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

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

FIGURE **PAGE**

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

This work mainly discusses algorithms that are present in the intersection of three subject areas namely, Reinforcement learning, Game Theory, and Online Learning. We provide some brief background to relevant concepts required to follow the ideas discussed in the following sections and point to more comprehensive resources when applicable.

2.1 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 designer wants the agent to learn to achieve without informing anything about how that goal should be achevied. 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), that 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:

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

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 by following a given policy: $V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$.

The objective can then be reformulated as to find a policy that maximizes the value of the initial state under the starting distribution μ:

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

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

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 that employs policy evaluation, and improvement steps to learn the optimal policy.

The policy evaluation step learns 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 step then uses the value function to greedily improve the current policy, $\pi_{k+1}(s) = \arg\max_a Q_k(s, a)$. Policy improvement always yields a strictly better policy except when the policy is already optimal [3]. Exact convergence of policy evaluation might make the learning process slower. 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 *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.

In formulating Reinforcement learning algorithms, there are two major approaches, and their derivatives.

2.1.1 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, which is a Monte Carlo policy gradient method.

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

2.1.2 PPO

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

2.1.3 Multi-agent Reinforcement Learning (MARL)

There are a few key challenges in extending Reinforcement learning algorithms to multi-agent settings. From a theoretical perspective, many of the function approximation based algorithms assume that the state distribution $\mu(s)$ remains stationary for a given policy. However, in multi-agent settings the state distribution can become non-stationary due to the changes in the behavior 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.

2.2 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.

2.2.1 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 representa-

tion 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)

2.2.2 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

2.3 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 4 *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 $\mathcal{H}$.

$$Regret_T(h^*) = \sum_{t=1}^{T} l(p_t, y_t) - \sum_{t=1}^{T} l(h^*(x_t), y_t), \tag{2.1}$$

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

$$Regret_T(\mathcal{H}) = \max_{h^* \in \mathcal{H}} Regret_T(h^*) \tag{2.2}$$

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 5 (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 2.1 in terms of convex optimization, we refer to a competing hypothesis here as some vector u from the convex set S.

$$Regret_T(u) = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(u) \tag{2.3}$$

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

$$Regret_T(U) = \max_{u \in U} Regret_T(u) \tag{2.4}$$

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,

$$w_{t+1} = -\eta \sum_{i=1}^{t} z_i = w_t - \eta z_t \tag{2.5}$$

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

$$\forall z \in \partial f(w), \forall u \in S, f(u) \geq f(w) + \langle z, u - w\rangle + \frac{\sigma}{2}\|u - w\|^2. \tag{2.6}$$

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 7 *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

There have been a few recent efforts in adapting mirror descent from a constrained convex optimization algorithm into a reinforcement learning algorithm. In this work, we study two such algorithms, namely MDPO (Mirror Descent Policy Optimization), and MMD (Magnetic Mirror Descent). We first describe these algorithms and their connection to Mirror Descent in this section. In the next section we propose some modifications to be applied on top of these algorithms to improve their convergence behaviors in two-player zero-sum settings.

For the following discussion, and the experiments, we consider a class of parameterized stochastic policies, i.e., $\Pi = \{\pi_\theta, \theta \in \Theta\}$. We also only consider on-policy learning.

3.1 Mirror Descent Policy Optimization

The first method we discuss is the Mirror Descent Policy Optimization [5] (MDPO). MDPO deals with the problem of trust-region based policy learning. Trust-region methods aim to stabilize learning by constraining the policy update at each iteration. State-of-the-art RL algorithms like PPO [6], TRPO [7] are instances of trust region optimization methods.

Mirror Descent as such can also be interpreted as a trust-region optimization algorithm due to the presense of proximal regularization. MDPO adapts the mirror descent objective **??** into RL terms is as follows:

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

This objective is then approximately solved at each iteration through gradient ascent. We focus on the on-policy variant of MDPO where, for a parameterized policy π_θ the above update rule is reframed into:

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

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

The gradient of the KL component of the above objective for one step of SGD is zero. Hence, it is necessary to take multiple steps of stochastic gradient updates to approximate the objective in Equation 3.1.

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

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

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

On-policy MDPO has strong connections to TRPO, and PPO, and has been shown to have better empirical performance in continuous control tasks, and Atari environments [5].

3.1.1 MDPO in MARL

Although MDPO shows strong empirical performance in single-agent RL, it is not directly extendable to multi-agent settings. The main challenge here is the non-stationarity problem that is discussed in 2.1.3.

In this section, we derive the update rule for a tabular all-actions MDPO and apply it to the PerturbedRPS problem discussed in section to show the convergence behavior of MDPO in a multi-agent setting.

derivation tbd

Fig add trajectory plots shows the convergence behavior of MDPO in the Perturbed RPS problem. We can see that MDPO fails to converge to a Nash equilibrium (indicated by the red dot). This is due to the non-stationarity problem caused by the simultaneous updates of both agents. In this next chapter we discuss how to overcome this problem, and how to adapt MDPO for last-iterate convergence in two-player zero-sum games.

3.2 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.

3.2.1 Connection between Variational Inequalities and QREs

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

Definition 8 *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.

3.2.2 MMD Algorithm

Various algorithms have been proposed to solve the VI problem 8. 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 $z_1 \in \text{int dom } \psi \cap \mathcal{Z}$, 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 $\eta \leq \frac{\alpha}{L^2}$.

3.2.3 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:

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

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

3.2.4 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]. 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 in exploring existing techniques in multiagent learning and last-iterate convergence literature, and studying their effectiveness when combined with mirror-descent based 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 how they fit into the problem that we are trying to address. In the next section we perform a detailed experimental evaluation of these algorithms in two-player zero-sum games and summarize our findings.

4.1 Neural Replicator Dynamics (NeuRD)

Replicator Dynamics is an idea from Evolutionary game theory (EGT) that defines operators to update the dynamics of a population in order to maximize some pay-off defined by a fitness function. Neural Replicator Dynamics (NeuRD) is an extension of Replicator Dynamics to function approximation settings.

The single-population replicator dynamics is defined by the following system of differntial equations:

$$\dot{\pi}(a) = \pi(a)[u(a, \pi) - \bar{u}(\pi)], \forall a \in \mathcal{A} \tag{4.1}$$

Hennes et.al, [13] show equivalence between Softmax Policy gradients 2.1.1 and continuous-time Replicator Dynamics [13, THEOREM 1, on p5].

With knowledge of this equivalence they detail how NeuRD can be implemented as a one line change to SPG. This can be viewed as a fix to the regular SPG update to make it more suited to multi-agent settings by making the policy updates more responsive to changes in dynamics.

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 no-regret properties of algorithms like Hedge. While NeuRD has average iterate convergence, the authors also induce last iterate convergence in imperfect information settings for NeuRD using reward transformation based regularization [16].

e As seen in Fig.., tabular SPG does not converge to the Nash Equilibrium, whereas the average policy of tabular NeuRD converges to the Nash Equlibrium.

4.1.1 Alternating vs Simultaneous updates

We also observe that while alternating updates shows average-iterate convergence, simultaneous updates do not converge. More background about this

4.1.2 NeuRD fix in MMD, and MDPO

The NeuRD fix can be applied to MMD, and MDPO in a similar way to SPG.

4.2 Extrapolation methods

The Extragradient method (EG) was first introduced by G.M.Korpelevich [14] as a modification of gradient descent methods in solving saddle point problems. EG is a classical method for solving smooth and strongly convex-concave bilinear saddle point problems with a linear rate of convergence. Extragradient and Optimistic Gradient Descent Ascent methods have been shown to be approximations of proximal-point method for solving saddle point methods [17].

EG also has linear last-iterate convergence guarantees for variational inequality problems with a strongly monotone operator. This [11] - Last iterate linear convergence using extragradient method for VI problems.

4.2.1 MMD-EG

4.2.2 MDPO-EG

4.3 Optimism

CHAPTER 5

EXPERIMENTS

We now evaluate our proposed methods experimentally in both tabular and function approximation settings. Though these algorithms can be applied both as approximate equilibrium solvers QRE and Nash equilibirum. We present some results for convergence to QRE in the tabular setting but focus on Nash convergence mainly for the tabular and function approximation setting.

Through the experiments, we aim to answer the following questions:

- How does the addition of NeuRD-fix, Extragradient updates, and Optimistic upates affect the convergence rate of these algorithms in solving for QREs, and Nash equilibrium?
- What is the last-iterate vs average-iterate convergence behavior of these algorithms in the presence of these modifications?
- Do these performance improvements scale well with the size of the game?

5.1 Evaluation Metrics

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

5.1.1 Divergence to the equilibrium

In settings with a known unique equilibrium such as for QREs, or Nash equilibrium in symmetric markov games, 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.

5.1.2 Exploitability

In general, there might not be a unique Nash equilibrium. In such cases, another notion of distance from the equilibrium is needed to measure convergence. Exploitability measures the gain in value by deviating from the current strategy. This is measured by computing the utility of a best response agent against the current policy, as an indiciation of incentive to deviate from the current policy.

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

5.2 Tabular Experiments

For the tabular domain, we study the learning behavior of these algorithms on PerturbedRPS and Matching Pennies. PerturbedRPS, a modified version of the standard Rock Paper Scissors, is a symmetric Normal form game that has a unique Nash Equlibrium. Although it is a simple problem, standard RL algorithms fail to converge due to the non-stationarity induced by the opponent. We employ the standard model of alternating updates to learn a parameterized policy for both agents. We use softmax parameterization, and perform updates to learn the logits associated with each action. The logits then act as action preferences which are sampled from using the softmax function to get a stochastic policy.

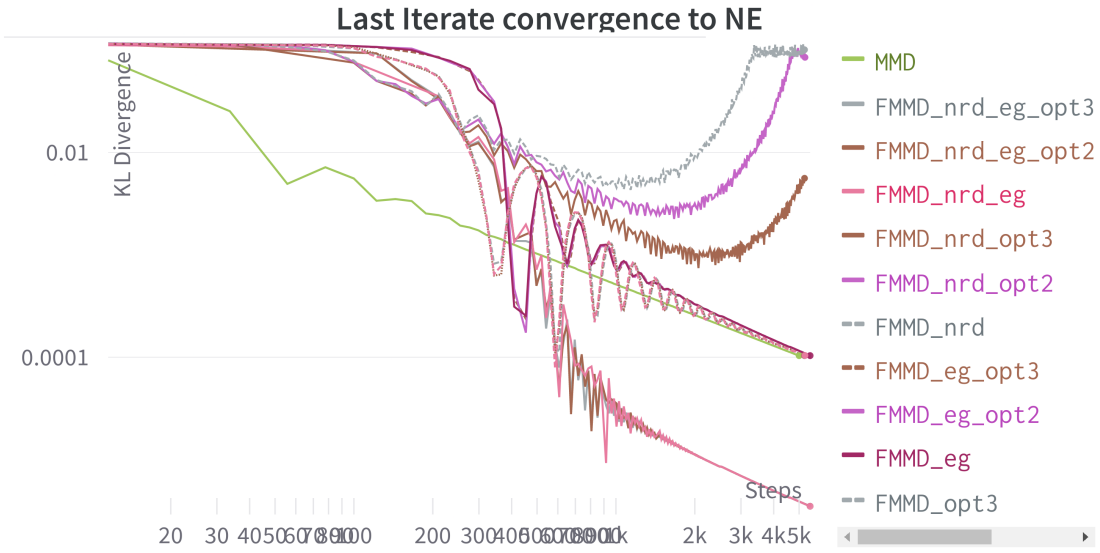
For PerturbedRPS, we use the QRE solutions computed through Gambit and, we also derive the unique Nash Equilibrium (please refer to **??** in the appendix for the derivation) for PerturbedRPS.

Table I summarizes the last and average iterate convergences behaviors of 3 different algorithms - MMD, MDPO, SPG under the proposed modifications. We present the results for NE, and QRE convergence (with 0.5 temperature). Fig

Algorithm	**Nash**				**QRE (α=0.5)**	
	MMD		**MDPO**		**MMD**	
	Avg	**Last**	**Avg**	**Last**	**Avg**	**Last**
Base	y	y	x	y	y	y
NeuRD	y	y	y	x	y	y
EG	y	y	y	y	y	y
OPT	y	y	y	n	y	y
EG-OPT	y	y	y	y	y	y
EG-N	y	y	y	y	y	y
OPT-N	y	y	y	n	y	y
EG-OPT-N	y	y	y	y	y	y

TABLE I

QRE and Nash convergence in Perturbed RPS for last, and average iterates.



- MDPO-EG has superlinear last iterate convergence to Nash equilibrium.
- MMD with NeuRD fix and extragradient

From the above experimental results, we note the following:

- MDPO with extragradient updates has superlinear convergence to Nash equilibrium
- Extragradient updates, and the NeuRD fix speedup convergence in both MMD and MDPO for convergence to both Nash equilibrium and QREs.

Based on the above findings, we test some of these variants in the function approximation setting for two-player zero-sum games with a large state space.

CHAPTER 6

NEURAL EXPERIMENTS

6.0.1 Approximate Exploitability

For larger games, it is not possible to compute the exact exploitability due to the large state space. However, we can approximate the exploitability by training a best response agent against the fixed current policy to be exploited. Then the exploitability can be approximated by sampling trajectories and measuring the average reward the best response agent acheived against the exploited policy.

6.0.2 Deep Multi-agent RL Experiments

Based on the observations from the Tabular NFG experiments, we evaluate the most promising combinations of these algorithms in the function approximation setting. As noted in [8], we implement MMD by modifying the PPO implementation in RLLib [18]. We also use RLLib's OpenSpiel adapter with some modifications to use information states as inputs as opposed to observations.

For the function approximation setting, we evaluate the algorithms on Kuhn Poker, Abrupt Dark Hex, and Phantom Tic-tac-toe. Kuhn Poker is a smaller extensive form game that allows for more introspection and exact exploitability computation. Whereas, Abrupt Dark Hex, and Phantom TTT are games with a large state-space that test the scalability of these algorithms.

In function approximation settings we compute exact exploitability for Kuhn Poker, and 2×2 Dark Hex. For the larger games, compute approximate exploitability by training a DQN best response agent to approximate the best response computation similar to [8].

We train these reinforcement learning agents i self-play for the environments mentioned above.

Implementation Details: - Implementation details (RLLib, PPO modifications, GAE) - Neural network architecture, hyperparameters

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. 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.

17. 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.

18. 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.