Bayesian Opponent Exploitation by Inferring the Opponent's Policy Selection Pattern

Kuei-Tso Lee
Institute of Electronics
National Yang Ming Chiao Tung University
Hsinchu, Taiwan
fuj30089@gmail.com

Sheng-Jyh Wang
Institute of Electronics
National Yang Ming Chiao Tung University
Hsinchu, Taiwan
shengjyh@nycu.edu.tw

***Abstract*—In a multi-agent competitive domain, the agent needs to anticipate the opponent's behavior and select a suitable policy to exploit the opponent. In this work, based on the BPR (Bayesian Policy Reuse) framework, we further assume the opponent may determine its policy depending on its previous observation. To deal with opponents of this kind, we discuss three different approaches for the agent, including learning from scratch, reasoning from experience, and reasoning accompanied by learning. The "reasoning accompanied by learning" approach turns out to be the most favorable method, in which the agent executes an iterative process that alternates between "updating the belief of each pre-collected model" and "progressively learning the opponent's policy selection pattern" based on the observed data. In our experiments, we simulate a simplified batter vs. pitcher game. The experimental results show that the "reasoning accompanied by learning" approach does receive a larger averaged utility value than the learn-from-scratch approach and the reason-from-experience approach.**

***Index Terms*—Bayesian Policy Reuse, Bayes rule, opponent exploitation, opponent modeling, non-stationary opponent.**

I. Introduction

Reinforcement learning (RL) has been employed to solve game playing [1]–[3] in stationary environments. To deal with non-stationary opponents, opponent models are typically incorporated into the RL framework [4]–[6]. Moreover, to quickly adapt to the opponent's changing behaviors, it would be more efficient if the agent can obtain some prior knowledge of the opponent's behaviors before the game. For example, Maruan et al. [7] regards the non-stationarity of an opponent's policy as a sequence of stationary tasks and adopts a meta-learning algorithm [8] to learn the Markovian transitions between consecutive tasks so that the agent can quickly adapt to similar non-stationarities at the gaming time.

On the other hand, the Bayesian Policy Reuse (BPR) framework [9] selects one responding policy from a pre-learned policy library when facing an unknown task. Based on the same concept, BPR+ [10] and DPN-BPR+ [14] further treats the task as the opponent's policy. In these two BPR-based methods, the agent continuously updates the belief of the opponent's policy via the Bayes rule and then adopts a suitable policy accordingly. However, they only assume the opponent adopts a very simple policy changing pattern, which randomly switches its policy among several stationary ones. To be more realistic, Yang et al. [11] assume the opponent may adopt either random switching or BPR reasoning technique to play the game and propose the Bayes-ToMoP framework that incorporates the concept of Theory of Mind (ToM) [12], [13] into the BPR framework to deal with such kind of opponents. However, Bayes-ToMoP framework only assumes the opponent has two possible policy selection patterns: random switching or BPR, and this framework uses a heuristic method based on the winning rate to detect the opponent's policy selection pattern.

In this work, we make a more reasonable and realistic assumption that the opponent may select its policy depending on its previous observation. Based on this assumption, we present three approaches to deal with opponents of this kind. In the first approach, the agent learns the opponent's policy selection pattern from scratch. In the second approach, the agent reasons the opponent's policy selection pattern based on pre-collected models. In the third approach, the agent combines both learning and reasoning approaches to infer the opponent's policy selection pattern. In the next section, we first explain the BPR framework, which works as the backbone of our approaches.

II. Background

The Bayesian Policy Reuse (BPR) framework was originally proposed in [9] for an agent to choose its policy π from a pre-learned policy library Π, when facing an unknown task $\tau \in T$. The task τ is defined as a Markov Decision Process and $\pi(a, s)$ represents the probability of choosing the action a given the state s. That is, $\pi(a, s) = p(a|s)$.

A BPR-based agent is equipped with an observation model $p(\sigma|\tau, \pi)$ and a performance model $p(U|\tau, \pi)$. The observation model $p(\sigma|\tau, \pi)$ represents the probability distribution of σ when the agent adopts the policy π to deal with the task τ. Here, $\sigma \in \Sigma$ denotes the information observed by the agent, such as the state-action-state tuples, instantaneous rewards, or terminal utility. On the other hand, the performance model $p(U|\tau, \pi)$ represents the probability distribution of the utility U when the agent adopts the policy π to deal with the task τ. When facing a new task τ^*, a BPR-based agent continuously updates the belief $\beta(\tau)$ through episodes, where $\beta(.)$ denotes the probability distribution of τ over the task space T that measures the similarity between τ^* and τ in T. After the

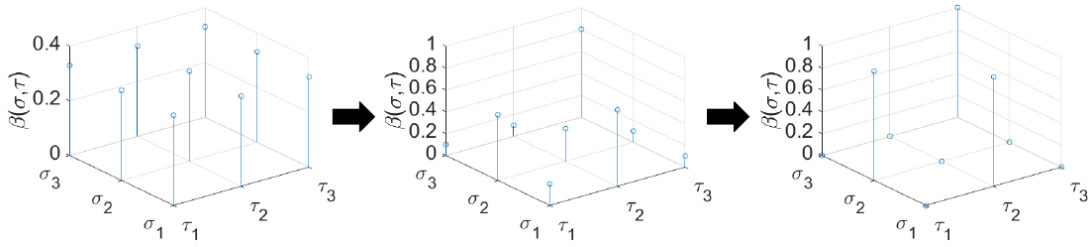


Fig. 1: An illustration of $\beta(\sigma, \tau)$ updating process.

episode at t, based on (σ^t,π^t), and the observation model $p(\sigma|\tau, \pi)$, the BPR-based agent updates the belief $\beta^t(\tau)$ based on the Bayes rule:

$$\beta^t(\tau) = \frac{\beta^{t-1}(\tau)p(\sigma^t|\tau, \pi^t)}{\sum\limits_{\tau' \in T} \beta^{t-1}(\tau')p(\sigma^t|\tau', \pi^t)}. \tag{1}$$

With the belief $\beta^t(\tau)$, there are several policy selection methods in order to select the best policy. For example, the agent may select a policy π^{t+1} from its policy library Π that maximizes the expected utility U under the belief $\beta^t(\tau)$ and the performance model $p(U|\tau, \pi)$. That is,

$$\pi^{t+1} = arg\max_{\pi \in \Pi} \sum_{\tau \in T} \beta^t(\tau)\mathbb{E}[U|\tau, \pi]. \tag{2}$$

Based on Equations (1) and (2), a BPR-based agent can efficiently respond to the unknown task, relying on a small number of episodes. With the same concept, BPR+ [10] and DPN-BPR+ [14] further treat the task τ as the opponent's policy and assume the opponent randomly switches its policy among several predefined policy candidates for every few episodes. As the opponent switches its policy τ, the BPR-based agent can gradually adjust its belief $\beta(\tau)$ based on (1) and thereby exploit the opponent.

Basically, the updating rule in (1) is based on a fundamental assumption that the opponent's current policy is the same as the previous policy and hence the agent can keep updating its belief on the opponent's policy based on every observation. However, in real life, an opponent may change its policy quickly during the game. In this situation, the updating rule in (1) becomes ineffective. In our work, instead, we assume the opponent may determine its policy τ^t based on the previous observation σ^{t-1}. This assumption is more realistic than the assumptions in the BPR variants. In the next section, we explain more details of the proposed framework.

III. Bayesian Opponent Exploitation

In the proposed framework, we model the opponent's policy selection pattern by the form $\tau^t = f_{op}(\sigma^{t-1})$, where the initial selection τ^1 is made in a random way. In this work, we present three different approaches. In the first approach, we assume the agent has no information of $f_{op}(.)$ at all and has to learn $f_{op}(.)$ from scratch. In the second approach, we assume the agent has a pre-collected library that contains several models of the opponent's policy selection pattern and the agent aims to infer $f_{op}(.)$ by referring to the library. In the third approach, we combines the above two approaches and assume the agent may not only refer to the pre-collected library but also try to learn $f_{op}(.)$ from scratch at the same time.

A. Approach 1: Learning from scratch

In the first approach, the agent aims to infer the opponent's adopted policy $\tau \in T$ under each observation $\sigma \in \Sigma$. Here we model the agent's belief of the opponent's current policy τ based on the previous observation σ as $\beta(\sigma, \tau) = p(\tau^t = \tau|\sigma^{t-1} = \sigma)$; that is, $\beta : \Sigma \times T \rightarrow [0, 1]$. Besides, we initialize $\beta(.)$ as a uniform distribution of τ for every possible value of σ.

During the game, after observing (σ^t, π^t) at episode t, the agent updates its belief of τ under the condition that the previous observation is $\sigma = \sigma^{t-1}$ based on the Bayes rule, which can be expressed as

$$\beta(\sigma, \tau) \leftarrow \frac{\beta(\sigma, \tau)p(\sigma^t|\tau, \pi^t)}{\sum\limits_{\tau' \in T} \beta(\sigma, \tau')p(\sigma^t|\tau', \pi^t)}, \tag{3}$$

where $\sigma = \sigma^{t-1}$.
After that, the agent predicts the opponent's next policy via $\beta(\sigma^t, \tau)$ and selects π^{t+1} based on the following equation:

$$\pi^{t+1}(\sigma^t) = arg\max_{\pi \in \Pi} \sum_{\tau \in T} \beta(\sigma^t, \tau)\mathbb{E}[U|\tau, \pi]. \tag{4}$$

As the episode proceeds, the agent dynamically updates $\beta(\sigma, \tau)$ via (3). As t becomes large, $\beta(\sigma, \tau)$ is expected to approach the real $\beta(.)$, which is defined as

$$\beta_{real}(\sigma, \tau) = \begin{cases} 1, \tau = f_{op}(\sigma) \\ 0, \tau \neq f_{op}(\sigma) \end{cases}. \tag{5}$$

That is, the agent may eventually get a close prediction of the opponent's next policy. With the predicted policy, the agent can select the optimal policy π^{t+1} to exploit the opponent. An illustration of the $\beta(\sigma, \tau)$ updating process is illustrated on Figure 1.

B. Approach 2: Reasoning from experience

In this approach, we assume the agent has obtained n models of $\beta(.)$ beforehand to form a library $B = \{\beta_1, \beta_2, \beta_3, ..., \beta_n\}$. As the game proceeds, the agent continuously reasons which β_i in B is most similar to the opponent's policy selection behavior based on the observed σ, together with the agent's policy π. With this reasoning process, the inference of $f_{op}(.)$ becomes a classification problem to classify

the opponent's policy selection behavior into one of the β models in the B library. Here, we no longer need to learn $f_{op}(.)$ from scratch and this speeds up the inference process greatly.

Here we use $\gamma(.)$ to represent the agent's belief over β_i in B and initialize $\gamma(.)$ with a uniform distribution. In our formulation, after observing (σ^t, π^t) at the episode at t, the agent updates $\gamma(\beta_i)$ by

$$\gamma(\beta_i) \leftarrow \frac{\gamma(\beta_i)L(\beta_i|\sigma^t,\pi^t,\sigma^{t-1})}{\sum_{i=1}^{n}\gamma(\beta_i)L(\beta_i|\sigma^t,\pi^t,\sigma^{t-1})}, \quad (6)$$

where $L(\beta_i|\sigma^t,\pi^t,\sigma^{t-1})$ represents the likelihood function of β_i.

To derive $L(\beta_i|\sigma^t,\pi^t,\sigma^{t-1})$, we combine the model $\beta_i(\sigma^{t-1},\tau)$ and the model $p(\sigma^t|\tau,\pi^t)$ and marginalize the variable τ out. That is, we define $L(\beta_i|\sigma^t,\pi^t,\sigma^{t-1})$ as $\sum_{\tau\in T}\beta_i(\sigma^{t-1},\tau)p(\sigma^t|\tau,\pi^t)$. Based on this formulation, (6) can be rewritten as

$$\gamma(\beta_i) \leftarrow \frac{\gamma(\beta_i)\sum_{\tau\in T}\beta_i(\sigma^{t-1},\tau)p(\sigma^t|\tau,\pi^t)}{\sum_{i=1}^{n}\gamma(\beta_i)\sum_{\tau\in T}\beta_i(\sigma^{t-1},\tau)p(\sigma^t|\tau,\pi^t)}. \quad (7)$$

With (7), the agent predicts τ^{t+1} by taking the weighted average of $\beta_i(\sigma^t,\tau)$, with the weight $\gamma(\beta_i)$, and selects π^{t+1} via the following equation:

$$\pi^{t+1}(\sigma^t) = arg\max_{\pi\in\Pi}\sum_{\tau\in T}[\sum_{i=1}^{n}\gamma(\beta_i)\beta_i(\sigma^t,\tau)]\mathbb{E}[U|\tau,\pi]. \quad (8)$$

As the episode proceeds, the agent adaptively updates $\gamma(\beta_i)$. As t becomes large, $\gamma(.)$ will be inclined to the β_i in B which is most similar to β_{real} than the other models. If β_{real} is not too far from this β_i, the agent may still effectively deal with the opponent. However, we would expect this β_i cannot predict the opponent's policy as accurately as the learn-from-scratch approach in Approach 1 does. For the fortunate case that β_{real} happens to be the same as the model β_i in the library B, $\gamma(\beta_{real})$ will be triggered to approach 1. In this case, the term $\sum_{i=1}^{n}\gamma(\beta_i)\beta_i(\sigma^t,\tau)$ in (8) can be simplified to $\beta_{real}(\sigma^t,\tau)$ and the agent can successfully predict the opponent's next policy and select the optimal π to exploit the opponent.

Here, in a conceptual way, we compare the behavior of the learn-from-scratch approach in Approach 1 and the reason-from-experience in Approach 2. With the learn-from-scratch approach, the agent can eventually learn the opponent's policy selection function $f_{op}(.)$ as t becomes large. However, the learning process takes time. On the other hand, with the reason-from-experience approach, the agent can quickly select the most similar β_i in B as the opponent's policy selection pattern. However, as t becomes large, the agent's received utility may be restricted to a lower value if β_{real} does not belong to the library B.

In Figure 2, we illustrate the agents' received utilities versus t of these two approaches. For the reason-from-experience approach, as shown in blue, its received utility value increases quickly in the beginning episodes but is restricted when t is large. On the other hand, for the learn-from-scratch approach, as shown in black, its utility value increases slowly in the beginning episodes but will reach a higher value than the reason-from-experience approach as t becomes large.

Now, if we may combine the advantages of these two approaches, the agent's received utility value versus t curve may look like the red line in Figure 2. This "Reasoning accompanied by learning" approach is to be explained in the next subsection.

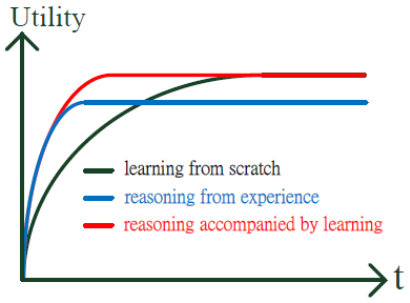


Fig. 2: An illustration of the agents' utility versus t.

C. Approach 3: Reasoning accompanied by learning

Similar to Approach 2, we assume the agent has obtained the library B before the game. Apart from this, we add in an extra adjustable model $\beta_*(.)$ into the B library. During the game, $\beta_*(.)$ is iteratively adjusted based on (3) to learn $f_{op}(.)$. On the other hand, the agent also updates $\gamma(\beta_i)$ to infer $f_{op}(.)$ from the extended B library. The flow of the proposed inference process is described in Algorithm 1.

Algorithm 1 Reasoning accompanied by learning

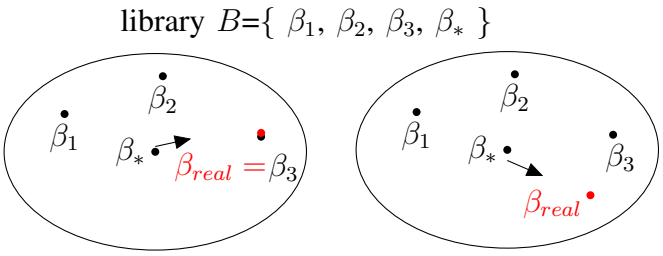
Require: agent's policy library Π, opponent's policy library T, opponent's model library B, performance model $p(U|\tau,\pi)$, observation model $p(\sigma|\tau,\pi)$.

1: Add $\beta_*(.)$ into B and initialize $\beta_*(.)$ with a uniform distribution of τ for every value of σ
2: Initialize $\gamma(.)$ as a uniform distribution
3: Randomly select π^1 from Π
4: **for** episodes t=1 to N **do**
5: adopt π^t to play and observe σ^t
6: $\gamma(\beta_i) \leftarrow \frac{\gamma(\beta_i)\sum_{\tau\in T}\beta_i(\sigma^{t-1},\tau)p(\sigma^t|\tau,\pi^t)}{\sum_{i=1}^{n+1}\gamma(\beta_i)\sum_{\tau\in T}\beta_i(\sigma^{t-1},\tau)p(\sigma^t|\tau,\pi^t)}$
7: $\beta_*(\sigma,\tau) \leftarrow \frac{\beta_*(\sigma,\tau)p(\sigma^t|\tau,\pi^t)}{\sum_{\tau'\in T}\beta_*(\sigma,\tau')p(\sigma^t|\tau',\pi^t)}$, where $\sigma=\sigma^{t-1}$
8: $\pi^{t+1}(\sigma^t) = arg\max_{\pi\in\Pi}\sum_{\tau\in T}\{\sum_{i=1}^{n+1}\gamma(\beta_i)\beta_i(\sigma^t,\tau)\}\mathbb{E}[U|\tau,\pi]$
9: **end for**

With the "reasoning accompanied by learning" algorithm, the agent updates $\gamma(\beta_i)$ and $\beta_*(\sigma,\tau)$ via (σ^t,π^t) alternately at the episode at t and then selects π^{t+1} based on the weighted prediction of $\beta_i(\sigma^t,\tau)$, $\sum_{i=1}^{n+1}\gamma(\beta_i)\beta_i(\sigma^t,\tau)$. As more and more

observation becomes available, $\gamma(\beta_i)$ and $\beta_*(\sigma,\tau)$ will be adjusted to make the prediction $\sum_{i=1}^{n+1}\gamma(\beta_i)\beta_i(\sigma^t,\tau)$ match the opponent's adopted policy. After that, the agent may select the optimal policy to exploit the opponent.

Next, we give an example and consider two extreme cases to explain the weighted prediction $\sum_{i=1}^{n+1}\gamma(\beta_i)\beta_i(\sigma^t,\tau)$. In this example, we assume there are three β models, β_1, β_2, β_3, in the library B and we add in one extra model β_* to form the extended library, where β_1, β_2, and β_3 remain unchanged while β_* continues to be updated via the learning-from-scratch process. In Case 1, we assume $\beta_{real}=\beta_3$; while in Case 2, we assume β_{real} is different from β_1, β_2, and β_3, as illustrated in Figure 3.



(a) Case1: $\beta_{real}=\beta_3$

(b) Case2: β_{real} is different from β_1, β_2, and β_3.

Fig. 3: An illustration of an example, where β_1, β_2, and β_3 remain unchanged while β_* is adjusted to approach β_{real}.

In Case 1, the agent's belief of β_3, $\gamma(\beta_3)$, will be quickly triggered to approach 1 before β_* is adjusted to β_{real}. In this case, the weighted prediction $\sum_{i=1}^{n+1}\gamma(\beta_i)\beta_i(\sigma^t,\tau)$ will converge to $\beta_3(\sigma^t,\tau)$.

In Case 2, the agent cannot find a clear match between β_{real} and the pre-collected β_i in B. As the episode proceeds, the learned β_* will gradually move toward β_{real}. As t becomes large, β_* will converge to β_{real} and $\gamma(\beta_*)$ will converge to 1. In this case, the weighted prediction $\sum_{i=1}^{n+1}\gamma(\beta_i)\beta_i(\sigma^t,\tau)$ will converge to $\beta_*(\sigma^t,\tau)$.

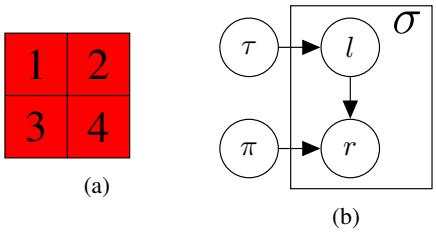
Based on the above analysis, we would expect that no matter whether β_{real} is in B or not, the "reasoning accompanied by learning" approach can help the agent to predict the opponent's policy as t becomes large and to select the proper policy to exploit the opponent.

IV. Experiment

In our experiments, we construct a simplified batter vs. pitcher game (BvPG) to compare the performance of different approaches.

A. Description of batter vs. pitcher game (BvPG)

In BvPG, the batter aims to predict the pitcher's targeted location in order to hit the ball, while the pitcher tries to avoid the ball being hit by the batter.This competing process repeats for several episodes. Here, we treat the batter as the agent and the pitcher as the opponent. The flow for a pitch in an episode is illustrated in Figure 4b. In an episode, the pitcher may choose the policy τ_1, τ_2, τ_3, or τ_4 that targets on the location 1 to 4 to pitch, respectively, as illustrated in Figure 4a. To make the game more realistic, each pitch contains a certain degree of uncertainty and the actually pitched location falls at the location l, where $1\le l\le 4$. On the other hand, the batter may choose the policy π_1, π_2, π_3, or π_4 that targets on the location 1 to 4 to hit, respectively. Depending on l and π, the result r may be "hit" or "miss", denoted by $r=1$ and $r=0$, respectively. For the batter, we define the value of the utility U to be 1 or 0 if r is "hit" or "miss", respectively. Here, we treat (l,r) as the observation σ and we define 8 different cases of σ, as listed in Table I.



(a) (b)

Fig. 4: Illustration of the batter vs. pitcher game (BvPG). (a) Pitch locations. (b) Flow for a pitch in an episode.

TABLE I: The corresponding (l,r) for each definition of the 8 observations.

$\sigma_1:(1,1)$	$\sigma_2:(1,0)$	$\sigma_3:(2,1)$	$\sigma_4:(2,0)$
$\sigma_5:(3,1)$	$\sigma_6:(3,0)$	$\sigma_7:(4,1)$	$\sigma_8:(4,0)$

The transition probability $p(l|\tau)$ and $p(r|l,\pi)$ are defined in detail in the appendix. Based on $p(l|\tau)$ and $p(r|l,\pi)$, we can derive the observation model $p(\sigma|\tau,\pi)$ and the performance model $p(U|\tau,\pi)$. With the defined $p(U|\tau,\pi)$, we may obtain $\mathbb{E}[U|\tau,\pi]$, as listed in Table II. Referring to Table II, if a batter knows the pitcher's next policy is τ_i, the batter will select π_i to play. The corresponding batter's expected utility value is 0.834. This is the upper bound of the batter's expected received utility value.

TABLE II: $\mathbb{E}[U|\tau,\pi]$, the batter's expected utility given τ and π.

	τ_1	τ_2	τ_3	τ_4
π_1	0.834	0.454	0.454	0.188
π_2	0.454	0.834	0.188	0.454
π_3	0.454	0.188	0.834	0.454
π_4	0.188	0.454	0.454	0.834

B. Compare Batters' performance against different pitchers

In this experiment, we define 4 types of batters (the agent), which include

Batter A: learning from scratch;

Batter B: reasoning from the library B;

Batter C: reasoning from the library B accompanied by learning from scratch; and
Batter D: playing based on the BPR mechanism.

In our experiments, we construct the library to be $B = \{\beta_1, \beta_2, \beta_3, ..., \beta_8\}$, as listed in Table III. This table shows the predicted opponent's policy τ at $t+1$ given the observation σ at t for each of the eight β models. For example, for the β_1 model, if σ_1 is observed at t, then the predicted opponent's policy at $t+1$ is τ_1. Similarly, for the β_8 model, if σ_5 is observed at t, the predicted opponent's policy at $t+1$ is τ_3.

TABLE III: Library B

	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8
$\beta_1(.)$	τ_1	τ_1	τ_1	τ_1	τ_1	τ_1	τ_1	τ_1
$\beta_2(.)$	τ_2	τ_2	τ_2	τ_2	τ_2	τ_2	τ_2	τ_2
$\beta_3(.)$	τ_3	τ_3	τ_3	τ_3	τ_3	τ_3	τ_3	τ_3
$\beta_4(.)$	τ_4	τ_4	τ_4	τ_4	τ_4	τ_4	τ_4	τ_4
$\beta_5(.)$	τ_3	τ_3	τ_3	τ_3	τ_1	τ_1	τ_1	τ_1
$\beta_6(.)$	τ_4	τ_4	τ_4	τ_4	τ_2	τ_2	τ_2	τ_2
$\beta_7(.)$	τ_4	τ_1	τ_3	τ_2	τ_2	τ_3	τ_1	τ_4
$\beta_8(.)$	τ_1	τ_4	τ_2	τ_3	τ_3	τ_2	τ_4	τ_1

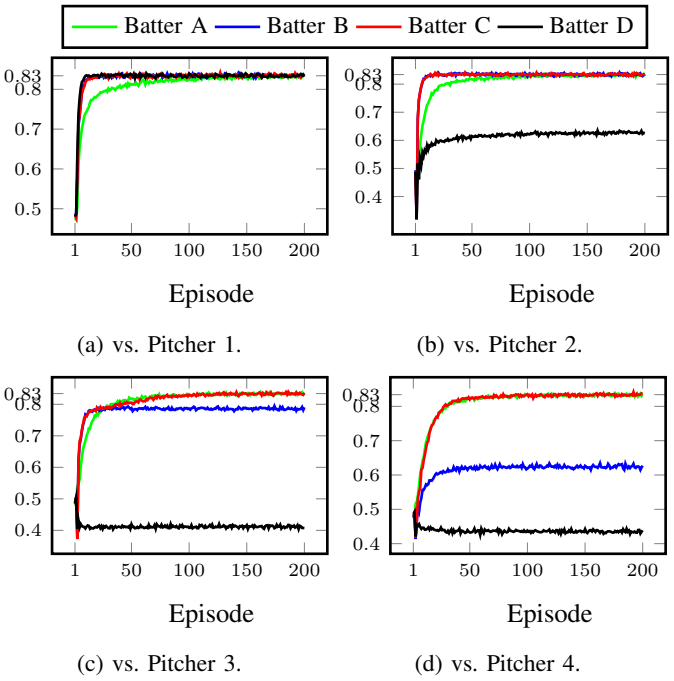
For β_1, β_2, β_3, and β_4, they always choose the same τ no matter what σ is observed. On the other hand, for β_5, it may predict τ_3 or τ_1 depending on the observation σ. For β_6, it may predict τ_4 or τ_2 depending on the observation σ. For β_7, when r is "miss" ($r = 0$) or "hit" ($r = 1$) at location i, β_7 will predict the same location or the diagonal location, respectively, at the next episode. In the contrast, the policy selection pattern of β_8 is opposite to that of β_7.

To evaluate Batters' performance, we define 4 types of pitchers (the opponent), including Pitcher 1, Pitcher 2, Pitcher 3, and Pitcher 4. Their corresponding β_{real} models are defined in Table IV.

TABLE IV: β_{real} for the 4 Pitchers.

	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8
$\beta_{real_1}(.)$	τ_1	τ_1	τ_1	τ_1	τ_1	τ_1	τ_1	τ_1
$\beta_{real_2}(.)$	τ_3	τ_3	τ_3	τ_3	τ_1	τ_1	τ_1	τ_1
$\beta_{real_3}(.)$	τ_4	τ_1	τ_3	τ_2	τ_2	τ_3	τ_2	τ_3
$\beta_{real_4}(.)$	τ_4	τ_1	τ_3	τ_2	τ_4	τ_1	τ_2	τ_3

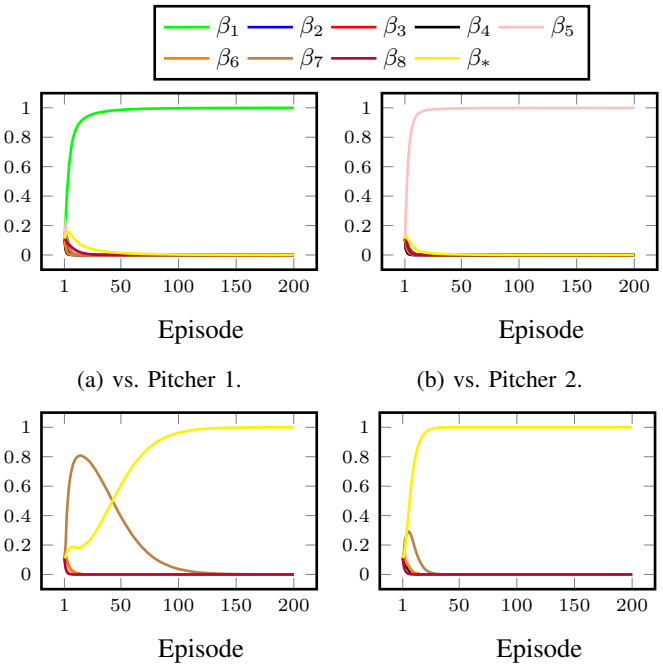
Comparing the batter's β model library B and the pitcher's actual policy selection model β_{real} in Table IV, β_{real_1} and β_{real_2} are included in B ($\beta_{real_1} = \beta_1$, $\beta_{real_2} = \beta_5$), while β_{real_3} and β_{real_4} have two and four elements, respectively, different from β_7 in B. In the following discussion, we use the number of different elements between two β models to represent the "distance" between these two models.
In our experiments, each batter plays against each type of pitcher. For each pairing of batter and pitcher, we simulate 20000 games, with each game containing 200 episodes. After that, we investigate each batter's received utility against Pitcher 1 to Pitcher 4, as shown in Figure 5a to Figure 5d, respectively.



(a) vs. Pitcher 1. (b) vs. Pitcher 2.

(c) vs. Pitcher 3. (d) vs. Pitcher 4.

Fig. 5: Batters' received utility values, averaged over 20000 games.



(a) vs. Pitcher 1. (b) vs. Pitcher 2.

(c) vs. Pitcher 3. (d) vs. Pitcher 4.

Fig. 6: Batter C's belief, $\gamma(\beta)$, averaged over 20000 games.

1) vs. Pitcher 1: Since Pitcher 1 always adopts τ_1, Batter D, who adopts BPR, can quickly detect the pitcher's policy and thereby its utility value increases rapidly to the expected upper bound 0.834. On the other hand, for Batter B and Batter C, since the pitcher's policy selection pattern β_{real_1}

is the same as the β_1 model in B, their belief of β_1 will be quickly triggered and their received utility values also increase quickly to approach 0.834. Especially, Batter C's belief of β_1, with respect to the other β models is plotted in Figure 6a. In comparison, for Batter A, its utility value increases slowly since it learns the opponent's policy selection model from scratch.

2) vs. Pitcher 2: Since Pitcher 2 may adopt τ_3 or τ_1, Batter D with the BPR mechanism cannot accurately predict the opponent's policy τ. Hence, its utility value will be restricted. On the other hand, since $\beta_{real_2} = \beta_5$, the utility values of Batter B and Batter C will increase quickly to 0.834 (Batter C's belief of β_5 is quickly triggered, as shown in Figure 6b.). For Batter A, its utility value increases slowly, comparing with the utility values of Batter B and Batter C.

3) vs. Pitcher 3: Since Pitcher 3 has many different choices of τ, Batter D will be confused and its utility value becomes very low. On the other hand, since β_{real_3} has only two elements different from the β_7 model in B, Batter B will select β_7 as the policy selection model of Pitcher 3 and its utility value increases quickly in the beginning episodes. However, as the episode proceeds, Batter B's utility value is restricted to a value lower than 0.834. In contrast, although Batter A's utility value increases slowly, the learn-from-scratch mechanism will help Batter A's utility value to approach the upper bound 0.834 eventually. For Batter C, since β_{real_3} is close to β_7, $\gamma(\beta_7)$ increases in the beginning episodes. At the same time, β_* continuous to be adjusted to approach β_{real_3}. As the episode proceeds, $\gamma(\beta_*)$ will gradually increase and exceed $\gamma(\beta_7)$ to reach 1, as shown in Figure 6c. With the above belief updating process, Batter C's utility value increases quickly in the beginning and continuous to increase to reach 0.834.

4) vs. Pitcher 4: For Batter D, its utility value is low since it cannot deal with the opponent with many different selections of τ. Since β_{real_4} is very different from the β models in B, the mechanism of reasoning from B does not help too much to increase the utility value. Hence, Batter B's utility value is restricted to only about 0.6. For Batter A and Batter C, with the learning mechanism, their utility values may still reach 0.834 (Batter C's belief of β_* dominates the others, as shown in Figure 6d.).

Until now, we have compared Batters' received utility values with respect to the situation $\beta_{real} \in B$ or $\beta_{real} \notin B$. Next, we further test the case when the batter plays against a pitcher with different degrees of "distance" with respect to the β models in the library B.

C. Batters' performance against pitchers with different degrees of "distance" with respect to B

In this experiment, we first design a procedure to generate the pitcher's β_{real} model in a somewhat random way. Here, we define a parameter q to control the degree of "distance" with respect to the β_i in B defined in Table III.

Random generation of β_{real} with a "distance" q away from B

1. Randomly select a β model β_i in B and copy it to form the prototype model of the β_{real} model.
2. For this prototype model, randomly select q σ-values out of the 8 σ-values, σ_1 to σ_8.
3. For each selected σ-value, randomly change its corresponding τ-value to a different τ-value. For each of the remaining $(8-q)$ σ-values, keep its corresponding τ value unchanged.
4. Check whether the "distance" between the newly modified prototype model and each of the β models in B is equal to or larger than q. If yes, assign the newly modified β model to be the β_{real} model; if no, repeat the above process again.

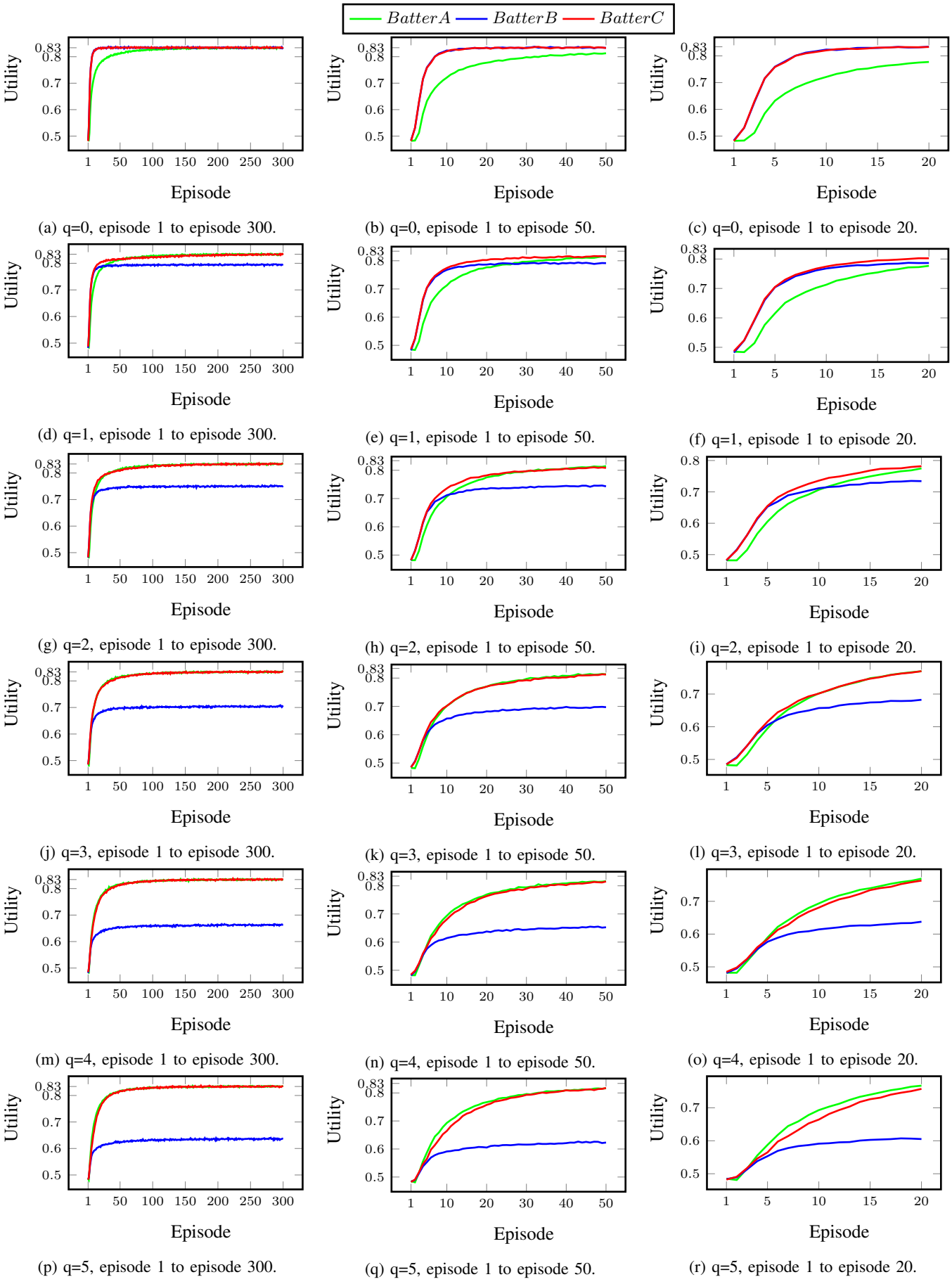
In our experiments, we choose the range of q to be $0 \leq q \leq 5$ and compare the received utility value along episodes for Batter A, Batter B, and Batter C, as shown in Figure 7. For each case, we simulate 100000 games, with each game containing 300 episodes. In each game, β_{real} is randomly generated based on the aforementioned process.

For Batter A and Batter C, who have adopted the learning mechanism, their received utility values always converge to the expected upper bound 0.834 no matter what q-value we choose, as shown in the left column of Figure 7. On the other hand, for Batter B, who only reasons from the pre-collected library B, the saturation value of its utility value decreases as q increases.

Now we investigate Batters' performance in the beginning episodes, as shown in the middle column of Figure 7. Batter B's utility value increases faster than Batter A. However, with the learning-from-scratch mechanism, Batter A's utility value exceeds Batter B's utility value as the game proceeds. If we denote the overtaking moment as t_*, then t_* becomes smaller as q becomes larger. On the other hand, with the "reasoning accompanied by learning" algorithm, Batter C's utility value increases as quickly as Batter B's and converges to 0.834, like Batter A does.

Next, we further investigate Batters' performance from episode 1 to episode 20, as shown in the right column of Figure 7. Similarly, t_* gets smaller as q becomes larger and Batter C combines the advantages of Batter A and Batter B for $0 \leq q \leq 3$. Note that when $q = 4$ or $q = 5$, as shown in Figure 7o and Figure 7r, Batter C's utility value is slightly smaller than Batter A's utility value. This could be due to the fact that β_{real} is very far away from β_i in B and the reason-from-B mechanism causes some incorrect classification results in Batter C's prediction. Fortunately, as the episode proceeds, Batter C's $\gamma(\beta_*)$ still approaches 1 and Batter C's utility value will catch up with Batter A's utility value, as shown in Figure 7q.

Finally, we average each batter's utility value over $0 \leq q \leq 3$ from Figure 7, as shown in Figure 8. Basically, Batter C receives a larger averaged utility value than Batter A and Batter B. This result matches the expectation illustrated in Figure 2.



(a) q=0, episode 1 to episode 300.

(b) q=0, episode 1 to episode 50.

(c) q=0, episode 1 to episode 20.

(d) q=1, episode 1 to episode 300.

(e) q=1, episode 1 to episode 50.

(f) q=1, episode 1 to episode 20.

(g) q=2, episode 1 to episode 300.

(h) q=2, episode 1 to episode 50.

(i) q=2, episode 1 to episode 20.

(j) q=3, episode 1 to episode 300.

(k) q=3, episode 1 to episode 50.

(l) q=3, episode 1 to episode 20.

(m) q=4, episode 1 to episode 300.

(n) q=4, episode 1 to episode 50.

(o) q=4, episode 1 to episode 20.

(p) q=5, episode 1 to episode 300.

(q) q=5, episode 1 to episode 50.

(r) q=5, episode 1 to episode 20.

Fig. 7: Batter's utility value against randomly selected Pitchers, averaged over 100000 games.

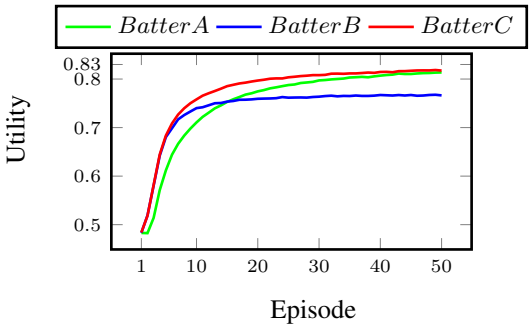


Fig. 8: Batter's utilities, averaged over $0 \leq q \leq 3$.

V. Conclusion

In this work, based on the BPR framework, we further assume the opponent may determine its policy based on the previous observation. Here, we present three different approaches and construct a simplified batter vs. pitcher game to compare their performance. The agent with the reason-from-experience approach may increase its utility value quickly by referring to the pre-collected policy selection models. However, if the opponent's policy selection model is different from the pre-collected models, the agent's utility value will be restricted. On the other hand, the agent with the learn-from-scratch approach can accurately learn the opponent's policy as the episode proceeds and its utility value will reach the expected upper bound. However, this learning-from-scratch approach takes time to learn. In comparison, with the reason-accompanied-by-learn approach, the agent's utility value can not only increase quickly in the beginning but also reach the expected upper bound eventually. This "reason-accompanied-by-learn" approach appears to be the most favorable one among these three approaches when dealing with an opponent with unknown policy selection pattern.

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Anda, M. G., Bellemare, J. V., et al. Human level control through deep reinforcement learning. Nature, 518(7540), 529–533, 2015.

[2] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., and Guez, A. Mastering the game of go without human knowledge. Nature, 550:354–359, 2017.

[3] Silver, D., Hubert, T., Schrittwieser, J., and Antonoglou, I. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science, 362, 2018.

[4] He, H., Boyd-Graber, J., and Kwok, K. Opponent modeling in deep reinforcement learning. In International Conference on Machine Learning (ICML), 2016.

[5] Everett, R. and Roberts, S. Learning against non-stationary agents with opponent modelling and deep reinforcement learning. In Association for the Advancement of Artificial Intelligence (AAAI), 2018.

[6] Albrecht, S. V. and Stone, P. Autonomous agents modelling other agents: A comprehensive survey and open problems. Artificial Intelligence, 2018.

[7] Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. Continuous adaptation via meta- learning in nonstationary and competitive environments. In Proceedings of International Conference on Learning Representations, 2018.

[8] Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In International Conference on Machine Learning (ICML), 2017.

[9] Rosman, B., Hawasly, M., and Ramamoorthy, S. Bayesian policy reuse. Machine Learning, pp. 99–127, 2016.

[10] Hernandez-Leal, P., E. Taylor, M., and Rosman, B. Identifying and tracking switching, non-stationary opponents: A bayesian approach. In Association for the Advancement of Artificial Intelligence(AAAI) Workshop, 2016.

[11] Yang, T., Hao, J., Meng, Z., and Zheng, Y. Towards efficient detection and optimal response against sophisticated opponents. In Joint Conference on Artificial Intelligence (IJCAI), 2019.

[12] de Weerd, H., Verbrugge, R., and Verheij, B. How much does it help to know what she knows you know? Artificial Intelligence, pp. 67–92, 2013.

[13] Baker, C. L., R., S. R., and B., T. J. Bayesian theory of mind: Modeling joint belief-desire attribution. In Annual Meeting of the Cognitive Science Society, 2011.

[14] Zheng, Y., Hao, J., Meng, Z., and Yang, T. Efficient policy detecting and reusing for non-stationarity in markov games. Autonomous Agents and Multi-Agent Systems (AAMAS), 2021.

VI. Appendix

A. Definition of the transition probability $p(l|\tau)$

We define the transition probability $p(l|\tau)$ with the following rules with the parameters κ and λ, where $\lambda = \frac{1-\kappa}{5}$ and $\kappa > 2\lambda$:

1. $p(l = i|\tau_i) = \kappa$.
2. $p(l = j|\tau_i) = 2\lambda$, where the position j is next to the position i.
3. $p(l = k|\tau_i) = \lambda$, where the position k is at the diagonal position of the position i.

For example, for $i = 1$:

1. $p(l = 1|\tau_1) = \kappa$.
2. $p(l = 2|\tau_1) = 2\lambda; p(l = 3|\tau_1) = 2\lambda$.
3. $p(l = 4|\tau_1) = \lambda$.

B. Definition of the transition $p(r|l, \pi)$

We define the transition probability $p(r|l, \pi)$ with the following rules with the parameters μ and ν, where $\nu = 0.1 + \frac{2}{5}(\mu - 0.1)$ and $\mu > \nu > 0.1$:

1. $p(r = 1|l = i, \pi_i) = \mu$.
2. $p(r = 1|l = j, \pi_i) = \nu$, where the position j is next to the position i.
3. $p(r = 1|l = k, \pi_i) = 0.1$, where the position k is at the diagonal position of the position i.

For example, for $i = 1$:

1. $p(r = 1|l = 1, \pi_1) = \mu$
2. $p(r = 1|l = 2, \pi_1) = \nu$; $p(r = 1|l = 3, \pi_1) = \nu$
3. $p(r = 1|l = 4, \pi_1) = 0.1$

Actually, κ represents the pitcher's control capability and μ represents the batter's batting capability. In this experiment, we let κ=0.8, μ=0.95.