Modified upates 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

CHAPTER PAGE

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF NOTATIONS

θ	Parameters of a function approximator.

SUMMARY

Put your summary of thesis.

CHAPTER 1

INTRODUCTION

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 presents opportunities in creating learning algorithms for agents in Multi-agent settings. Multiagent Learning (MAL) is an area of research that studies the application of machine learning to Multiagent Systems. Reinforcement learning (RL), an area of study within machine learning is a popular choice of learning technique due to its compatability in terms of learning through interaction. Although RL algorithms have gained applications in various domains, their direct applicability in multi-agent setting is limited due to the non-stationarity problem. Multiagent Reinforcement Learning (MARL) as a research area deals with designing efficient and performant RL algorithms for general multiagent problems by incorporating ideas from game theory, online learning, and evolutionary biology etc. Apart from the non-stationarity problem There are a range of problems that are currently being studied in designing Multi-agent systems, including communication,

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. Specifically we study Mirror Descent Policy Optimization, and Magnetic Mirror Descent, two recent algorithms that extends Mirror Descent as Single RL algorithms, and approximate equilibrium solvers. We propose novel improvements to these algorithms by incorporating existing techniques and study their convergence behaviors in normal form games. We also evaluate the performance of these algorithms in large extensive form games under function approximation. We 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 these study we provide some recommendations in designing MARL algorithms and close with some remarks about future research directions.

? Recent developments have demonstrated a vast potential in application of machine learning, and deep learning to a wide range of domains with efforts in creating more general large-scale foundational models to smaller specialized models that are optimized for specific use cases. Due to such progress it can be expected that there will soon be a prevalance of such learnt applications deployed in a variety of context gaining increasing power of autonomy and execution. It can be forseen that soon these models will have to be adaptive in the presence of other such models that are either competing or cooperative in nature.

Motivation:

- MARL
- Last iterate convergence
- NeuRD
- Extrapolation methods
- 2p0s game applications like GANs

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 background on the topics relevant to this work. We first discuss foundational concepts within reinforcement learning that serves as a base for the rest of the discussion. The framework of multiagent learning problems have also been conventionally rooted in Game Theory, and hence we cover some key ideas that help establishing this consistency. 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 is the mathematical study of interaction between agents to produce outcomes while trying to uphold certain individual or group preferences. It has a wide range of applications including economics, biology, and computer science. In overcoming the challenges mentioned above, many algorithms have adopted game-theoretic constrtucts and ideas when designing Reinforcement learning algorithms for multi-agent settings. It is also a common practice to use game theoretic constructs in evaluating the performance of multi-agent algorithms and provide theoretical guarantees. In this work, we mainly focus on a branch of game theory called non-cooperative game theory in which each agent has their own indivudual preference.

2.1.1 Problem Representations

In discussing agent interactions, and preferences, we need a formal notion of how agents act, and how agent preferences can be defined. In game theory, agent preferences are formalized using Utility theory, where each agent has a utility function that maps the agents preferences over outcomes to a real value.

The primary way of modeling problems in game theory is through a *game* that encodes information about the agents, possible actions agents can take in different situations, their preferences, and the outcome of an interaction. There are many types of such representations, a few relevant of which we introduce below. Before we introduce such representations, we first cover some preliminary concepts that will be helpful in formally defining those representations and agent preferences.

- what are utilities, and strategies?

Normal-Form Games: Normal-Form games are a popular way of representing situations in which all agents act simultaneously and the outcome is revealed after each agent has taken their action. A few popular games that can be represented in this form are rock-paper-scissors, matching pennies, prisoner's dilemma etc. A more formal definition of a normal-form game is as follows:

Definition 1 (Normal-form games) *A (N, A, u) tuple is a n-player normal form game, where N is the set of players, $A = A_1 \times A_2 \ldots \times A_n$, with A_i being the set of actions available to player i, and $u = (u_i \forall i \in N)$ is the set of utility functions that map an action profile to a real utility value for each agent, $u_i : A \mapsto \mathbb{R}$.*

Normal-form games are typically represented using a n-dimensional tensor, where each dimension represents the possible actions available to each agent, and every entry represents an outcome. The actual entries of the

An NFG that is widely popular in the literature is the **Biased/Perturbed RPS** problem. add details.

Sequential Games

Although normal-form games provide a neat representation, many real-world scenarios necessitate agents act sequentially which is difficult to represent as a matrix. These problems require a tree-like representation where each node is an agent's turn to make a choice, and each edge is a possible action. There are a few ways to represent such scenarios, one being normal-form games themselves. A downside is that the size of the normal-form representation for sequential games explode exponentially in the size of the game tree. Other possible representations include the Extensive-form, and Sequence-form.

Definition 2 (Extensive-form games)

Definition 3 (Sequence form games)

2.1.2 Solution Concepts

Now that we have - what are solution concepts? - what are the common solution concepts? Nash equilibrium, Quantal response equilibrium. - what are the relevant information related to solution conepts for this work? Existence of a nash equilibrium Uniqueness of QRE

2.2 Reinforcement Learning

Reinforcement learning (RL) is sub-domain of machine learning that deals with designing interactive *agents* that 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 acheived. Reinforcement learning has been shown to be effective in many application domains to learn and solve an arbitrary problem as long as it can be encapsulated into an interactive environment with a well-defined reward signal. - cite RL applications

Markov Decision Process. Reinforcement learning problems are formally modeled as Markov Decision Processes (MDPs), usually represented as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu)$ where: $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0,1]$ is the transition or the dynamics function, $R(s,a) \subset \mathbb{R}$ is the reward function, $\gamma \in [0,1]$ is the discount factor and μ is the initial state distribution.

The objective is to maximize the expected discounted reward:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) \tag{2.1}$$

In learning to take actions that maximize this objective, algorithms usually involve learning value functions and policies. A policy is a mapping from a state to an action distribution $\pi : \mathcal{S} \mapsto \mathcal{A}$, and a value function, $V_\pi(s)$ estimates the expected reward that can be achieved from the current state under the policy π: $V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$.

Now, the problem of learning a policy that maximizes the objective Equation 2.1 can stated in terms of the value function as follows.

Definition 4 (Reinforcement Learning (RL)) *Given an MDP:* $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu)$*, the Reinforcement learning problem is to find an optimal policy mapping* $\pi^* : \mathcal{S} \mapsto \mathcal{A}$*, such that:*

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s_0 \sim \mu}[V_\pi(s_0)]$$

There maybe more than one optimal policy, but they all share the same value function $V_{\pi^*} = \max_\pi V_\pi(s), \forall s \in \mathcal{S}$.

Apart from the value function, a 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. The objective can also framed in terms of these alternate value functions.

Generalized Policy Iteration (GPI). For small, finite state spaces optimal policies can be learnt through tabular methods and dynamic programming. Policy iteration is an iterative method to learn the optimal policy. The process of a policy iteration step involves both policy evaluation, and policy improvement. Policy evaluation is done to estimate the value function for the current policy using the following iterative update until it converges.

$$V_{k+1}(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V_k(S_t + 1 | S_t = s)]$$

Policy improvement then uses this estimated value function to greedily improve the current policy, $\pi_{k+1}(s) = \arg\max_{\pi(s)} V_{k+1}(s)$. Policy improvement always yields a strictly better policy except when the policy is already optimal [3]. Requiring exact convergence of policy evaluation step might make this learning process slower. However, this step can be truncated to only evaluate the value for the immediate states (i.e. perform only one step of policy evaluation). This is known as **value iteration**. In a more general algorithm called the *Generalized policy iteration (GPI)*, these steps are run independently, and asynchronously until convergence.

For very large state spaces that is common in many practical applications, there is a need to learn approximate value functions and parameterized policies. This is typically done through function approximation.

- Bellman eqn
-

Fixed points of the Bellman quation are optimal policies and optimal value functions. For small MDPs that strictly satisfy the Markov property, Bellman equiation can be explicity solved. In settings where the transitions and rewards can be represented explicitly in a tabular manner, dynamic programming is one approach to solve the Bellman equation. However, for most practical applications it is common to use parameterized policies and approximate value functions.

2.2.1 Multi-agent Reinforcement Learning (MARL)

Reinforcement learning algorithms have also been studied in settings that have more than one agent where the agents try to maximize some objective in a cooperative or competitive manner. Athough many of these algorithms have exhibited strong empirical performance in multi-agent settings, there are a few key challenges when extending Reinforcement learning algorithms to multi-agent settings. From a theoretical perspective, the foundational assumption for RL algorithms is that the MDP's transition dynamics remain stationary. However, in multi-agent settings non-stationary is induced due to the presence of the other agents. This makes it difficult in adapting the algorithms disucssed above directly to multi-agent settings. From an algorithm design perspective, the action space explodes exponentially in multi-agent settings making it computationally challenging to apply reinforcement algorithms without decomposing the problem into more managable sub-problems first.

2.3 Online Learning and Mirror Descent

Online learning is the study of designing algorithms that use historical knowledge in making predictions for future rounds while trying to minimize some loss function in an adaptive (possibly adversarial) setting.

The problem of training agents in a potentially non-stationary setting can be cast into an online learning problem. Hence, it is useful to study online learning algorithms from the perspective of designing reinforcement learning algorithms as they provide a framework for the analysis of the algorithms and deriving theoretical guarantees. The brief background provided in this section closely follows the details as presented in [4]. For a more in-depth introduction into Online learning and Online Convex Optimization, please refer to the above work.

In Online Learning, a learner is tasked with predicting the answer to a set of questions over a sequence of consecutive rounds. We now define an Online learning problem more formally as follows:

Definition 5 *For each round t, given an instance $x_t \in \mathcal{X}$, and a prediction $p_t \in \mathcal{Y}$, a loss function $l(p_t, y_t) \mapsto \mathbb{R}$*

At each round t, a question x_t is taken from an instance domain $\mathcal{X}$, and the learner is required to predict an answer, p_t to this question. After the prediction is made, the correct answer y_t, from a target domain $\mathcal{Y}$ is revealed and the learner suffers a loss $l(p_t, y_t)$. The prediction p_t could belong to $\mathcal{Y}$ or a larger set, $\mathcal{D}$.

The main aim of an online learning algorithm A, is to minimize the cumulative regret of a learner with respect to the best competing hypothesis h^* from the assumed hypothesis class $\mathcal{H}$.

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

The regret of A with $\mathcal{H}$ is,

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

A popular framework for studying and designing Online learning algorithms is through Convex optimization. The assumptions made around the framework provides properties that are useful in deriving convergence guarantees. We define a few terms below that are used in this section and the following ones.

Definition 6 (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 $L - smooth$.*

The typical structure of online learning expressed as an online convex optimization problem is as follows:

```
AlgorithmOCO
input: a convex set S for t = 1, 2, ...
predict a vector $w_t \in S$
receive a convex loss function $f_t : S \mapsto \mathbb{R}$
```

Reframing Equation 2.2 in terms of convex optimization, we refer to a competing hypothesis here as some vector u from the convex set S.

$$Regret_T(u) = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(u) \tag{2.4}$$

and similarly, the regret with respect to a set of competing vectors U is,

$$Regret_T(U) = \max_{u \in U} Regret_T(u) \tag{2.5}$$

The set U can be same as S or different in other cases. Here we assume $U = S$ and $S = \mathbb{R}$ unless specified otherwise.

2.3.1 FoReL

Follow-the-Regularized-leader (FoReL) is a classic online learning algorithm that acts as a base in deriving various regret mimization algorithms. The idea of FoReL is to include a regularization term to stabilize the updates in each iteration leading to better convergence behaviors.

The learning rule can be written as,

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

where $R(w)$ is the regularization term. The choice of the regularization function lead to different algorithms with varying regret bounds.

2.3.2 Gradient Descent

In the case of linear loss functions with respect to some z_t, i.e., $f_t(w) = \langle w, z_t \rangle$, and $S = \mathbb{R}^d$, if FoReL is run with l_2-norm regularization $R(w) = \frac{1}{2\eta}\|w\|_2^2$, then the learning rule can be written as,

$$w_{t+1} = -\eta \sum_{i=1}^{t} z_i = w_t - \eta z_t \tag{2.6}$$

Since, $\nabla f_t(w_t) = z_t$, this can also be written as, $w_{t+1} = w_t - \eta \nabla f_t(w_t)$. This update rule is also commonly known as Online Gradient Descent. The regret of FoReL run on Online linear optimization with a euclidean-norm regularizer is:

$$Regret_T(U) \leq BL\sqrt{2T}.$$

where $U = u : \|u\| \leq B$ and $\frac{1}{T}\sum_{t=1}^{T} \|z_t\|_2^2 \leq L^2$ with $\eta = \frac{B}{L\sqrt{2T}}$.

Beyond Euclidean regularization, FoReL can also be run with other regularization functions and yield similar regret bounds given that the regularization functions are strongly convex.

Definition 7 *For any σ-strongly-convex function $f : S \mapsto \mathbb{R}$ with respect to a norm $\|.\|$, for any $w \in S$,*

$$\forall z \in \partial f(w), \forall u \in S, f(u) \geq f(w) + \langle z, u - w\rangle + \frac{\sigma}{2}\|u - w\|^2. \tag{2.7}$$

Lemma 1 *For a FoReL algorithm producing a sequence of vectors $w_1, \ldots, w_T$ with a sequence of loss functions $f_1, \ldots, f_T$, for all $u \in S$,*

$$\sum_{t=1}^{T}(f_t(w_t) - f_t(u)) \leq R(u) - R(w_1) + \sum_{t=1}^{T}(f_t(w_t) - f_t(w_{t+1}))$$

2.3.3 Hedge

- what is hedge?

Gradient descent as a FTRL variant

2.4 Mirror Descent

In this section we discuss about Mirror Descent, and two mirror descent-based reinforcement learning algorithms (MDPO, and MMD). Mirror descent is a popular first order optimization algorithm that has seen wide applications in machine learning, and reinforcement learning.

There are different views of arriving at Mirror Descent as an optimization algorithm, here we present the Mirror Descent framework through the lens of mirror maps.

2.4.1 Mirror Maps

Definition 8 *Given a convex set $\mathcal{X} \subset \mathbb{R}^n$, and a differentiable convex function $f : \mathcal{X} \mapsto \mathbb{R}$, the Bregman Divergence associated with the function f is defined as,*

$$D_f(x, y) = f(x) - f(y) - \nabla f(y)^T (x - y).$$

For an alternate view of deriving Mirror Descent as an improvement of FoReL, please refer to [4][Section 2.6].

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 PG methods in RL to set a base for the rest of the chapter before moving on to the above algorithms.

3.1 Policy Gradients

3.1.1 Policy gradient methods

Policy gradient method involve directly learning a parameterized policy that enables action selection without the use of a value function. These are generally called Policy gradient methods, and are a major area of study within Reinforcement learning. Policy gradient methods sometimes have the advantage that the policy space could be simpler to learn compared to the value function space. In this case, the policy is parameterized by $\theta \in \Theta$ and $\pi(s, a) = P(s, a; \theta)$.

Given some objective $J(\theta)$, these methods seek to learn the parameters that maximize this objective through gradient ascent.

One key challenge in Policy gradient methods is that the evaluation of performance of a policy depends on the state distribution which could be unknown. However, the Policy Gradient Theorem establishes that the gradient is independent of underlying environment's state distribution as long as it is stationary conditioned on the current policy. The most popular policy gradient method is Reinforce uses monte-carlo estimations to approximate the value estimation. A value function may still be used in guiding the policy learning, and these are called actor-critic methods. Here the actor refers to the policy, and the critic refers to the value function that evaluates the action taking by the policy guiding the policy learning.

Reinforce: - Widely popular PG algorithm, many insantiations possible including variants with baselines to reduce variance. - Convergence is proven using PG theorem.

Softmax Policy Gradients - Parametrizing policies - PG with softmax parameterization is referred to as Softmax Policy gradients. - General significance of softmax

3.1.2 PPO

- Trust region methods - PPO an approximation of TRPO with hueristic objective

3.2 Mirror Descent Policy Optimization

The first method we discuss is the Mirror Descent Policy Optimization [5] (MDPO). MDPO tackles the problem developing a stable reinforcement learning algorithm through the framework of trust-region optimization. Trust-region optimization stabilizes learning by constraining the policy update at each iteration. Instances of trust region optimization methods such as PPO [6], TRPO [7] and their variants have found large success in single agent RL with state-of-the-art performances in a wide array of tasks.

Due to the presence of a proximal regularization, mirror descent in itself can be interpreted as a trust-region optimization method. Building on top of existing theoretical guarantees for mirror descent in tabular RL settings, MDPO derives practical RL algorithms that is performant in function approximation settings. Using the formulation of a parameterized policy from 3.1.1, MDPO performs the following update at each iteration:

$$\pi_{k+1}(.|s) \leftarrow \arg\max_{\pi \in \Pi} \mathbb{E}_{a\sim\pi}[A_{\pi_k}(s,a)] - \frac{1}{t_k} KL(s;\pi,\pi_k) \tag{3.1}$$

This update is approximately solved at each iteration by taking a few stochastic gradient steps. The gradient of the KL component of the above objective for one step of SGD is zero, and hence it is necessary to take multiple gradient steps every iteration to correctly approximate the mirror descent objective. Tomar et.al, derive on-policy and off-policy variants of MDPO, and in this work we mainly focus on the on-policy algorithm.

For a parameterized policy π_θ the on-policy MDPO update is given by:

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

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

For m steps MDPO uses the following gradient to update the policy parameters,

$$\nabla_\theta J(\theta,\theta_k)|_{\theta=\theta_k^{(i)}} = \mathbb{E}_{s\sim\rho_{\theta_k} a\sim\theta_k}[\frac{\pi^{(i)}_{\theta_k}}{\pi_{\theta_k}} \nabla_\theta \log \pi^{(i)}_{\theta_k}(a|s) A_{\theta_k}(s,a)] - \frac{1}{t_k}\mathbb{E}_{s\sim\rho_{\theta_k}}[\nabla_\theta KL(s;\pi_\theta,\pi_{\theta_k})_{|\theta=\theta_k^{(i)}}] \tag{3.2}$$

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

MDPO strong connections to the other constrained optimization RL algorithms and entropy regularized algorithms. We refer to [5] for a detailed discussion of connections between on-policy MDPO to PPO and TRPO, and off-policy MDPO to SAC.

3.2.1 MDPO in MARL

Although MDPO exhibits strong empirical performance in single-agent RL, it is not directly extendable to multi-agent settings due to the non-stationarity of the dynamics.

We demonstrate the behavior of MDPO using the Perturbed RPS problem discussed in Section 2.1.1. Normal form games allow for tabular policy representations with direct or softmax parameterizations. Algorithms can typically use exact value estimations for all possible actions (payoffs) given the opponent policy, as opposed to a sample-based expected value estimations. This is 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 3.2 becomes:

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

From the trajectories in Fig add trajectory plots we can see that MDPO fails to converge to a Nash equilibrium (marked by a red dot). In the next chapter we discuss ways to adapt MDPO for last-iterate convergence in two-player zero-sum games.

3.3 Magnetic Mirror Descent

Another extension of Mirror Descent to reinforcement learning is Magnetic Mirror Descent [8], that attempts to create a unified approach to work well in both single and multiagent settings. This work studies the relation between equilibrium solving and Variational inequalities with composite structures. Taking advantage of this connection, a equilibrium solving algorithm has linear last-iterate convergence guarantees is proposed. Moreover, this approach extends well as a reinforcement learning algorithm in single agent, and multiagent settings. In this section, first we outline the connection between Varational inequalities and equlibrium solving as presented in [8]. 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 9 *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 [9]. 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 following entropy-regularized saddle point problem. Given $\mathcal{X} \subseteq \mathbb{R}^n$, $calY \subseteq \mathbb{R}^m$, and $g_1 : \mathbb{R}^n \mapsto \mathbb{R}$, $g_2 : \mathbb{R}^m \mapsto \mathbb{R}$, find:

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \alpha g_1(x) + f(x, y) + \alpha g_2(y), \tag{3.3}$$

The solution (x_*, y_*) to the saddle point problem Equation 3.3 has the following first-order optimality conditions:

$$\begin{aligned} \langle \alpha \nabla g_1(x_*) + \nabla_{x_*} f(x_*, y_*), x - x_* \rangle \geq 0, \forall x \in \mathcal{X}. \\ \langle \alpha \nabla g_2(y_*) + \nabla_{y_*} f(x_*, y_*), y - y_* \rangle \geq 0, \forall y \in \mathcal{Y}. \end{aligned} \tag{3.4}$$

The reguarlized saddle point problem Equation 3.3 of solving for QREs is equivalent to the VI problem 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 $\nabla g = [\nabla_y g_1(x), \nabla_x g_2(y)]$. The optimality conditions Equation 3.4 are equivalent to the VI$(\mathcal{Z}, G)$, and thus the solution to the VI: $z^* = (x^*, y^*)$ is also the solution to the saddle point problem stated in Equation 3.3.

3.3.2 MMD Algorithm

Various algorithms have been proposed to solve the VI problem 9. In particular, the proximal point method has linear last iterate convergence for Variational inequality problems with a monotone operator [10]. This algorithm was extended to composite objectives [11], and to non-euclidean spaces with Bergman divergence as a proximity measure, that allows for non-euclidean proximal regularization [12].

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:

$$z_{t+1} = \arg\min_{z \in \mathcal{Z}} \eta(\langle F(z_t), z\rangle + \alpha g(z)) + B_\psi(z; z_t). \tag{3.5}$$

where 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 3.5 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.

We now restate the main algorithm as stated in Sokota et.al, [8],

AlgorithmMMD [8, (Algorithm 3.6)] Starting with $z_1 \in$ int dom $\psi \cap \mathcal{Z}$, at each iteration t do

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

Algorithm 2 provides the following convergence guarantees.

Theorem 1 *[8, 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 from Algorithm 2 admits closed form in some instances, while requires approximation through gradient updates in other cases. For a parameterized policy pi_θ, when ψ is negative entropy the update rule from Algorithm 2 can be restated in RL terms as follows:

$$\pi_{\theta_{k+1}}(s,a) = \arg\max_{\theta} \mathbb{E}_{s\sim\rho_{\theta_k}} \left[\mathbb{E}_{a\sim\pi_{\theta_k}}[Q_{\theta_k}(s,a)] + \alpha H(\pi_\theta) - \frac{1}{\eta} KL(\pi_\theta, \pi_{\theta_k}) \right], \tag{3.6}$$

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,

$$\nabla_\theta J(\theta,\theta_k)|_{\theta=\theta_k^{(i)}} = \mathbb{E}_{s\sim\rho_{\theta_k} a\sim\theta_k}[\frac{\pi_{\theta_k}^{(i)}}{\pi_{\theta_k}} \nabla_\theta \log \pi_{\theta_k}^{(i)}(a|s) A_{\theta_k}(s,a)] + \alpha H(\pi_{\theta_k}^{(i)}) - \frac{1}{\eta} \mathbb{E}_{s\sim\rho_{\theta_k}}[\nabla_\theta KL(s;\pi_\theta,\pi_{\theta_k})_{|\theta=\theta_k^{(i)}}] \tag{3.7}$$

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

To examine the effect of the choice of m, we compare the performance of the behavioral form update from Equation 3.6 to the closed-form for MMD with a negative entropy mirror map [8, equation (12)]:

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

For a two-player zero-sum normal form game, the expected Q-values are the expected payoffs for the max and min players respectively. As there is only one state, we can replace the expectations in the update with an all-action Q-value computation that is only dependent upon the opponent's policy.

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 $m = 10$.

3.3.4 Comparison of MMD to other Mirror Decent based methods

The non-euclidean proximal gradient method Equation 3.5, has strong connections to Mirror Descent [8, Appendix D.3]. TBD: other works also discuss this relationship between mirror descent and proximal gradient methods applied to VIs. Consequently negative entropy based MMD is also equivalent to MDPO with an added entropy regularization as detailed in [8, Appendix L] and as can be seen from Equation 3.1 and Equation 3.6.

MMD as a reinforcement learning algorithm performs on par with CFR. In single agent settings MMD's performance is competitive with PPO in Atari and MuJoCo environments.

CHAPTER 4

MODIFIED UPDATES FOR MIRROR-DESCENT BASED METHODS

Our main contribution is adapting existing techniques for inducing convergence in multiagent settings with the algorithms discussed in the previous section. More specifically, we study the following three techniques - Neural Replicator Dynamics [13], the Extragradient algorithm [14], and the Optimistic gradient algorithm [15]. In this section we describe these techniques in more detail and provide our proposed modifications to the mirror descent algorithms.

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 [13, 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.

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 the single-state setting.

Motivated by this observation, Hennes et.al [13] 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. Replicator dynamics also have a strong connection to no-regret algorithms, and in this work, the authors also establish the equivalence between single-state all actions tabular NeuRD, Hedge, and discrete time RD [13, Statement 1, p5]. Due to this equivalence, NeuRD also inherits the property of no-regret algorithms that their time-average policy converges to the Nash equilibrium.

Neural Replicator Dynamics (NeuRD) is an adaptation of discrete-time Replicator Dynamics to reinforcement learning under function approximation settings. NeuRD can be implemented as a one line change to SPG, bypassing the gradients with respect to the softmax operator and computing the gradients of the parameters directly with respect to the logits.

Since most RL algorithms, including the ones discussed in the previous chapter utilize Softmax parameterization, the NeuRD fix is directly applicable to the policy loss component of their respective loss functions without any major changes. In our work, we apply the NeuRD fix to the policy loss component of the MMD and MDPO loss functions.

4.1.1 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.

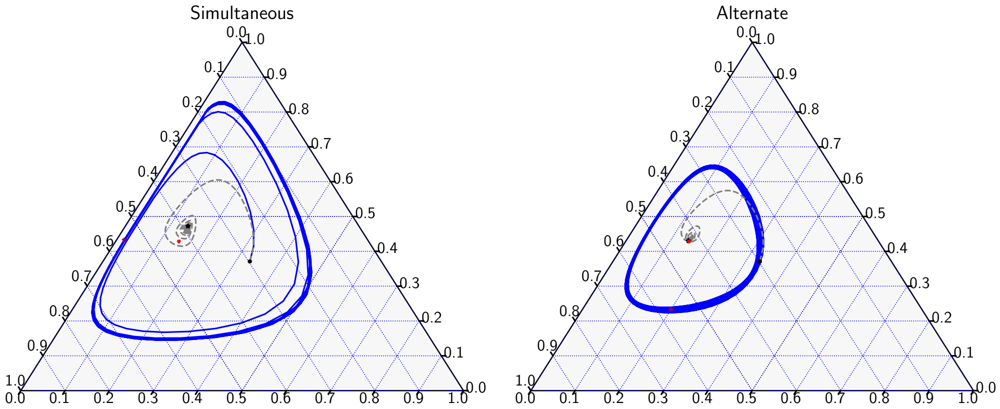


Figure 1: NeuRD converges with alternating updates, but not with simultaneous updates.

We note that [13] also use alternating updates in all their experiments evaluating NeuRD. A similar behavior of simultaneous gradient updates diverging was also observed in [16] for unconstrained min-max problems. As noted in the earlier work, the sequence of iterates have bounded error, and thus the average policy converges to the optimal point. 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 update in a similar manner. Asymmetric updates have also been studied in a similar context and has been shown to improve the speed of convergence [17].

4.2 Extragradient and Optimistic Gradient

First-order optimization algorithms are studied extensively in varying contexts including solving variational inequalities, saddle-point problems, and convex optimization. While gradient based methods including the classic Arrow-Hurwicz method work well for many optimization problems, for saddle point problems they converge only under strong-convexity even in the unconstrained case [18].

The **Extragradients (EG)**: G.M.Korpelevich [14] introduced EG as a modification of 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:

$$\begin{aligned}
\bar{x}_k &= P_{\mathcal{X}}[x_k - \eta f(x_k, y_k)] \\
\bar{y}_k &= P_{\mathcal{Y}}[y_k + \eta f(x_k, y_k)] \\
x_{k+1} &= P_{\mathcal{X}}[x_k - \eta f(\bar{x}_k, \bar{y}_k)] \\
y_{k+1} &= P_{\mathcal{Y}}[x_k + \eta f(\bar{x}_k, \bar{y}_k)]
\end{aligned} \tag{4.1}$$

If f is L-smooth, and $0 < \eta < 1/L$ is the step size, EG has asymptoticlast-iterate convergence. [11] - Last iterate linear convergence using extragradient method for VI problems.

Optimistic gradients (OPT): For the same problem [15] also proposed a modification to the Arrow-Hurwicz method that differs from the EG method by reusing the *extragradients* of previous iteration instead of computing an additional gradient in each iteration. This form of update, popularly known as **Optimistic updates** can be expressed through the following sequence:

$$\begin{aligned} \bar{x}_k &= P_{\mathcal{X}}[x_k - \eta f(\bar{x}_{k-1}, \bar{y}_{k-1})] \\ \bar{y}_k &= P_{\mathcal{Y}}[y_k + \eta f(\bar{x}_{k-1}, \bar{y}_{k-1})] \\ x_{k+1} &= P_{\mathcal{X}}[x_k - \eta f(\bar{x}_k, \bar{y}_k)] \\ y_{k+1} &= P_{\mathcal{Y}}[x_k + \eta f(\bar{x}_k, \bar{y}_k)] \end{aligned} \tag{4.2}$$

Variations of the above algorithms have been studied throughout the literature. Most notably, Optimistic Gradient Descent Ascent and Optimistic Multiplicative Weights Update are well-established algorithms that use Optimistic udpates.

[19] studied this algorithm in the context of Online learning with predictable sequences under the name Optimistic Mirror Descent. Optimistic updates coincides with the idea of using a predictor for the performance of next iteration to guide the updates in Online learning []. While this assumption is valid for the problem of learning with predictable sequences, it is also valid in multiagent settings if the objective function is convex and both players use regularized learning algorithms. Within this work, OMD was also studied in the context of zero-sum games with strongly uncoupled dynamics. This work showed a convergence rate of $O(logT/T)$ for all-actions setting and hypothesized that a better rate than $O(1/\sqrt{T})$ is not possible for the sample based setting. Optimistic Mirror Descent (OMD) is an extragradient version of Mirror

Descent, and OGDA can be shown to be an instantiation of OMD that uses *extrapolation from past*. OMD updates are given by:

$$\begin{aligned} x_{k+1} &= P[x_k - 2\eta f(x_k, y_k) + \eta f(x_{k-1}, y_{k-1})] \\ y_{k+1} &= P[x_k + 2\eta f(x_k, y_k) - \eta f(x_{k-1}, y_{k-1})] \end{aligned} \tag{4.3}$$

[17] studied OMD to tackle the problem of instability in training GANs. The also prove last-iterate convergence for OMD in bilinear games and detail an optimistic version of the Adam optimizer. OGDA has linear last-iterate convergence [17].

OMD was also studied in [16] for the problem of convergence in GANs under the framework of Variational Inequalities, dubbed *extrapolation from the past.* Here, extrapolation refers to the EG algorithm, and Optimism is interpreted as EG with the extrapolation done using the past iterations gradient as an estimate of the extragradient. They also prove convergence for stochastic VI problems with strong monotone operators for the average of the iterates. [20] 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.

Optimism is often interpreted as an adaptation of the Extragradient method with the idea to avoid the additional call to a gradient oracle by estimating the extrapolation using the previous iteration's gradient. Extragradient and Optimistic Gradient Descent Ascent methods have been shown to be approximations of proximal-point method for solving saddle point methods [21].

In our work, we study the effects of adapting extragradient and optimistic updates into the algorithms discussed in the previous chapter for 2p0s 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?
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.

Perturbed RPS

Matching Pennies

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.

5.2.3 Experiment Setup

The policies that are logit-parameterized with a softmax projection and all the algorithms use full-feedback gradient updates as discussed in Chapter 3. Being symmetric matrix games, both games have a unique Nash Equilibrium: Perturbed RPS $(\frac{1}{7}, \frac{3}{7}, \frac{3}{7})$, Matching Pennies $(\frac{1}{2}, \frac{1}{2})$. 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 schedule is motivated by mirror-descent theory, while MMD's schedule is closer to the type of annealing schedule used for the original MMD experiments found through a hyperparameter sweep. So for both of these methods we cap the KL-coefficient at 0.2 because very low values destabilize the updates, especially for the NeuRD version of the algorithms.

5.3 Results

Table I summarizes the last and average iterate convergences behaviors of 3 different algorithms - MMD, MDPO, SPG under the proposed modifications. Fig Figure 2 summarizes the results for convergence to the Nash Equilibrium.

Algorithm	**PG**		**MDPO**		**MMD**	
	Avg	**Last**	**Avg**	**Last**	**Avg**	**Last**
Base	×	×	×	×	✓	✓
NeuRD	✓	×	✓	×	✓	✓
EG	✓	✓	✓	✓	✓	✓
OPT	✓	✓	✓	✓	✓	✓
EG-OPT	✓	✓	✓	✓	✓	✓
EG-N	✓	✓	✓	✓	✓	✓
OPT-N	✓	✓	✓	✓	✓	✓
EG-OPT-N	✓	✓	✓	✓	✓	✓

TABLE I: Convergence in Perturbed RPS for last, and average iterates.

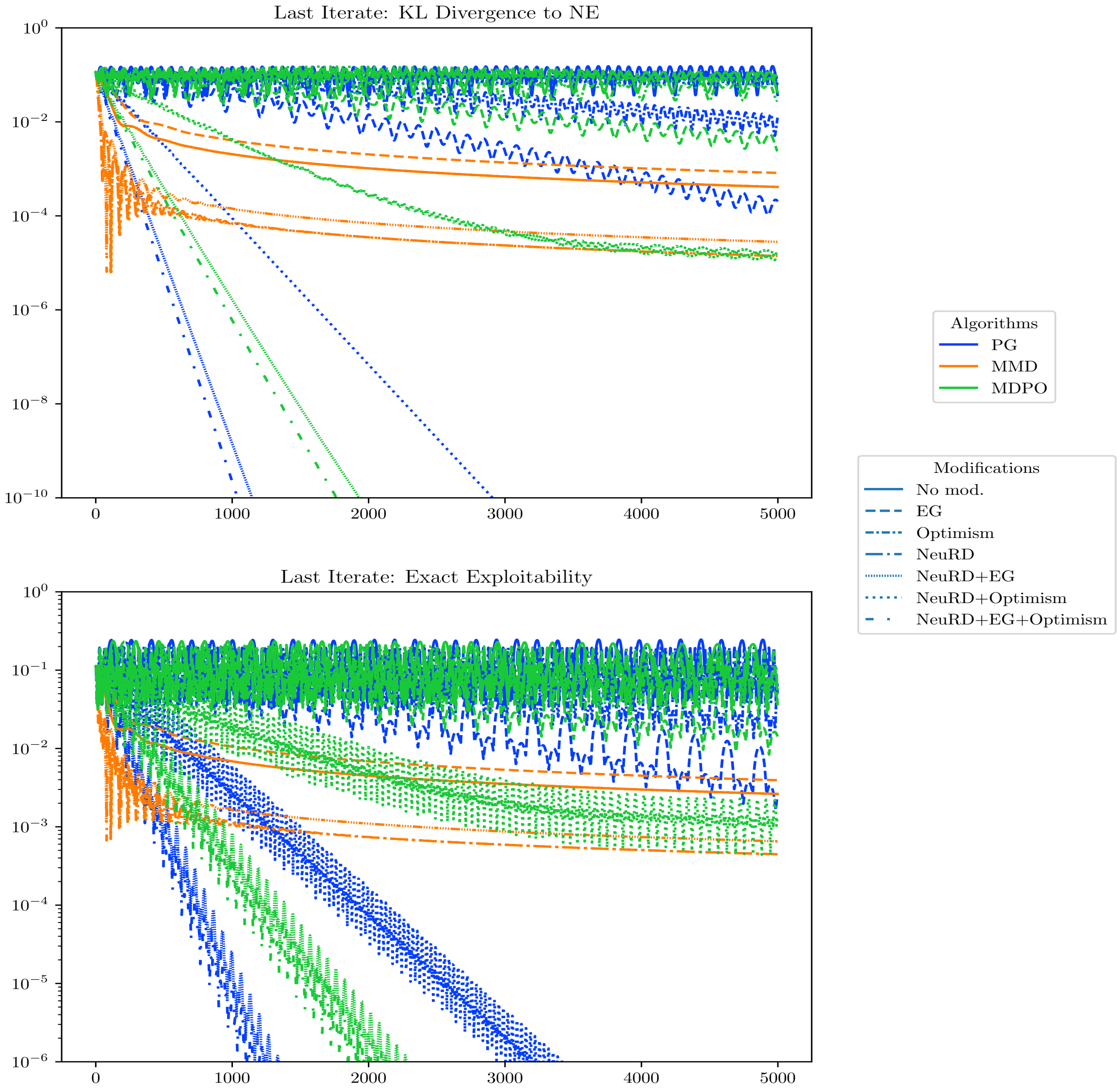


Figure 2: Last-iterate converegence plots in PerturbedRPS.

5.3.1 Observations

From the above experimental results, we note the following:

- NeuRD-fix speeds up convergence in all the algorithms with or without EG and Optimism.
- Most notably, Optimistic-NeuRD (PG-nrd-opt2) exhibits faster last-iterate convergence.
- MMD shows no performance improvements with the addition of Extragradient or Optimistic updates.
- In the presence of EG/Optimism to enable last-iterate convergence, trust-region constraints and entropy regularization slow-down the learning, as evidenced by the better performacne of SPG variants compared to the MDPO, and MMD variants.
- Combining Optimism with EG updates provides minimal improvements as they fulfill a similar role.

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

CHAPTER 6

DEEP MULTI-AGENT RL EXPERIMENTS

Expanding on our the observations from the tabular experiments, we proceed to evaluate the better performing algorithms under function approximation in large-scale 2p0s games to answer question 3. For the neural experiments, we study the effect of NeuRD addition, and the utilization of Optimistic-variants of popular optimizers (Optimistic-SGD, and Optimistic-Adam) when paired with deep RL algorithms. We do not evaluate extragradient updates for the Neural experiments.

6.1 Experimental Domains

For these experiments we choose three 2p0s EFGs, namely - Kuhn Poker, Abrupt Dark Hex, and Phantom Tic-tac-toe.

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)

We train all the algorithms through self-play reinforcement learning.

6.1.1 Approximate Exploitability

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 by training a local best response agent [22] against the fixed joint policy that was trained through self-play. Then the exploitability can be approximated by sampling trajectories and measuring the average reward the best response agent acheived against the exploited policy. Let π_{BR} be a learnt best-response approximator, and $\pi_{fixed} = (\pi_1, \pi_2)$ be the exploitee-joint policy, then the approximate exploitability is given by:

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

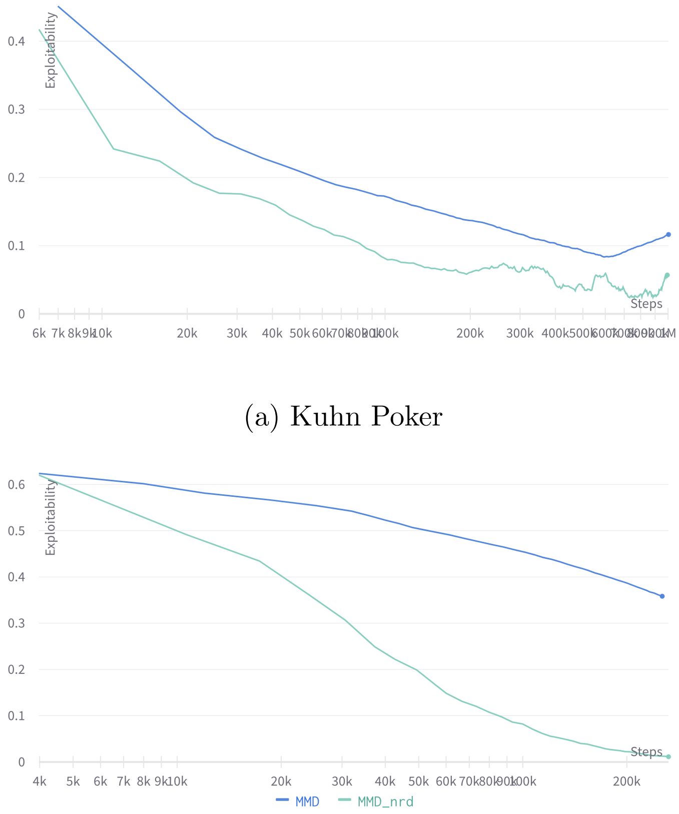
TBD: rephrase the equation in terms of cumulative episodic rewards.

6.1.2 Experiment Setup

Our experiment setup is akin to that of [8], we use RLLib [23] for implementing the algorithms, and interface with the game implementations in OpenSpiel using RLLib's environment adapter. For all the experiments we employ actor-critic version of the alogrithms without shared parameters for the policy and value networks. For all the experiments, we a use 2-layered MLP with (128, 128) hidden units. We use GAE for computing the advantage estimates. Please refer to the table **??** in the appendix for a more exhaustive list of hyperparameter values. For Kuhn Poker, and Abrupt Dark Hex 2×2, we train the agents in self-play for 1M steps and measure exact exploitability at each iteration. For Abrupt Dark Hex 3×3, and Phantom TTT we train the agents for 5M steps and compute approximate exploitability(Equation 6.1) of the final joint-policy at the end of training. We train a DQN agent against the fixed joint policy as the best-response appoximator π_{BR}, for 5M steps, and evaluate its performance over N=1000 episodes.

6.2 Results

Figure 3 plots the exact exploitabililty of the joint policy as a function of the iterations.

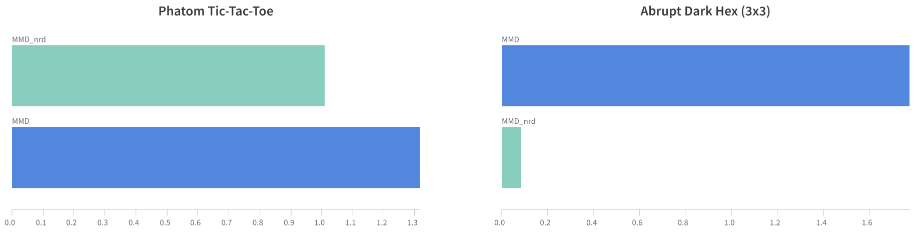


(a) Kuhn Poker

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

Figure 3: Performance in small EFGs, measured by exact exploitability.

Figure 4 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) Kuhn Poker

(b) Abrupt Dark Hex (3×3)

Figure 4: Performance in larger EFGs, measured by approximate exploitability.

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

6.2.1 Observations

- Similar to the tabular experiments, NeuRD-fix speeds up convergence of MMD in the Neural experiments as well.

CHAPTER 7

DISCUSSION

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. Chhablani et.al, [24] showed improved performance in identical-interest games by applying the NeuRD fix to COMA [25]. Perolat et.al, [26] used NeuRD loss to approximate Replicator dynamics as a part of the DeepNash algorithm. They used the same reward transformation based adaptive regularization [27] to induce last-iterate convergence in training agents for the game of Stratego. 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.

Entropy Regularization: As noted in the tabular experiments, and as observed from the performance of MDPO in the deep MARL experiments, there exist faster methods to induce last-iterate convergence that adaptive regularization.

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.

CHAPTER 8

CONCLUSION

APPENDICES

Appendix A

SOME ANCILLARY STUFF

Ancillary material should be put in appendices.

Appendix B

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. 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.

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

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

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

7. 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.

8. 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.

9. 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.

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

11. 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.

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

13. 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.

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

15. 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.

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

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

18. 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.

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

20. 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.

21. 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.

22. 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.

23. 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.

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

25. 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.

26. 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.

27. 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.