R}\boldsymbol{\pi}_j^* - \boldsymbol{\pi}_i^{*T}\mathbf{R}\boldsymbol{\pi}_j^t) \\
&\quad + (\boldsymbol{\pi}_i^{tT}\mathbf{R}\boldsymbol{\pi}_j^t - \boldsymbol{\pi}_i^{tT}\mathbf{R}\boldsymbol{\pi}_j^*) \\
&= \boldsymbol{\pi}_i^{*T}\mathbf{R}(\boldsymbol{\pi}_j^* - \boldsymbol{\pi}_j^t) + \boldsymbol{\pi}_i^{tT}\mathbf{R}(\boldsymbol{\pi}_j^t - \boldsymbol{\pi}_j^*) \\
&\le \left(\max\{|\boldsymbol{\pi}_i^{*T}\mathbf{R}|\} + \max\{|\boldsymbol{\pi}_i^{tT}\mathbf{R}|\}\right) \|\boldsymbol{\pi}_j^t - \boldsymbol{\pi}_j^*\|_1 \\
&\le 4\beta\sqrt{2\varepsilon},
\end{aligned}$$

where $\beta \triangleq \max_{a_i,a_j} \mathbf{R}_{a_i,a_j}$.

Since the situation (2) happens with probability no less than $1-\delta$ and it has lower expected regret than the situation (1), the expected regret is bounded as $\mathbb{E}[r_t] \le 4\beta\sqrt{2\varepsilon} + \delta\tau$.

References

Bowling, M., and Veloso, M. 2002. Multiagent learning using a variable learning rate. *Artificial Intelligence* 136(2):215–250.

Bowling, M. 2005. Convergence and no-regret in multiagent learning. In *Advances in neural information processing systems 17*, 209.

Brown, G. 1951. Iterative solution of games by fictitious play. *Activity analysis of production and allocation* 13(1):374–376.

Conitzer, V., and Sandholm, T. 2007. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning* 67(1):23–43.

Johanson, M., and Bowling, M. 2009. Data biased robust counter strategies. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*.

Markovitch, S., and Reger, R. 2005. Learning and exploiting relative weaknesses of opponent agents. *Autonomous Agents and Multi-Agent Systems* 10(2):103–130.

McCracken, P., and Bowling, M. 2004. Safe strategies for agent modelling in games. In *AAAI Fall Symposium on Artificial Multi-agent Learning*, 103–110.

Muelling, K.; Kober, J.; and Peters, J. 2010. A Biomimetic Approach to Robot Table Tennis. In *Proceedings of IROS*.

Powers, R.; Shoham, Y.; and Vu, T. 2007. A general criterion and an algorithmic framework for learning in multi-agent systems. *Machine Learning* 67(1):45–76.

Seldin, Y., and Tishby, N. 2010. PAC-Bayesian Analysis of Co-clustering and Beyond. *Journal of Machine Learning Research* 11:3595–3646.

Simon, H. 1991. Bounded rationality and organizational learning. *Organization science* 2(1):125–134.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the International Conference on Machine Learning*.