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.

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 begin by providing some brief background to familiarize the reader with the relevant concepts required to follow the ideas discussed in the following sections. We also point to more comprehensive resources for a more detailed discussion of the same.

2.1 Reinforcement Learning

Reinforcement learning 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 problems are formalized as Markov Decision Processes (MDPs) that are defined as follows MDP definition

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. examples

Reinforcement learning algorithms typically involve learning value functions and policies. A policy is the action plan followed by the agent that dictates how the agent acts given a

particular state. Formally, a policy maps a state to a probability distribution over the set of possible actions $\pi : S \mapsto A$. A value function estimates the worth (better word?) of the agent being in a given state while following a particular policy π, i.e., $v_\pi : S \mapsto \mathbb{R}$. The worth is typically determined by the expected return following the policy from that state.

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s]$$

Similarly, we can also specify the value of taking a specific action a at a given state, as opposed to the value of the entire state. These are typically known as action values or q-values

$$a_\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a]$$

A foundational property is that the value functions satisfy the Bellman equation:

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. However, for larger state spaces and settings that do not satisfy the assumptionsthere is a need to design algorithms that solve this equation approximately.

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. For most practical applications it is necessary to develop parameterized policies and approximate value functions.

In formulating Reinforcement learning algorithms, there are two major approaches, and their derivatives. One approach is to

Most of the tabular, and iterative methods require an explicit model of an environment with access to the transition functions. In the absence of a model or transition functions, one can use Monet Carlo methods to approximate these probabilities through sampling. This method of learning from experience by iteracting with a blackbox enivornment or a simulator is the setting in most of the recent efforts in machine learning. This is because it is difficult to explictly design a model that accurately captures all the properties of a real-world artifact, and exhaustively model the effects of taking all the actions that are available to the agent. It is also sometimes impossible or unknown. However, it is possible to have access to such a real-world artifact, and we only need to provide an interface for the agent to the artifact and expose available actions to the agent. We can use the real-world artifact to evaluate these actions and their effects and convert them into rewards to provide a signal for the agent to learn from.

2.1.1 Policy gradient methods

Apart from the value-based methods discussed above, another approach is to directly optimize a parameterized policy that maximizes the expected reward. A value function maybe used in learning the policy, but the agent always uses the policy for deciding on actions. These are generally called Policy gradient methods, and methods that use a value function in addition to learn the policy 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. Policy gradient methods generally have the advantage that the policy space could be simpler to learn compared to the value function space.

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.

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 - One way of projecting parametrized policy to a probability simplex is Softmax

PG with softmax parameterization is referred to as Softmax Policy gradients.

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

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)

2.2.2 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 Online Convex Optimization

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 [3]. 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.

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 5 *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

CHAPTER 3

MIRROR DESCENT IN REINFORCEMENT LEARNING

3.1 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.

3.1.1 Mirror Maps

Definition 6 *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,*

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

We can also define a few properties that the function f has based on the Bregman Divergence.

Definition 7 (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 an alternate view of deriving Mirror Descent as an improvement of FoReL, please refer to [3][Section 2.6].

There have been a few well-established efforts in adapting mirror descent from a general first-order optimization algorithm as described above into a reinforcement learning algorithm. In this work, we mainly consider two such algorithms, namely MDPO (Mirror Descent Policy Optimization), and MMD (Magnetic Mirror Descent). We provide a high-level description of the approach taken in arriving at these algorithms and how they are connected to Mirror Descent before applying modifications on top of these algorithms in an effort to improve convergence behaviors of these algorithms in two-player zero-sum settings.

3.1.2 Mirror Descent Policy Optimization

The first of these algorithms Mirror Descent Policy Optimization [4] addresses the problem of trust-region solving to stabilize reinforcement learning updates. Reinforcement learning algorithms, especially Policy gradient methods are notoriously known for having high-variance updates causing troubles in converging to the optimal policy cite. One approach to tackle this problem from an optimization perspective is to ensure that the policy does not change drastically with each gradient update to stabilize the learning. This is popularly known as trust region optimization. PPO, TRPO are popular single agent reinforcement learning algorithms that incorporate trust region optimization into . . . outline PPO, and TRPO. [5]

While MDPO differs from the above approaches in that they arrive at a similar algorithm from the perspective of Mirror Descent, they also show strong connections to PPO, TRPO, and SAC.

3.1.3 Mirror Descent as an RL algorithm

RL objective is not convex; can still use MD.. [6]

$$\pi_{k+1} < -\arg\max_{\pi\in\Pi} \mathbb{E}_{s\sim\rho_{\pi_k}}$$

Gradient of KL-Divergence

Let's consider a reference policy Q and P_θ, a policy that is being optimized. The reverse KL-Divergence between the two is given by

$$D_{KL}(P_\theta \| Q) = \sum_{a\in A} P_\theta(a) \log \frac{P_\theta(a)}{Q(a)}$$

The gradient of KL-Divergence with respect to the parameters theta is given by:

$$\begin{aligned}
\nabla_\theta D_{KL} &= \sum_{a\in A} \nabla_\theta [P_\theta(a) \log P_\theta(a)] - \nabla_\theta [P_\theta(a) \log Q(a)] \\
&= \sum_{a\in A} [\nabla_\theta P_\theta(a) \log P_\theta(a) + \nabla_\theta P_\theta(a)] - \nabla_\theta P_\theta(a) \log Q(a) \\
&= \sum_{a\in A} \nabla_\theta P_\theta(a) (\log P_\theta(a) - \log Q(a) + 1)
\end{aligned}$$

3.2 Magnetic Mirror Descent

Another extension of Mirror Descent to Reinforcement learning is Magnetic Mirror Descent [7]. The main aim of this work is creating an RL algorithm that performs well in both single, and multiagent settings. The authors start-off by casting QRE solving as a bilinear saddle point problem and establish its equivalence to solving Variational Inequality problems with specific structures.

3.2.1 Connection between Variational Inequalities and QREs

Variational inequalities are a general class of problems that have a wide range of applications. A Variational Inequality problem is generally of the following form:

Definition 8 *Given $\mathcal{Z} \subseteq \mathbb{R}^n$ and mapping $G : \mathcal{Z} \to \mathbb{R}^n$, the variational inequality problem VI $(\mathcal{Z}, G)$ is to find $z_* \in \mathcal{Z}$ such that,*

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

Solving for QREs in two-player zero-sum games can be represented as the following bilinear saddle point problem:

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \alpha g_1(x) + f(x, y) + \alpha g_2(y), \tag{3.1}$$

where $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^m$ are closed and convex, and $g_1 : \mathbb{R}^n \to \mathbb{R}$, $g_2 : \mathbb{R}^m \to \mathbb{R}$, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$.

The solution (x_*, y_*) to the saddle point problem Equation 3.1, corresponds to the Nash equilibrium of the reguarlized game with the following first-order optimality conditions:

$$\langle \nabla g_1(x_*) + \nabla_{x_*} f(x_*, y_*), x - x_* \rangle \geq 0, \forall x \in \mathcal{X}. \tag{3.2}$$

$$\langle \nabla g_2(y_*) + \nabla_{y_*} f(x_*, y_*), y - y_* \rangle \geq 0, \forall y \in \mathcal{Y}. \tag{3.3}$$

These optimality conditions are equivalent to VI($\mathcal{Z}, G$), where $G = F + \alpha\nabla g$, $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, and $g : \mathcal{Z} \to \mathbb{R}$. Hence, the solution the the VI problem ($z_* = (x_*, y_*)$), corresponds to the solution of the saddle point problem stated in Equation 3.1.

3.2.2 MMD Algorithm

Based on the above connection they propose a non-euclidean proximal gradient method as an equilibrium finding algorithm in two-player zero-sum games. They also prove a linear last-iterate convergence guarantee for the algorithm as a QRE solver.

The authors first propose a non-Euclidean proximal gradient method to solve the VI ($\mathcal{Z}, F + \alpha\nabla g$) with the following update at each iteration:

$$z_{t+1} = \arg\min_{z \in \mathcal{Z}} \eta(\langle F(z_t), z\rangle + \alpha g(z)) + B_\psi(z; z_t). \tag{3.4}$$

With the following assumptions, z_{t+1} is well defined:

- ψ is 1-strongly convex with respect to $\|.\|$ over $\mathcal{Z}$, and for any l, stepsize $\eta > 0$, $\alpha > 0$, $z_{t+1} = \arg\min_{z \in \mathcal{Z}} \eta(\langle l, z\rangle + \alpha g(z)) + B_\psi(z; z_t) \in \text{int dom } \psi$.
- F is monotone and L-smooth with respect to $\|.\|$ and g is 1-strongly convex relative to ψ over $\mathcal{Z}$ with g differentiable over int dom ψ.

The algorithm that is termed MMD uses the same update as Equation 3.4, with g taken to be ψ or $B_\psi(.; z')$ for some z'.

We now restate the main algorithm as stated in Sokota et.al, [7],

AlgorithmMMD [7, (Algorithm 3.6)] Starting with $z_1 \in \text{int dom } \psi \cap \mathcal{Z}$, at each iteration t do

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

or, Given some z', do

$$z_{t+1} = \arg\min_{z \in \mathcal{Z}} \eta(\langle F(z_t), z \rangle + \alpha B_\psi(z; z')) + B_\psi(z, z_t).$$

Algorithm 2 provides the following convergence guarantees.

Theorem 1 *[7, 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

Algorithm 2 can be either implemented in closed form for a given set of parameters or by performing stochastic gradient descent at each step to approximate the closed form. This approximation is especially useful when it is not possible to compute the closed form such as in function approximation settings.

In the two form of updates specified in Algorithm 2, the second form involves a z', term that is referred to as the magnet. This can either be a reference policy such as the ones used in Imitation learning, or one that is trailing the current policy to stabilize learning. Alternatively, one can also use a uniform policy in which case the term reduces to entropy and acts as a regularization term and encourages exploration. For the rest of our discussion on MMD, we assume the magnet to always be a uniform policy.

With ψ taken to be negative entropy, the behavioral form of MMD is to perform the following update at each information state,

$$\pi_{t+1} = \arg\max_{\pi} \mathbb{E}_{A\sim\pi} q_t(A) + \alpha H(\pi) - \frac{1}{\eta} KL(\pi, \pi_t), \tag{3.5}$$

where π_t is the current policy, q_t is a vector containing the q-values following the policy π_t, and $H(\pi)$ is the entropy of the policy being optimized.

In single-agent settings MMD's performance is competitive with PPO in Atari and MuJoCo environments. And, in the multi-agent setting the performance of tabular MMD is on par with CFR, but worse than CFR+.

3.2.4 Equivalence of MMD and MDPO

- MDPO with an added entropy term is equivalent to MMD with negative entropy mirror map and uniform magnet.

CHAPTER 4

MODIFIED UPDATES FOR MIRROR-DESCENT BASED METHODS

The main work of this thesis is exploring a few existing techniques that have been proven to be effective in multi-agent settings in the context of the above alogrithms. These techniques are namely - Neural Replicator Dynamics [8], Extragradient methods [9], and Optimistic updates [?]. We propose combining these techniques to the methods discussed in the previous section and investigate their behavior through experiments in two-player zero-sum games. We now discuss these techniques and how they have generally been applied in the literature. While discussing each technique, we also outline how they fit into the algorithms discussed in the previous section and hypothesize about the potential effects these modifications.

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, [8] show equivalence between Softmax Policy gradients 2.1.1 and continuous-time Replicator Dynamics [8, 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 [8, 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 [10].

Since the typical neural network representation of policies use softmax projection on the logits, the idea of fixing the gradient updates can be applied more generally to algorithms beyond SPG. The NeuRD loss has been adapted into other algorithms to improve performance or induce convergence in competitive and cooperative settings. Chhablani et.al, [11] showed improved performance in identical-interest games by applying the NeuRD fix to COMA [12]. Perolat et.al, [13] used NeuRD loss to approximate Replicator dynamics as a part of the DeepNash algorithm. They used the same reward transformation based adaptive regularization [10] to induce last-iterate convergence in training agents for the game of Stratego.

4.1.1 NeuRD fix in MMD, and MDPO

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

4.2 Extragradient updates

The Extragradient method was first introduced by G.M.Korpelevich [9] as a modification of gradient descent methods in solving saddle point problems. Extragradient 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 [14].

4.2.1 MMD-EG

4.2.2 MDPO-EG

4.3 Optimism

4.3.1 Optimistic Mirror Descent

4.3.2 OMMD

4.3.3 OMDPO

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?

We evaluate the converegence behavior of all these algorithms on Perturned RPS. 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.

5.1 Evaluation Methods

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

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.

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.

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 at the equilibrium is zero by definition since no player wants to deviate from their current strategy.

5.1.3 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.

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 [7].

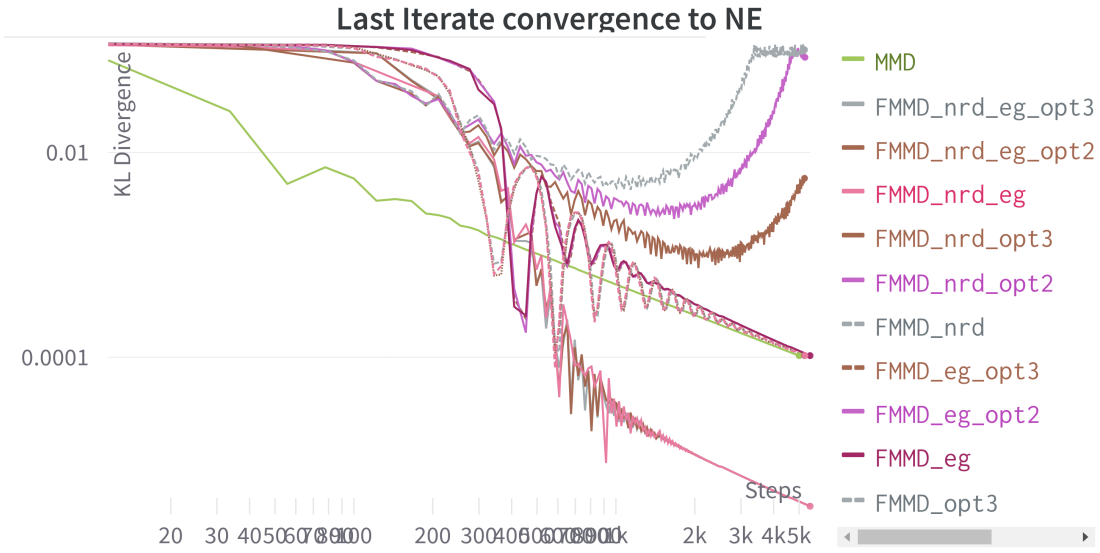
5.2 Tabular Experiments

Below, we give an overview of Last, and Average iterate convergences of 3 different algorithms - MMD, MDPO, SPG, and their variants obtained by applying combinations of the modified updates. We give the convergence results for the more interesting behaviors observed, a more exhaustive list of all combinations and their convergence behaviors can be found in the appendix.

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

Convergence in Perturbed RPS



- 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.

5.2.1 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 [7], we implement MMD by modifying the PPO implementation in RLLib [15]. We also use RLLib's OpenSpiel adapter with some modifications to use information states as inputs as opposed to observations. We train these reinforcement learning agents in self-play for the environments mentioned above.

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

CHAPTER 6

DISCUSSION

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

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

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

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

7. 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.

8. 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.

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

10. 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.

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

12. 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.

13. 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.

14. 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.

15. 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.