Modified Updates for Mirror Descent based methods in two-player zero-sum games.

by

Thiruvenkadam Sivaprakasam Radhakrishnan

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master's in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2023

Chicago, Illinois

Defense Committee:
Prof. Ian Kash, Chair and Advisor
Prof. Anastasios Sidiropoulos
Prof. Ugo Buy

ACKNOWLEDGMENTS

The thesis has been completed. .. (INSERT YOUR TEXTS)

YOUR INITIAL

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF NOTATIONS

π	A policy.
θ	Parameters of a function approximator.

SUMMARY

Remove overlaps with introduction, and shorten a bit.

Multiagent Systems (MAS) [1] are frameworks of problems of distributed nature with independent actors called agents that cooperate or compete to achieve some outcome. Due to the complexity of such problems, designing efficient algorithms for them can be challenging. Machine learning and Reinforcement learning present opportunities for creating learning algorithms for agents in Multi-agent settings. Multiagent Reinforcement Learning (MARL) as a research area deals with designing efficient and performant RL algorithms for general multiagent problems. Two-player zero-sum games are instances of multiagent systems that are purely competitive in nature. Due to their unique structure, two-player zero-sum games can also be represented as saddle-point problems that provide certain analytical properties. This presents an opportunity to study and design optimization algorithms under the dynamics of two-player zero-sum games, with an aim to extend these algorithms to general multiagent settings. Two-player zero-sum games also model other learning problems [2] and as such these advances can also be equivalently applied or adapted to solve them. Mirror Descent is a popular first-order optimization algorithm that has wide applications. In this work, we study mirror-descent-based reinforcement learning algorithms in the context of two-player zero-sum games. We evaluate the performance of these algorithms in benchmark extensive-form games under function approximation and summarize our findings regarding the effectiveness of mirror-descent-based reinforcement learning algorithms in the multiagent setting and the effect of the modifications

we apply. Through this study, we provide some recommendations for designing MARL algorithms and close with some remarks about future research directions.

CHAPTER 1

INTRODUCTION

Multi-agent Systems (MAS) [1] are frameworks of problems of distributed nature with independent actors called agents that cooperate or compete to achieve some outcome. Due to the complexity of such problems, designing efficient algorithms for them can be challenging. Machine learning and Reinforcement learning present opportunities for creating learning algorithms for agents in Multi-agent settings. Multi-agent Reinforcement Learning (MARL) as a research area deals with designing efficient and performant RL algorithms for general multi-agent problems. Due to the inherent issues in multi-agent settings such as the violation of the non-stationarity assumption that underlies most RL algorithms, their direct adaptation to the multi-agent setting can be limiting. Also, the multi-agent setting presents new complications in the form of large state and action spaces, high-dimensional problem representations, and credit assignment to name a few. There have been continued efforts to apply popular policy gradient and value-based single-agent RL methods to multi-agent setting through novel adaptions that account for some of the above limitations. On the other hand, various works have adopted ideas from areas like online learning, game theory, and evolutionary biology to design novel RL algorithms that circumvent the above limitations by providing new theoretical guarantees, and displaying strong empirical performances. Of these, a vast majority of methods rely on regret minimization and self-play learning where theoretical convergence guarantees necessitate maintaining an average of past strategies of the agents. While this is easier in tabular settings,

maintaining such averages of past policies can become burdening for large-scale problems especially when utilizing function approximation methods like neural networks to represent the policy parameters. Hence, there is a strong incentive in designing algorithms that have convergence guarantees for the last-iterate policy as opposed to having to maintain an average policy.

In terms of the problem setting that concerns our work, we focus on the game-theoretic construct of two-player zero-sum games (2p0s). There are various reasons for the interesting nature of two-player zero-sum games. Firstly, within various game-theoretic problem formulations, these games present a simple enough structure while still retaining the multi-agent setting. This presents an opportunity to study and design algorithms under the dynamics of two-player zero-sum games, with an aim to extend these algorithms to general multi-agent settings. Second, two-player zero-sum games are saddle-point problems, and provide an opportunity to connect with optimization algorithms that have been studied extensively for solving saddle-point problems. Finally, two-player zero-sum games also model other learning problems such as generative adversarial networks (GANs) [2] and as such any improvements in methods for the games can provide insights into designing algorithms for applications like GANs.

Our work primarily revolves around algorithms that were proposed or can be interpreted as extensions of mirror descent as a reinforcement learning update rule. Mirror Descent is a popular first-order optimization algorithm that has seen wide application in various problem settings such as online learning, convex optimization, optimal transport, etc. In this work, we study adaptations of mirror descent to reinforcement learning algorithms in single, and

multi-agent settings. Specifically, we study Mirror Descent Policy Optimization (MDPO), and Magnetic Mirror Descent (MMD), two recent algorithms that extend mirror descent as RL algorithms, and approximate equilibrium solvers. While there have been many works studying mirror descent with the aim of designing RL algorithms, MDPO presents a practical deep RL algorithm and studies its similarities in structure, and connections to existing work in the literature. MDPO does not directly apply to two-player zero-sum games as it is a single-agent RL algorithm, and as such has no convergence guarantees. However, in our work, we study its performance in this setting and also propose some modifications to the update that MDPO uses to make it more amenable to the multi-agent setting. MMD, on the other hand, approaches a different problem of entropy regularized equilibrium (QREs) solving in two-player zero-sum games. Finding such equilibrium is cast as variational inequality problems and an algorithm from the variational inequality literature is applied to this problem setting under the name of MMD. MMD enjoys novel linear last-iterate convergence by taking advantage of problem-specific assumptions in finding QREs. Although there are no theoretical guarantees, MMD is also empirically studied as an algorithm for finding approximate Nash equilibria by annealing the strength of the regularization. MMD shows strong empirical performance as both tabular and deep reinforcement learning algorithms in self-play to converge to the Nash equilibrium. In this work, we use MMD as a baseline method and empirically study the effect of the same proposed modifications to understand how it improves MMD's performance. Given the already strong baseline, any improvements observed can guide further algorithmic design choices, and direct more studies to understand the effect of the proposed modification in this setting.

The modifications that we apply in this work are two-fold - one that modifies the loss being used by the algorithm, and the other that modifies the update structure of the algorithms, thereby affecting the problem's geometry in an effort to improve, or induce last-iterate convergence. The first modification is the Neural Replicator Dynamics (NeuRD), an extension of the Replicator dynamics framework from evolutionary game theory to function approximation and deep reinforcement learning. NeuRD can be interpreted as a modification of softmax policy gradients (SPGs) to handle the plateauing effect of the nonlinearity in adapting to the non-stationary rewards in multi-agent settings. Due to this structure, NeuRD can be incorporated into other algorithms that are extensions of SPGs. The second proposed modification is the incorporation of Extragradient and Optimistic updates to the above algorithms. The idea of using extra-gradients, or reusing past gradients is independent of the loss function being used, and is therefore extendable to the algorithms discussed in this work. However, there can be inherent limitations in applying these algorithms due to the theoretical assumptions behind them when applied to specific problem settings. We only study the effect of the modifications empirically and do not provide any theoretical convergence guarantees for the proposed updates. However, we take the potential theoretical limitations into consideration when discussing the results of the empirical evaluations.

We evaluate the proposed algorithms on the classic game-theoretic normal-form game of biased rock-paper-scissor to understand their empirical performance. We show that in the presence of certain modifications for last-iterate convergence policy gradient methods outperform the newer methods in this basic setting. Through this comprehensive study, we then present

these algorithms as extensions of policy gradient methods with various design choices. This presentation mainly aims at identifying the components of these extensions that contribute more towards the performance and potential components that detract from the performance. Based on our findings, we proceed to implement and evaluate a subset of the modifications as deep multi-agent reinforcement learning algorithms by training the players in self-play. We use standard extensive-form games like Kuhn poker and large-scale benchmarks like Abrupt Dark Hex to test the applicability and scalability of the modifications in two-player zero-sum extensive form games. We summarize our findings regarding the effectiveness of mirror-descent-based reinforcement learning algorithms in the multi-agent setting and the effect of the modifications we apply. Through this study, we provide some recommendations for designing MARL algorithms and close with some remarks about future research directions.

1.1 Outline

The rest of the thesis is organized as follows. We begin by providing some background and definitions in section 2 that are useful for the understanding of the algorithms and methods described in section 3 and 4. Section 3 introduces Mirror Decsent and expands on Mirror Descent based methods for solving Reinforcement learning problems. Section 4 discusses combining novel improvements on top of these methods and discusses their structure and the expected effects. In Section 5, we dive into some experimental results and discuss the performance of these algorithms in different settings. We then close the thesis with some discussion.

CHAPTER 2

BACKGROUND

In this section, we lay out some ideas, and concepts from Game Theory, Reinforcement Learning, Online learning, and Optimization that are relevant to this work. We only introduce fundamental notions within these areas that are useful in following the thesis, and we direct the reader to more comprehensive resources for an in-depth discussion wherever necessary. The framework of multi-agent learning has been conventionally rooted in Game Theory, and so we begin by discussing preliminary constructs from game theory and detail the setting that is pertinent to this work, namely, two-player zero-sum games. We then discuss foundational concepts within reinforcement learning that serves as a base for the algorithms discussed in this work. We also talk about the current limitations of Reinforcement learning methods in multi-agent settings and motivate the need for the development of better algorithms in this setting. Finally, we also present some preliminary concepts from online learning and optimization algorithms that are useful in understanding the approaches we study in this work.

2.1 Game Theory

Game theory [3] is the study of modeling interactions between decision-makers. Problems are represented as a game in which decision-makers are players taking actions that adhere to a set of rules in accordance with some real-world situation. Games intuitively capture the nature of the interaction, while abstracting away irrelevant details and complexity of real-world

problems that do not affect the outcome of interactions. Game theory is widely applied in modeling problems that are of economic, social, and political importance. In this section, we briefly introduce concepts from game theory that are relevant to this work. For an in-depth and more rigorous discussion of game theory and how it relates to problems in multi-agent learning, we refer the reader to [4, 5].

2.1.1 Problems Formulations in Game Theory

The primary components of modeling a problem using game theory are *actions* available to the players, and a specification of each player's preferences over outcomes. Players can choose from a set of available actions, and the availability of actions only depends on the situation, not on a player's preferences. Also, there is no restriction placed upon a player's preferences, and as such they can be risk-preferring, risk-averse, or even altruistic. It is also assumed that players follow *rational choice*, i.e., given the same information, an agent consistently chooses the same action irrespective of the qualitative nature of that action. The set of actions of all the players in a specific situation is jointly referred to as the *action profile*. Preferences are formalized through a *payoff function* (or utility function) that maps outcomes to scalar payoffs (or utilities). Payoff functions only induce an ordering over outcomes and are not reflective of the relative intensity or magnitude of such preferences.

A *strategy* describes how the player acts when given some information, and strategies can be deterministic (*pure strategies*) or distributions over pure strategies (*mixed strategies*). The expected payoff that can be achieved by a player following a given strategy is conditioned on the strategies of all the other players. While each player aims to learn strategies to maximize

their expected payoffs, the expected payoffs are themselves affected by the changing strategies, giving rise to the complexity in learning good strategies.

2.1.2 Game Representations

Two fundamental game-theoretic representations are the normal form and the extensive form games. In a normal-form (or a strategic) game, all players act simultaneously to reveal an outcome immediately after, and a payoff is assigned to each player. A more formal definition of a normal-form game is as follows:

Definition 1 (Normal-form games) *A n-player, general-sum normal form game is given by the* (N, A, u) *tuple, where:*

- $N = \{1 \dots n\}$ *is the set of players*
- $A = A_1 \times A_2 \dots \times A_n$*, with* A_i *being the set of actions available to player* i
- $u = u_i, \forall i \in N$ *is the set of payoff functions that map an action profile to a real payoff value for each agent,* $u_i : A \mapsto \mathbb{R}$*.*

For a normal-form game, the set of mixed strategies available to player i is given by $S_i = \Pi(A_i) \in \Delta^{k_i}$. The set of mixed-strategy profiles of all the players in the game is called the mixed strategy profile $S = S1 \times \dots \times S_n$. Normal-form games are typically represented as an n-dimensional matrix, where each dimension corresponds to the pure-strategy space of each player, and the entries correspond to the payoffs of action profiles. For this reason, normal-form games are also referred to as *matrix games*. Canonical examples of normal-form-games from

game theory literature include Matching pennies, Battle of the sexes, Prisoner's dilemma, and Rock-paper-scissors (RPS), etc.

Normal-form games do not explicitly model the temporal nature of decision-making that occurs in many real-world problems. As a result, while these problems can still be modeled as NFGs, this representation can be computationally expensive to work with. These problems are more naturally modeled as an Extensive-form game (EFG), a representation that explicitly incorporates the sequential nature of interactions in the form of a game tree.

Definition 2 (Extensive-form games) *An extensive form game is specified by the tuple* $G = (N, A, H, Z, \chi, \rho, \sigma, u)$, *where:*

- $N = \{1 \dots n\}$ *is the set of players*
- A *is the set of actions*
- H *is the set of non-terminal decision nodes*
- Z *is the set of terminal nodes;* $H \cap Z = \emptyset$
- $\chi : H \mapsto 2^A$ *is the action function that assigns each decision node a set of actions available to the player*
- $\rho : H \mapsto N$, *is the player function*
- $\sigma : H \times A \mapsto H \cup Z$ *is the successor function*
- $u = (u_i, \forall i \in N)$ *is the set of payoff functions that map terminal nodes to a real payoff value for each agent,* $u_i : Z \mapsto \mathbb{R}$.

The above representation definition assumes that all the relevant information for decision-making is available and accessible to all the players equally. This is known as the *perfect-information* setting, as opposed to the *imperfect-information* setting that occurs in many real-world problems where relevant information is obscured to the players.

Definition 3 (Imperfect-information Extensive-form games) *An imperfect-information extensive-form game,* $G = (N, A, H, Z, \chi, \rho, \sigma, u, I)$ *corresponds to a perfect-information extensive-form game* $G = (N, A, H, Z, \chi, \rho, \sigma, u)$*, where* $I = \{I_i, \forall i \in N\}$*, and* $I_i = (I_{i,1}, \ldots, I_{i,k_i})$ *defines a partition over all non terminal-histories* $\{h \in H : \rho(h) = i\}$*, such that* $\chi(h) = \chi(h')$*, and* $\rho(h) = \rho(h')$*, whenever* $\exists j$ *for which* $h, h' \in I_{i,j}$*.*

A crucial assumption in such a setting is a *perfect recall* of the sequence of actions taken by a player to reach the current node of the game tree. If all players possess perfect recall, then the game is said to be of perfect recall. While mixed strategies in extensive-form games are distributions over the pure strategies of the entire game tree, *behavioral strategies* are distributions over actions at each decision node. Normal-form representation of an extensive-form game is referred to as a reduced-NFG. Here the dimensions of the matrix represent pure strategies over the entire sequence of interactions, leading to an exponential blowup of the action space for the reduced-form NFGs. A more efficient *sequence-form* representation makes use of the varying depth of the game tree by defining the payoff matrix with respect to the subsets of histories instead of pure-strategy profiles. The payoffs achieved in the terminal histories are passed up the trees to non-terminal histories as well.

Beyond the above representations, many variants have been developed to add an account for specific characteristics of problems. More generally, definitions of game representations are typically based on the number of agents (Single-agent, Two-player, Multi-agent), the length of the game (single-shot, sequential), type of interaction (simultaneous-move, turn-based), utility values of players (general-sum, constant-sum, zero-sum), etc. In this work, we mainly deal with two-player zero-sum games (interchangeably referred to in the text as 2p0s), a purely competitive setting that only involves interactions between two agents whose payoff functions are a negation of each other.

Definition 4 (Two-player Zero-sum games) *A two-player normal-form game is zero-sum if for each strategy profile* $a \in A_1 \times A_2$, $u_1(a) + u_2(a) = 0$.

As motivated in the introduction for studying this problem setting, two-player zero-sum games are an important class of problems that coincide with other important domains including saddle point optimization, variational inequalities, and image generation in deep learning.

2.1.3 Equilibrium Concepts

Equilibrium concepts are points in the joint strategy space of the players that define a notion of optimality of a player's strategies. They allow us to reason about behaviors of decision makers in real-world problem settings, and analyze the worst case payoffs in potentially adversarial settings. Based on the definition of optimality, the problem setting, and assumptions made about player behaviors there exist various well-established equilibrium concepts in game theory literature.

Nash Equilibrium: Perhaps the most prevalent, and well-known concept is the Nash equilibrium which models perfectly rational decision makers as players and reasons about the best worst-case payoff one can achieve even in a completely adversarial scenario. The best response of player i to the strategy profile s_{-i} is a mixed strategy such that $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i}), \forall s_i \in S_i$.

Definition 5 (Nash equlibrium (NE)) *A strategy profile $s = (s_1, \ldots, s_n), s \in S$ in an n-player normal-form game is a Nash-equilibrium if s_i is a best response to s_{-i}, for all i.*

A major result in game theory ensures the existence of a Nash equilibrium for any finite n-person normal-form game [6], and extensive-form games with perfect recall [7]. While Nash equilibrium is the most prevalent solution concept, various other solution concepts with interesting properties have also been proposed and studied. Minmax and Maxmin strategies assure payoffs for players in the most aggressive (player i playing to minimize player j's payoffs) and most conservative (player $-i$ playing to minimize player i's payoffs) scenarios. The seminal work of von-Neumann showed that in two-player zero sum games Nash equilibrium strategies fetch payoffs that are equivalent to their minmax and maxmin values.

Quantal Response Equilibrium: While the solution concept of Nash equilibrium assumes perfect rationality, the Quantal response equilibrium (QRE) models agents with bounded rationality allowing for some errors in action predictions. So instead of modeling agent actions as best-response functions, QRE instead models player actions according to quantile response functions of their expected payoffs. A quantal response function $\sigma : \mathbb{R}^n \mapsto \Delta^n$ is nonempty in the interior of the simplex, continuous on $\mathbb{R}^n$, and monotonically increasing. QRE places all

action choices within the interior of the probability simplex thereby allowing the possibility of any action occurring within observed data. QRE was first proposed for normal-form games [8], and then later extended to extensive form games using the sequence form representation [9].

Definition 6 (Quantal-response Equilibrium (QRE)) *For a given normal form game with payoff functions $u = (u_1, \ldots, u_n)$, a strategy profile $S \in \Delta^{k_i}$ is a quantal response equilibrium (QRE) if*

$$S_i = \sigma_{ij}(u_i(S)), for 1 \leq j \leq k$$

An instance of QRE is the logit-QRE (which we simply refer to as QRE going forward) where the logistic quantal response function is used to model the player's action choices.

$$\pi_{ij} = \frac{e^{\lambda u_{ij}(\pi)}}{\sum_{j'=1}^{k} e^{\lambda u_{ij'}(\pi)}}$$

For extensive-form games, if the behavioral strategies of all players are QREs, then the joint policy is said to be an Agent QRE (AQRE). QRE predicts systematic deviations from Nash equilibrium when fit with experimental data of game-theoretic problems. There exist many other prominent equilibrium concepts including the coarse-correlated equilibrium, the sub-game perfect equilibrium, and we refer the reader to [3, 5] for a more detailed discussion on these.

2.2 Reinforcement Learning

Connect RL with 2.1; can define various settings to be instances of GT problems like stochastic games.

Reinforcement learning (RL) [10] deals with designing algorithms for *agents* that interactively learn to maximize a *reward* signal in an *environment*. The reward signal encodes information about the goal that the agent has to learn to achieve without any specific directions about how that goal should be achieved.

Markov Decision Process. Markov Decision Processes (MDPs) provide a framework to formalize reinforcement learning problems. A γ-discounted MDP ($\mathcal{M}$) is given by the tuple, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where: $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s', r|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition dynamics function, $R(s, a) \subset \mathbb{R}$ is the reward function, $\gamma \in [0, 1]$ is the discount factor. In this work, we discuss finite MDPs where the sequence of interactions is characterized as episodes that have a terminal state. The expected return, G_t, at time step t is the discounted sum of rewards accumulated by the agent starting from t till the end of the episode. Formally, $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$, where the discount rate γ indicates the importance placed upon future rewards over immediate rewards. The objective of an agent is to learn to act in a way that maximizes its *expected return*. A *Policy* is a mapping from a state to an action distribution $\pi : \mathcal{S} \mapsto \mathcal{A}$. A *Value function*, $V_\pi(s)$ estimates the expected reward that can be achieved from the current state under the policy π: $V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$. Analogously, the *Q-value function* estimates the expected reward of taking a specific action a at a given state s and then following the policy π: $Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$. The difference between Q and V functions is referred to as the advantage function $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$. This is the advantage of taking a particular action over following the average policy.

Value functions impose a ranking over policies in terms of the expected return achieved by an agent following a given policy. An *optimal policy* (π^*) is one that is at least as better as any other policy ($\pi \in \Pi$) in terms of this ranking, i.e. $V_{\pi^*}(s) \geq V_\pi(s), \forall s \in \mathcal{S}$. While there may be more than one optimal policy, they all share the same unique *optimal value function*: $V_{\pi^*} = \max_\pi V_\pi(s), \forall s \in \mathcal{S}$.

2.2.1 Tabular methods

For a given policy, the value function satisfies the **Bellman equation**:

$$V_\pi(s) \doteq \mathbb{E}_\pi \left[r + \gamma V_\pi(s') | R_t = r, S_{t+1} = s' \right] \tag{2.1}$$

For small state spaces, it is possible to learn optimal value functions by iterating using Dynamic programming. Starting with an arbitrary policy, we can learn an optimal policy interleaving Policy evaluation, and Policy improvement. In the *Policy Evaluation* step, we first estimate the value function until it satisfies Equation 2.1, i.e., $v_{k+1}(s) \doteq \mathbb{E}_\pi \left[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s \right]$. Next, in the *Policy Improvement* step, we update the policy greedily at every state, i.e., $\pi'(s) \doteq \arg\max_a Q_\pi(s, a)$. The Policy improvement theorem guarantees that this new policy is strictly better than the old policy, unless the policy has converged. This method is known as *Policy Iteration*, and if the policy evaluation is approximated instead of exact convergence, then it is known as *Value Iteration*.

Most of the tabular, and iterative methods require an explicit model of an environment with access to the transition functions. In the absence of a model or transition functions, one can use

Monet Carlo methods to approximate these probabilities through sampling. Tabular methods are realizable in settings with small state spaces, where we have access to the the perfect model of the environment in the form of MDPs. Tabular methods are also useful in establishing theory and guarantees for various algorithms. However, for most practical applications it is common to use parameterized policies and approximate value functions. Policy gradient methods allow for directly learning a parameterized policy that enables action selection without the use of a value function. Policy gradient methods have the advantage that in many problem settings the policy space could be relatively simpler to approximate compared to the value function space. We discuss Policy gradient methods in more detail in the next chapter as it serves as a base for the rest of the thesis. We refer the reader to [10] a more detailed discussion on tabular methods, and approximate value functions.

2.3 Multi-agent Reinforcement Learning (MARL)

Single-agent RL assumes the world to be Markovian, and this assumption is still valid in many real-world settings, particularly when the environment can be isolated from the outside world. Advancements in theoretical understanding, development of algorithms for learning function approximators and the availability of computational resources have led to a resonating success of reinforcement learning in tasks with high-dimensional state spaces, and a diverse range of applications including game playing, nuclear reactor control, video compression, assisting surgeons with anesthesia control, and aligning models with human preferences [11–14]. However, often in real-world problems, there are multiple agents with objectives that can be aligned or indifferent, or adversarial to each other. Studying and designing learning algorithms

for agents in this problem setting is termed *Multi-agent learning (MAL)* [1]. The area of Multi-agent reinforcement learning (MARL) aims to design reinforcement learning algorithms that are performant in the multi-agent setting, with strong theoretical guarantees, and empirical performance.

There are multiple approaches that have been explored in designing RL algorithms that are performant in the multiagent setting. Independent MARL treats every other agent as a part of the environment and tries to maximize the reward for an agent. This approach, although not grounded in theory, is a standard baseline approach for problem domains without theoretically grounded or empirically proven algorithms. Other approaches include designing fully-decentralized learning algorithms and centralized algorithms that either share information only during the training, or both during training and execution. Among these, the Centralized Training Decentralized Execution (CDTE) regime has been widely adopted especially in cooperative settings. Another focus in cooperative or mixed-cooperative settings is to encourage agents to learn to communicate information while satisfying some communication bottlenecks. However, there is still a huge gap to fill in designing algorithms that support practical implementation, strong performance in real-world applications, and well-rooted in theory.

2.3.1 Challenges in MARL

Beyond the problems that are prevalent in single-agent settings such as sample efficiency, and variance of updates, there is a new range of problems that arise in the multiagent setting. Many of the theoretical guarantees behind reinforcement learning, no longer hold in the multi-agent setting as the environment can be non-Markovian due to the presence of other agents

that affect the reward an agent gets for taking the same set of actions. Also, the joint action space explodes exponentially in the number of agents making it computationally challenging to apply reinforcement algorithms. In cooperative settings, as agents act toward a common goal it becomes harder to discern the actions that led to some global reward, which is known as the credit assignment problem. Hence, designing algorithms that tackle the above issues is critical for better performance in many applications. To this end, there have been many attempts at designing MARL algorithms that borrow from game theory, online learning, evolutionary biology, and economics.

2.4 Online Learning, and Convex Optimization

As discussed in previous sections, problems in Game theory and Reinforcement learning typically involve learning strategies or policies in an online manner through interaction between agents and environment. Online Learning [15] studies the problem of learning to make accurate predictions in an online manner with an aim to minimize a potentially adversarial loss function. Formally, an Online learning problem is stated as follows:

Definition 7 (Online Learning) *At each round t, given an instance or a question $x_t \in \mathcal{X}$, a learner makes a prediction $p_t \in \mathcal{D}$, and is revealed the answer $y_t \in \mathcal{Y}$. The learner receives a scalar loss according to some loss function $l(p_t, y_t) \mapsto \mathbb{R}$. Here, $\mathcal{X}$ is the instance domain, $\mathcal{Y}$ is the target domain, and $\mathcal{D}$ can be $\mathcal{Y}$ or some superset of $\mathcal{Y}$.*

The answer for each round can be chosen by some adversary before observing the learner's prediction. The goal of an online learning algorithm A is to make predictions that are competi-

tive against the best-fixed hypothesis from some hypothesis class ($\mathcal{H}$). The relative performance of A with respect to some fixed *competing hypothesis* (h^*) is called its *regret*:

$$Regret_T(h^*) = \sum_{t=1}^{T} l(p_t, y_t) - \sum_{t=1}^{T} l(h^*(x_t), y_t), \tag{2.2}$$

Then, the regret of A with respect to the hypothesis class $\mathcal{H}$ is,

$$Regret_T(\mathcal{H}) = \max_{h^* \in \mathcal{H}} Regret_T(h^*) \tag{2.3}$$

2.4.1 Convergence analysis

In discussing learning and optimization algorithms, it is of theoretical and practical interest to understand the convergence guarantees of iterative methods to ascertain the efficiency and computational demands in solving various problems. In analyzing a sequence of values produced by an iterative method x_k at each iteration k for an objective function f, against some solution point x^*, two commonly categorized convergence rates are linear, and sublinear convergences. The sequence is said to converge with a sublinear rate if $f(x_k) - f(x^*) \leq O(1/k)$, and with a linear rate if $f(x_k) - f(x^*) \leq O(q^k)$ for some $q \in [0, 1]$. Based on this rate, one can also derive a bound on number of iterations required to achieve a certain approximation accuracy of ϵ, i.e. $f(x_k) - f(x^*) \leq \epsilon$.

If the loss of an Online learning algorithm grows sublinearly with each round then, the learner's average regret goes to zero as t tends to infinity. Algorithms that possess this property are called *no-regret* or regret minimizing algorithms and are of great interest in online learning,

learning strategies in games, and other prediction problems. If the learner is restricted to deterministic predictions at each round, it can be shown that there problems where the learner's regret grows unbounded [16]. Hence, an important relaxation is to allow the learner to make randomized predictions and the adversary has to choose the answer y_t without knowledge of the learner's randomization for that round.

2.4.2 Convex Optimization

We begin this section with some relevant definitions for discussing convex optimization algorithms.

Definition 8 (μ-Strongly-Convex) *A convex function $f : S \mapsto \mathbb{R}$ is μ-strongly-convex with respect to a norm $\|.\|$ if for any $w \in S$*

$$\forall z \in \partial f(w), \forall u \in S, f(u) \geq f(w) + \langle z, u - w \rangle + \frac{\mu}{2}\|u - w\|^2. \tag{2.4}$$

Definition 9 (σ-Strongly-Smooth) *Given a convex set $\mathcal{X} \in \mathbb{R}^n$, a convex function $f : \mathcal{X} \mapsto \mathbb{R}$ is σ-strongly-smooth with respect to a norm $\|.\|$, if $\|\nabla f(x) - \nabla f(y)\|_* \leq \sigma\|x - y\|, \forall x, y \in \mathcal{X}$. For a given constant L, this is also referred to as $L-$ smooth.*

It is beneficial to use tools from the well-studied area of Convex Optimization in designing and analyzing online learning algorithms. In this case, predictions are vectors that belong to a convex set $w_t \in S$, and the learner's loss at each round is given by some function, $f_t : S \mapsto \mathbb{R}$,

is convex in the dom(S). Similarly, the competing hypothesis is also a vector from a convex set $u \in U$, and the Regret (Equation 2.2) is redefined as follows:

$$Regret_T(u) = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(u) \tag{2.5}$$

And, the regret with respect to a set of all competing vectors in U is,

$$Regret_T(U) = \max_{u \in U} Regret_T(u) \tag{2.6}$$

A common assumption is that $U = S = \mathbb{R}^d$. The convex optimization framework is useful in deriving performance guarantees for Online Learning algorithms under these more restrictive assumptions. Also, for a non-convex problem, there are various ways one can convexify the objective with the loss of some accuracy in the solution to take advantage of algorithms with better performance guarantees. A preliminary algorithm is to choose the prediction vector that minimizes the cumulative loss of all previous rounds, this is called *Follow-the-leader*(FTL).

$$\forall t, w_t = \arg\min_{w \in S} \sum_{i=1}^{t-1} f_i(w)$$

As it turns out, this algorithm can have constant regret for some adversarial loss functions due to its unstable predictions.

2.4.3 FTRL

Follow-the-Regularized-leader (FTRL) extends FTL by including a regularization term to stabilize the predictions at each round. The learning rule can be written as,

$$\forall t, w_t = \arg\min_{w \in S} \sum_{i=1}^{t-1} f_i(w) + R(w).$$

where $R : S \mapsto \mathbb{R}$ is the regularization function. The choice of the regularization function leads to different algorithms with varying regret bounds. For instance, with a euclidean norm regularization, $R(w) = \frac{1}{2\eta}\|w\|_2^2$, the FTRL update becomes:

$$w_{t+1} = w_t - \eta \nabla f_t(w_t) \tag{2.7}$$

leading to the familiar gradient descent udpate, referred to as the Online Gradient Descent.

For constrained problems (e.g., the predictions have to be a valid probability distribution), it is necessary to use non-euclidean regularization functions as FTRL with Euclidean regularization (OGD) is no longer applicable. Mirror descent is a generalization of FTRL that relies on link functions or mirror maps to simplify FTRL updates. For the discussion of mirror descent, we move away from the framework of Online learning, and state the ideas from an optimization perspective following- [**?**, 17].

2.4.4 Mirror Descent

Consider the following convex optimization problem:

$$x^* = \min_{x \in X} f(x) \tag{2.8}$$

where $X \subseteq \mathbb{R}^n$ is closed and convex, and $f : X \mapsto \mathbb{R}$ is convex, L-Lipschitz with respect to some norm $\|.\|$.

Given some $u \in \mathbb{R}^n$, the Euclidean projection of u to X is given by:

$$P_{\mathcal{X}}(u) = \arg\min_{x \in \mathcal{X}} \|u - x\|. \tag{2.9}$$

A popular approach to solve Equation 2.8 is using the **Projected subgradient method**, by iteratively applying the following update:

$$x_{k+1} = P_{\mathcal{X}}(x_k - t_k f'(x_k)) \tag{2.10}$$

where $f'(x_k) \in \partial f(x_k)$ is a subgradient. Mirror Descent is an extension of the projected subgradient method Equation 2.10 to non-euclidean spaces that utilize Bergman divergences instead of the Euclidean norm.

Definition 10 (Bergman Divergence) *Given a convex set $\mathcal{X} \subset \mathbb{R}^n$, and a differentiable σ-strongly convex potential function $\phi : \mathcal{X} \mapsto \mathbb{R}$, the Bregman Divergence associated ϕ is defined as,*

$$B_\phi(x, y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y \rangle.$$

Bergman Divergences are not true norms, as they can be asymmetric, and need not satisfy the triangle inequality. Given a strongly-convex potential function ϕ, the mirror descent update rule is as follows:

$$x_{k+1} = \arg\min_{x \in \mathcal{X}} \left\{ \langle t_k \nabla f(x_k), x \rangle + B_\phi(x, x_k) \right\}. \tag{2.11}$$

One can recover the original projected subgradient method by letting $\phi = \|.\|$.

2.4.5 Proximal gradient methods

Consider the following convex optimization problem with a composite objective:

$$x^* = \min_{x \in X} f(x) + g(x)$$

where f is L-smooth g is closed and convex. Applying projected subgradient method to the above problem results in the following update:

$$x_{k+1} = \arg\min_{x \in \mathcal{X}} \left\{ t_k g(x) + \frac{1}{2} \|x - (x_k - t_k \nabla f(x_k))\|^2 \right\}$$

.

Definition 11 (Proximal operator) *The proximal operator or proximal mapping of a function f is given by:*

$$prox_{t_k,g}(y_k) = \arg\min_{x \in \mathcal{X}} \left\{ t_k g(x) + \frac{1}{2}\|x - y_k\|^2 \right\}$$

.

Simplifying the update rule using the proximal operator results in the **Proximal gradient method**:

$$x_{k+1} = prox_{t_k,g}(x_k - t_k \nabla f(x_k))$$

.

Analogous to the extension of the projected subgradient method to non-euclidean settings, the proximal gradient method can also be extended to non-euclidean settings by substituting the Euclidean distance in the proximal operator with a Bergman divergence resulting in the **Non-euclidean proximal gradient method** [17, 18].

2.4.6 Regret Minimization and CFR

As discussed so far, regret quantifies the relative performance of an online learning algorithm in hindsight as compared to an alternative hypothesis or a best fixed action. This form of regret analysis is also useful in the problem of learning in games (such as repeated matrix games), and algorithms that aim to take actions that minimize the cumulative regret are known as *regret minimization* algorithms. The above notion of regret as compared to a best fixed alternative is commonly referred to as *external regret* in this domain. Another notion of regret called

internal or swap regret compares the actions taken while allowing the swapping of one or many actions with alternatives that can lead to a lower regret. An important connection between regret minimizing algorithms and game theory is that if at iteration T, if both player's average regret is less than ϵ, then the average joint-strategy $\bar{\sigma}^T$ (average policy in RL terms) is a 2ϵ Nash equilibrium. Hence, one approach to find approximate Nash equilibria is to design strong regret minimizing algorithms, and employ it as a strategy producing technique self-play learning. *Regret matching* is a regret minimization algorithm where each player takes actions with probability distribution given by the normalized positive regrets associated with each action. Counterfactual Regret minimization (CFR) is an extension of regret minimization to extensive-form games that involves defining regret at the level of an information set, and allows for accumulating regrets along the game tree.

CHAPTER 3

MIRROR DESCENT IN REINFORCEMENT LEARNING

Given the generality of the mirror descent algorithm as a first order optimization method, there has been continued efforts to incorporate it as a single/multi-agent reinforcement learning algorithm. Extensive studies of mirror descent under various settings and assumptions help in deriving much needed theoretical guarantees for mirror descent based reinforcement learning algorithms in terms of sample complexity, and convergence rates. In this work, we study two such algorithms, namely MDPO (Mirror Descent Policy Optimization), and MMD (Magnetic Mirror Descent). We first begin with a discussion of Softmax Policy Gradients to set a base framework for the rest of the chapter before moving on to the above algorithms.

3.1 Softmax Policy Gradients

In case of a parameterized policy π_θ, where $\theta \in \Theta$ represent the policy parameter space, the aim of policy gradient methods is to maximize some objective $J(\theta)$. Similar to the value-based approximation methods, the policy can also be directly learnt using gradient ascent. A prototypical performance measure is simply the value of the initial state under the current policy: $J(\theta) = V_{\pi_\theta}(s_0)$. The Policy Gradient Theorem [10, Chapter 13.2] establishes that the gradient of this objective function can be estimated without knowledge of the environment's state distribution as long as it is stationary conditioned on the current policy.

$$\nabla_\theta J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_\theta \pi_t heta(a|s)$$

Softmax Policy Gradients For discrete action settings, it is common to use a parameterized function $y_\theta(s,a)$ to represent action preferences or *logits*, and the policy is then extracted using a softmax operator on top: $\pi(a|s,\theta) \doteq \frac{e^{y_\theta(s,a)}}{\sum_b e^{y_\theta(s,b)}}$. Softmax parameterization is the most popular form of policy representation in RL under function approximation settings. Policy gradient method with a softmax parameterization is typically referred to as Softmax Policy Gradients (SPG).

Reinforce: The most fundamental policy gradient method *REINFORCE*, uses monte-carlo estimations to approximate the performance measure.

$$\nabla_\theta J(\theta_t)_{|\theta=\theta_t} \propto \mathbb{E}_\pi \big[G_t \nabla_\theta \log \pi_{\theta_t}(S_t, A_t) \big] \tag{3.1}$$

Although Monte-carlo estimates of the returns are unbiased, they can be of high variance. A baseline that is independent of the action can be used to reduce this variance. A popular choice for such a baseline an approximate value function such as the one from value-based approximation methods ($V(s;w)$).

Actor-Critic Methods: Apart from using this approximate value function as a baseline, we can also use them to better estimate the peformance measure used in the objective. This leads to *actor-critic* methods, that learn a parameterized policy (called the actor), guided by an approximate value function is referred to as the critic.

3.1.1 Trust-region methods

As discussed in the previous section, Policy gradient methods typically use the following objective to perform gradient updates. Kakade et. al [19] note that this gradient update is heavily down weighted for unlikely states and actions, requiring very high sample complexity to make policy improvements for these states. Also, it is not clear how to decide on the stepsize α, as too large of a step can move the policy away from a good space, and too low of a step size can take a long time to converge. They instead propose Conservative Policy Iteration (CPI), to instead optimize the objective with respect to an alternate objective measure that guarantees improvement at every state when learning from an on-policy distribution.

Trust region policy optimization (TRPO) [20] is a policy gradient method that expands on the idea of CPI using constrained policy updates instead of using mixture policies. TRPO guarantees monotonic improvement with every update and can be extended to function approximation settings. The TRPO algorithm uses a conjugate gradient algorithm and performs a line search to update the policy in a way that satisfies the KL-constraint. Although TRPO shows good performance in complex continuous control tasks, it is difficult to adapt this idea when working with RNNs, and parameter sharing.

Proximal Policy Optimiation (PPO) [21] instead approximates this KL-constrained objective by using a hueristic objective and uses a simple first-order optimization algorithm (SGD) instead of the conjugate gradient algorithm. The simple nature of the clipped objective makes it easier to extend this to other settings like RNNs, and parameter sharing where it was difficult to apply TRPO.

3.2 Mirror Descent Policy Optimization

(For trust region methods: This seems like it mostly belongs in 3.1.2 and then you need to rework this introduction;)

(For MDPO citation: Make sure you include the citation at least the first time you reference them by name so that the reader can get evrything synced up)

(for tabular mirror descent guarantees: Which are? need to establish some background either in 2 or here)

(On-policy only: Needs an explanation why).

(Needs founddations somewhere).

Now, we move on to the first mirror descent-based algorithm, Mirror Descent Policy Optimization [22] (MDPO). Although the trust-region methods discussed in the previous section approach the problem of performing conservative policy updates, they can also be treated as instances of mirror descent. Shani et. al [23] show sample-based TRPO as an instance of Mirror Descent with negative entropy regularization along with an adaptive scaling for the proximal regularization. A similar study of trust-region methods in the case of policies represented using neural networks was done in [24].

Following these theoretical results, Tomar et. al [22] present MDPO as a practical implementation of extending mirror descent as a deep reinforcement learning algorithm, that takes a mirror descent step at each iteration to update the policy:

$$\pi_{k+1}(.|s) \leftarrow \arg\max_{\pi \in \Pi} \mathbb{E}_{a\sim\pi}[A_{\pi_k}(s,a)] - \frac{1}{t_k} KL(s;\pi,\pi_k) \tag{3.2}$$

This form of update is analagous to the trust-region methods discussed above, however is obtained from an optimization perspective instead of tackling the problem of performing conservative policy updates. For a parameterized policy π_θ, the on-policy MDPO update is given by:

$$\theta_{k+1} \leftarrow \arg\max_{\theta} J(\theta, \theta_k)$$

$$\text{where, } J(\theta, \theta_k) = \mathbb{E}_{s\sim\rho_{\theta_k}}[\mathbb{E}_{a\sim\pi_\theta}[A_{\theta_k}(s,a)] - \frac{1}{t_k} KL(s; \pi_\theta, \pi_{\theta_k}]$$

Instead of solving this objective exactly at each iteration, MDPO approximates it using stochastic gradient updates. As noted in [22], the gradient of the KL component in Equation 3.2 for one step is zero, and hence it is necessary to take multiple gradient steps every iteration to properly approximate the mirror descent objective. For m steps MDPO uses the following gradient to update the policy parameters,

$$\nabla_\theta J(\theta, \theta_k)|_{\theta=\theta_k^{(i)}} = \mathbb{E}_{s\sim\rho_{\theta_k}, a\sim\theta_k}\left[\frac{\pi_{\theta_k}^{(i)}}{\pi_{\theta_k}} \nabla_\theta \log \pi_{\theta_k}^{(i)}(a|s) A_{\theta_k}(s,a)\right] - \frac{1}{t_k}\mathbb{E}_{s\sim\rho_{\theta_k}}\left[\nabla_\theta KL(s; \pi_\theta, \pi_{\theta_k})_{|\theta=\theta_k^{(i)}}\right] \tag{3.3}$$

where $i = 0, 1, \ldots, m-1$.

Where, $\frac{1}{\pi_{\theta_k}}$ is the importance sampling factor that adjusts the updates for the trajectories sampled using π_{θ_k}.

MDPO also accomodates off-policy learning, and we refer the reader to [22] for details on the off-policy algorithm, and a discussion of MDPO's connection to popular RL algorithms.

3.2.1 Tabular MDPO

In this section, we formulate a tabular version of MDPO with exact policy parameterization, where the parameters are the logits, or action preferences associated with each action. The parameters are converted to a distribution over the action space using a softmax operator. We also assume that exact action-values are available for the entire action space at each state. In normal-form games, these action-values correspond to the expected payoffs associated with each action given a fixed opponent policy. This is also referred to as the all-actions setting (also known as *full-feedback* or first-order information setting). In this setting, the gradient computation in Equation 3.3 becomes:

$$\nabla\theta_{|\theta=\theta_k^{(i)}} = \nabla_\theta \sum_{a\in A}[\pi_{\theta_k}^{(i)}(a)A_{\theta_k}(a)] - \frac{1}{t_k}[\nabla_\theta KL(\pi_\theta, \pi_{\theta_k})]$$

We use this update for the tabular experiments and the original on-policy formulation for the neural experiments.

3.3 Magnetic Mirror Descent

(It isn't immediately clear where the entropy and regularization are in (3.4)) (For (3.7): This is an equation, not al algorithm, because it doesn't tell you how to compute the minimumum) (For discussion on MMD vs CFR performance: Cite? Also, have you introduced CFR at some point?)

The second extension of mirror descent that we study is Magnetic Mirror Descent (MMD). Sokota et. al [25], study relation between equilibrium solving and Variational Inequalities with

composite structures. They use this connection to propose strategy learning algorithms for two-player zero-sum games. Moreover, the proposed algorithm also extends well as a reinforcement learning algorithm that is performant in single, and multiagent settings. In this section, first we outline the connection between Varational inequalities and equlibrium solving as presented in [25]. Then we outline the Magnetic Mirror Descent algorithm and its convergence properties.

3.3.1 Connection between Variational Inequalities and QREs

A Variational Inequality (VI) problem, written as $VI(Z, F)$ is generally defined as follows:

Definition 12 *Given $\mathcal{Z} \subseteq \mathbb{R}^n$ and mapping $F : \mathcal{Z} \to \mathbb{R}^n$, the variational inequality problem VI($\mathcal{Z}, F$) is to find $z_* \in \mathcal{Z}$ such that,*

$$\langle F(z_*), z - z_* \rangle \geq 0 \quad \forall z \in \mathcal{Z}.$$

The VI problem described above is very general, and as such a wide range of problems can be cast into this framework [26]. We mainly focus on the relation between VI problems with a similar structure, and QREs.

Finding the QRE of a two-player zero-sum game can be represented as the regularized saddle point problem. Given $\mathcal{X} \subseteq \mathbb{R}^n$, $\mathcal{Y} \subseteq \mathbb{R}^m$, and $g_1 : \mathbb{R}^n \mapsto \mathbb{R}$, $g_2 : \mathbb{R}^m \mapsto \mathbb{R}$, find:

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \alpha g_1(x) + f(x, y) + \alpha g_2(y), \tag{3.4}$$

where g_1, and g_2, are strongly-convex functions. For a two-player zero-sum game, $f(x, y)$ represents the payoff matrix, and g_1, g_2 represent entropy-regularization of the strategies of

the two players. The solution (x_*, y_*) to Equation 3.4 has the following first-order optimality conditions:

$$\begin{aligned} \langle \alpha \nabla g_1(x_*) + \nabla_{x_*} f(x_*, y_*), x - x_* \rangle \geq 0, \forall x \in \mathcal{X}. \\ \langle \alpha \nabla g_2(y_*) + \nabla_{y_*} f(x_*, y_*), y - y_* \rangle \geq 0, \forall y \in \mathcal{Y}. \end{aligned} \tag{3.5}$$

The optimality conditions Equation 3.5 are equivalent to the optimality conditions of a VI$(\mathcal{Z}, G)$ with the following composite objective $G = F + \alpha \nabla g$, where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, $F(z) = [\nabla_x f(x, y) - \nabla_y f(x, y)]$, and $\nabla g = [\nabla_y g_1(x), \nabla_x g_2(y)]$. Hence, the solution to the VI: $z^* = (x^*, y^*)$ is also the solution to the saddle point problem stated in Equation 3.4. So, the reguarlized saddle point problem Equation 3.4 of finding QREs can be cast as a VI problem allowing us to tap into the vast array of existing techniques for solving Variational inequalities.

3.3.2 MMD Algorithm

Various algorithms have been proposed to solve the VI problem 12. In particular, the proximal point method has linear last iterate convergence for Variational inequality problems with a strongly monotone operator [27]. This algorithm was extended to composite objectives [28], and to non-euclidean spaces with Bergman divergence as a proximity measure, that allows for non-euclidean proximal regularization [18].

The non-euclidean proximal gradient algorithm, that is more generally applicable to any VI problem with a monotone operator, performs the following update at each iteration:

$$z_{t+1} = \arg\min_{z \in \mathcal{Z}} \eta(\langle F(z_t), z \rangle + \alpha g(z)) + B_\psi(z; z_t). \tag{3.6}$$

where $z_1 \in \text{int dom } \psi \cap \mathcal{Z}$, and ψ is a strongly convex function with respect to $\|.\|$ over $\mathcal{Z}$.

The algorithm that is termed Magnetic Mirror Descent (MMD) uses the same update as Equation 3.6 with g taken to be either ψ, or $B_\psi(.; z')$. In the former, ψ is as a strongly convex regularizer that makes the objective smoother and encouranges exploration. In the latter form, B_ψ is another proximity term that forces the iterates (z_{t+1}) to stay close to some *magnet* (z'). For all our discussion, we only consider the former update rule which is more widely applicable. In this case, the main MMD algorithm [25, (Algorithm 3.6)] becomes,

$$z_{t+1} = \arg\min_{z \in \mathcal{Z}} \eta(\langle F(z_t), z\rangle + \alpha\psi(z)) + B_\psi(z; z_t). \tag{3.7}$$

The MMD algorithm using the above update rule has the following linear convergence guarantee.

Theorem 1 *[25, Theorem 3.4] Assuming that the solution z_* to the problem VI $(\mathcal{Z}, F + \alpha\nabla g)$ lies in the int dom ψ, then*

$$B_\psi(z_*; z_{t+1}) \leq \left(\frac{1}{1+\eta\alpha}\right)^t B_\psi(z_*; z_1),$$

if $\alpha > 0$, and $\eta \leq \frac{\alpha}{L^2}$.

3.3.3 Behavioral form MMD

The update rule Equation 3.7 admits closed form in some instances, while requires approximation through gradient updates in other cases. For a parameterized policy π_θ, when ψ is negative entropy it can be restated in RL terms as follows:

$$\pi_{\theta_{k+1}}(s,a) = \arg\max_{\theta} \mathbb{E}_{s\sim\rho_{\theta_k}} \left[\mathbb{E}_{a\sim\pi_{\theta_k}}[Q_{\theta_k}(s,a)] + \alpha H(\pi_\theta) - \frac{1}{\eta} KL(\pi_\theta, \pi_{\theta_k}) \right], \tag{3.8}$$

where $H(\pi_\theta)$ is the entropy of the policy being optimized.

This behavioral form update can also be approximated using gradient updates similar to MDPO. For m steps MMD uses the following gradient to update the policy parameters,

$$\nabla_\theta J(\theta,\theta_k)|_{\theta=\theta_k^{(i)}} = \mathbb{E}_{s\sim\rho_{\theta_k} a\sim\theta_k}[\frac{\pi_{\theta_k}^{(i)}}{\pi_{\theta_k}} \nabla_\theta \log \pi_{\theta_k}^{(i)}(a|s) A_{\theta_k}(s,a)] + \alpha H(\pi_{\theta_k}^{(i)}) - \frac{1}{\eta} \mathbb{E}_{s\sim\rho_{\theta_k}}[\nabla_\theta KL(s;\pi_\theta,\pi_{\theta_k})_{|\theta=\theta_k^{(i)}}] \tag{3.9}$$

where $i = 0, 1, \ldots, m-1$.

Tabular MMD: Similar to tabular MDPO, the gradient computation for behavioral-form MMD in single-state all-action setting becomes:

$$\nabla\theta_{|\theta=\theta_k^{(i)}} = \nabla_\theta \sum_{a\in A} [\pi_{\theta_k}^{(i)}(a) A_{\theta_k}(a)] + \alpha \nabla_\theta H(\pi_{\theta_k}) - \frac{1}{\eta} [\nabla_\theta KL(\pi_\theta, \pi_{\theta_k})] \tag{3.10}$$

3.3.4 Closed-form vs Behavioral-form

MMD with a negative entropy mirror map [25, equation (12)] has the following closed-form:

$$\pi_{k+1} \propto [\pi_t \rho^{\alpha\eta} e^{\eta Q_{\pi_k}}]^{\frac{1}{1+\alpha\eta}} \tag{3.11}$$

To examine the effect of the choice of m, we compare the performance of tabular MMD Equation 3.10 to the above closed-form.

In Fig **??**, we plot the norm of the difference between the closed-form and behavioral form policies at each iteration. It can be seen that the behavioral form approximates the closed form well as expected for most choices of step size and m. For large number of gradient steps, or large step sizes the updates are unstable. For all our tabular experiments, we use a step-size of 0.1, and $m = 10$.

3.3.5 Comparison of MMD to other Mirror Decent based methods

The non-euclidean proximal gradient method Equation 3.6, has strong connections to Mirror Descent [25, Appendix D.3]. Consequently negative entropy based MMD is also equivalent to MDPO with an added entropy regularization as detailed in [25, Appendix L]. Sokota et. al [25] experimentally demonstrate MMD's strong performance in single, and multi-agent settings by evaluating it against popular baselines. In single agent settings MMD's performance is competitive with PPO in Atari and MuJoCo environments. MMD instantiated as a reinforcement learning algorithm performing behavioral form update at each information state performs on the same level as CFR, but relatively worse compared to CFR+.

CHAPTER 4

MODIFIED UPDATES FOR LAST-ITERATE CONVERGENCE

Our main contribution is adapting existing techniques for inducing convergence in multiagent settings with the algorithms discussed in the previous section. (Too brief. Slow down and explain to the reader what the problem is that these modified updates are intended to solve.) Last-iterate convergence of algorithms is an attractive property that has been studied widely in the context of optimization, and equilibirium learning algorithms. While there have been numerous methods proposed for equilibrium solving in games, many of them only guarantee convergence in the ergodic sense, i.e., only the average of their iterates converge to an equilibrium. However, in cases where the strategies are represented using function approximators, this presents a problem as it becomes complicated to maintain a history of past strategies or in terms of averaging them.

The main contribution of our work is in studying a few modifications to the algorithms discussed in the previous section, with the objective of either inducing and speeding up last-iterate convergence in two-player zero-sum settings. More specifically, we study the following three techniques, namely - Neural Replicator Dynamics [29], Extragradient updates [30], and Optimistic gradient updates [31], that have been previously studied in the context of learning in games, and solving saddle point problems. In this section we describe these techniques in more detail and provide our proposed modifications to the algorithms from the previous section.

4.1 Neural Replicator Dynamics (NeuRD)

Consider the problem of learning a parameterized policy π_θ in a single-state all-action problem setting. In such a setting, SPG employs the following update at each iteration t [29, Section A.1]:

$$y_{\theta_t}(a) = y_{\theta_{t-1}}(a) + \eta_t \pi_{\theta_{t-1}}(a)[r_t(a) - \bar{r_t}], \forall a \in A$$

where y represents the logits, $r_t(a)$ is the immediate reward associated with action a, and $\bar{r}_t$ is the average reward of the state.

4.1.1 Replicator Dynamics and No-regret Learning

The above SPG update is equivalent to the instant regret scaled by the current policy. This scaling makes it difficult for SPG to adapt to sudden changes in rewards associated with actions that are already less likely under the current policy. This problem is more evident in multiagent settings where the opponent's policy can change arbitrarily affecting rewards associated with actions even in single-state settings.

Motivated by this observation, Hennes et. al [29] propose to make modifications to SPG using the idea of Replicator dynamics from Evolutionary game theory. Replicator Dynamics defiens operators that describe the dynamics of a population's evolution when attempting to maximize some arbitrary utility function (cite). Replicator dynamics also have a strong connection to no-regret algorithms, and in [29, Statement 1], the authors establish the equivalence between Hedge, and discrete time RD. Due to this equivalence, RD also inherits the property of no-regret algorithms that their time-average policy converges to the Nash equilibrium (cite).

4.1.2 NeuRD

Neural Replicator Dynamics (NeuRD) is an adaptation of discrete-time Replicator Dynamics to reinforcement learning for function approximation settings (Include the math, not just the english). For a parameterized policy π_θ, the NeuRD-update rule is given by:

$$\theta_t \leftarrow \theta_{t-1} + \eta_t \sum_{a'} \nabla_\theta \mathrm{Y}_{\theta_{t-1}}(s, a') A(s, a') \tag{4.1}$$

where $Y_{\theta_{t-1}}$ represent the logits of the parameterized policy at timestep $t-1$, and A represents the advantage value.

The only difference between the NeuRD update, and the SPG update is that the gradient of the parameters are computed directly with respect to the logits. This can be viewed as a modification to SPG making it more adaptive in non-stationary settings. To prevent numerical issues stemming from accumulating the advantages into the logits resulting in unstable gradients, a logit gap based thresholding is proposed in [29]. This thresholding operator is: $\nabla_\theta(f(\theta), \eta, \beta) \doteq [\eta \nabla_\theta f(\theta)] \mathbb{I}\{f(\theta + \eta \nabla_\theta f(\theta)) \in [-\beta, \beta]\}$, where β is the *NeuRD-threshold* that determines how arbitrarily close to 0 or 1 the probabilities can get to. Since most PG methods, including the ones discussed in the previous chapter utilize softmax parameterization, the NeuRD fix is easily extendable to the policy loss component of their respective loss functions without any major changes. In our work, we also apply the NeuRD fix to the policy loss component of the MMD and MDPO loss functions (rewrite this sentence).

4.1.3 Relation to Natural Policy Gradients

4.1.4 Alternating vs Simultaneous gradient updates

As an aside, we wish to discuss two possible update schemes for discrete learning dynamics, namely - Simultaneous and Alternating updates. We also observe that while alternating updates shows average-iterate convergence, simultaneous updates do not converge.

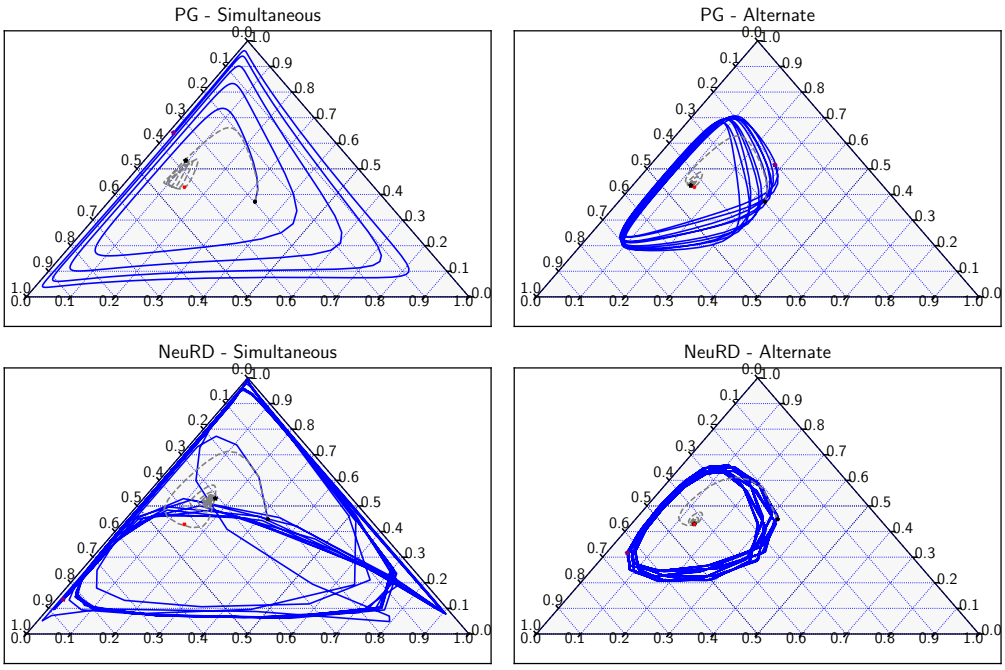


Figure 1: NeuRD converges with alternating updates, but not with simultaneous updates.

We note that all the experiments in [29] also employed alternating updates to evaluate NeuRD and SPG. A similar behavior of simultaneous gradient updates diverging was also observed in [32] for unconstrained min-max problems. They note that the sequence of iterates under alternating updates have a bounded error, and thus the average of the iterates converges to the solution. Based on these observations, we use alternating udpates for all our tabular experiments. Both of these are symmetric update schemes in the sense that both the players perform equal number of updates every iteration. Asymmetric updates have also been studied in a similar context and have been shown to improve the speed of convergence [33].

4.2 Extragradient and Optimistic Gradient

(Since you say EG builds on this (Arrow-Hurwicz) a few lines later, you need to tell the reader at least a bit more about what it is) (For opt updates: explain what has changed) (Make sure you have defined the various notions of convergence somewhere, probably in 2 or 3)

While the NeuRD fix modifies the loss function to better adjust to changing dynamics in multiagent settings, there is no inherent change to the learning procedure as such. Iterative methods such as fictitous play, are part of a broader set of learning dynamics studied to identify a simple sequence of iterative updates that each of the player in a multiagent setting can follow to converge to some solution set like the set of equilibrium points. It is common for these methods to borrow from first-order optimization algorithms [17] that have been studied extensively in varying contexts including solving variational inequalities, saddle-point problems, and convex optimization. In particular, saddle-point problems are of much interest due to their

application in modeling various problems including learning in games, generative models, and robust optimization problems.

The classic Arrow-Hurwicz (inexact Uwaza) method [34] is a popular gradient-based for saddle point problems that can be expressed as a sequence of updates:

$$\begin{aligned} x_{k+1} &= P_{\mathcal{X}}(x - \nabla f_x(x_k, y_k)) \\ x_{k+1} &= P_{\mathcal{Y}}(y + \nabla f_y(x_{k+1}, y_k)) \end{aligned} \tag{4.2}$$

While first-order gradient based iterative methods like the above work very well for function minimization problems, convergence for saddle point problems is only possible under strong-convexity assumptions even in the unconstrained case [35]. However, extensions to the above have been proposed and analyzed for solving of variational inequalities, and saddle point problems, that have better convergence guarantees under restrictive assumptions.

Extragradient Updates (EG): The Extragradient method was introduced by G. M. Korpelevich [30] as a modification to the Arrow-Hurwicz method for solving convex-concave saddle point problems, and variational inequality problems with strongly monotone operators. The EG algorithm follows the sequence of updates as given below:

$$\begin{aligned} \bar{x}_k &= P_{\mathcal{X}}[x_k - \eta f(x_k, y_k)] \\ \bar{y}_k &= P_{\mathcal{Y}}[y_k + \eta f(x_k, y_k)] \\ x_{k+1} &= P_{\mathcal{X}}[x_k - \eta f(\bar{x}_k, \bar{y}_k)] \\ y_{k+1} &= P_{\mathcal{Y}}[y_k + \eta f(\bar{x}_k, \bar{y}_k)] \end{aligned} \tag{4.3}$$

Optimistic updates (OPT): Optimistic updates are another modification to the Arrow-Hurwicz gradient method proposed by L. D. Popov [31] for solving convex-concave saddle point problems. The algorithm follows the below sequence of updates:

$$\begin{aligned} \bar{x}_k &= P_{\mathcal{X}}[x_k - \eta f(\bar{x}_{k-1}, \bar{y}_{k-1})] \\ \bar{y}_k &= P_{\mathcal{Y}}[y_k + \eta f(\bar{x}_{k-1}, \bar{y}_{k-1})] \\ x_{k+1} &= P_{\mathcal{X}}[x_k - \eta f(\bar{x}_k, \bar{y}_k)] \\ y_{k+1} &= P_{\mathcal{Y}}[y_k + \eta f(\bar{x}_k, \bar{y}_k)] \end{aligned} \tag{4.4}$$

Contrast to the EG method, optimistic updates reuse the gradients of the leading points of the previous iteration $f(\bar{x}_{k-1}, \bar{y}_{k-1})$ instead of computing an *extragradient* in each iteration $f(\bar{x_k}, \bar{y_k})$ as done in Equation 4.3. Optimistic and Extragradient updates are also interpreted as a variants extrapolation methods where EG uses an additional gradient computation, and Optimistic updates approximates this extrapolation by reusing previous iteration's gradient. Optimistic updates were studied in the context of online learning under the name *Optimistic Mirror Descent (OMD)* [36]. OMD when applied to saddle-point problems leads to two popularly studied variants - Optimistic Gradient Descent Ascent (OGDA) and Optimistic Multiplicative Weight Updates (OMWU), that use euclidean and entropic mirror maps respectively.

Due to their broad applicability, and theoretical grounding, convergence rates for these algorithms have been studied under various settings. For VIs with strongly monotone operators or composite structures, EG has a linear last-iterate convergence rate [28]. A linear last-iterate convergence for OGDA in bilinear games and an optimistic version of the Adam optimizer

was also given in [33]. [37] also study the EG method for training GANs by establishing a framework of *coherence*, and show that EG has last-iterate convergence for coherent VI problems. A unified analysis of both these algorithms was also presented [38] studied both these algorithms in a unified way as an approximation of the Proximal point method, and showed that both these algorithms have linear last-iterate convergence for the strongly-convex strongly-convex and bilinear cases. More recently, these results have been extended to monotone VIs, constrained saddle point problems, markov games, and extensive form games under various conditions [39–43]. In our work, we incorporate extragradient and optimistic updates into the algorithms discussed in the previous chapter and experimentally evaluate the improvement gains when applied to two-player zero-sum games.

CHAPTER 5

TABULAR EXPERIMENTS

We now evaluate our proposed methods experimentally in both tabular and function approximation settings as approximate equilibrium solvers. Through the experiments, we aim to answer the following questions:

1. What is the last-iterate and average-iterate convergence behavior of these algorithms?
2. How does the addition of NeuRD-fix, Extragradient updates, and Optimistic upates affect the convergence rate of these algorithms (alone and in combination)?
3. Do these performance improvements scale well with the size of the game?

MMD has theoretical converegence guarantees only as a QRE solver, but shows a strong empirical performance for finding approximate Nash Equilibriums. In this work our main focus is convergence to the Nash Equilibrium, and as such we focus our main experimental results for the same. We provide some additional results for the performance of different algorithms as QRE solvers in the appendix.

5.1 Experimental Domains

We evaluate the algorithms on two normal form games namely, Perturbed RPS and Matching Pennies.

(Not sure what goes here. This seems likely to at least in part duplicate 2 as laid out, so be careful about that)

	H	T
H	1	-1
T	-1	1

(a) Matching Pennies Payoffs.

	R	P	S
R	0	-v	v
P	v	0	-1
S	-v	1	0

(b) Perturbed RPS Payoffs.

TABLE I: Tabular NFG Payoffs

Matching Pennies Matching Pennies is a classic example from Game Theory that demonstrates decision making in multiagent settings in the simplest form. In matching pennies, two players toss coins independently and the row player wins if they both land on the same side, the column player wins otherwise. The payoffs for Matching Pennies are shown in Table I.

Perturbed RPS Perturbed/Biased RPS is a modified version of the canonical normal form game Rock-Paper-Scissors with biased payoffs for each action. Table I shows the payoff matrix for Perturbed RPS.

Being symmetric matrix games, both games have a unique Nash Equilibrium: Perturbed RPS $(\frac{1}{7}, \frac{3}{7}, \frac{3}{7})$, Matching Pennies $(\frac{1}{2}, \frac{1}{2})$.

5.2 Evaluation Metrics

Our main focus being the convergence behaviors and rates, we need a notion of distance from the equilibrium point to measure the performance of these algorithms.

5.2.1 Divergence to the equilibrium

In settings with a known unique equilibrium, we can compute the distance of the current policy to the known equilibrium using a measure of distance in the policy space such as the KL-Divergence. Given the policy at iteration t, pi_t, and an equilibrium policy π_*, the metric we measure is: $KL(\pi_t||\pi_*) = \sum_{a \in A} \pi_t(a) \log\left(\frac{\pi_t(a)}{\pi_*(a)}\right)$.

5.2.2 Exploitability

In general, there might not be a unique Nash equilibrium and we might not know which equilibrium point the current policy is converging towards. This makes it tricky to use a direct measure of distance as the above metric. Exploitability is another metric that is commonly used as a notion of optimality in game theory. Exploitability measures the gain in value a player can achieve by deviating from the current policy. We measure the value that a worst-case opponent can achieve by keeping the current policy fixed by computing a best response at every state. The difference between the value that this best-response opponent can acheive and the game value is the exploitability of the current policy.

- Exploitability formal expression in terms of best responses, and value.

5.3 Experiment Setup

(something wrong; this sentence doesn't parse. Has enough facts in it that it is probably better to split into multiple sentences.) (Explain how this was chosen as you did for other parameters below.) In the tabular experiments we use softmax-parameterized tabular policies and assume an all-action setting (full-feedback updates). For all the algorithms, we train both the players for 5000 training steps with alternating updates using the exact payoff vectors. For

the EG variants, we train the players for only 2500 steps as they use two gradient computation per step for a fair comparison. We use the $m = 10$ gradient steps per iteration with a learning rate of 0.1 for all the runs. For MMD and MDPO, we anneal the temperature (entropy coefficient) with the schedule $\alpha_t = 1/\sqrt{t}$. For the KL-coefficient, we use different schedules for MMD ($\eta_t = \max(1/\sqrt{t}, 0.2)$), and MDPO ($\eta_t = \max(1 - t/T, 0.2)$). MDPO's KL schedule is motivated by mirror-descent theory, while the entropy annealing and MMD's KL schedule are closer to the annealing schedule used in the original MMD experiments found through a hyperparameter sweep. For both of these methods we cap the KL-coefficient at 0.2 as very low values destabilize the updates, especially for the NeuRD version of the algorithms.

5.4 Results

(iankash: Currently the writing here is very choppy. Even when presenting results like this you should be telling the reader a story, not just spitting a list of facts at them.)

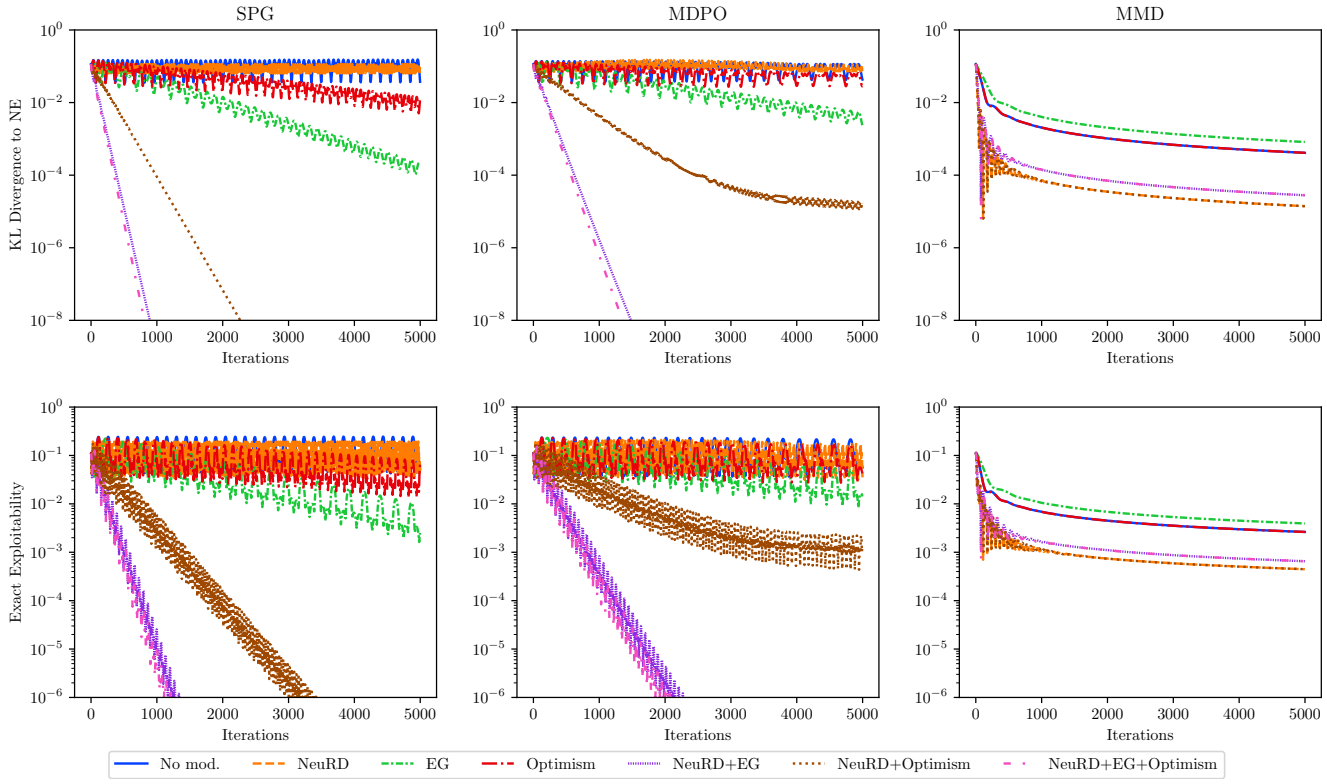
(iankash: Keep this as just part of 5.3)

(iankash: Relatedly, one plot with 21 lines in it is illegible (at least in places). Can you break this giant plot into multiple plots from the same data that tell each part of the story in an easier to digest way?)

(iankash: Seems like we should also have plots for the average iterate?) (iankash: Figures are missing x-axis label; Explain EG correction in them)

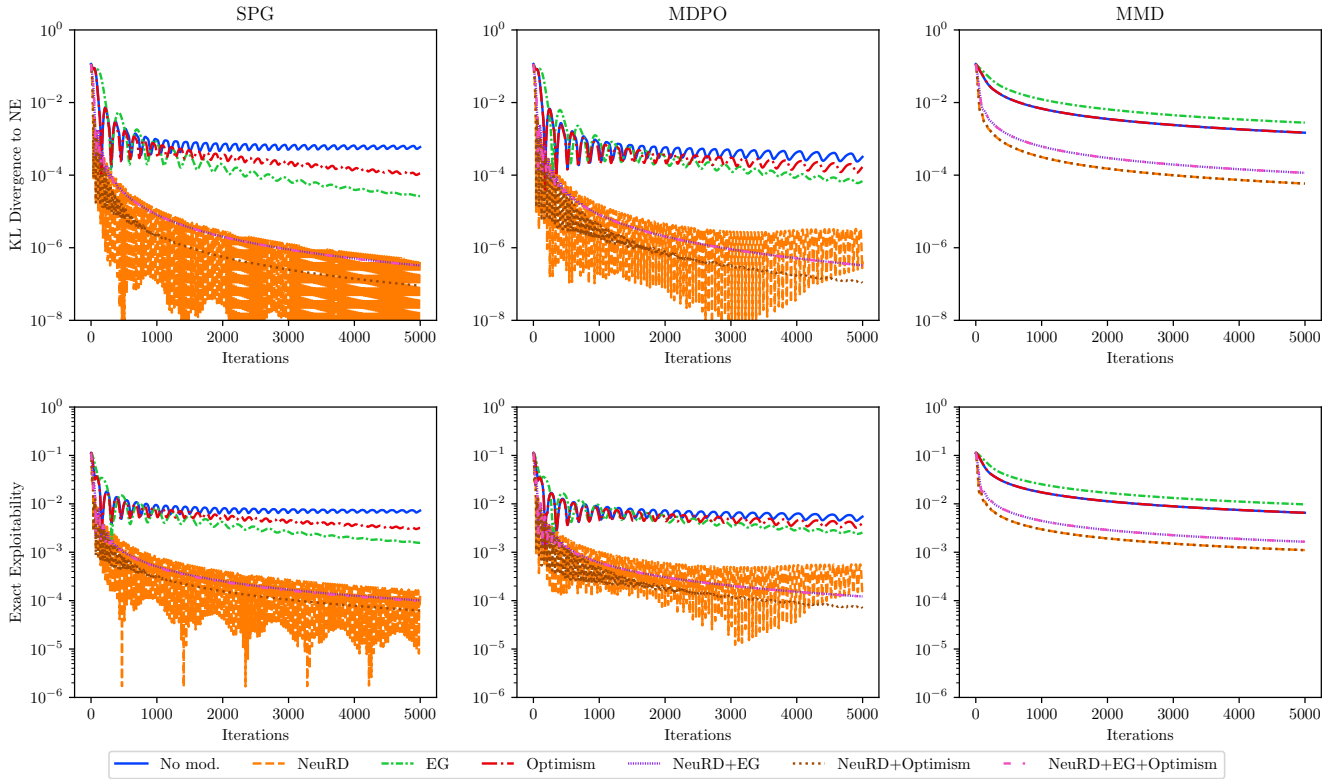
We plot the last-iterate convergence in terms of both evaluation metrics in Figure 2 as a function of the iterations. We also plot the average-iterate convergence for all the methods in Figure 3. As mentioned in the experiment setup, the experiment was run for only 2500 steps

and one iteration counts as 2. "No mod." indicates the base version of these algorithms without any modifications that serves as a baseline for comparison.



Note: One iteration counts as 2 for all EG variants.

Figure 2: Last-iterate converegence in PerturbedRPS.



Note: One iteration counts as 2 for all EG variants.

Figure 3: Average-iterate converegence in PerturbedRPS.

There are many observations that can be made from the performances of these algorithms in the presence of these modifications. We group these observations by the modifications, and contrast their effects on SPG, MDPO, and MMD.

NeuRD Fix: SPG, and MDPO do not have last-iterate or average-iterate convergence guarantees, and the same is observed experimentally for their *No mod.* baselines. The NeuRD variants of SPG, and MDPO display average-iterate convergence Figure 3, as guaranteed by its no-regret properties. However, as expected the addition of NeuRD fix does not induce last-iterate convergence for SPG, and MDPO. For MMD, which already has average iterate convergence guarantees, it speeds up the convergence speed. In fact, NeuRD-fix is the only modification that has a significant impact in terms of convergence for MMD.

EG, and Optimism: The baseline version of MMD has last-iterate convergence as guaranteed by its theory. As seen in Figure 2, EG, and Optimistic updates either induces last-iterate convergence or speeds it up in most cases. Interestingly, there are a few algorithm specific differences in terms of the improvement gained through these modifications. EG updates induce last-iterate convergence in SPG, and MDPO. On the contrary, the addition of EG updates slows down MMD. While Optimistic updates work in SPG, they do not help with convergence for MDPO.

Combining the modifications: We also experiment with combination of these modifications to see if this leads to a superior performance. Combining EG, and Optimism leads to marginal or no improvement, as they both perform a similar modification of extrapolating the gradients. However, we can see a significant improvement in convergence speeds when EG,

and Optimistic updates are combined with the NeuRD fix. Hence, contrary to trust-region constraints, the NeuRD-fix provides an improvement that is orthogonal to EG/Optimistic Updates that also help in reducing the cycling around the equilibrium. For both SPG, and MDPO, 'NeuRD + EG + Optimism' provides the fastest last-iterate convergence. Whereas, for MMD, 'NeuRD', 'NeuRD + Optimism' are the fastest variants.

Average vs Last Iterate: A primary motivation for our study is achieving last-iterate convergence, but it is also important to discuss the behavior of average iterate convergence and how it compares to the last-iterate convergence. For instance, it was shown that the last-iterate convergence is slower than the average-iterate for the EG method for smooth convex-concave saddle point problems in [44] by proving a tight lower bound for the last-iterate. We note that from our tabular experimental results, the last-iterate convergence is better than the average iterate for all the algorithms.

The tabular experiments, and the above observations provide some answers to questions 1, and 2.

5.5 Algorithm Design Choices

5.5.1 NeuRD-fix

The NeuRD-loss has also been previously applied on top of algorithms to improve performance or induce convergence in competitive and cooperative settings. Perolat et. al, [12] introduced Regularized Nash Dynamics (R-NAD) that utilizes NeuRD as a fixed-point approximator along with adaptive regularization [45] to achieve last-iterate convergence in the game of Stratego. Chhablani et. al, [46] motivated the choice of an advantage baseline in COMA [47]

through no-regret literature and demonstrated that applying the NeuRD loss to COMA's objective improved its performance even in cooperative settings such as identical-interest games. The improved performance of adapting the NeuRD-fix with mirror-descent based methods, and other techniques for last-iterate convergence is also evident from our experimental evaluations. Through this, we reinforce the idea that the NeuRD-loss can be more generally adapted into loss functions of various algorithms in multi-agent settings. is there a benefit to adding NeuRD in single-agent setting? even if the reward function is not dynamic.

5.5.2 Entropy Regularization

As noted in the tabular experiments, in some problem settings there might be faster methods to induce last-iterate convergence that adaptive regularization. However, entropy regularization is more widely studied within single, and multi-agent RL. Adaptive entropy regularization is also relatively easier to implement, and less constraining in terms of how each player updates their policy as compared to modifications like EG, or Optimism. In single-agent RL, entropy regularization is motivated as a means to encourage exploration, and make the policies more robust. In multi-agent RL, it has been studied more specifically to induce convergence even including in NeuRD, and MMD. Given the universal inclusion, and multiple perceived benefits, entropy regularization is considered a standard design choice to include in most algorithms.

5.5.3 Trust-region constraints

Trust-region constraints form the basis of state-of-the-art RL algorithms like PPO. The presence of a relatively strongly-convex proximal operator is necessary for deriving algorithms with strong performance guarantees.

5.5.4 EG and Optimism

Finally, we highlight that the best performing variants are on top of SPG as opposed to MDPO, or MMD as can be seen from Figure 2. This is surprising as the latter methods were proposed as improvements over SPG in single, and multi-agent settings. But, this indicates that EG/Optimistic updates provide a better last-iterate convergence rates compared to adaptive entropy regularization with trust-region constraints. While the convergence rates of EG, and Optimistic updates are well-studied, it is not the case for adaptive regularization. Most notably, Optimistic-NeuRD (SPG + NeuRD + Optimism) exhibits a faster last-iterate convergence that is straightforward to extend to more complicated settings.

Based on the above findings, and discussion, we cast the various algorithms discussed as extensions of policy gradient methods with different added components. In the next section we perform an empirical study of these extensions under function approximation.

CHAPTER 6

DEEP MULTI-AGENT RL EXPERIMENTS

All of the methods discussed in this work are typically implemented in function approximation settings. Hence, it is important to evaluate if the observed performance improvements carry over when the policy parameters, and the value approximation are both represented using neural networks.

Expanding on our the observations from the tabular experiments, we proceed to evaluate the algorithms in large-scale 2p0s games to answer question 3. For the neural experiments, we study the effect of the NeuRD-fix, and the utilization of Optimistic-variants of popular optimizers (Optimistic-SGD, and Optimistic-Adam) when paired with deep RL algorithms.

6.1 Experimental Domains

In the tabular experiments, we evaluated the algorithms as reinforcement learning methods in normal-form games. In this section, we evaluate the deep RL variant of these algorithms in extensive-form games. EFGs have been used as common benchmark and milestones in designing and evaluating multi-agent reinforcement learning. Should I expand on this? We follow the suit of [25], and choose four 2p0s EFGs for the evaluation, namely - Kuhn Poker, Abrupt Dark Hex, and Phantom Tic-tac-toe. Kuhn Poker, and 2x2 Dark Hex are smaller benchmarks that were used to test, and tune the algorithms. Phantom Tic-tac-toe, and 3x3 Dark Hex, are large-scale

benchmarks that are used here to evaluate the scalability, and stability of these algorithms in converging to an equilibrium. Please refer to Appendix B for an overview of these games.

6.2 Evaluation Metrics

The measures used in tabular experiments are harder to compute for large-scale EFGs. Distance to an equilibrium point is harder to compute due to the absence of a unique equilibrium point. For larger games, the computation of exact best responses to measure the exact exploitability is prohibitive due to the large state space. However, we can approximate the exploitability computation if we can approximate the best response for a given fixed policy.

6.2.1 Approximate Exploitability

For poker games, a local best response approximator (LBR) [48] has been commonly used in place of an exact best response computation by combining local search along with truncation based on a heuristic value function. However, the value function is based on poker domain-specific knowledge, and does not translate to other games. Instead, an alternative LBR formulation was proposed in [49] that uses MCTS [1] and an approximate learnt value function. The exploitability is then computed by

Then the exploitability can be approximated by sampling trajectories and measuring the average reward the best response agent acheived against the exploited policy. In our experiments, we train the best response agent against the fixed joint-policy learn through self-play.

---

[1]It uses an extension of MCTS to imperfect-information settings called information state MCTS (ISMCTS) [50]

Let π_{BR} be a learnt best-response approximator, and $\pi_{fixed} = (\pi_1, \pi_2)$ be the joint policy to be exploited, then the approximate exploitability is given by:

$$\text{Exp}_{appx}(\pi_{BR}, \pi_{fixed}) = \frac{1}{N}\left[\sum_{i=1}^{N/2} \mathbb{E}_{a\sim\pi_1,\pi_{BR}}[R] + \sum_{i=1}^{N/2} \mathbb{E}_{a\sim\pi_{BR},\pi_2}[R]\right] \tag{6.1}$$

TBD: rephrase the equation in terms of cumulative episodic rewards.

6.3 Experiment Setup

Due to the strong performance exhibited by MMD [25] as a deep MARL algorithm for 2p0s games, we consider vanilla-MMD to be our baseline. For a proper comparison of MMD's performance against the proposed algorithms, we setup our experiments following [25], and we outline the various implementation details, and experimental choices made in the former work here.

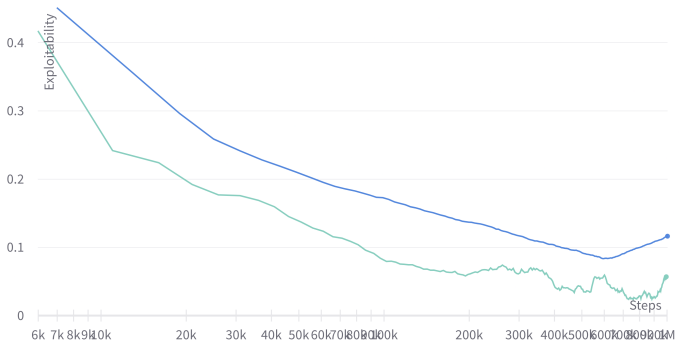
MMD was implemented as a modification of RLLib's [51] PPO, by modifying the entropy, and KL annealing schedules. RLLib's PPO implementation uses MLPs to represent the policy, and value networks, which is sufficient for the vector representation of information states in OpenSpiel. The networks have 2 fully connected layers with [256, 256] hidden units, and use tanh activations. We also note that RLLib's implementation makes use of Generalized Advantage Estimates (GAE) [52] instead of regular advantage estimates. The experiments used OpenSpiel [53] for all the benchmark game implementations. RLLib's environment adapter was used for the algorithms to interact with the OpenSpiel environments, and the RLLib's adapter implementation was modified to work with information states instead of observations as required

by the problem specification. All algorithms were trained using self-play, and the evaluations were done on the final trained joint-policy. We retain all the above choices with PyTorch as the framework of choice within RLLib, while only making changes for the addition the proposed changes. For NeuRD, we modify the policy loss component of MMD implementation to use logit-thresholded advantages, with a β of 2.0 (which is the default in OpenSpiel, and we find this to work well among a few choices that we experimented with [1.0, 2.0, 5.0, 10.0]). For Optimistic variants of the algorithms, we adapt existing PyTorch implementations of optimistic versions of popular neural network optimizers (SGD, and Adam). We use a training batch size of 4000 trajectories, and within each training iteration, we sample mini-batches of size 128 to perform mini-batch SGD updates. We perform 30 SGD updates per iteration, and use a constant learning rate of 5e-5 for all the experiments.

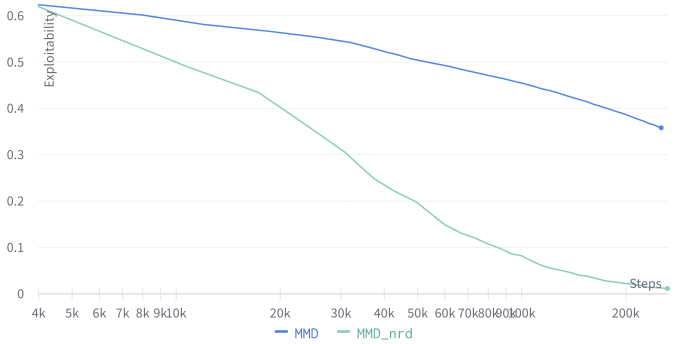
For the smaller games (Kuhn Poker, and Abrupt Dark Hex 2×2), we train the agents for 1M steps, and for the larger games we train the agents for 5M steps. For Kuhn Poker, and 2x2 Dark Hex, we report the exact exploitability curves across the training iterations, For 3x3 Dark Hex, and Phantom TTT, due to the difficulty in computing exact exploitability we only report the approximate exploitability at the end of training. To compute the approximate exploitability, we train a DQN approximate best-response agent against the trained joint-policy of the players for 5M steps. Then, we evaluate the performance of the DQN agent against the fixed joint-policy for 1000 episodes, with the DQN agent starting first in 500, and the exploitee player starting first in the other 500. We report the win-rate of the DQN agent across these episodes as the approximate exploitability which ranges between 0 and 1, as done in [25].

6.4 Results

Figure 4 plots the exact exploitabililty of the joint policy as a function of the iterations, and Figure 4 shows the approximate exploitability at the end of training for these games. These results show correspondence between exact and approximate exploitability as expected qualifying the latter as a valid evaluation metric for the larger games. In agreement to the tabular experiments, the NeuRD versions of these algorithms have a better performance.



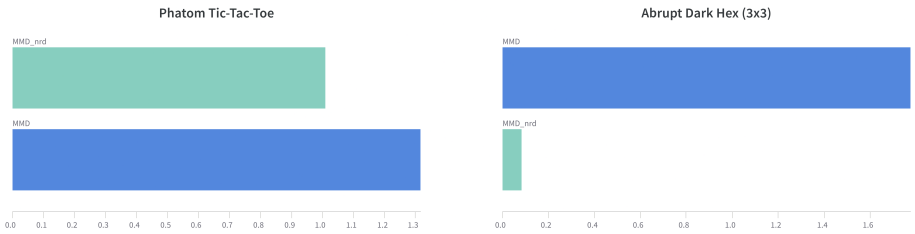
(a) Kuhn Poker



(b) Abrupt Dark Hex (2×2) (1M steps version to be added)

Figure 4: Performance in small EFGs, measured by exact exploitability.

Figure 5 shows improvement in performance by applying the NeuRD-fix in Abrupt Dark Hex (3×3) and Phantom TTT, as measured by the approximate exploitability.



(a) Phantom Tic-tac-toe (b) Abrupt Dark Hex (3×3)

Figure 5: Performance in larger EFGs, measured by approximate exploitability.

Firstly, echoing the observations from the tabular experiments, we observe that the addition of NeuRD-fix improved the performance of MMD. Secondly, there is no consistent benefit of using the optimistic version of the optimizers in contrast to their effect in normal-form games.

We also evaluate these algorithms by having the trained agents play against each other in a head-to-head manner.

CHAPTER 7

DISCUSSION

CHAPTER 8

CONCLUSION

In this thesis, we studied mirror-descent based methods applied to two-player zero-sum games, and proposed modifications to these algorithms to achieve last-iterate convergence. We experimentally We discussed how the various modifications are reoccuring components in general reinforcement learning and game-theoretic algorithms. We also raise interesting observations that stand out from our experimental evaluations some of which coincide with previous findings, and others are less understood phenomena. We believe these findigns are useful in the design of reinforcement learning algorithms for two-player zero-sum games and beyond.

APPENDICES

Appendix A

SOME ANCILLARY STUFF

Ancillary material should be put in appendices.

Appendix B

GAME DESCRIPTIONS

Kuhn Poker Kuhn Poker is a smaller extensive form game that allows for more introspection and exact exploitability computation.

Abrupt Dark Hex Abrupt Dark Hex

Phantom Tic-Tac-Toe Phantom Tic-Tac-Toe (Phantom TTT)

Appendix C

IMPLEMENTATION DETAILS

Appendix D

SOME MORE ANCILLARY STUFF

CITED LITERATURE

1. Tuyls, K. and Weiss, G.: Multiagent Learning: Basics, Challenges, and Prospects. *AI Magazine* , 33(3):41–41, September 2012.

2. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.: Generative Adversarial Nets. In *Advances in Neural Information Processing Systems* , volume 27. Curran Associates, Inc., 2014.

3. Osborne, M. J.: *An Introduction to Game Theory* . New York, Oxford University Press, 2004.

4. eds. N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani *Algorithmic Game Theory* . Cambridge, Cambridge University Press, 2007.

5. Shoham, Y. and Leyton-Brown, K.: *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* . Cambridge University Press, December 2008.

6. Nash, J. F.: Equilibrium Points in n-Person Games. *Proceedings of the National Academy of Sciences of the United States of America* , 36(1):48–49, 1950.

7. Kuhn, H. W.: Extensive Games. *Proceedings of the National Academy of Sciences of the United States of America* , 36(10):570–576, 1950.

8. McKelvey, R. D. and Palfrey, T. R.: Quantal Response Equilibria for Normal Form Games. *Games and Economic Behavior* , 10(1):6–38, July 1995.

9. Mckelvey, R. D. and Palfrey, T. R.: Quantal Response Equilibria for Extensive Form Games. *Experimental Economics* , 1(1):9–41, June 1998.

10. Sutton, R. S. and Barto, A. G.: *Reinforcement Learning: An Introduction* . Adaptive Computation and Machine Learning Series. Cambridge, Massachusetts, The MIT Press, second edition edition, 2018.

11. Schamberg, G., Badgeley, M., Meschede-Krasa, B., Kwon, O., and Brown, E. N.: Continuous action deep reinforcement learning for propofol dosing during general anesthesia. *Artificial Intelligence in Medicine* , 123:102227, January 2022.

12. Perolat, J., de Vylder, B., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., McAleer, S., Elie, R., Cen, S. H., Wang, Z., Gruslys, A., Malysheva, A., Khan, M., Ozair, S., Timbers, F., Pohlen, T., Eccles, T., Rowland, M., Lanctot, M., Lespiau, J.-B., Piot, B., Omidshafiei, S., Lockhart, E., Sifre, L., Beauguerlange, N., Munos, R., Silver, D., Singh, S., Hassabis, D., and Tuyls, K.: Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning. *Science* , 378(6623):990–996, December 2022.

13. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R.: Training language models to follow instructions with human feedback, March 2022.

14. Mandhane, A., Zhernov, A., Rauh, M., Gu, C., Wang, M., Xue, F., Shang, W., Pang, D., Claus, R., Chiang, C.-H., Chen, C., Han, J., Chen, A., Mankowitz, D. J., Broshear, J., Schrittwieser, J., Hubert, T., Vinyals, O., and Mann, T.: MuZero with Self-competition for Rate Control in VP9 Video Compression, February 2022.

15. Shalev-Shwartz, S.: Online Learning and Online Convex Optimization. *Foundations and Trends® in Machine Learning* , 4(2):107–194, 2012.

16. Cover, T.: Behavior of sequential predictors of binary sequences. In *Proc. of the 4th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes* , pages 263–272. Publishing House of the Czechoslovak Academy of Sciences, 1965.

17. Beck, A.: *First-Order Methods in Optimization* . MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, October 2017.

18. Tseng, P.: Approximation accuracy, gradient methods, and error bound for structured convex optimization. *Mathematical Programming* , 125(2):263–295, October 2010.

19. Kakade, S. and Langford, J.: Approximately Optimal Approximate Reinforcement Learning. In *International Conference on Machine Learning* , July 2002.

20. Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P.: Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning* , pages 1889–1897. PMLR, June 2015.

21. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O.: Proximal Policy Optimization Algorithms, August 2017.

22. Tomar, M., Shani, L., Efroni, Y., and Ghavamzadeh, M.: Mirror Descent Policy Optimization. In *International Conference on Learning Representations* , January 2022.

23. Shani, L., Efroni, Y., and Mannor, S.: Adaptive Trust Region Policy Optimization: Global Convergence and Faster Rates for Regularized MDPs. *Proceedings of the AAAI Conference on Artificial Intelligence* , 34(04):5668–5675, April 2020.

24. Liu, B., Cai, Q., Yang, Z., and Wang, Z.: Neural Trust Region/Proximal Policy Optimization Attains Globally Optimal Policy. In *Advances in Neural Information Processing Systems* , volume 32. Curran Associates, Inc., 2019.

25. Sokota, S., D'Orazio, R., Kolter, J. Z., Loizou, N., Lanctot, M., Mitliagkas, I., Brown, N., and Kroer, C.: A Unified Approach to Reinforcement Learning, Quantal Response Equilibria, and Two-Player Zero-Sum Games. In *The Eleventh International Conference on Learning Representations* , February 2023.

26. eds. F. Facchinei and J.-S. Pang *Finite-Dimensional Variational Inequalities and Complementarity Problems* . Springer Series in Operations Research and Financial Engineering. New York, NY, Springer, 2004.

27. Rockafellar, R. T.: Monotone Operators and the Proximal Point Algorithm. *SIAM Journal on Control and Optimization* , 14(5):877–898, August 1976.

28. Tseng, P.: On linear convergence of iterative methods for the variational inequality problem. *Journal of Computational and Applied Mathematics* , 60(1):237–252, June 1995.

29. Hennes, D., Morrill, D., Omidshafiei, S., Munos, R., Perolat, J., Lanctot, M., Gruslys, A., Lespiau, J.-B., Parmas, P., Duenez-Guzman, E., and Tuyls, K.: Neural Replicator Dynamics, February 2020.

30. Korpelevich, G. M.: The extragradient method for finding saddle points and other problems. *Matecon* , 12:747–756, 1976.

31. Popov, L. D.: A modification of the Arrow-Hurwicz method for search of saddle points. *Mathematical notes of the Academy of Sciences of the USSR* , 28(5):845–848, November 1980.

32. Gidel, G., Berard, H., Vignoud, G., Vincent, P., and Lacoste-Julien, S.: A Variational Inequality Perspective on Generative Adversarial Networks, August 2020.

33. Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H.: Training GANs with Optimism, February 2018.

34. Arrow, K. J.: *Studies in Linear and Non-Linear Programming* . Stanford Mathematical Studies in the Social Sciences. Stanford, Calif., Stanford University Press, 1958.

35. He, B., Xu, S., and Yuan, X.: On Convergence of the Arrow–Hurwicz Method for Saddle Point Problems. *Journal of Mathematical Imaging and Vision* , 64(6):662–671, July 2022.

36. Rakhlin, A. and Sridharan, K.: Optimization, Learning, and Games with Predictable Sequences, November 2013.

37. Mertikopoulos, P., Lecouat, B., Zenati, H., Foo, C.-S., Chandrasekhar, V., and Piliouras, G.: Optimistic Mirror Descent in Saddle-Point Problems: Going the Extra (Gradient) Mile. 2019.

38. Mokhtari, A., Ozdaglar, A., and Pattathil, S.: A Unified Analysis of Extra-gradient and Optimistic Gradient Methods for Saddle Point Problems: Proximal Point Approach. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics* , pages 1497–1507. PMLR, June 2020.

39. Wei, C.-Y., Lee, C.-W., Zhang, M., and Luo, H.: Linear Last-iterate Convergence in Constrained Saddle-point Optimization, March 2021.

40. Cen, S., Chi, Y., Du, S. S., and Xiao, L.: Faster Last-iterate Convergence of Policy Optimization in Zero-Sum Markov Games. In *The Eleventh International Conference on Learning Representations* , September 2022.

41. Cai, Y., Oikonomou, A., and Zheng, W.: Tight Last-Iterate Convergence of the Extragradient and the Optimistic Gradient Descent-Ascent Algorithm for Constrained Monotone Variational Inequalities, May 2022.

42. Gorbunov, E., Loizou, N., and Gidel, G.: Extragradient Method: O(1/K) Last-Iterate Convergence for Monotone Variational Inequalities and Connections With Cocoercivity. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics* , pages 366–402. PMLR, May 2022.

43. Gorbunov, E., Taylor, A., and Gidel, G.: Last-Iterate Convergence of Optimistic Gradient Method for Monotone Variational Inequalities. In *Advances in Neural Information Processing Systems* , May 2022.

44. Golowich, N., Pattathil, S., Daskalakis, C., and Ozdaglar, A.: Last Iterate is Slower than Averaged Iterate in Smooth Convex-Concave Saddle Point Problems. In *Proceedings of Thirty Third Conference on Learning Theory* , pages 1758–1784. PMLR, July 2020.

45. Perolat, J., Munos, R., Lespiau, J.-B., Omidshafiei, S., Rowland, M., Ortega, P., Burch, N., Anthony, T., Balduzzi, D., Vylder, B. D., Piliouras, G., Lanctot, M., and Tuyls, K.: From Poincaré Recurrence to Convergence in Imperfect Information Games: Finding Equilibrium via Regularization. In *Proceedings of the 38th International Conference on Machine Learning* , pages 8525–8535. PMLR, July 2021.

46. Chhablani, C. and Kash, I. A.: Counterfactual Multiagent Policy Gradients and Regret Minimization in Cooperative Settings. In *AAAI* , 2021.

47. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S.: Counterfactual Multi-Agent Policy Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence* , 32(1), April 2018.

48. Lisy, V. and Bowling, M.: Equilibrium Approximation Quality of Current No-Limit Poker Bots, January 2017.

49. Timbers, F., Bard, N., Lockhart, E., Lanctot, M., Schmid, M., Burch, N., Schrittwieser, J., Hubert, T., and Bowling, M.: Approximate Exploitability: Learning a Best Response. In *Thirty-First International Joint Conference on Artificial Intelligence* , volume 4, pages 3487–3493, July 2022.

50. Cowling, P. I., Powley, E. J., and Whitehouse, D.: Information Set Monte Carlo Tree Search. *IEEE Transactions on Computational Intelligence and AI in Games* , 4(2):120–143, June 2012.

51. Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I.: RLlib: Abstractions for Distributed Reinforcement Learning, June 2018.

52. Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P.: High-Dimensional Continuous Control Using Generalized Advantage Estimation, October 2018.

53. Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., Hennes, D., Morrill, D., Muller, P., Ewalds, T., Faulkner, R., Kramár, J., De Vylder, B., Saeta, B., Bradbury, J., Ding, D., Borgeaud, S., Lai, M., Schrittwieser, J., Anthony, T., Hughes, E., Danihelka, I., and Ryan-Davis, J.: OpenSpiel: A Framework for Reinforcement Learning in Games, September 2020.