Learning Dynamics of Non-cooperative Agents in Dynamic Environments

Benjamin Chasnov BCHASNOV@UW.EDU
Electrical and Computer Engineering
University of Washington
Seattle, WA 98195-4322, USA

Abstract

We consider the setting in which multiple agents learn to act in a dynamic environment and study the resulting dynamics of the learning process. Specifically, we formulate the problem as a continuous game where agents seek local minimizers of their individual cost functions in a coupled dynamical system. Since agents are non-cooperative, the Jacobian of the learning dynamics is asymmetric; we show that this asymmetry can lead to spurious limit cycles, stable non-Nash equilibria, and unstable Nash equilibria. Yet, when the symmetric part of the Jacobian is positive-definite, we derive non-asymptotic rates for convergence to stable Nash equilibria.

1. Introduction

A significant focus in the study of *self-interested* decision-makers and non-cooperative game theory is the characterization and computation of equilibria such as *Nash equilibria*. A natural question that arises is how non-cooperative players find or learn such equilibria. With this question in mind, a variety of fields have focused their attention on the problem of learning in games which has lead to a plethora of learning algorithms including multi-agent reinforcement learning, gradient play, fictitious play and best response among others [9]. While convergence has been studied for many of these algorithms, the results tend to asymptotic.

More recently, game theoretic models of algorithm interaction are being adopted in machine learning applications. For instance, game theoretic tools are being used to improve the robustness and generalizability of machine learning algorithms; e.g., generative adversarial networks have become a popular topic of study demanding the use of game theoretic ideas to provide performance guarantees [7]. In other work from the learning community, game theoretic concepts are being leveraged to analyze the interaction of learning agents—see, e.g., [10, 12, 1, 15].

Despite this activity, we still lack a complete understanding of the dynamics and limiting behaviors of coupled, competing learning algorithms. In particular, it is important to know when to terminate the algorithms in order to ensure certain performance guarantees or to obtain a finite time bound on the error that can be used to provide guarantees on subsequent control or incentive policy synthesis.

One may imagine that the myraid results on convergence of gradient descent in optimization readily extend to the game setting. Yet, they do not since gradient-based learning schemes in games *do not correspond to gradient flows*. Gradient flows are a very narrow class of flows admitting *nice* convergence guarantees—e.g., almost sure convergence to local minimizers—due to the fact that they preclude flows with the *worst geometries* [14]. In particular, the gradient-based learning dynamics for competitive, multi-agent settings have a *non-symmetric Jacobian* and as a consequence their dynamics may admit complex eigenvalues and non-equilibrium limiting behavior such as periodic orbits. In short, this fact makes it difficult to extend many of the optimization approaches, whose primary technique depends on a cost decreasing at each update, to convergence in single-agent optimization settings to multi-agent settings. In fact, in games, as our example highlights, a player's cost can increase when they follow the gradient of their own cost. This behavior is due to the coupling between the agents.

Project overview The goal of this project is to develop and analyze learning algorithms for control of multiple decision-making agents. In classical optimal control problems, a policy is sought that minimizes a cost function $c(x, u)$ subject to dynamical and state constraints:

$$\min_u c(x, u) \qquad \text{subject to } x^+ = f(x, u),$$

where x is the state of the system, u is the control input or policy and $f(x, u)$ is the dynamics of the system, costs are twice continuously-differentiable and the dynamics once continuously-differentiable. In practice, the cost function is often a combination of the agents' goals and efforts. My project extends the classical problem to n agents, where each agent selects a control input u_i or policy to minimize its respective cost,

$$\min_{u_i} c_i(x, u_1, \ldots, u_n) \qquad \text{subject to } x^+ = f(x, u_1, \ldots, u_n), \qquad \text{for all } i \in \{1, \ldots n\}. \tag{P1}$$

In this formulation, the agents may be non-cooperative because their goals, as represented by the cost functions $c_i(x, u)$, may not be aligned (or may even be directly opposed).

Notation The joint action is $u = (u_1, \ldots, u_n) \in \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_n} = \mathbb{R}^d$. The shared state of the system is $x \in \mathbb{R}^m$. We use the following notation for partial first derivatives, $D_j c_i(u) = \frac{\partial c_i}{\partial u_j} \in \mathbb{R}^{d_j}$ and second derivatives $D_{jk} c_i(u) = \frac{\partial}{\partial u_j} \frac{\partial}{\partial u_k} c_i(u) \in \mathbb{R}^{d_j \times d_k}$. Costs are real-valued functions $c_i : \mathbb{R}^d \to \mathbb{R}$ for settings without dynamics and $c_i : \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}$ for settings with dynamics.

Cooperative vs. non-cooperative Before we begin, an important distinction needs to be made with regards to optimization problems with multiple costs, that is the notion of cooperative versus non-cooperative settings. In *cooperative* games, as the name suggests, agents wish to and *are able to* cooperate with each other. Not only do they have similar goals, they are capable of trusting and communicating with each other to coordinate strategies to obtain a Pareto optimum. To borrow Stephen Boyd's notation [3] of multi-criterion optimization, a cooperative game can be formulated as such,

$$\min \text{ (w.r.t } \mathbb{R}^n) \quad (c_1(u), \cdots, c_n(u)) \tag{P2}$$

For simplicity, however, we will first consider the case without dynamics. The conditions for stationarity of this unconstrained problem is the following.

Definition 1 (Pareto Optimum) *The stationary condition of unconstrained cooperative game is a Pareto optimum,*

$$\sum_{i \in \mathcal{I}} \theta_i D_i c_i(u^*) = 0, \ \sum_{i \in} \theta_i D_{ii} c_i(u^*) > 0$$

where $\theta_i \in \mathbb{R}^+$ *and* $\sum_{i=1}^n \theta_i = 1$.

At a Pareto optimum, an individual's gradient can non-zero, that is, for some agent i, $D_i c_i(u^*) \neq 0$, which means that the individual agent has an incentive to deviate from the point to improve its cost holding other agents' actions constant. Also, the location on the Pareto front, θ, needs to be communicated and agreed upon amongst the agents, which is unrealistic if agents cannot trust each other. This motivates us to study another stationary condition for *non-cooperative* continuous games, that is a differential Nash equilibrium.

Definition 2 (Differential Nash Equilibrium) *The fixed point* u^* *is a differential Nash equilibrium if*

$$D_i c_i(u^*) = 0, \ D_{ii} c_i(u^*) > 0$$

for all agents $i = 1, \ldots, n$. *The derivative of an agent's cost with respect to its own action is zero and locally curves upwards.*

A Nash equilibrium is a concept of stationarity for self-interested agents who are only concerned with value of its own cost. An example of such a scenario is a *zero-sum game* where one agent minimizes a cost and the adversary tries to maximize that cost. However, not all non-cooperative games have strict adversaries. A game can be non-cooperative even if agents' costs are "aligned". Suppose two agents wish to accomplish the same task, but are unable to communicate or trust each other. These scenarios are non-cooperative because of the agents' lack of ability to cooperate. If an agent sees the possibility of decreasing its cost, it will.

In this work, we seek to develop gradient-based methods to solve non-cooperative games. The settings of these games are dynamic, in the sense that there's a Markovian process at work with a state $x \in \mathbb{R}^m$ that is changing in time. In the following section, we analyze the learning dynamics of simultaneous gradient descent in the unconstrained problem. We will present numerical examples from LQ games [2] to illustrate the convergence of simultaneous gradient descent to the unique Nash equilibrium as well as to a non-Nash attractor.

The work that follows has been accepted at the AAAI and SPIE conferences and oral presentations were given by the main author, see references [5], [6]. A paper extending this work to stochastic gradient estimates and non-uniform learning rates has been submitted to UAI [4].

2. Learning dynamics in games

In this section, we will provide a non-asympototic convergence guarantee to local Nash equilibrium assuming that the fixed point is a local asymptotically stable equilibrium of the dynamics of `SIMGRAD`. We notice the analysis quickly differs from typical convergence of gradient descent in optimization problems primarily because of two facts: i) the cost is not guaranteed to decrease at each step and ii) the Jacobian of the vector field, or game Jacobian, is non-symmetric.

2.1 A gradient-based method for solving games

If we treat the unconstrained coupled optimization problem with no dynamics,

$$\min_{u_i} \; c_i(u_i, u_{-i}), \quad \forall i \in [1, n],$$

as n individual optimization problems, a naive method to solve it would be to have each agent perform gradient descent on their own cost function.

Algorithm 1 (Simultaneous Gradient Descent, `SIMGRAD`) *Initialize $u^{(0)}$ in a ball of radius r around a locally stable Nash equilibrium. Choose step-size γ_i for each agent appropriately. Simultaneously update each agent's actions by descending its own gradient*

$$u_i^{(k+1)} = u_i^{(k)} - \gamma_i D_i c_i(u^{(k)}), \quad \textit{for all agents } i \in \mathcal{I},$$

for $k \geq 0$ until convergence.

An important object to study from the algorithm is its vector field denoted as

$$\omega(u) \equiv (D_1 c_1(u), \cdots, D_n c_n(u))^\top.$$

With appropriately chosen step sizes γ_i, we approximate the above discrete time update equations as a continuous time autonomous dynamical system

$$\dot{u} = -\omega(u), \tag{1}$$

and therefore by studying the Jacobian of this vector field we are able to analyze its convergence properties.

Definition 3 (Game Jacobian) *The Jacobian of the vector field ω of* SIMGRAD *is*

$$J(u) \equiv D\omega(u) = \begin{bmatrix} D_{11}c_1(u) & \cdots & D_{1n}c_1(u) \\ \vdots & \ddots & \vdots \\ D_{n1}c_n(u) & \cdots & D_{nn}c_n(u) \end{bmatrix} \in \mathbb{R}^{d\times d},$$

where $D_{jk}c_i(u) = \frac{\partial}{\partial u_j}\frac{\partial}{\partial u_k}c_i(u)$ *and* $D_{jk}c_i = D_{kj}c_i^T$.

Observe that the since the off-diagonals are partial derivatives of non-cooperative costs, the game Jacobian in general is non-symmetric, i.e. $D\omega \neq D\omega^T$. For two player, zero-sum games, note that $D_{12}c_1 = -D_{21}c_2^T$.

2.2 Non-asymptotic convergence guarantees for SIMGRAD

The multi-agent learning framework we analyze is such that each agent's rule for updating their choice variable consists of the agent modifying their action u_i in the direction of their individual gradient D_ic_i. Let us first consider the setting in which each agent i has oracle access to D_ic_i. We consider the case where the agents have a constant *uniform* learning rate—i.e., $\gamma_i \equiv \gamma$.

Proposition 1 *Consider an n–player continuous game $(c_1, \ldots, c_n)$ where the costs are twice continuously-differentiable and ω is $L-$Lipschitz. Let $u^* \in U$ be a stable differential Nash equilibrium. Suppose agents use the gradient-based learning rule $u^{(k+1)} = u^{(k)} - \gamma\omega(u^{(k)})$ with learning rates $0 < \gamma < \tilde{\gamma}$ where $\tilde{\gamma}$ is the smallest positive h such that the eigenvalues λ_j of J satisfy $\max_j |1-h\lambda_j(J(u^*))| = 1$. Then, for $u^{(0)} \in B_r(u^*)$, $u^{(k)} \to u^*$ exponentially.*

The above result provides a range for the possible learning rates for which simultaneous gradient descent converges to a stable differential Nash equilibrium assuming agents initialize in a ball contained in the region of attraction of u^*. Note that the usual assumption in gradient-based approaches to single-objective optimization problem is that $\gamma < 1/L$.

The convergence guarantees in Proposition 1 is asymptotic in nature. The next result provides a finite-time convergence guarantee to an $\varepsilon-$differential Nash equilibrium.

Let $S(u) = \frac{1}{2}(J(u)+J(u)^T)$ be the symmetric part of $J(u)$. The following constants are useful in the analysis of the convergence properties of SIMGRAD algorithm. They are the squared minimum singular value of the symmetric part of J and the squared maximum singular value of the full Jacobian at a specific fixed point u^*. Define

$$\alpha = \min_{u\in B_r(u^*)} \lambda_d\big(S(u)^T S(u)\big) \quad \text{and} \quad \beta = \max_{u\in B_r(u^*)} \lambda_1(J(u)^T J(u))$$

where $B_r(u^*)$ is a r–radius ball around u^* and λ_d is the dth largest eigenvalue. With a judicious choice of learning rate γ, Algorithm (1) will converge (at an exponential rate) to a locally stable equilibrium of the dynamics. Our recent work [5] provides the following result.

Theorem 1 *Consider a game $(c_1, \ldots, c_n)$ on $U = U_1 \times \cdots \times U_n$ where costs are twice continuously-differentiable. Let $u^* \in U$ be a stable differential Nash equilibrium. Suppose $u^{(0)} \in B_r(u^*)$ and that $\alpha < \beta$. Then the gradient-based learning dynamics with learning rate $\gamma = \sqrt{\alpha}/\beta$ obtains an ε–differential Nash for all*

$$k \geq \left\lceil 2\frac{\beta}{\alpha}\log\frac{r}{\varepsilon} \right\rceil.$$

To prove this result, we must determine the ranges of learning rates γ such that the dynamics are contracting. We do so by bounding the two norm of $I - \gamma J$ using the values of α and β.

Proof. First, note that $\|u^{(k+1)} - u^*\| = \|g(u^{(k)}) - g(u^*)\|$ where $g(u) = u - \gamma\omega(u)$. Now, given $u^{(0)} \in B_r(u^*)$, by the mean value theorem,

$$\|g(u^{(0)}) - g(u^*)\| = \| \int_0^1 Dg(\tau u^{(0)} + (1-\tau)u^*)(u^{(0)} - u^*)d\tau\| \leq \sup_{u \in B_r(u^*)} \|Dg(u)\|\|u^{(0)} - u^*\|.$$

Hence, it suffices to show that for the choice of γ, the eigenvalues of $I - \gamma J(u)$ are in the unit circle. Indeed, since $\omega(u^*) = 0$, we have that

$$\|u^{(k+1)} - u^*\|_2 = \|u^{(k)} - u^* - \gamma(\omega(u^{(k)}) - \omega(u^*))\|_2 \leq \sup_{u \in B_r(u^*)} \|I - \gamma J(u)\|_2 \|u^{(k)} - u^*\|_2$$

If $\sup_{u \in B_r(u^*)} \|I - \gamma J(u)\|_2$ is less than one, then the dynamics are contracting. For notational convenience, we drop the explicit dependence on u. Since $\lambda_d(S) \geq \sqrt{\alpha}$ on $B_r(x^*)$,

$$\begin{aligned}(I - \gamma J)^T(I - \gamma J) &= I - \gamma(J^T + J) + \gamma^2 J^T J \\ &\leq (1 - 2\gamma\lambda_d(S) + \gamma^2\lambda_1(J^T J))I \\ &\leq (1 - \tfrac{\alpha}{\beta})I\end{aligned}$$

where the last inequality holds for $\gamma = \sqrt{\alpha}/\beta$. Hence,

$$\begin{aligned}\|u^{(k+1)} - u^*\|_2 &\leq \sup_{u \in B_r(u^*)} \|I - \gamma J(x)\|_2 \|u^{(k)} - u^*\|_2 \\ &\leq (1 - \tfrac{\alpha}{\beta})^{1/2} \|u^{(k)} - u^*\|_2.\end{aligned}$$

Since $\alpha < \beta$, we have that $(1 - \alpha/\beta) < \exp(-\alpha/\beta)$ so that

$$\|u^{(T)} - u^*\|_2 \leq \exp(-T\alpha/(2\beta))\|u^{(0)} - u^*\|_2.$$

This, in turn, implies that $u^{(k)} \in B_\varepsilon(u^*)$ for all $k \geq T = \lceil 2\frac{\beta}{\alpha}\log(r/\varepsilon)\rceil$. ■

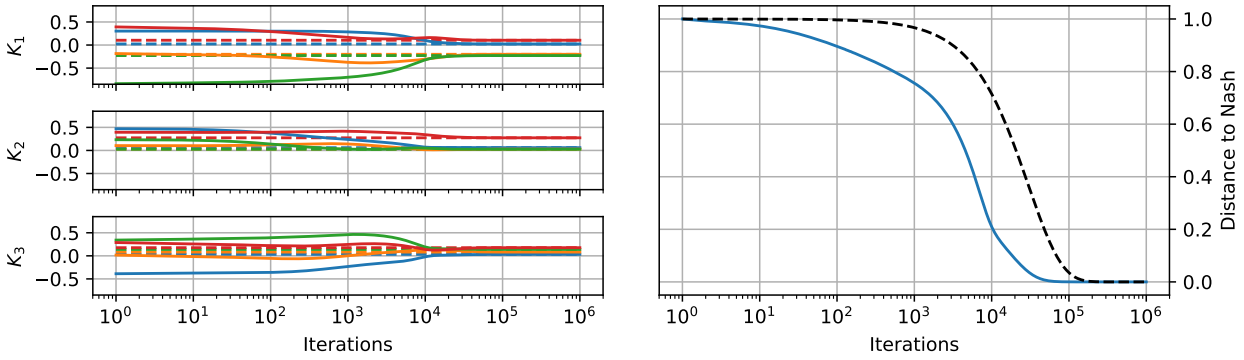
Note that $\gamma = \sqrt{\alpha}/\beta$ is selected to minimize $1 - 2\gamma\lambda_d(S) + \gamma^2\lambda_1(J^T J)$. Hence, this is the fastest learning rate given the worst case eigenstructure of J over the ball $B_r(x^*)$ for the choice of operator norm $\|\cdot\|_2$. We note, however, that faster convergence is possible as indicated by Proposition 1 and observed in the examples in Section 3.1. Indeed, we note that the spectral radius $\rho(\cdot)$ of a matrix is always less than its maximum singular value—i.e. $\rho(I - \gamma J) \leq \|I - \gamma J\|_2$—so it is possible to contract at a faster rate. We remark that if J was symmetric (i.e., in the case of a potential game [13] or a single-agent optimization problem), then $\rho(I - \gamma J) = \|I - \gamma J\|_2$. In non-cooperative games, however, J is not symmetric.

3. Application to agents in dynamic environments

The analysis above assumes oracle access to deterministic gradients. At each iteration of `SIMGRAD` agents play joint action $u = (u_i)_{i=0}^n$ and observe cost $(c_i(u))_{i=0}^n$ and its gradients. In this section, we will describe how we expect to use this framework in environments with a shared state x.

3.1 A Gradient Method for Linear Quadratic Dynamic Games

The first example we explore is a linear quadratic (LQ) game with three players in the space of linear feedback policies. This game serves as a useful benchmark since it has a unique global



(a) (b)

Figure 1: Convergence of policy gradient in LQ dynamic games to the Nash policy. (a) Each player's linear feedback gain matrix K_i converges to the unique Nash policies (dotted lines). (b) The black dashed line shows upper bound of the number of iterations required to converge within ε distance from Nash (2-norm). The actual convergence for this random initialization is shown as the solid line.

equilibrium that we can compute via a set of coupled algebraic Riccati equations [2]. The gradient-based learning rule for each of the agents is a multi-agent version of policy gradient in which agents have oracle access to their gradients at each iteration.

We first consider agents with deterministic linear feedback policies. Consider a four state discrete time linear dynamical system,

$$x(t+1) = Ax(t) + B_1u_1(t) + B_2u_2(t) + B_3u_3(t)$$

where $x(t) \in \mathbb{R}^4$ and, for each $i \in \{1,2,3\}$, $u_i(t) \in \mathbb{R}$ is the control for player i. The policy for each player is parameterized by a linear feedback gain matrix, $u_i(t) = -K_ix(t)$. Moreover, each player seeks to minimize a quadratic cost

$$c_i(x, u_i, u_{-i}) = \sum_{t=0}^{\infty} \left(x(t)^T Q_i x(t) + \textstyle\sum_{j=1}^{n} u_j(t)^T R_{ij} u_j(t) \right)$$

which is a function of the coupled state variable $x(t)$, their own control $u_i(t)$ and all other agents' control $u_{-i}(t)$ over an infinite time horizon. In an effort to learn a Nash equilibrium, each agent employs policy gradient. In particular, they update their feedback policy via

$$K_i^{(k+1)} = K_i^{(k)} - \gamma_i \nabla_{K_i} c_i(x, u_i^{(k)}, u_{-i}^{(k)}).$$

We compute the gradient of c_i with respect to K_i, the feedback gain that parameterizes player i's control input u_i. The computation of this gradient for the single agent case can be found in [8]. Indeed,

$$\nabla_{K_i} c_i(x(t), u_i(t), u_{-i}(t)) = 2(R_{ii}K_i - B_i^T P_i \widetilde{A}) \sum_{t=0}^{\infty} x(t)x(t)^T,$$

where the closed loop dynamics are $\widetilde{A} = A - B_1K_1 - B_2K_2 - B_3K_3$ for a given joint policy (K_1, K_2, K_3). Hence, the collection of the agents' individual gradients is given by

$$\omega(K_1, K_2, K_3) = \left(2(R_{ii}K_i - B_i^T P_i \widetilde{A}) \sum_{t=0}^{\infty} x(t)x(t)^T \right)_{i=1}^{3}$$

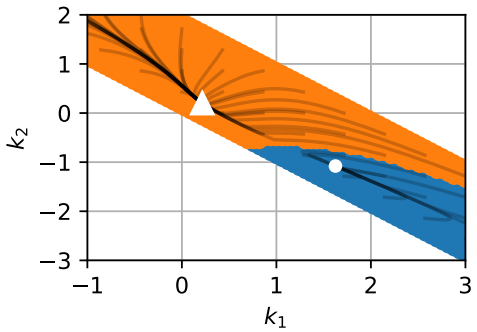


Figure 2: Scalar LQ game with policies parameterized by linear feedback gain k_1 and k_2. The colored band indicates pairs of (k_1, k_2) of which the dynamics are stable. The colors represent the two region of attractions for the Nash equilibrium (triangle) and non-Nash stable attractor (circle) under simultaneous gradient descent. All initiazations in the orange region converges to the unique Nash equilibrium, whereas initializations in the blue region converges to the non-Nash equilibrium where agent 2 is at a local maximum of its own cost.

The states $z(t)$ are obtained from simulating the system, or performing "roll-outs". For each i, the Riccati matrix P_i is computed by solving the Riccati equation

$$P_i = \widetilde{A}^T P_i \widetilde{A} + Q_i + \sum_{j=1}^{n} K_j R_{ij} K_j.$$

Note that this Riccati equation is only used to compute the gradient of the cost functions with respect to a specific set of feedback gains. The system parameters used in this example are listed in Appendix A.1.

For the purpose of validating convergence, we can compute the Nash policies (K_1^*, K_2^*, K_3^*) by an established method with coupled Riccati equations, explained in Appendix A.1. We use the learning rate $\gamma_i = \gamma$ defined as in Theorem 1. To compute γ we first compute the game Jacobian $J(K_1^*,\ K_2^*,\ K_3^*)$ at the Nash feedback gains and then find the maximum eigenvalue of $J^T J$ and minimum eigenvalue of $(J^T + J)^T (J^T + J)$ in a neighborhood of (K_1^*, K_2^*, K_3^*) to determine the constants α and β as defined in Section 2.2.

Figure 1 shows the convergence of the gradient updates to the Nash policies. The K_i are randomly initialized in a neighborhood of the known Nash equilibrium and such that $\widetilde{A}$ is stable. The number of iterations required to converge to an ε–differential Nash is bounded by the dashed black line in Figure 1b, which shows the curve of (ε, T) pairs determined by Theorem 1. However, this learning rate is not optimal, as choosing a larger γ will result in faster convergence as empirically observed.

3.1.1 Convergence to a non-Nash stable attractor

Despite having demonstrated that `SIMGRAD` converges to the global unique Nash equilibrium, we can only guarantee convergence if the algorithm is initialized in the region of attraction of the Nash equilibrium. In this example, we show a simple two-player, scalar example of an LQ game where `SIMGRAD` converges a stable equilibrium that is non-Nash, i.e. where one agent's second derivative is negative definite.

Consider a LQ game with scalar state $x \in \mathbb{R}$ and two players with scalar actions $u_1, u_2 \in \mathbb{R}$. The discrete time dynamics of the system is $x(t+1) = x(t) + u_1(t) + u_2(t)$ and the actions are parameterized by k_1, k_2 where $u_i = k_i x$. The costs for each player are $c_1(x, u_1, u_2) = \sum_{t=0}^{\infty} x(t)^2 +$

$5u_1(t)^2$ and $c_2(x, u_1, u_2) = \sum_{t=0}^{\infty} x^2 + 4u_2(t)^2 - 5u_1(t)^2$. Figure 2 shows the resulting learning dynamics of this game with various different initializations of $k^{(0)}$. The Nash equilibrium of this LQ game is $(k_1^*, k_2^*) = (0.21, 0.20)$, yet we observe that it is possible for the gradient dynamics to converge to a non-Nash equilibrium at $(1.62, -1.08)$. This example motivates the need for further study of these gradient dynamics in games and looks forward to possible modified update equations that avoid such spurious equilibria.

4. Discussion

The work detailed in this document merely scratches the surface of gradient methods for solving games. Our framework readily extends to various other settings beyond having oracle access to deterministic gradients and learning with uniform learning rates. For instance, in [4] we consider the stochastic setting where agents have unbiased estimators, $\widehat{D_i c_i}$, which can applied to a variety of problems such as multi-agent reinforcement learning or multi-armed bandit problems. We also consider scenarios where agents learn with non-uniform learning rates, which distorts the vector field of the learning dynamics and can cause agents to converge to different equilibria from same initializations. Various other numerical examples of multi-agent control [5] and human-machine dynamic interaction [6] have been presented, demonstrating the utility of this learning algorithm.

Future Work There are many new ideas to explore with regards to settings with multiple learning agents optimizing their own costs. By leveraging the framework developed in this work, we will explore new theoretical ideas and experimental validation of the theory. On the theoretical front, we wish to develop new learning updates that allow for constraints on the agents' actions, for example, coupled projected gradient descent or coupled interior point methods. We are also developing modified update rules based on "conjectures" of other agents' learning rules to allow for faster convergence and to avoid non-Nash equilibria or spurious limit cycles. These modified updates take into consideration the partial derivatives $D_i c_j$ where $i \neq j$, the effect a change in agent i's action has on the agent j's cost.

Experimentally, we plan to test these learning algorithms on sensorimotor games involving a human and a machine, each optimizing for their own cost. We predict that similar behaviors may arise in these dynamic games that we see in simulation, namely convergence to limit cycles and non-Nash equilibria. Finally, we aim to investigate the application of these learning algorithms to multi-agent autonomous quadcopters, in which the dynamics of these quadcopters are coupled through a shared state x.

5. Conclusion

We provide asymptotic and finite-time convergence guarantees for gradient-based learning in general-sum, continuous games. Specifically, we leverage the limiting continuous-time dynamical system and its Jacobian to construct a learning rate γ such that if the agents uniformly adopt this learning rate, they will be guaranteed to converge to a neighborhood of a stable local Nash equilibrium in finite-time. Despite not being an optimal learning rate, this method shines light on the theoretical basis of interaction of gradient-based learning dynamics in multi-agent settings where agents have their own individual objective that depends on the actions of others. Beyond analysis, the results are also useful for synthesis. One can use them to design games with desirable properties; this includes incentive design, control theory, and even machine learning, e.g., where game theoretic techniques are being employed to learn neural networks such as generative adversarial networks. We empirically verify the theoretical bounds by testing them with an LQ dynamic game with known Nash equilibria.

References

[1] David Balduzzi, Sébastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. *CoRR*, abs/1802.05642, 2018.

[2] T. Basar and G. Olsder. *Dynamic Noncooperative Game Theory*. Society for Industrial and Applied Mathematics, 2nd edition, 1998. doi: 10.1137/1.9781611971132.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Benjamin Chasnov, Lillian Ratliff, Eric Mazumdar, and Samuel Burden. Convergence analysis of gradient-based learning in continuous games. The Conference on Uncertainty in Artificial Intelligence (in submission), 2019.

[5] Benjamin Chasnov, Lillian J Ratliff, Daniel Calderone, Eric Mazumdar, and Samuel A Burden. Finite-time convergence of gradient-based learning in continuous games. *AAAI Workshop on Reinforcement learning in Games*, 2019.

[6] Benjamin Chasnov, Momona Yamagami, Behnoosh Parsa, Lillian J Ratliff, and Samuel A. Burden. Experiments with sensorimotor games in dynamic human/machine interaction. In *SPIE Defense + Security*. International Society for Optics and Photonics, 2019.

[7] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Traning GANs with Optimism. *arxiv:1711.00141*, 2017.

[8] Maryam Fazel, Rong Ge, Sham M. Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for linearized control problems. *arxiv:1801.05039*, 2018.

[9] Drew Fudenberg and David K Levine. *The theory of learning in games*, volume 2. MIT press, 1998.

[10] J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. *arxiv:1603.01121*, 2016.

[11] T-Y. Li and Z. Gajic. Lyapunov iterations for solving coupled algebraic riccati equations of nash differential games and algebraic riccati equations of zero-sum games. In Geert Jan Olsder, editor, *New Trends in Dynamic Games and Applications*, pages 333–351, Boston, MA, 1995. Birkhäuser Boston. ISBN 978-1-4612-4274-1.

[12] E. Mazumdar and L. J. Ratliff. On the convergence of competitive, multi-agent gradient-based learning algorithms. *arxiv:1804.05464*, 2018.

[13] Dov Monderer and Lloyd S. Shapley. Potential games. *Games and Economic Behavior*, 14(1): 124–143, 1996. doi: 10.1006/game.1996.0044.

[14] Robin Pemantle. A survey of random processes with reinforcement. *Probability Surveys*, 4 (1–79), 2007.

[15] Karl Tuyls, Julien Pérolat, Marc Lanctot, Georg Ostrovski, Rahul Savani, Joel Z Leibo, Toby Ord, Thore Graepel, and Shane Legg. Symmetric decomposition of asymmetric games. *Scientific Reports*, 8(1):1015, 2018. doi: 10.1038/s41598-018-19194-4.

Appendix A. Additional Examples

In this appendix, we include additional examples and information about examples contained in the main body of the text.

A.1 LQ game system parameters

The following are the system parameters and resulting Nash feedback gains computed using the coupled Riccati equations:

$$A = \begin{bmatrix} 0.402 & 1.037 & -0.565 & 0.115 \\ -0.021 & -0.990 & -0.584 & 0.457 \\ 0.377 & 1.105 & 0.698 & 1.192 \\ -0.177 & -0.332 & 0.237 & -0.286 \end{bmatrix},\ B_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix},\ B_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}\ B_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} 0.48 & 0 & 0 & 0 \\ 0 & 0.64 & 0 & 0 \\ 0 & 0 & 0.74 & 0 \\ 0 & 0 & 0 & 0.71 \end{bmatrix},\ Q_2 = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.41 & 0 & 0 \\ 0 & 0 & 0.71 & 0 \\ 0 & 0 & 0 & 0.44 \end{bmatrix},\ Q_3 = \begin{bmatrix} 1.00 & 0 & 0 & 0 \\ 0 & 0.55 & 0 & 0 \\ 0 & 0 & 0.86 & 0 \\ 0 & 0 & 0 & 0.63 \end{bmatrix},$$

$$R_{11} = \begin{bmatrix} 5.47 \end{bmatrix},\ R_{12} = \begin{bmatrix} 7.16 \end{bmatrix},\ R_{13} = \begin{bmatrix} 5.31 \end{bmatrix},\ R_{21} = \begin{bmatrix} 5.21 \end{bmatrix},\ R_{22} = \begin{bmatrix} 5.36 \end{bmatrix},$$
$$R_{23} = \begin{bmatrix} 7.63 \end{bmatrix},\ R_{31} = \begin{bmatrix} 9.71 \end{bmatrix},\ R_{32} = \begin{bmatrix} 2.34 \end{bmatrix},\ R_{33} = \begin{bmatrix} 5.26 \end{bmatrix},$$

and

$$K_1 = \begin{bmatrix} 0.023 \\ -0.201 \\ -0.228 \\ 0.104 \end{bmatrix}^T, K_2 = \begin{bmatrix} 0.060 \\ 0.029 \\ 0.026 \\ 0.274 \end{bmatrix}^T, K_3 = \begin{bmatrix} 0.033 \\ 0.082 \\ 0.138 \\ 0.177 \end{bmatrix}^T.$$

We use the following values for constants used in the LQ game: $\alpha = 19.4$ and 2.90×10^5; hence, we use $\gamma = 1.52 \times 10^{-5}$.

A.2 Coupled Riccati equations

We require the following standard assumption adopted in LQ games. Either $(A, B_1, \sqrt{Q_1})$ or $(A, B_2, \sqrt{Q_2})$ is stabilizable-detectable.

Without loss of generality, we assume $(A, B_1, \sqrt{Q_1})$ is stabilizable-detectable. We employ the following iterative Lyapunov algorithm for finding the Nash equilibrium to the linear quadratic game [11]:

step 1. Initialize $P_1^{(0)}$ to be the unique positive definite solution to the Riccati equation,

$$P_1^{(0)} = A^T P_1^{(0)} A - A^T P_1^{(0)} B_1 \big(R_1 + B_1^T P_1^{(0)} B_1\big)^{-1} B_1^T P_1^{(0)} A + Q_1, \tag{2}$$

and compute the corresponding gain matrix for player 1 by

$$K_1^{(0)} = (R_1 + B_1^T P_1^{(0)} B_1)^{-1} B_1^T P_1^{(0)} A. \tag{3}$$

Solve for $P_2^{(0)}$ by

$$P_2^{(0)} = \bar{A}^T P_2^{(0)} \bar{A} - \bar{A}^T P_2^{(0)} B_2 \big(R_2 + B_2^T P_2^{(0)} B_2\big)^{-1} B_2^T P_2^{(0)} \bar{A} + Q_1 \tag{4}$$

where $\bar{A} = A - B_1 K_1^{(0)}$ and compute the corresponding gain matrix for player 2 by

$$K_2^{(0)} = (R_2 + B_2^T P_2^{(0)} B_2)^{-1} B_2^T P_2^{(0)} \big(A - B_1 K_1^{(0)}\big). \tag{5}$$

We note that initializing using this method ensures that the initial closed loop matrix $A - B_1 K_1^{(0)} - B_2 K_2^{(0)}$ is stable.

step 2. Given $P_1^{(k)}$, $P_2^{(k)}$, $K_1^{(k)}$, and $K_2^{(k)}$, update the feedback gains using the following update rules:

$$K_1^{(k+1)} = (R_1 + B_1^T P_1^{(k)} B_1)^{-1} B_1^T P_1^{(k)} (A - B_2 K_2^{(k)}) \tag{6}$$

$$K_2^{(k+1)} = (R_2 + B_2^T P_2^{(k)} B_2)^{-1} B_2^T P_2^{(k)} (A - B_1 K_1^{(k)}) \tag{7}$$

step 3. Update the cost-to-go matrices by solving the Lyapunov equations:

$$\begin{aligned} P_1^{(k)} &= (A - B_1 K_1^{(k)} - B_2 K_2^{(k)})^T P_1^{(k+1)} (A - B_1 K_1^{(k)} - B_2 K_2^{(k)}) + (K_1^{(k)})^T R_1 K_1^{(k)} + Q_1 \\ P_2^{(k)} &= (A - B_1 K_1^{(k)} - B_2 K_2^{(k)})^T P_2^{(k+1)} (A - B_1 K_1^{(k)} - B_2 K_2^{(k)}) + (K_2^{(k)})^T R_2 K_2^{(k)} + Q_2 \end{aligned}$$

step 4. Repeat **steps 2–3** until the gains converge.

The extension to n-players is fairly straightforward; more detail can be found in the seminal reference [2].