L2E: LEARNING TO EXPLOIT YOUR OPPONENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Opponent modeling is essential to exploit sub-optimal opponents in strategic interactions. Most previous works focus on building *explicit* models to directly predict the opponents' styles or strategies, which require a large amount of data to train the model and lack adaptability to unknown opponents. In this work, we propose a novel Learning to Exploit (L2E) framework for *implicit* opponent modeling. L2E acquires the ability to exploit opponents by a few interactions with diverse opponents during training, thus can adapt to new opponents with unknown styles during testing quickly. We propose a novel opponent strategy generation algorithm that produces effective opponents for training automatically. We evaluate L2E on two poker games (*i.e.*, Leduc and Limit Texas Hold'em), a grid-world soccer environment, and a 3D simulated robot environment, which are very challenging benchmarks for opponent modeling. Comprehensive experimental results indicate that L2E quickly adapts to diverse styles of unknown opponents.

1 INTRODUCTION

One core research topic in modern artificial intelligence is creating agents that can interact effectively with their opponents in different scenarios. To achieve this goal, the agents should have the ability to reason about their opponents' behaviors, goals, and beliefs. Opponent modeling has been extensively studied in past decades (Albrecht & Stone, 2018) and many different approaches have been proposed. These methods have achieved great success in many practical applications, such as dialogue systems (Grosz & Sidner, 1986), intelligent tutor systems (McCalla et al., 2000), and security systems (Jarvis et al., 2005).

The existing opponent modeling algorithms vary greatly in their underlying assumptions and methodology. For example, policy reconstruction based methods (Powers & Shoham, 2005; Banerjee & Sen, 2007) explicitly fit an opponent model to reflect the opponent's observed behaviors. Type reasoning based methods (Dekel et al., 2004; Nachbar, 2005) reuse pre-learned models of several known opponents by finding the one that most resembles the current opponent's behavior. Classification based methods (Huynh et al., 2006; Sukthankar & Sycara, 2007) build models that predict the opponent's play style, and employ the counter-strategy, which is effective against that particular style. Some recent works combine opponent modeling with deep learning or reinforcement learning (He et al., 2016; Foerster et al., 2018; Wen et al., 2018). Although these algorithms have achieved some success, they also have two obvious disadvantages: 1) constructing accurate opponent models requires a lot of data, which is problematic since the agent may not have the time or opportunity to collect enough data about its opponent in most real applications; and 2) most of these algorithms perform well only when the opponents during testing are similar to the ones used for training, thus it is difficult for them to adapt to opponents with new styles.

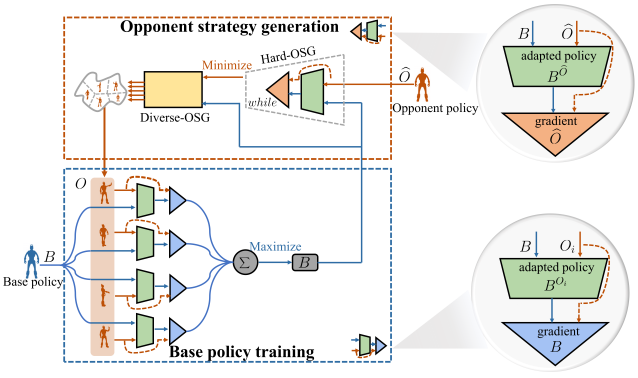


Figure 1: The overview of our proposed L2E framework.

To overcome these shortcomings, we propose a novel Learning to Exploit (L2E) framework for implicit opponent modeling, which has two desirable advantages. First, L2E does not build an explicit model for the opponent, so it does not require a large amount of interactive data and simultaneously eliminates the modeling errors. Second, L2E can quickly adapt to new opponents with unknown styles, with only a few interactions with them. L2E's key idea is to train a *base policy* to exploit various opponents of different styles by using only a few interactions between them during training, such that it acquires the ability to exploit different opponents quickly during testing. As shown in Fig. 1, the **base policy training part** (Section 3.1) maximizes the base policy's adaptability by continually interacting with *automatically generated* opponents from the opponent strategy generation (OSG) part. The **OSG part** first generates hard-to-exploit opponents for the current base policy (Hard-OSG, Section 3.2.1), then generates diverse opponent policies to improve the generalization ability of the base policy (Diverse-OSG, Section 3.2.2). The resulting base policy can fast adapt to completely new opponents with only a few interactions with them.

The main contributions of this paper are:

- We present a novel L2E framework to exploit sub-optimal opponents without explicitly building opponent models. It quickly adapts to unknown opponents using only a few interactions.
- We propose an adversarial training procedure to generate challenging opponents automatically. These hard-to-exploit opponents help L2E eliminate the weakness in its adaptability effectively.
- We design a diversity-regularized policy optimization procedure to automatically generate diverse opponents, improving L2E's generalization ability significantly.

We conduct extensive experiments to evaluate L2E in a diverse set of challenging environments. Comprehensive experimental results demonstrate that the base policy trained with L2E quickly exploits a wide range of opponents compared to other competitive baseline algorithms.

2 Related Work

Opponent modeling The main goal of opponent modeling is to interact more effectively with other agents by building models to reason about their intentions, predicting their next moves or other properties (Brown, 1951; Albrecht & Stone, 2018). The commonly used opponent modeling methods can be roughly divided into the following four categories. Policy reconstruction methods (Mealing & Shapiro, 2015) reconstruct the opponents' decision-making process by building models that make explicit predictions about their actions. Classification methods (Weber & Mateas, 2009; Synnaeve & Bessiere, 2011) produce models that assign class labels to the opponent and employ a precomputed strategy that is effective against that particular class of opponent. Type-based reasoning methods (He et al., 2016; Albrecht & Stone, 2017) assume that the opponent has one of several known types and update the belief using the new observations obtained during the real-time interactions. Recursive reasoning methods (Muise et al., 2015; de Weerd et al., 2017) model the nested beliefs and simulate the opponents' reasoning processes to predict their actions. Unlike these existing methods, L2E does not explicitly model the opponent and acquires the ability to exploit different opponents by training with limited interactions with different styles of opponents.

Meta-Learning Meta-learning is a new trend of research in the machine learning community that tackles learning to learn (Hospedales et al., 2020). It leverages experiences in the training phase to learn how to learn, acquiring the ability to generalize to new environments or new tasks. Recent progress in meta-learning has achieved impressive results ranging from classification and regression in supervised learning (Finn et al., 2017; Nichol et al., 2018) to new task adaption in reinforcement learning (Wang et al., 2016; Xu et al., 2018). Some recent works have also initially explored the application of meta-learning in opponent modeling. For example, the theory of mind network (ToMnet) (Rabinowitz et al., 2018) uses meta-learning to improve the predictions about the opponents' future behaviors. Another related work (Al-Shedivat et al., 2018) uses meta-learning to handle the non-stationarity problem in multi-agent interactions. Unlike these methods, we focus on improving the agents' ability to quickly adapt to different unknown opponents. L2E can be seen as a particular case of meta-learning. The meta-learning algorithm, such as MAML (Finn et al., 2017), is initially designed for single-agent environments. It requires the manual design of training tasks, and the final performance largely depends on the user-specified training task distribution. The L2E

framework is designed explicitly for multi-agent competitive environments, which automatically generates effective training tasks (opponents).

Strategy Generation The automatic generation of effective opponents for training is a critical step in L2E. How to generate diverse strategies has been preliminarily studied in the reinforcement learning community. In specific, diverse strategies can be obtained in various ways, including adding some diversity regularization to the optimization objective (Abdullah et al., 2019), randomly searching in some diverse parameter space (Plappert et al., 2018; Fortunato et al., 2018), using information-based strategy proposal (Eysenbach et al., 2018; Gupta et al., 2018), and searching diverse strategies with evolutionary algorithms (Agapitos et al., 2008; Wang et al., 2019; Jaderberg et al., 2017; 2019). More recently, researchers from DeepMind propose a league training paradigm to obtain a Grandmaster level StarCraft II AI (*i.e.*, AlphaStar) by training a diverse league of continually adapting strategies and counter-strategies (Vinyals et al., 2019). Unlike these methods, L2E uses adversarial training and diversity-regularized policy optimization to produce challenging and diverse opponents automatically.

3 METHOD

We propose a novel L2E framework to endow the agents with the ability to adapt to its opponents quickly. As shown in Fig. 1, L2E mainly consists of two modules, *i.e.*, the base policy training part and the opponent strategy generation part. The opponent strategy generation part provides the base policy training part with challenging and diverse training opponents automatically. Next, we introduce these two modules in detail.

3.1 BASE POLICY TRAINING

Our goal is to find a base policy B that can fast adapt to an unknown opponent O by updating B's parameters using only a few interactions between B and O. The key idea is to train B against many opponents to maximize its payoffs by using only a small amount of interactive data, such that it acquires the ability to exploit different opponents quickly. In effect, L2E treats each opponent as a training example. After training, the resulting base policy B can quickly adapt to new opponents using only a few interactions.

Algorithm 1 L2E's base policy training part.

Input: Step size hyper parameters α, β;
base policy B with parameters θ;
opponent policy O with parameters ϕ.
Output: An adaptive base policy B.
Randomly initialize θ, ϕ;
Initialize policy pool $\mathcal{M} = \{O\}$;
for $1 \leq e \leq epochs$ **do**
  O = **Hard-OSG**(B); ▷ 3.2.1
  $\mathcal{P}$ =**Diverse-OSG**(B, O, N); ▷ 3.2.2
  Update opponent pool $\mathcal{M} = \mathcal{M} \cup \mathcal{P}$;
  Sample a batch of opponents $O_i \sim \mathcal{M}$;
  for Each opponent O_i **do**
    Construct a single-player MDP $M_B^{O_i}$;
    Use Eq. (2) to obtain B^{O_i};
  end for
  Update B's parameters θ according to Eq. (5).
end for

Without loss of generality, the base policy B is modeled by a deep neural network, *i.e.*, a parameterized function π_θ with parameters θ. Similarly, the opponent O for training is also a deep neural network π_ϕ with parameters ϕ. We model the base policy as playing against an opponent in a two-player Markov game (Shapley, 1953). This Markov game $M = (S, (A_B, A_O), T, (R_B, R_O))$ consists of the state space S, the action space A_B and A_O, and a state transition function $T : S \times A_B \times A_O \to \Delta(S)$ where $\Delta(S)$ is a probability distribution on S. The reward function $R_i : S \times A_B \times A_O \times S \to \mathbb{R}$ for each player $i \in \{B, O\}$ depends on the current state, the next state and both players' actions. Given a training opponent O whose policy is known and fixed, this two-player Markov game M reduces to a single-player Markov Decision Process (MDP), *i.e.*, $M_B^O = (S, A_B, T_B^O, R_B^O)$. The state and action space of M_B^O are the same as in M. The transition and reward functions have the opponent policy embedded:

$$T_B^O(s, a_B) = T(s, a_B, a_O), \quad R_B^O(s, a_B, s') = R_B(s, a_B, a_O, s'), \tag{1}$$

where the opponent's action is sampled from its policy $a_O \sim \pi_\phi(\cdot \mid s)$. Throughout the paper, M_X^Y represents a single-player MDP, which is reduced from a two-player Markov game (*i.e.*, player X and player Y). In this MDP, the player Y is fixed and can be regarded as part of the environment.

Suppose a set of training opponents $\{O_i\}_{i=1}^N$ is given. For each training opponent O_i, an MDP $M_B^{O_i}$ can be constructed as described above. The base policy B, *i.e.*, π_θ is allowed to query a limited number of sample trajectories τ to adapt to O_i. In our method, the adapted parameters θ^{O_i} of the base policy are computed using one or more gradient descent updates with the sample trajectories τ. For example, when using one gradient update:

$$\theta^{O_i} = \theta - \alpha \nabla_\theta \mathcal{L}_B^{O_i}(\pi_\theta), \tag{2}$$

$$\mathcal{L}_B^{O_i}(\pi_\theta) = -\mathbb{E}_{\tau \sim M_B^{O_i}}[\sum\nolimits_t \gamma^t R_B^{O_i}(s^{(t)}, a_B^{(t)}, s^{(t+1)})]. \tag{3}$$

$\tau \sim M_B^{O_i}$ represents that the trajectory $\tau = \{s^{(1)}, a_B^{(1)}, s^{(2)}, \ldots, s^{(t)}, a_B^{(t)}, s^{(t+1)}, \ldots\}$ is sampled from the MDP $M_B^{O_i}$, where $s^{(t+1)} \sim T_B^{O_i}(s^{(t)}, a_B^{(t)})$ and $a_B^{(t)} \sim \pi_\theta(\cdot \mid s^{(t)})$.

We use B^{O_i} to denote the updated base policy, *i.e.*, $\pi_{\theta^{O_i}}$. B^{O_i} can be seen as an *opponent-specific* policy, which is updated from the base policy through fast adaptation. Our goal is to find a *generalizable* base policy whose opponent-specific policy B^{O_i} can exploit its opponent O_i as much as possible. To this end, we optimize the parameters θ of the base policy to maximize the rewards that B^{O_i} gets when interacting with O_i. More concretely, the *learning to exploit* objective function is:

$$\min_\theta \sum\nolimits_{i=1}^N \mathcal{L}_{B^{O_i}}^{O_i}(\pi_{\theta^{O_i}}) = \min_\theta \sum\nolimits_{i=1}^N \mathcal{L}_{B^{O_i}}^{O_i}(\pi_{\theta - \alpha \nabla_\theta \mathcal{L}_B^{O_i}(\pi_\theta)}). \tag{4}$$

It is worth noting that the optimization is performed over the base policy's parameters θ, whereas the objective is computed using the adapted based policy's parameters θ^{O_i}. The parameters θ of the base policy are updated as follows:

$$\theta = \theta - \beta \nabla_\theta \sum\nolimits_{i=1}^N \mathcal{L}_{B^{O_i}}^{O_i}(\pi_{\theta^{O_i}}). \tag{5}$$

In effect, our L2E framework aims to find a base policy that can significantly exploit the opponent with only a few interactions with it (*i.e.*, with a few gradient steps). The resulting base policy has learned how to adapt to different opponents and exploit them quickly. An overall description of the base policy training procedure is shown in Alg. 1 which consists of three main steps. First, generating hard-to-exploit opponents through the Hard-OSG module (Section 3.2.1). Second, generating diverse opponent policies through the Diverse-OSG module (Section 3.2.2). Third, training the base policy with these opponents to obtain fast adaptability. When facing a new opponent O_i after training, base policy π_θ is allowed to query a limited number of sample trajectories τ to adapt to O_i. The adapted parameters θ^{O_i} of the base policy are computed using one or more gradient descent updates with τ using Eqn. 2. The pseudo-code of the testing procedure is in Appendix B.3.

3.2 Automatic Opponent Generation

Previously, we assumed that the set of opponents had been given. How to automatically generate effective opponents for training is the key to the success of our L2E framework. The training opponents should be challenging enough (*i.e.*, hard to exploit). By learning to exploit these hard-to-exploit opponents, the base policy B can eliminate the weakness in its adaptability and become more robust. Besides, they should be sufficiently diverse. The more diverse they are, the stronger the generalization ability of the resulting base policy. We propose a novel opponent strategy generation (OSG) algorithm to achieve these goals.

3.2.1 Hard-to-Exploit Opponents Generation

We use the idea of adversarial learning to generate challenging training opponents for the base policy B. From the perspective of the base policy B, giving an opponent O, B first adjusts itself to obtain an adapted policy, *i.e.*, the opponent-specific policy B^O, the base policy is then optimized to *maximize* the rewards that B^O gets when interacting with O. Contrary to the base policy's goal, we want to find a hard-to-exploit opponent $\widehat{O}$ for the current base policy B, such that even if B adapts to $\widehat{O}$, the adapted policy $B^{\widehat{O}}$ cannot take advantage of $\widehat{O}$. In other words, the hard-to-exploit opponent $\widehat{O}$ is trained to *minimize* the rewards that $B^{\widehat{O}}$ gets when interacting with $\widehat{O}$. The base policy attempts to increase its adaptability by learning to exploit different opponents, while the hard-to-exploit opponent adversarially tries to minimize the base policy's adaptability, *i.e.*, maximize its *counter-adaptability*.

More concretely, the hard-to-exploit opponent $\widehat{O}$ is also a deep neural network $\pi_{\widehat{\phi}}$ with randomly initialized parameters $\widehat{\phi}$. At each training iteration, an MDP $M_B^{\widehat{O}}$ can be constructed. The base policy B first query a limited number of trajectories to adapt to $\widehat{O}$. The parameters $\theta^{\widehat{O}}$ of the adapted policy $B^{\widehat{O}}$ are computed using one gradient descent update,

$$\theta^{\widehat{O}} = \theta - \alpha \nabla_\theta \mathcal{L}_B^{\widehat{O}}(\pi_\theta). \tag{6}$$

$$\mathcal{L}_B^{\widehat{O}}(\pi_\theta) = -\mathbb{E}_{\tau \sim M_B^{\widehat{O}}}[\sum\nolimits_t \gamma^t R_B^{\widehat{O}}(s^{(t)}, a_B^{(t)}, s^{(t+1)})]. \tag{7}$$

$\widehat{O}$'s parameters $\widehat{\phi}$ are optimized to minimize the rewards that $B^{\widehat{O}}$ gets when interacting with $\widehat{O}$. This is equivalent to maximizing the rewards that $\widehat{O}$ gets since we consider the two-player zero-sum competitive setting in this work. More concretely, the parameters $\widehat{\phi}$ are updated as follows:

$$\widehat{\phi} = \widehat{\phi} - \alpha \nabla_{\widehat{\phi}} \mathcal{L}_{\widehat{O}}^{B^{\widehat{O}}}(\pi_{\widehat{\phi}}) \tag{8}$$

$$\mathcal{L}_{\widehat{O}}^{B^{\widehat{O}}}(\pi_{\widehat{\phi}}) = -\mathbb{E}_{\tau' \sim M_{\widehat{O}}^{B^{\widehat{O}}}}[\sum\nolimits_t \gamma^t R_{\widehat{O}}^{B^{\widehat{O}}}(s^{(t)}, a_{\widehat{O}}^{(t)}, s^{(t+1)})]. \tag{9}$$

After several rounds of iteration, we can obtain a hard-to-exploit opponent $\pi_{\widehat{\phi}}$ for the current base policy B. The pseudo code of this hard-to-exploit opponent generation algorithm is in Appendix B.1.

3.2.2 Diverse Opponents Generation

Training an effective base policy requires not only the hard-to-exploit opponents but also diverse opponents of different styles. From a human player's perspective, the opponent style is usually defined as different types, and the most significant difference between different types of opponents lies in the actions taken in the same state. Based on the above analysis, we formalize the difference between opponent policies as the difference between the distribution of trajectories induced by each policy when interacting with the base policy. We use the Maximum Mean Discrepancy (MMD) metric (Gretton et al., 2007) to measure the differences between trajectory distributions.

Definition 1 Let $\mathcal{F}$ be a function space $f : \mathcal{X} \to \mathbb{R}$. Suppose we have two distributions p and q, $X := \{x_1, ..., x_m\} \sim p$, $Y := \{y_1, ..., y_n\} \sim q$. Then MMD and its empirical estimate are defined as:

$$\mathrm{MMD}[\mathcal{F}, p, q] := \sup_{f \in \mathcal{F}} \left(\mathbf{E}_{x \sim p}[f(x)] - \mathbf{E}_{y \sim q}[f(y)] \right). \tag{10}$$

$$\mathrm{MMD}[\mathcal{F}, X, Y] := \sup_{f \in \mathcal{F}} \left(1/m \sum\nolimits_{i=1}^{m} f(x_i) - 1/n \sum\nolimits_{i=1}^{n} f(y_i) \right). \tag{11}$$

By picking a suitable function space $\mathcal{F}$, we get the following important theorem (Gretton et al., 2007).

Theorem 1 *Let $\mathcal{F} = \{f \mid \|f\|_{\mathcal{H}} \leq 1\}$ be a unit ball in Reproducing Kernel Hilbert Space $\mathcal{H}$, with associated kernel $k(\cdot, \cdot)$. Then $\mathrm{MMD}[\mathcal{F}, p, q] = 0$ if and only if $p = q$.*

Intuitively, we can expect $\mathrm{MMD}[\mathcal{F}, X, Y]$ to be small if $p = q$, and the quantity to be large if distributions are far apart. Thus, MMD is a suitable metric which can act as a regularization term to generate strategies with diverse styles. Formally, given a base policy B, *i.e.*, π_θ and an opponent policy O_i, *i.e.*, π_{ϕ_i}, our goal is to generate a new opponent O_j, *i.e.*, π_{ϕ_j} whose style is different from O_i. We first construct two MDPs, *i.e.*, $M_{O_i}^B$ and $M_{O_j}^B$, and then sample two sets of trajectories, *i.e.*, $\mathrm{T}_i = \{\tau \sim M_{O_i}^B\}$ and $\mathrm{T}_j = \{\upsilon \sim M_{O_j}^B\}$ from this two MDPs. The overall objective of our proposed diversity-regularized policy optimization algorithm is:

$$\mathcal{L}^{\phi_i}(\phi_j) = -\mathbb{E}_{\tau \sim M_{O_j}^B}[\sum\nolimits_t \gamma^t R_{O_j}^B(s^{(t)}, a_{O_j}^{(t)}, s^{(t+1)})] - \alpha_{\text{mmd}} \, \mathrm{MMD}^2[\mathcal{F}, \mathrm{T}_i, \mathrm{T}_j]. \tag{12}$$

The first term in Eq.(12) is to maximize the rewards that O_j gets when interacting with the base policy B. The second one measures the difference between O_j and the existing opponent O_i. By this

diversity-regularized policy optimization, the resulting opponent O_j is not only good in performance but also diverse relative to the existing policy.

Remark 1 *Computing the derivative of the MMD metric between trajectories with respect to the policy parameters is tractable.*

Based on Remark 1, the objective function in Eqn. 12 can be optimized via gradient descent. Appendix A details the gradient calculation of the MMD term, *i.e.*, $\mathrm{MMD}^2[\mathcal{F}, \mathrm{T}_i, \mathrm{T}_j]$ in Eqn. 12.

We can iteratively apply the above algorithm to find a set of N distinct and diverse opponents. In specific, subsequent opponents are learned by encouraging diversity with respect to previously generated opponent set S. The distance between an opponent O_m and an opponent set S is defined by the distance between O_m and O_n, where $O_n \in S$ is the most similar policy to O_m. Suppose we have obtained a set of opponents $S = \{O_m\}_{m=1}^{M}, M < N$. The M+1-th opponent, *i.e.*, $\pi_{\phi_{M+1}}$ can be obtained by optimizing:

$$\mathcal{L}^S(\phi_{M+1}) = -\alpha_{\text{mmd}} \min_{O_i \in S} \mathrm{MMD}^2[\mathcal{F}, \mathrm{T}_i, \mathrm{T}_{M+1}] - \mathbb{E}_{\tau \sim M^B_{O_{M+1}}}[\textstyle\sum_t \gamma^t R^B_{O_{M+1}}(s^{(t)}, a^{(t)}_{O_{M+1}}, s^{(t+1)})]. \tag{13}$$

By doing so, the resulting $M+1$-th opponent remains diverse relative to the opponent set S. Appendix B.2 provides the pseudo code of the diverse training opponent generation algorithm.

4 Experiments

In this section, we conduct extensive experiments to evaluate L2E. We first verify that the trained base policy using L2E quickly exploit a wide range of opponents with only a few gradient updates. Then, we compare with other state-of-the-art baseline methods to show L2E's superiority. Finally, we conduct a series of ablation experiments to demonstrate the effectiveness of L2E's key modules. Specifically, we evaluate L2E's performance on the Leduc poker (Southey et al., 2005) and a Grid Soccer environment (He et al., 2016), which are the commonly used benchmarks for opponent modeling. Moreover, we also validate L2E's generalization ability on more complex and challenging environments, *i.e.*, the Limit Texas Hold'em poker (Bowling et al., 2015) and a 3D RoboSumo[1] simulation game (Bansal et al., 2017). Details of each environment's state, action space representation, and reward design are described in Appendix B.2. We report most of the low-level details of the training and adaptation process in Appendix D.

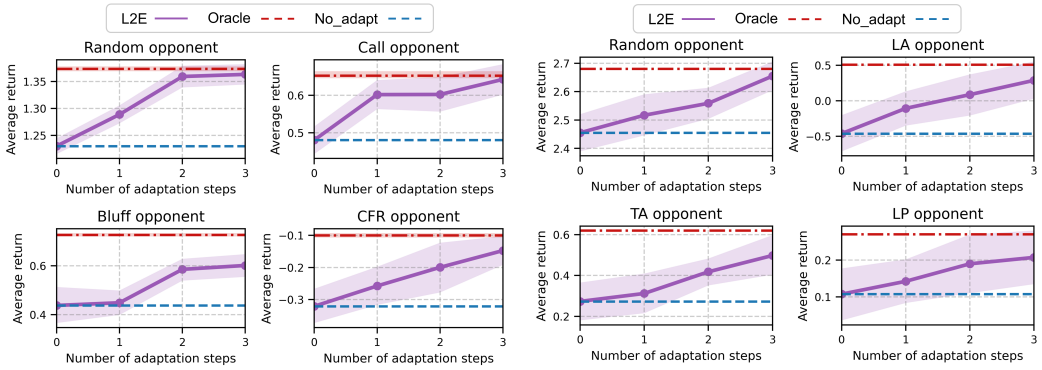
4.1 Rapid Adaptability

In this section, we first verify the trained base policy's ability to quickly adapt to different opponents in the Leduc poker environment. We provide a series of opponents with different styles and strengths. 1) The **Random** opponent randomly takes actions whose strategy is relatively weak and does not have an evident decision-making style. 2) The **Call** opponent makes decisions not based on its hand strength but always takes the call action. 3) The **Bluff** opponent takes actions usually based on its hand strength, but it also bluffs. This opponent is relatively strong and hard to exploit. 4) The **NFSP** (Heinrich & Silver, 2016) opponent is an approximate Nash equilibrium strategy generated by fictitious self-play with deep reinforcement learning. 5) The **CFR** (Zinkevich et al., 2008) opponent is also an approximate Nash equilibrium strategy generated iteratively by the CFR algorithm.

As shown in Fig. 2(a), L2E achieves a rapid increase in its performance with a few gradient updates against all opponents. The base policy can quickly approximate the best response strategies for the Call and the Random opponents with relatively weak strength or monotonous styles. For the Bluff and the CFR opponents that are difficult to exploit, L2E can also quickly improve its average returns.

Moreover, we use a larger and more challenging Limit Texas Hold'em (LHE) poker game to further verify L2E's effectiveness on large-scale problems. We provide three different types of high-performance opponents, *i.e.*, the **LooseAggressive** (LA) opponent, the **TightAggressive** (TA) opponent, and the **LoosePassive** (LP) opponent (Appendix C.1). These opponents are designed by some skilled Texas Hold'em player which can handle lots of different scenarios that are likely to

[1] https://github.com/openai/robosumo



(a) Leduc poker (b) Limit Texas Hold'em poker

Figure 2: L2E's base policy can quickly adapt to various opponents of different styles and strengths. The red dashed line represents the approximate best response for a given fixed opponent. The blue dashed line represents the base policy before adaptation. Shaded regions are 95% confidence intervals.

occur in the real play of LHE. Fig. 2(b) shows that L2E can also quickly exploit challenging opponents in large-scale complex problem like LHE. In the high-dimensional continuous RoboSumo environment, the base policy quickly adapt to the opponent's attack style and successfully force the opponent out of the arena[2]. In the grid soccer environment, L2E can also quickly adapt to opponents with aggressive and defensive styles (Appendix E.1).

4.2 Comparisons with Other Baseline Methods

As discussed in Section 1, most previous opponent modeling methods require constructing explicit opponent models from a large amount of data before learning to adapt to new opponents. To the best of our knowledge, L2E is the first attempt which learns to exploit opponents without building explicit opponent models. To demonstrate L2E's superiority, we design several competitive baseline methods. 1) **MAML**. The seminal meta-learning algorithm MAML (Finn et al., 2017) is designed for single-agent environments. We have redesigned and reimplemented the MAML algorithm for the two-player competitive environments. The MAML baseline trains a base policy by continually sampling the opponent's strategies, either manually specified or randomly generated. 2) **DRON**. The DRON baseline (He et al., 2016) combines opponent modeling with deep reinforcement learning to infer and exploit the opponents. 3) **EOM**. The Explicit Opponent Modeling (EOM) baseline (Albrecht & Stone, 2018) collects interaction data during the adaptation process to explicitly fit an opponent model M_O. The best response trained for M_O is used to interact with the opponent again to evaluate EOM's performance. We also add two additional Nash equilibrium baselines. 4) **CFR**. CFR (Zinkevich et al., 2008) is a popular equilibrium finding algorithm based on regret-minimization. 5) **NFSP**. NFSP (Heinrich & Silver, 2016) is a scalable end-to-end approach to learning approximate Nash equilibrium based on fictitious self-play and deep reinforcement learning. 6) **Oracle**. The Oracle represents each opponent's approximate best response which are obtained using the DQN (Mnih et al., 2015) algorithm trained separately with each fixed opponent. Oracle's result represents the upper bound of the performance.

As shown in Table 1(a), compared with MAML which is trained by using manually specified or randomly generated opponents, L2E achieves better performance in most cases thanks to its carefully designed opponent generation module. Different from the explicit opponent modeling baselines (*i.e.*, DRON and EOM) which require a lot of data to construct accurate opponent models to achieve good performance, L2E requires far less data (*i.e.*, using three gradient updates) to achieve comparable or better results. Meanwhile, L2E can exploit suboptimal opponents and obtain more payoffs compared to the equilibrium finding baselines, *i.e.*, CFR and NFSP. Moreover, L2E can quickly adjust its strategy to avoid being severely exploited by the approximate equilibrium opponents. Table 1(b) demonstrates that L2E also shows superiority in the more challenging LHE environment. Next,

[2]The corresponding video is available at `https://imgur.com/a/YbYgn0f`

Table 1: The average return of L2E and baseline methods when adapting different opponents. The $\pm$ shows the standard deviation across 3 random seeds, evaluated 10k hands each time.

(a) Leduc poker

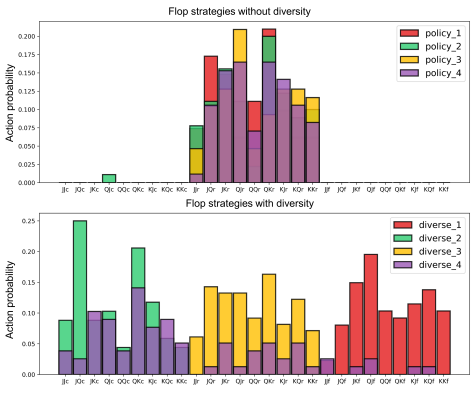
Method \ Opponent	Random	Call	Bluff	NFSP	CFR
L2E	1.362±0.063	**0.657±0.070**	**0.601±0.082**	**0.313±0.023**	-0.147±0.071
MAML	**1.372±0.028**	0.328±0.013	0.323±0.044	0.089±0.051	-0.409±0.010
DRON	1.323±0.014	0.418±0.011	0.409±0.052	0.212±0.080	-0.347±0.031
EOM	1.348±0.015	0.635±0.007	0.444±0.024	-0.012±0.023	-0.270±0.042
CFR	0.749±0.014	0.364±0.010	0.283±0.028	0.144±0.007	**0.010±0.024**
NFSP	0.780±0.019	0.132±0.024	0.029±0.022	0.011±0.027	-0.412±0.040
Oracle	1.373±0.007	0.662±0.014	0.727±0.012	0.338±0.041	-0.089±0.016

(b) Limit Texas Hold'em poker

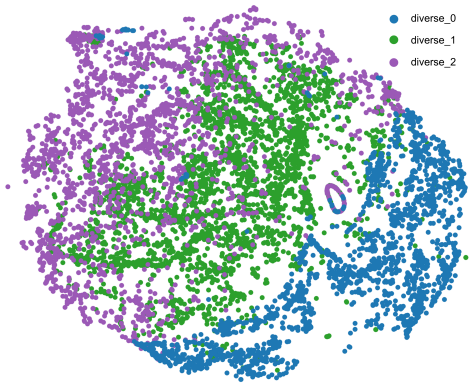
Method \ Opponent	Random	LA	TA	LP
L2E	**2.657±0.084**	**0.394±0.031**	**0.501±0.102**	**0.212±0.045**
MAML	2.633±0.035	0.037±0.047	0.231±0.057	0.089±0.051
DRON	2.131±0.672	-0.609±0.176	0.294±0.057	0.022±0.028
EOM	2.555±0.020	-0.014± 0.013	0.237±0.023	0.203±0.089
NFSP	1.342±0.033	-0.947± 0.012	-0.352±0.094	0.144±0.128
Oracle	2.682±0.033	0.513±0.009	0.624±0.011	0.270±0.026

we conduct some ablation experiments to analyze each component of L2E and its convergence properties.

4.3 Effects of the Diversity-regularized Policy Optimization



(a) Visualization of the styles of the strategies generated with or without the MMD regularization term in Leduc poker.



(b) t-SNE visualizes the policies generated by Diverse-OSG in RoboSumo, each point represents a state in a trajectory.

Figure 3: Visualization of policies generated by Diverse-OSG.

In this section, we verify whether our proposed diversity-regularized policy optimization algorithm can effectively generate policies with different styles. In Leduc poker, hand-action pairs represent different combinations of hands and actions. In the pre-flop phase, each player's hand has three possibilities, *i.e.*, J, Q, and K. Meanwhile, each player also has three optional actions, *i.e.*, Call (c), Rise (r), and Fold (f). For example, 'Jc' means to call when getting the jack. Action probability is the probability that a player will take a corresponding action with a particular hand. Fig. 3(a) shows that without the MMD regularization term, the generated strategies have similar styles. By optimizing with the MMD regularization term, the generated strategies are diverse enough which cover a wide range of different states and actions. In RoboSumo, considering its huge state and action spaces, it is no longer feasible to visualize the state-action probabilities. Instead, we use the t-SNE (Van der Maaten & Hinton, 2008) technique to visualize the strategies generated by Diverse-OSG. Fig. 3(b) visualizes the trajectories generated by three strategies interacting with the pre-trained opponent for 2000 steps in RoboSumo. It is clear that the generated opponents are also diverse in the larger RoboSumo environment. We further analysis the effect of the hyperparameter α_{mmd} in Eqn. 13 on

L2E's final performance at Appendix E.2. L2E can achieve good performance as long as α_{mmd} is not too large or too small.

4.4 Effects of the Hard-OSG and the Diverse-OSG

A crucial step in L2E is the automatic generation of training opponents. The Hard-OSG and Diverse-OSG modules are used to generate opponents that are difficult to exploit and diverse in styles. Fig. 4 shows the impact of each module on L2E's performance. 'L2E-H' is L2E without the Hard-OSG module, 'L2E-D' is L2E without the Diverse-OSG module, and 'L2E-HD' removes both modules altogether. The results show that both Hard-OSG and Diverse-OSG have a crucial influence on L2E's performance. It is clear that the Hard-OSG module helps to enhance the stability of the base policy, and the Diverse-OSG module can further improve the base policy's performance. Specifically, the performance of 'L2E-HD' is unstable, *e.g.*, its two-step adaptation performance is roughly the same as its one-step adaptation performance. With the addition of Hard-OSG, 'L2E-D' alleviates this problem. By further incorporating Diverse-OSG, L2E achieves the best performance.

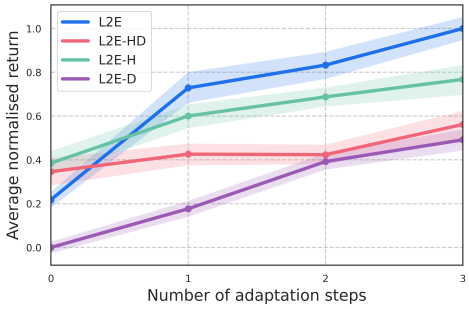


Figure 4: Each curve shows the average normalized returns of the base policy trained with different variants of L2E in the grid soccer environment. Shaded regions are 95% confidence intervals.

4.5 Convergence Analysis

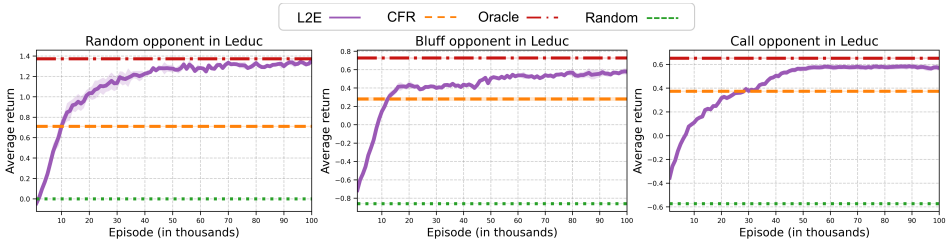


Figure 5: L2E's performance converges after a certain number of iterations in Leduc poker.

One may wonder whether L2E is supposed to converge at all. If L2E can converge, what is the relationship between the obtained base policy and the equilibrium concepts, such as Nash equilibrium? Since convergence is difficult to analyze theoretically, we have designed a series of experiments to empirically analyze L2E's convergence properties. From the experimental results in Fig. 5, it can be seen that as the training progresses, L2E's adaptability becomes stronger and stronger. After reaching a certain number of iterations, the improvement eventually reaches a plateau, which provides some empirical evidence for the convergence of L2E. Compared to the approximate Nash equilibrium policy obtained by the CFR method, the base policy can achieve higher average returns against different stationary opponents. Similar performance convergence curves can also be observed in LHE poker and RoboSumo game (Appendix E.3).

5 Conclusion

We propose a Learning to Exploit (L2E) framework to exploit sub-optimal opponents without building explicit opponent models. L2E acquires the ability to exploit opponents by a few interactions with different opponents during training to adapt to new opponents during testing quickly. We propose a novel opponent strategy generation algorithm that produces challenging and diverse training opponents for L2E automatically. Detailed experimental results in four challenging environments demonstrate the effectiveness of the proposed L2E framework.

6 REPRODUCIBILITY STATEMENT

We report all the low-level details that contribute to reproducibility for the training and adaptation process in Appendix D. The anonymous source code is available at `https://anonymous.4open.science/r/L2E-ACD0/`.

REFERENCES

Mohammed Amin Abdullah, Hang Ren, Haitham Bou Ammar, Vladimir Milenkovic, Rui Luo, Mingtian Zhang, and Jun Wang. Wasserstein robust reinforcement learning. *arXiv preprint arXiv:1907.13196*, 2019.

Alexandros Agapitos, Julian Togelius, Simon M Lucas, Jurgen Schmidhuber, and Andreas Konstantinidis. Generating diverse opponents with multiobjective evolution. In *IEEE Symposium On Computational Intelligence and Games*, pp. 135–142, 2008.

Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018.

Stefano V Albrecht and Peter Stone. Reasoning about hypothetical agent behaviours and their parameters. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 547–555, 2017.

Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

Dipyaman Banerjee and Sandip Sen. Reaching pareto-optimality in prisoner's dilemma using conditional joint action learning. *Autonomous Agents and Multi-Agent Systems*, 15(1):91–108, 2007.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*, 2017.

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

George W Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 13(1):374–376, 1951.

Harmen de Weerd, Rineke Verbrugge, and Bart Verheij. Negotiating with other minds: the role of recursive theory of mind in negotiation with incomplete information. *Autonomous Agents and Multi-Agent Systems*, 31(2):250–287, 2017.

Eddie Dekel, Drew Fudenberg, and David K Levine. Learning to play bayesian games. *Games and Economic Behavior*, 46(2):282–303, 2004.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135, 2017.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130, 2018.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, et al. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems*, pp. 513–520, 2007.

Barbara Grosz and Candace L Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 1986.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1804–1813, 2016.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

Trung Dong Huynh, Nicholas R Jennings, and Nigel R Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13(2):119–154, 2006.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Peter A Jarvis, Teresa F Lunt, and Karen L Myers. Identifying terrorist activity with ai plan recognition technology. *AI Magazine*, 26(3):73–73, 2005.

Gordon McCalla, Julita Vassileva, Jim Greer, and Susan Bull. Active learner modelling. In *International Conference on Intelligent Tutoring Systems*, pp. 53–62, 2000.

Richard Mealing and Jonathan L Shapiro. Opponent modeling by expectation–maximization and sequence prediction in simplified poker. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1):11–24, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Christian Muise, Vaishak Belle, Paolo Felli, Sheila McIlraith, Tim Miller, Adrian R Pearce, and Liz Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In *AAAI Conference on Artificial Intelligence*, pp. 3327–3334, 2015.

John H Nachbar. Beliefs in repeated games. *Econometrica*, 73(2):459–480, 2005.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018.

Rob Powers and Yoav Shoham. Learning against opponents with bounded memory. In *International Joint Conference on Artificial Intelligence*, pp. 817–822, 2005.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International Conference on Machine Learning*, pp. 4218–4227, 2018.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: opponent modelling in poker. In *Uncertainty in Artificial Intelligence*, pp. 550–558, 2005.

Gita Sukthankar and Katia Sycara. Policy recognition for multi-player tactical scenarios. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1–8, 2007.

Gabriel Synnaeve and Pierre Bessiere. A Bayesian model for opening prediction in rts games with application to StarCraft. In *IEEE Conference on Computational Intelligence and Games*, pp. 281–288, 2011.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

Ben G Weber and Michael Mateas. A data mining approach to strategy prediction. In *IEEE Symposium on Computational Intelligence and Games*, pp. 140–147, 2009.

Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2018.

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2396–2407, 2018.

Daochen Zha, Kwei-Herng Lai, Yuanpu Cao, Songyi Huang, Ruzhe Wei, Junyu Guo, and Xia Hu. Rlcard: A toolkit for reinforcement learning in card games. *arXiv preprint arXiv:1910.04376*, 2019.

Yan Zheng, Jianye Hao, Zongzhang Zhang, Zhaopeng Meng, Tianpei Yang, Yanran Li, and Changjie Fan. Efficient policy detecting and reusing for non-stationarity in markov games. *Autonomous Agents and Multi-Agent Systems*, 35(1):1–29, 2021.

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, pp. 1729–1736, 2008.

A GRADIENT CALCULATION OF MMD

Computing MMD is tractable when the function space $\mathcal{F}$ is a unit-ball in a reproducing kernel hilbert space defined by a kernel $k(\cdot,\cdot)$ and is given by:

$$\mathrm{MMD}^2[\mathcal{F},p,q]=\mathbb{E}_{x,x'\sim p}[k(x,x')]-2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)]+\mathbb{E}_{y,y'\sim q}[k(y,y')], \tag{14}$$

So, $\mathrm{MMD}^2[\mathcal{F},\mathrm{T}_i,\mathrm{T}_j]$ in Eqn. 12 can be calculated as:

$$\mathrm{MMD}^2[\mathcal{F},\mathrm{T}_i,\mathrm{T}_j]=\mathbb{E}_{\tau,\tau'\sim M^B_{O_i}}\left[k\left(\tau,\tau'\right)\right]-2\mathbb{E}_{\tau\sim M^B_{O_i},\upsilon\sim M^B_{O_j}}\left[k\left(\tau,\upsilon\right)\right]+\mathbb{E}_{\upsilon,\upsilon'\sim M^B_{O_j}}\left[k\left(\upsilon,\upsilon'\right)\right]. \tag{15}$$

Here, we use the Gaussian radial basis function kernel, *i.e.*, k is defined over a pair of trajectories and is calculated as:

$$k(\tau,\upsilon)=\exp(-\frac{\|g(\tau)-g(\upsilon)\|^2}{2h}). \tag{16}$$

In our experiments, we found that simply setting the bandwidth h to 1 produced satisfactory results. g stacks the states and actions of a trajectory into a vector. For trajectories with different length, we clip the trajectories when both of them are longer than the minimum length L. Usually, the trajectory's length does not exceed L, and we apply the masking based on the done signal from the environment to make them the same length.

The gradient of the MMD term with respect to the policy's parameter ϕ_j can be calculated as follows:

$$\begin{aligned}
\nabla_{\phi_j}\mathrm{MMD}^2(\mathrm{T}_i,\mathrm{T}_j)&=\nabla_{\phi_j}\mathbb{E}_{\tau,\tau'\sim M^B_{O_i}}\left[k\left(\tau,\tau'\right)\right] && (①)\\
&\quad-2\nabla_{\phi_j}\mathbb{E}_{\tau\sim M^B_{O_i},\upsilon\sim M^B_{O_j}}\left[k\left(\tau,\upsilon\right)\right] && (②)\\
&\quad+\nabla_{\phi_j}\mathbb{E}_{\upsilon,\upsilon'\sim M^B_{O_j}}\left[k\left(\upsilon,\upsilon'\right)\right] && (③)\\
&=\mathbb{E}_{\tau,\tau'\sim M^B_{O_i}}\left[k\left(\tau,\tau'\right)\nabla_{\phi_j}\log(p(\tau)p(\tau'))\right] && (\text{from } ①)\\
&\quad-2\mathbb{E}_{\tau\sim M^B_{O_i},\upsilon\sim M^B_{O_j}}\left[k\left(\tau,\upsilon\right)\nabla_{\phi_j}\log(p(\tau)p(\upsilon))\right] && (\text{from } ②)\\
&\quad+\mathbb{E}_{\upsilon,\upsilon'\sim M^B_{O_j}}\left[k\left(\upsilon,\upsilon'\right)\nabla_{\phi_j}\log(p(\upsilon)p(\upsilon'))\right]. && (\text{from } ③)
\end{aligned}$$

where $p(\cdot)$ is the probability of the trajectory. The second equation is obtained by the policy gradient theorem. Since $\mathrm{T}_i=\{\tau\sim M^B_{O_i}\}$, O_i is the known opponent policy that has no dependence on ϕ_j. The gradient with respect to the parameters ϕ_j in first term is 0. The gradient of the second and third terms can be easily calculated as follows:

$$\nabla_{\phi_j}\log(p(\upsilon))=\sum\nolimits_{t=0}^{T}\nabla_{\phi_j}\log\pi_{\phi_j}(a_t|s_t). \tag{17}$$

B ALGORITHM

B.1 HARD-OSG

Alg. 2 is the overall description of the hard-to-exploit training opponent generation algorithm.

$$\theta^{\widehat{O}}=\theta-\alpha\nabla_\theta\mathcal{L}^{\widehat{O}}_B(\pi_\theta). \tag{18}$$

$$\mathcal{L}^{\widehat{O}}_B(\pi_\theta)=-\mathbb{E}_{\tau\sim M^{\widehat{O}}_B}[\sum\nolimits_t\gamma^t R^{\widehat{O}}_B(s^{(t)},a^{(t)}_B,s^{(t+1)})]. \tag{19}$$

$$\widehat{\phi}=\widehat{\phi}-\alpha\nabla_{\widehat{\phi}}\mathcal{L}^{B^{\widehat{O}}}_{\widehat{O}}(\pi_{\widehat{\phi}}) \tag{20}$$

$$\mathcal{L}^{B^{\widehat{O}}}_{\widehat{O}}(\pi_{\widehat{\phi}})=-\mathbb{E}_{\tau'\sim M^{B^{\widehat{O}}}_{\widehat{O}}}[\sum\nolimits_t\gamma^t R^{B^{\widehat{O}}}_{\widehat{O}}(s^{(t)},a^{(t)}_{\widehat{O}},s^{(t+1)})]. \tag{21}$$

Algorithm 2 Hard-OSG, the hard-to-exploit training opponent generation algorithm.

Input: The latest base policy B with parameters θ.
Output: A hard-to-exploit opponent $\widehat{O}$ for B.
Randomly initialize $\widehat{O}$'s parameters $\widehat{\phi}$;
for $1 \leq i \leq epochs$ **do**
  Construct a single-player MDP $M_B^{\widehat{O}}$;
  Sample a small number of trajectories $\tau \sim M_B^{\widehat{O}}$ using B against $\widehat{O}$;
  Use Eq. 18 to update the parameters of B to obtain an adapted policy $B^{\widehat{O}}$;
  Sample trajectories $\tau' \sim M_{\widehat{O}}^{B^{\widehat{O}}}$ using $\widehat{O}$ against $B^{\widehat{O}}$;
  Update the parameters $\widehat{\phi}$ of $\widehat{O}$ according to Eq. 20.
end for

Algorithm 3 Diverse-OSG, the proposed diversity-regularized policy optimization algorithm to generate diverse training opponents.

Input: The latest base policy B, an existing opponent O_1, the total number of opponents that to be generated N.
Output: A set of diverse opponent $S = \{O_m\}_{m=1}^{N}$.
Initialize the opponent set $S = \{O_1\}$;
for $i = 2$ to N **do**
  Randomly initialize an opponent O_i's parameters ϕ_i;
  for $1 \leq t \leq steps$ **do**
    Calculate the objective function $\mathcal{L}^S(\phi_i)$ according to Eq. 22;
    Calculate the gradient $\nabla_{\phi_i}\mathcal{L}^S(\phi_i)$;
    Use this gradient to update ϕ_i;
  end for
  Update the opponent set S, *i.e.*, $S = S \cup O_i$.
end for

B.2 DIVERSE-OSG

Alg. 3 is the overall description of the diverse training opponent generation algorithm.

$$\mathcal{L}^S(\phi_{M+1}) = -\alpha_{\text{mmd}} \min_{O_i \in S} \text{MMD}^2[\mathcal{F}, \text{T}_i, \text{T}_{M+1}] - \mathbb{E}_{\tau \sim M_{O_{M+1}}^B}[\textstyle\sum_t \gamma^t R_{O_{M+1}}^B(s^{(t)}, a_{O_{M+1}}^{(t)}, s^{(t+1)})]. \tag{22}$$

B.3 L2E'S TESTING PROCEDURE

Alg. 4 is the testing procedure of our L2E framework.

$$\theta^{O_i} = \theta - \alpha \nabla_\theta \mathcal{L}_B^{O_i}(\pi_\theta), \tag{23}$$

Algorithm 4 The testing procedure of our L2E framework.

Input: Step size hyper-parameters α; The trained base policy B with parameters θ; An unknown opponent O.
Output: The updated base policy B^O that has been adapted to the opponent O.
Construct a single-player MDP M_B^O;
for $1 \leq step \leq steps$ **do**
  Sample trajectories τ from the MDP M_B^O;
  Update the parameters θ of B according to Eq. 23.
end for

B.4 LIMITATIONS

Although L2E's base policy can quickly adapt to the opponent with limited interaction data, its payoff during the adaptation process is not guaranteed. Therefore, how to guarantee the base policy's payoff during the adaptation process is an important further work. One possible solution is to use a Nash equilibrium strategy to collect data during the adaptation process, since Nash strategy is a safe strategy which guarantees not to lose in expectation in two-player zero-sum games. Meanwhile, the opponent's strategy may change at any time, and how to deal with this problem is also a very interesting research topic. One simple solution is to incorporate an off-the-shelf opponent strategy change detection module (*e.g.*, (Zheng et al., 2021)), when a change in the opponent's strategy is detected, the base policy then quickly re-adapt to it.

C ADDITIONAL DETAILS ON THE ENVIRONMENTS

C.1 POKER GAME

Leduc Poker The Leduc poker generally uses a deck of six cards that includes two suites, each with three ranks (Jack, Queen, and King of Spades, Jack, Queen, and King of Hearts). The game has a total of two rounds. Each player is dealt with a private card in the first round, with the opponent's deck information hidden. In the second round, another card is dealt with as a public card, and the information about this card is open to both players. If a player's private card is paired with the public card, that player wins the game; otherwise, the player with the highest private card wins the game. Both players bet one chip into the pot before the cards are dealt. The following two betting rounds, with a maximum of two raises per round, whose values are 2 and 4 chips, respectively.

Limit Texas Hold'em Poker We use a larger and more challenging Limit Texas Hold'em (LHE) poker to further verify the effectiveness of our L2E framework. LHE is a poker game of real-world scale with a card deck of 52 cards. LHE has a total of four rounds. Each player is dealt with two private cards in the first round, and then three public cards are dealt in three stages (the flop, the turn, and the river). The raise amount in LHE is fixed to the big blind for the first two rounds and doubled for the second two rounds, and the maximum number of raises per round is 4. There are four actions in Leduc and LHE: call, check, raise and fold.

Environment Representation The states in Leduc and LHE are encoded in the same way as (Zha et al., 2019). The states in Leduc are encoded as a vector of length 36, with the first six dimensions corresponding to the private and public cards, respectively. The final 30 dimensions correspond to the number of chips of both players. Similarly, an information state of LHE can be encoded as a vector of length 72. The first 52 dimensions represent cards, and the last 20 dimensions correspond to the betting history. The reward is measured in big-blinds won per hand, i.e., bb/h. To provide some intuition for win rates in LHE, a player that always folds will lose -0.7 bb/h.

We provide three different types of high-performance opponents in the LHE poker based on the PokerStove[3]. We use the hand equities' cache matrix to calculate the winning probability, *i.e.*, the expected percentage of the time each hand wins at showdown. The LA opponent takes an aggressive strategy in a relatively large winning range. The TA opponent raises only in a relatively small winning range, while there is also bluffing. The LP opponent takes a defensive strategy in a relatively large winning range. The specific winning range boundaries and rules are designed by some skilled Texas Hold'em player.

C.2 GRID SOCCER

This game contains a board with a 6×9 grid, two players, and their respective target areas. The position of the target area is fixed, and the two players appear randomly in their respective areas at the start of the game. One of the two players randomly has the ball. The goal of all players is to move the ball to the other player's target position. When the two players move to the same grid, the player with the ball loses the ball. Players gain one point for moving the ball to the opponent's area.

[3]PokerStove is a highly hand optimized C++ poker hand evaluation library: `https://github.com/andrewprock/pokerstove`

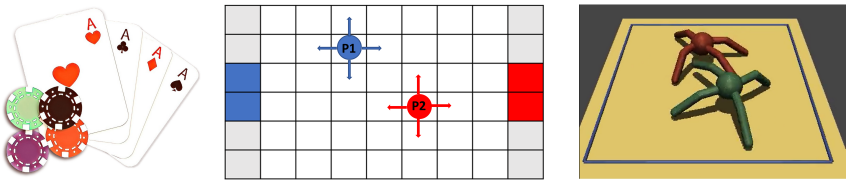


Figure 6: Illustrations of the two-player zero-sum games we use for evaluation. **Left**: Leduc Poker and Limit Texas Hold'em Poker. **Middle**: Grid Soccer. **Right**: RoboSumo Ants.

The player can move in all four directions within the grid, and action is invalid when it moves to the boundary.

We train the L2E algorithm in this soccer environment in which both players are modeled by a neural network. Inputs to the network include information about the position of both players, the position of the ball, and the boundary. We provide two types of opponents to test the effectiveness of the resulting base policy. 1) A defensive opponent who adopts a strategy of not leaving the target area and preventing opposing players from attacking. 2) An aggressive opponent who adopts a strategy of continually stealing the ball and approaching the target area with the ball. Facing a defensive opponent will not lose points, but the agent must learn to carry the ball and avoid the opponent moving to the target area to score points. Against an aggressive opponent, the agent must learn to defend at the target area to avoid losing points.

C.3 RoboSumo Ants

RoboSumo Ants is a high-dimensional continuous simulated robot environment (Bansal et al., 2017), where two Ants wrestle in an arena. The Ants compete against each other, and the side that forces its opponent out of the arena wins. If the time limit is reached, the game will end in a tie. Both agents observed the position, speed, and contact forces of the joints in their bodies, as well as the position of their opponent's joints. We initialize the base policy using a set of pre-trained policy weights published in the *agent zoo* (Bansal et al., 2017), and the rest of the sets are used as adapted opponents of the base policy after training.

D Implementation Details

D.1 Network Architectures and Hyperparameters

Following the MAML paper (Finn et al., 2017), we use TRPO (Schulman et al., 2015a) with Generalized Advantage Estimation (Schulman et al., 2015b) as the meta-optimizer and used a 2-layer fully connected network with ReLU nonlinearities for function approximation. The experiments in LHE poker and RoboSumo involve more complex neural network architectures and parameter designs than Leduc poker and Grid Soccer. For RoboSumo game, we utilized PPO (Schulman et al., 2017) as the meta-optimizer, and used a 2-layer fully connected network of size 128. All experiments are conducted on a machine with NVIDIA TITAN Xp GPU and 20 CPU cores (Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz). For the selection of hyperparameters, we first search randomly in a coarse range and then do a grid search in a smaller hyperparameter space. The other detailed hyperparameters are listed in Table 2. *Opponents-batch-size* refers to "number of opponents sampled per batch". *Trajectories-batch-size* refers to "number of trajectories to sample for each opponent".

D.2 Baseline Implementation Details

Table 3 shows the details of the critical parameters of the baseline methods. As discussed in Section 4.2, we define each individual static opponent policy as a task and fuse the opponent and environment into a single MDP for MAML training. Both DRON and Oracle use DQN as the basic reinforcement learning algorithm. Following the original paper, DRON uses the DRON-Concat network architecture for extracting and fusing opponent features. For a fair comparison, the parameters of MAML were set the same as L2E. And the baselines that require online adaptation, such as MAML and EOM, use the same amount of sampled data as L2E shown in Table 2.

Table 2: Hyperparameters of L2E.

Module	Parameter	Leduc	Soccer	LHE	RoboSumo
	Input Dimension	36	15	72	120
	Output Dimension	4	5	4	8
	Policy network size	[64,64]	[64,64]	[128,128]	[128,128]
Training	Training step size α	0.1	0.1	0.05	0.001
	Training step size β	0.01	0.01	0.01	5e-4
	Discount factor γ	0.99	0.99	0.995	0.995
	GAE λ	1.0	1.0	0.99	0.98
	Opponents-batch-size	20	20	40	40
	Trajectories-batch-size	20	20	200	200
	Gradient steps in training	1	1	1	1
Testing	Testing step size γ	0.1	0.1	0.05	0.001
	Gradient steps in testing	3	3	3	3
	Trajectories-batch-size in testing	10	10	20	20
OSG	Hard-OSG training epochs	5	5	20	20
	Hard-OSG learning rate	1e-2	1e-2	1e-3	1e-3
	Sample size	20	20	100	100
	Diverse-OSG training epochs	10	10	20	20
	Weight of the MMD term α_{mmd}	1.0	1.0	1.2	1.2
	Bandwidth of RBF kernel	1	1	1	1

Table 3: Hyperparameters of Baseline.

Baseline	Parameter	Leduc	LHE
NFSP	RL learning rate	0.1	0.01
	SL learning rate	5e-3	1e-3
	Optimizer	Adam	Adam
	Reservoir buffer size	30k	200k
	Anticipatory parameter	0.1	0.1
Oracle&DRON	RL learning rate	5e-5	1e-5
	Buffer size	30k	200k
	Optimizer	Adam	Adam
	Architecture in DRON	DRON-Concat	DRON-Concat
MAML	Step size α	0.1	0.05
	Step size β	0.01	0.01
	Adapt learning rate	0.1	0.05
	Meta batch size	20	40
	Trajectory batch size	20	200

E ADDITIONAL RESULTS

E.1 RAPID ADAPTABILITY

Fig. 7 shows the comparisons between L2E, MAML, and TRPO. L2E adapts quickly to both types of opponents; TRPO works well against defensive opponents but loses many points against aggressive opponents; MAML is unstable due to its reliance on task specification during the training process.

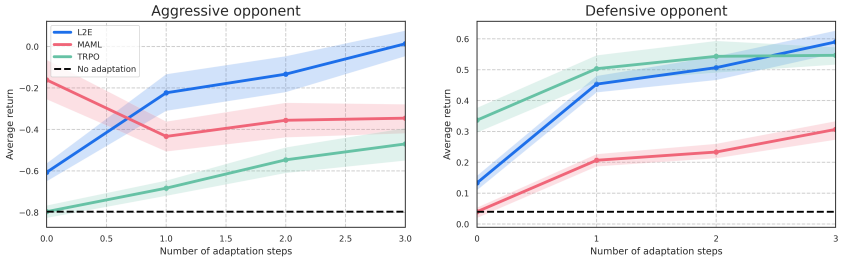


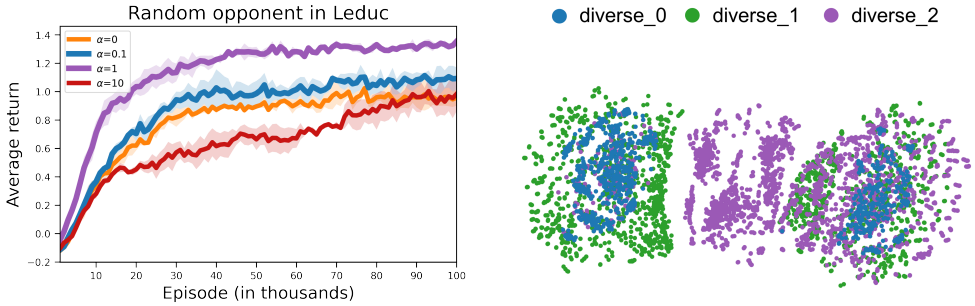
Figure 7: The trained base policy using L2E can quickly adapt to opponents with different styles in the Grid Soccer environment.

The average win rates of L2E against different opponents in RoboSumo before and after training are given in Table 4. As discussed in Appendix C.3, L2E uses the pre-training weights of *AgentZoo1* to initialize the base policy, and the rest of the weight sets are used as adapted opponents for the base policy after training.

Table 4: Average win-rates (95% CI) of 500-rounds between L2E and different opponents in RoboSumo Ants.

Method \ Opponent	AgentZoo1	AgentZoo2	AgentZoo3
Initial	53.3±3.6%	42.7±5.9%	45.7±9.0%
L2E	86.4±3.1%	69.2±4.6%	73.6±6.8%

E.2 EFFECTS OF THE HYPERPARAMETER α_{MMD}



(a) Effects of the hyperparameter α_{mmd} on L2E's final performance in Leduc poker.

(b) t-SNE (Van der Maaten & Hinton, 2008) visualizes the policies generated by Diverse-OSG in LHE poker, each point represents a state in a trajectory.

Figure 8: The effects of the hyperparameter α_{mmd}.

The hyperparameter α_{mmd} in Eq.(12) controls the generated policies' diversity, with larger α_{mmd} corresponding to policies with more diverse styles. Fig. 8(a) demonstrates that α_{mmd} has a significant impact on L2E's final performance. When α_{mmd} is in a reasonable range, L2E's performance is improved as α_{mmd} increases. When α_{mmd} is too large, it will adversely affect L2E's performance.

In our experiments, we found that L2E performs best when α_{mmd} is around 1. Fig. 8(b) visualize the trajectories generated by three strategies interacting with the random opponent for 1000 steps, respectively.

E.3 L2E'S PERFORMANCE CONVERGENCE CURVES

Fig. 9 shows the full results of L2E's performance convergence curves.

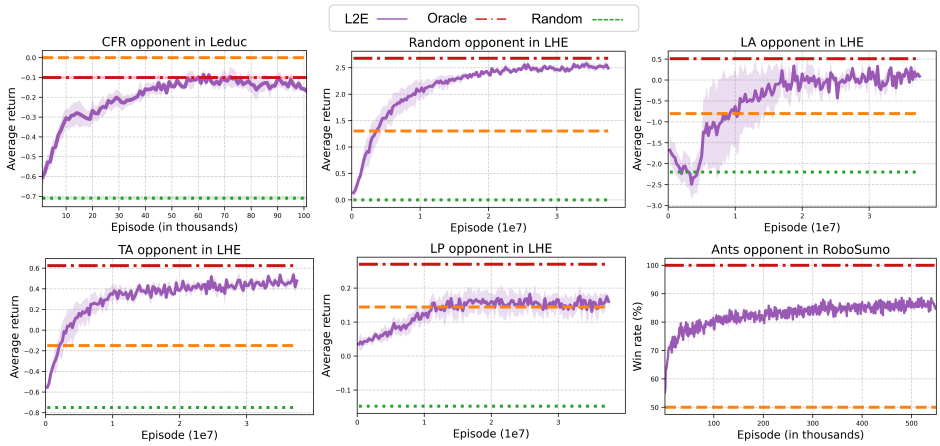


Figure 9: The solid line shows L2E's adaptability in all environments.