# CSC 2541: Neural Net Training Dynamics

## Lecture 10 - Differentiable Games

Roger Grosse
(based on a lecture by Guodong Zhang)

University of Toronto, Winter 2022

# Differentiable Games

- So far, we have been focusing on optimization, where the parameters of a neural net are chosen to minimize a single cost function.
- What if multiple networks (or other optimization variables) are being optimized simultaneously to different objectives?

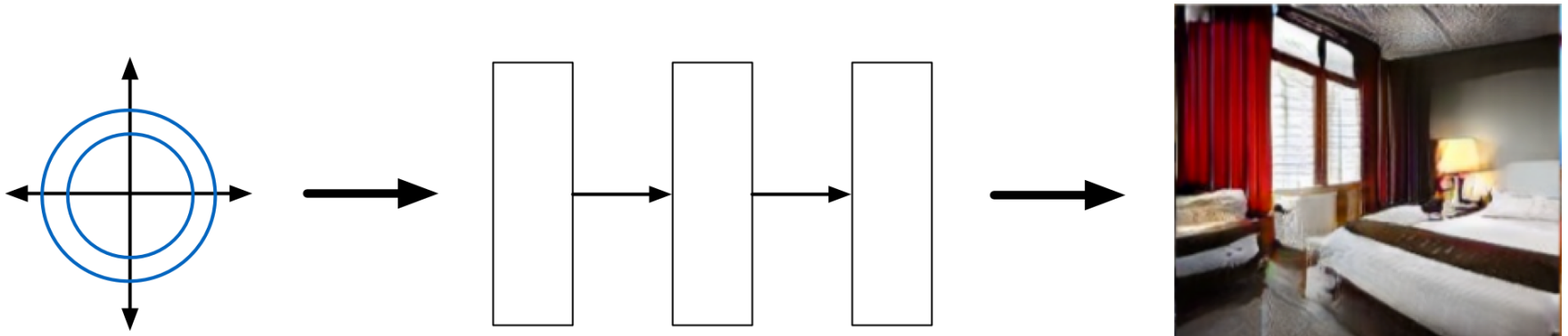
$$\mathbf{z}_i^* \in \arg\min_{\mathbf{z}_i} f_i(\mathbf{z}_i, \mathbf{z}_{-i}^*)$$

- The different networks are like players in a game.
  - This week: simultaneous games
  - Next two weeks: sequential games

# Motivation: GANs

# Implicit Generative Models

Implicit generative models learn a mapping from random noise vectors to things that look like, e.g., images:



Each dimension of the code vector is sampled independently from a simple distribution, e.g. Gaussian or uniform.

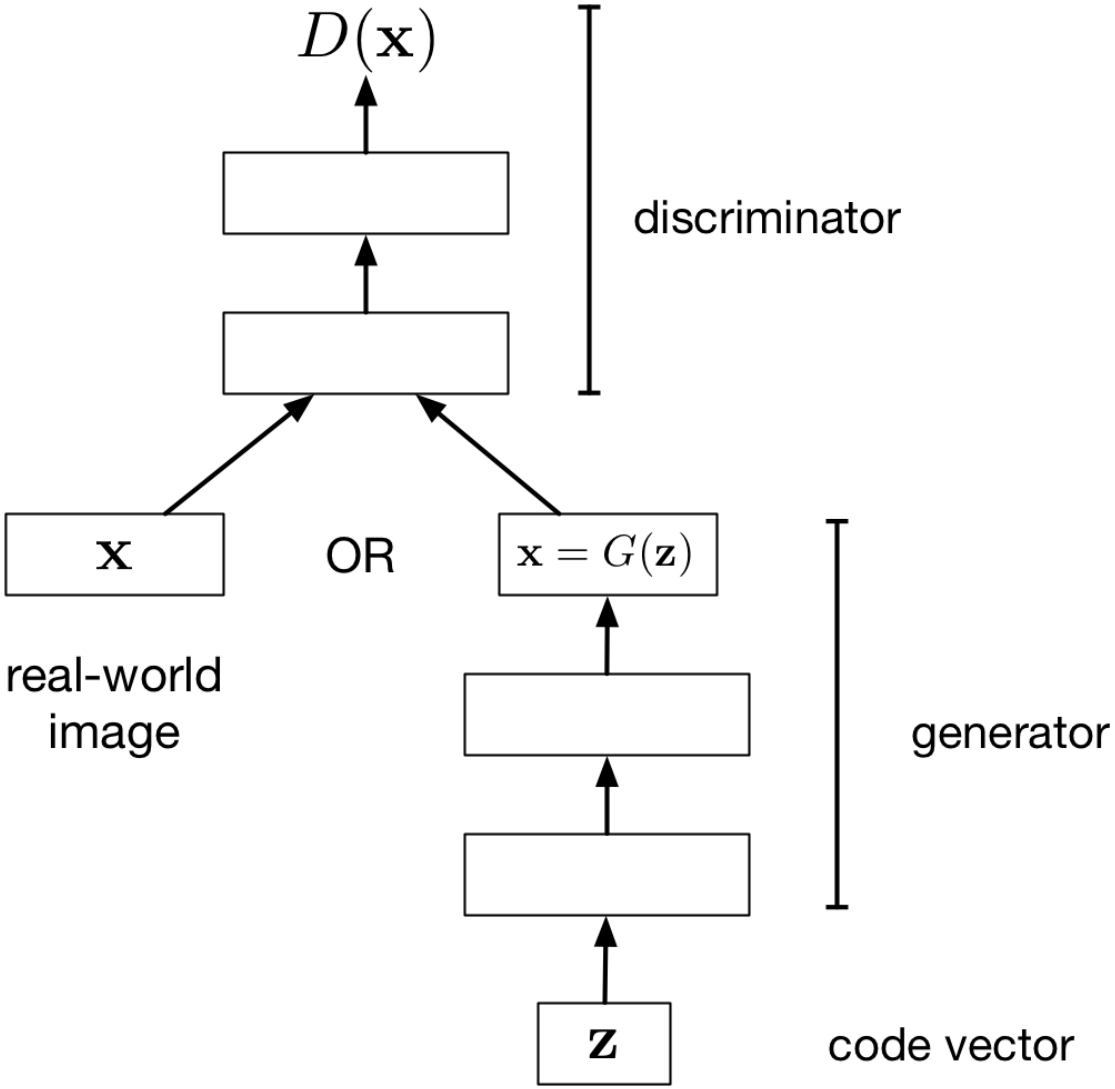
This is fed to a (deterministic) generator network.

The network outputs an image.

# Generative Adversarial Networks

- The advantage of implicit generative models: if you have some criterion for evaluating the quality of samples, then you can compute its gradient with respect to the network parameters, and update the network's parameters to make the sample a little better
- The idea behind Generative Adversarial Networks (GANs): train two different networks
  - The generator network tries to produce realistic-looking samples
  - The discriminator network tries to figure out whether an image came from the training set or the generator network
- The generator network tries to fool the discriminator network

# Generative Adversarial Networks



# Generative Adversarial Networks

- Let $D$ denote the discriminator's predicted probability of being data
- Discriminator's cost function: cross-entropy loss for task of classifying real vs. fake images

$$\mathcal{J}_D = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

- One possible cost function for the generator: the opposite of the discriminator's

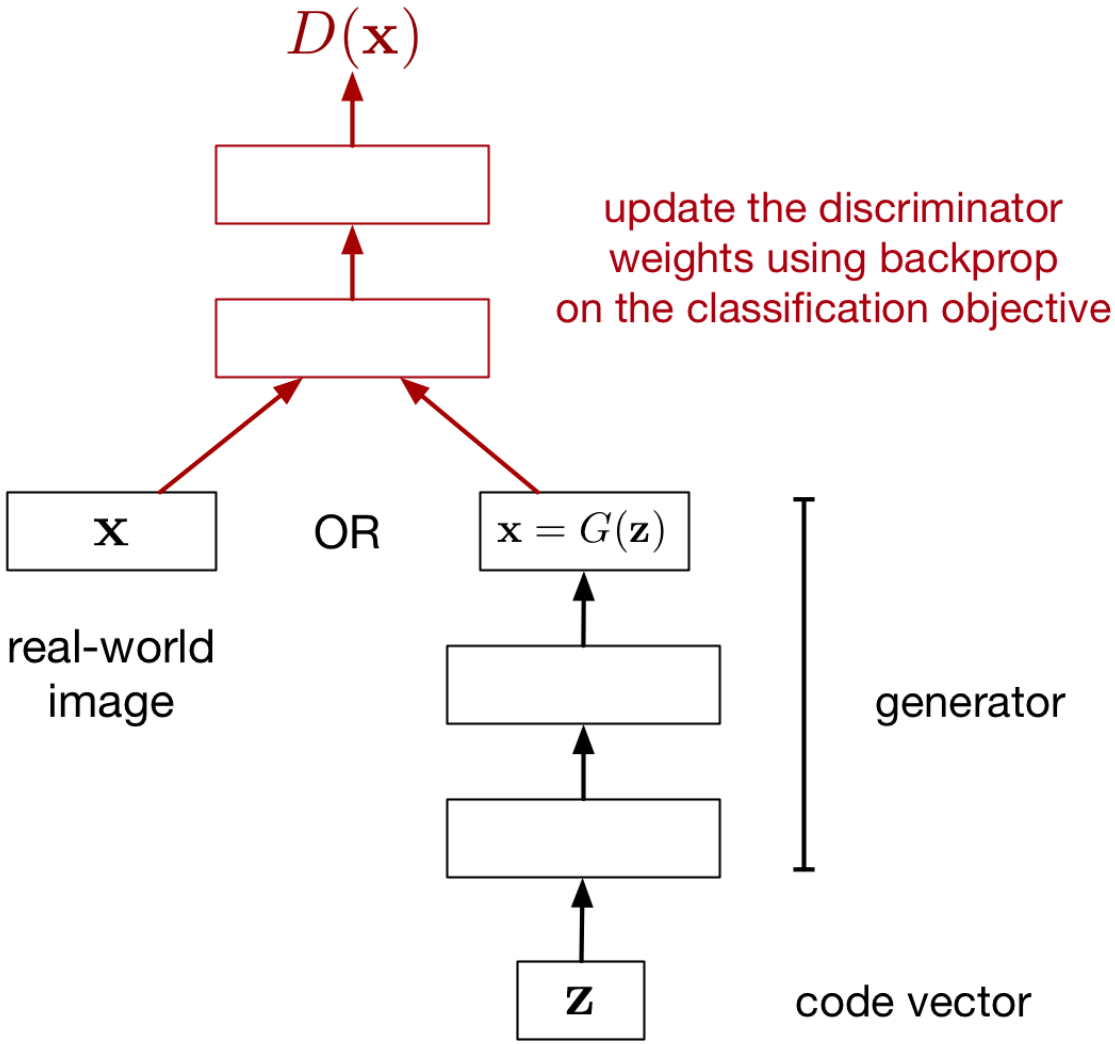
$$\begin{aligned}\mathcal{J}_G &= -\mathcal{J}_D \\ &= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]\end{aligned}$$

- This is called the minimax formulation, since the generator and discriminator are playing a zero-sum game against each other:

$$\max_G \min_D \mathcal{J}_D$$

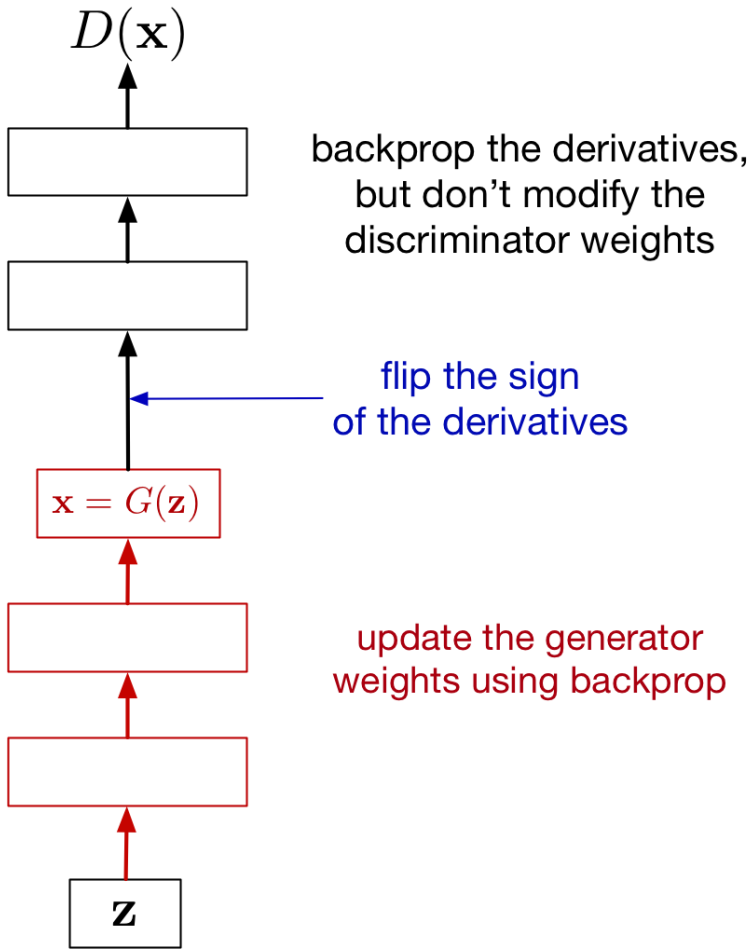
# Generative Adversarial Networks

Updating the discriminator:



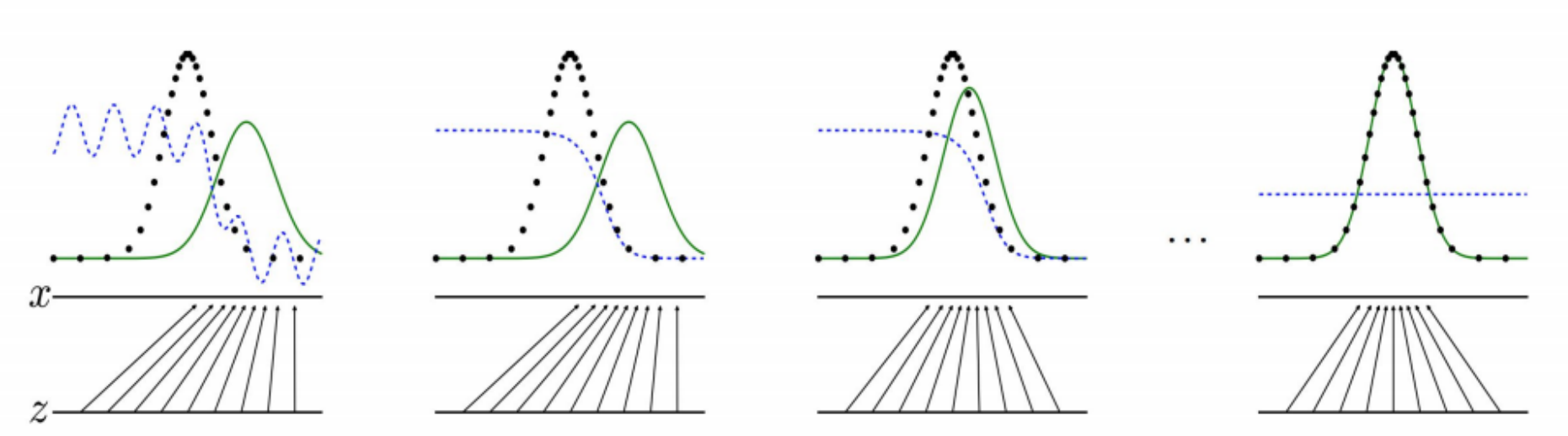
# Generative Adversarial Networks

Updating the generator:



# Generative Adversarial Networks

Alternating training of the generator and discriminator:




(Goodfellow et al., “Generative adversarial nets”)

# GAN Samples

Bedrooms:

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

# GAN Samples

ImageNet object categories (by BigGAN, a much larger model with a bunch more engineering tricks):

Brock et al., 2019. Large scale GAN training for high fidelity natural image synthesis.

# Generative Adversarial Networks

- **Caveat:** I introduced the minimax (zero-sum) formulation, which is how GANs are typically introduced. This is why they're often used as a motivation for minimax optimization.
- In practice, the generator is typically trained with a different objective which doesn't saturate. This is the non-saturating formulation. Hence typical GANs aren't actually zero-sum.
  - Original minimax cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

  - Modified generator cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$$

  - This fixes the saturation problem.
- A variant of GANs called the Wasserstein GAN (WGAN) uses a genuine minimax formulation, and is also well-motivated from a statistical standpoint.
  - While it doesn't seem to work quite as well as ordinary GANs in practice, it's a useful test case for minimax optimization algorithms.

# Differentiable Games

# Differentiable Games

- A differentiable game involves two or more players, indexed by $i$. We use $-i$ as a shorthand to denote all players *except* Player $i$.
- Each player chooses a parameter vector $\mathbf{z}_i$, and wants to minimize its own cost function $f_i(\mathbf{z}_i, \mathbf{z}_{-i})$.
- If the $f_i$ are all equal, this is a perfectly cooperative game, and essentially reduces to ordinary optimization.
- The next simplest case is the minimax, or perfectly competitive, setting, as with GANs. Here, there are two players, typically denoted $\mathbf{x}$ and $\mathbf{y}$, and the losses are zero-sum, i.e., $f_2(\mathbf{x}, \mathbf{y}) = -f_1(\mathbf{x}, \mathbf{y})$.
  - I.e., player $\mathbf{x}$ minimizes a function $f$ and player $\mathbf{y}$ maximizes $f$.

# Differentiable Games

- In the most basic dynamics, each player simultaneously updates their parameters opposite the gradient direction for their own loss, treating the other players as fixed:

$$\mathbf{z}_i^{(k+1)} \leftarrow \mathbf{z}_i^{(k)} - \eta \nabla_{\mathbf{z}_i} f_i(\mathbf{z}_i^{(k)}, \mathbf{z}_{-i}^{(k)}) \qquad \text{for all } i.$$

- In the minimax setting, this is known as gradient descent-ascent (GDA):

$$\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \eta \nabla_{\mathbf{x}} f(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$$
$$\mathbf{y}^{(k+1)} \leftarrow \mathbf{y}^{(k)} + \eta \nabla_{\mathbf{y}} f(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$$

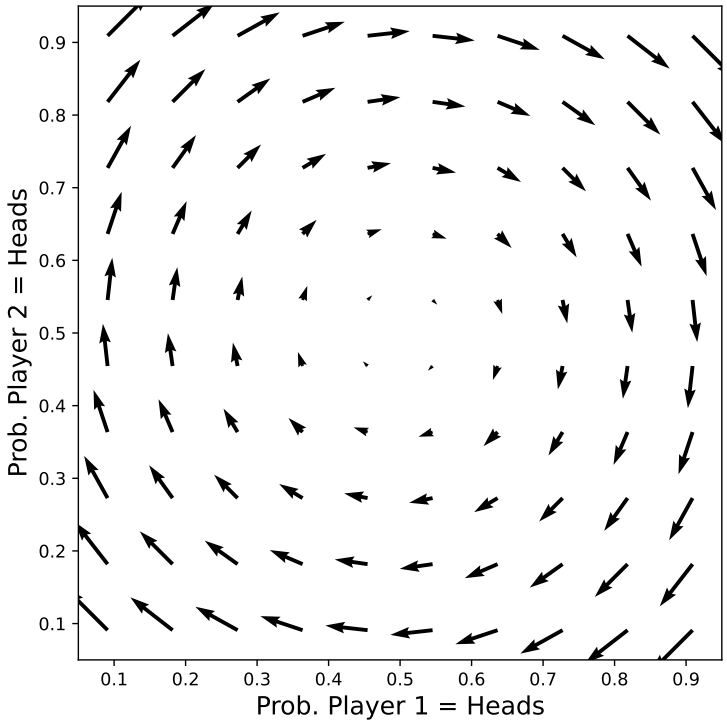
- As before, we can also consider the gradient flow:

$$\dot{\mathbf{z}}_i = -\eta \nabla_{\mathbf{z}_i} f_i(\mathbf{z}_i, \mathbf{z}_{-i}) \qquad \text{for all } i.$$

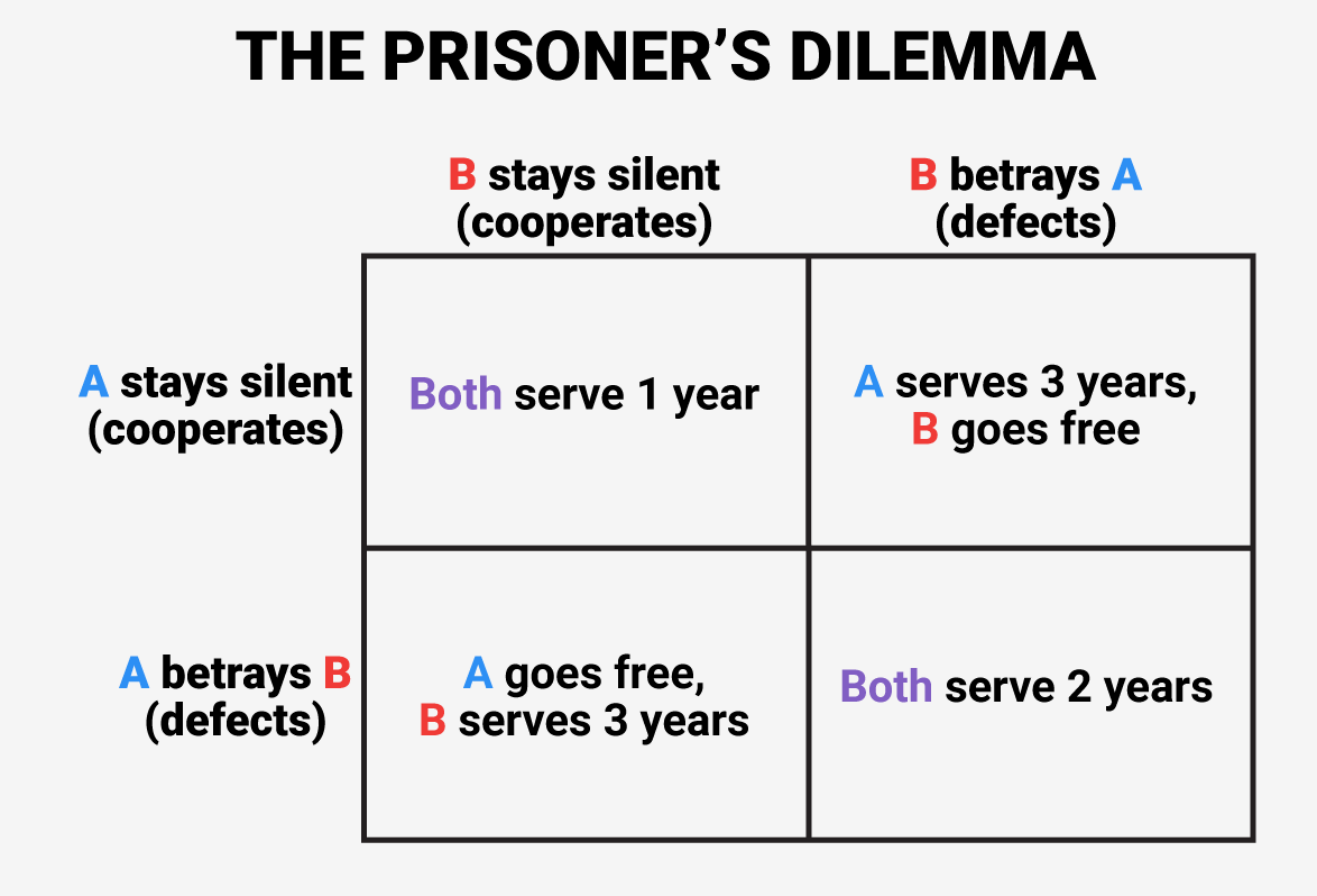
# Differentiable Games

**Example: Matching Pennies**

- Each player secretly turns a penny to Heads or Tails, and then reveals the choice. If the pennies match, then player $\mathbf{x}$ wins, otherwise $\mathbf{y}$ wins.
- This is a zero-sum game.
- Each player chooses a mixed strategy parameterized by $z_i \in [0, 1]$, the probability of choosing Heads.



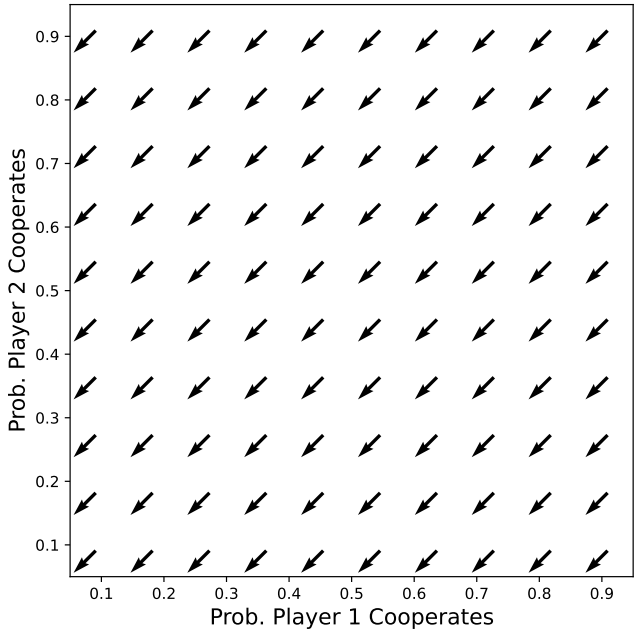
**Example: Prisoner's Dilemma**



# Differentiable Games

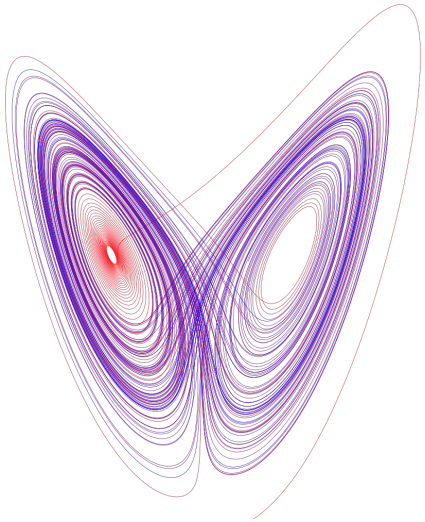
**Example: Prisoner's Dilemma**

- Each player chooses a mixed strategy parameterized by $z_i \in [0, 1]$, the probability of cooperation.
- This is a non-zero-sum game. E.g., if both players cooperate, then they're both better off than if they both defect.
- However, defection is a dominant strategy, i.e. each player is better off defecting regardless of the opponent's strategy.
- Gradient dynamics lead to mutual defection:



# Differentiable Games

- Remember the Lorenz system?



$$\dot{x} = a(y - x)$$
$$\dot{y} = x(b - z) - y$$
$$\dot{z} = xy - cz$$

Image: By Dschwen - Own work, CC BY 2.5 https://commons.wikimedia.org/w/index.php?curid=494694

- It's easy to write down a differentiable game such that the gradient flow has these dynamics:

$$f_x(x, y, z) = -a(xy - \tfrac{1}{2}x^2)$$
$$f_y(x, y, z) = -xy(b - z) + \tfrac{1}{2}y^2$$
$$f_z(x, y, z) = -xyz + \tfrac{c}{2}z^2$$

# Solving Differentiable Games

# Solving Differentiable Games

- In optimization, we were searching for a (local) optimum. What's the analogue for differentiable games?
- Assume for simplicity that the action space is unconstrained.
- The most basic requirement is to be at a fixed point:

$$\nabla_{\mathbf{z}_i} f_i(\mathbf{z}_i, \mathbf{z}_{-i}) = \mathbf{0} \qquad \text{for all } i$$

- A Nash equilibrium occurs when each player minimizes their own loss, i.e. no player has an incentive to deviate.

$$\mathbf{z}_i^* \in \arg\min_{\mathbf{z}_i} f_i(\mathbf{z}_i, \mathbf{z}_{-i}^*)$$

- This may be too much to ask if each player's loss is nonconvex (and hence we can't expect to find global optima).
  - A local Nash equilibrium occurs which each player minimizes their loss within a neighborhood.

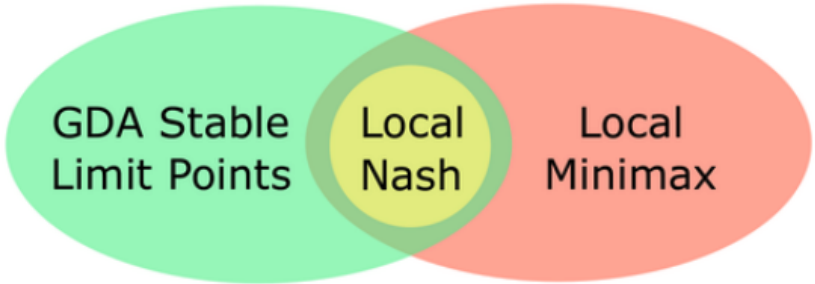
# Solving Differentiable Games

- In the minimax case, we can distinguish several solution concepts:
  - Local Nash equilibrium (as before):

$$\mathbf{x}^* \in \arg\min_{\mathbf{x} \in \mathcal{N}(\mathbf{x}^*)} f(\mathbf{x}, \mathbf{y}^*)$$

$$\mathbf{y}^* \in \arg\max_{\mathbf{y} \in \mathcal{N}(\mathbf{y}^*)} f(\mathbf{x}^*, \mathbf{y})$$

  - Local minimax, a solution concept for a sequential game where $\mathbf{x}$ moves first. (Sequential games are covered next week.)
  - Stable limit points of GDA, i.e. fixed points where the GDA flow is convergent.
- Amazingly, these solution concepts are not equivalent!

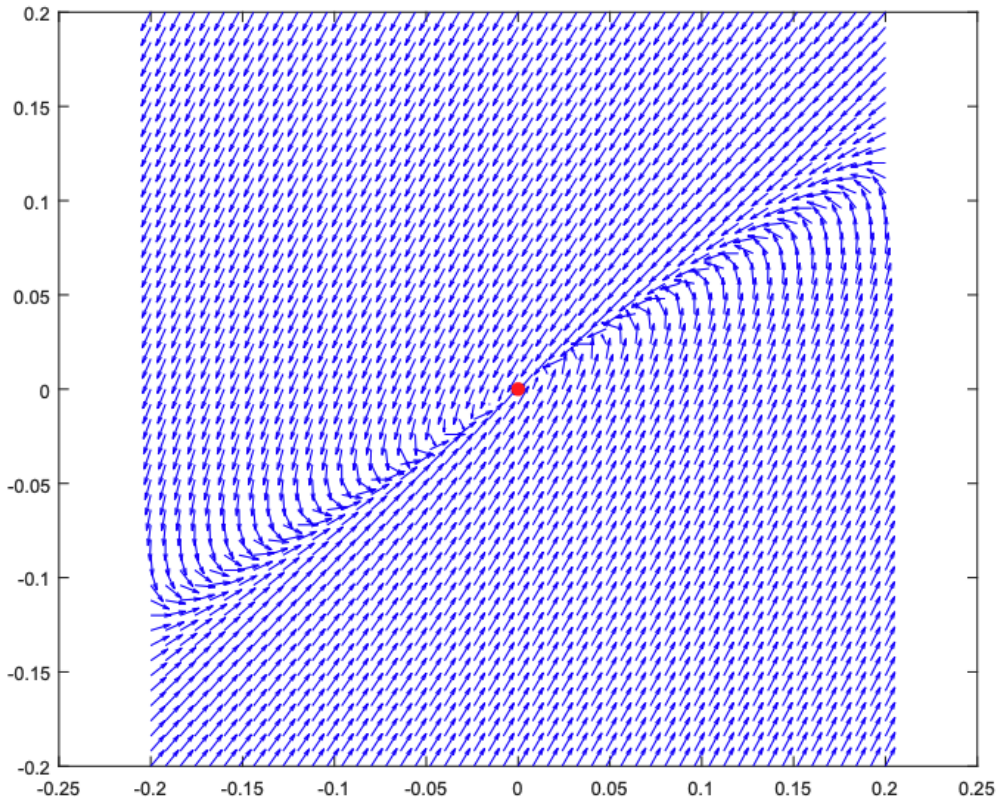



Wang et al., "On solving minimax optimization locally"

# Solving Differentiable Games

- The following function has a stable limit point of GDA which isn't a local Nash equilibrium:

$$f(x, y) = -\tfrac{1}{8}x^2 - \tfrac{1}{2}y^2 + \tfrac{6}{10}xy^{10}$$



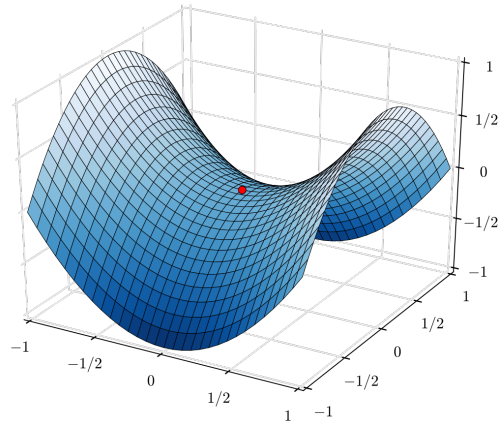

Daskalakis and Panageas, "The limit points of (optimistic) gradient descent in min-max optimization"

# Solving Differentiable Games

- To get efficient convergence guarantees, we need stronger assumptions.
- Let's focus on two-player, strongly-convex strongly-concave, zero-sum games.

$$\min_{\mathbf{x}} \max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$$

(many insights carry over to more general settings)




- Strong duality (minimax theorem) holds, i.e.,

$$\min_{\mathbf{x}} \max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{y}} \min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})$$

- Local Nash equilibrium is global and it is unique.
- Even for this simple setting, convergence can be slow because the "rotational force" necessitate extremely small learning rates.

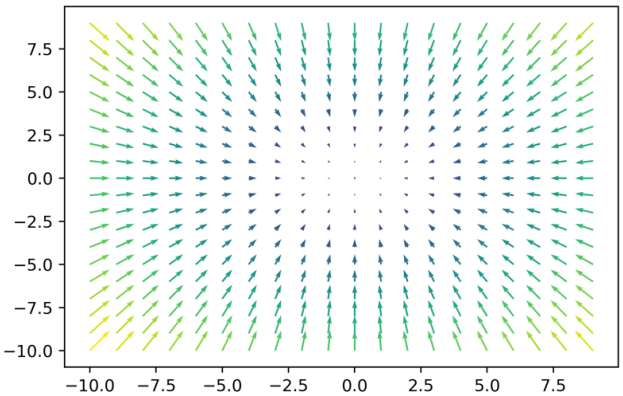
# Solving Differentiable Games

- Consider the general dynamics:

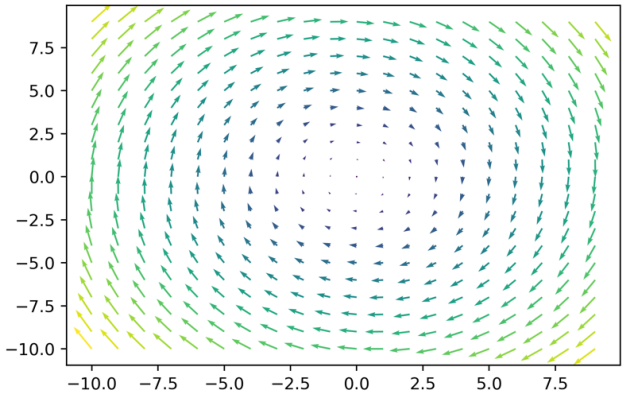
$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - \eta F(\mathbf{z}^{(k)})$$

(where $F$ is a vector field)

- Linear case: $F(\mathbf{z}) = \mathbf{Hz}$
  - **Minimization: H** is symmetric and all eigenvalues are real
  - **Differentiable Games: H** is non-symmetric and can have complex eigenvalues (with large imaginary parts)




$\min f(x, y) = 0.5x^2 + 0.5y^2$




$\min_x \max_y f(x, y) = 0.5x^2 + 10xy - 0.5y^2$

# Simultaneous Gradient Descent-Ascent

# Simultaneous Gradient Descent-Ascent

- Now I'll refer to GDA as Simultaneous GDA (Sim-GDA) to emphasize that both players update simultaneously:

$$\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \eta \nabla_{\mathbf{x}} f(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$$
$$\mathbf{y}^{(k+1)} \leftarrow \mathbf{y}^{(k)} + \eta \nabla_{\mathbf{y}} f(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$$

- We can compactly write it as $\mathbf{z}^{(k+1)} \leftarrow \mathbf{z}^{(k)} - \eta F(\mathbf{z}^{(k)})$ where $\mathbf{z} = [\mathbf{x}^\top, \mathbf{y}^\top]^\top$ and $F(\mathbf{z}) = [\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})^\top, -\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})^\top]^\top$.
- Assuming a quadratic problem $f(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{x}^\top \mathbf{B}\mathbf{y} - \frac{1}{2}\mathbf{y}^\top \mathbf{C}\mathbf{y}$
  - We have the dynamics:

$$\mathbf{z}^{(k+1)} \leftarrow (\mathbf{I} - \eta \mathbf{H})\mathbf{z}^{(k)}$$

where $\mathbf{z} = [\mathbf{x}^\top, \mathbf{y}^\top]^\top$ and $\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ -\mathbf{B}^\top & \mathbf{C} \end{bmatrix}$

- $\mathbf{H}$ is often called the game Hessian by analogy with the ordinary Hessian, though in general it's not symmetric.

# Convergence Analysis of Sim-GDA

- Setting: Smooth and strongly-monotone games
  - Lipschitz Smooth: a vector field $F$ is Lipschitz if for any $\mathbf{z}_1, \mathbf{z}_2$ and a parameter $L$:

$$\|F(\mathbf{z}_1) - F(\mathbf{z}_2)\| \leq L\|\mathbf{z}_1 - \mathbf{z}_2\|$$

  - Strongly Monotone: a vector field $F$ is strongly monotone if for any $\mathbf{z}_1, \mathbf{z}_2$ and a parameter $\mu$:

$$(F(\mathbf{z}_1) - F(\mathbf{z}_2))^\top (\mathbf{z}_1 - \mathbf{z}_2) \geq \mu\|\mathbf{z}_1 - \mathbf{z}_2\|^2$$

    - This property is analogous to strong convexity.
  - Condition number: $\kappa \triangleq \frac{L}{\mu}$
- Quadratic case: $F(\mathbf{z}) = \mathbf{H}\mathbf{z}$ where $\mathbf{H} \succeq \mu\mathbf{I}$ and $\|\mathbf{H}\| \leq L$

# Convergence Analysis of Sim-GDA

- Recall that the dynamics of Sim-GDA: $\mathbf{z}^{(k+1)} \leftarrow (\mathbf{I} - \eta\mathbf{H})\mathbf{z}^{(k)}$
- Its convergence rate is $\min_\eta \rho(\mathbf{I} - \eta\mathbf{H}) = \min_\eta \max_{\lambda \in \mathrm{Sp}(\mathbf{H})} \|1 - \eta\lambda\|$

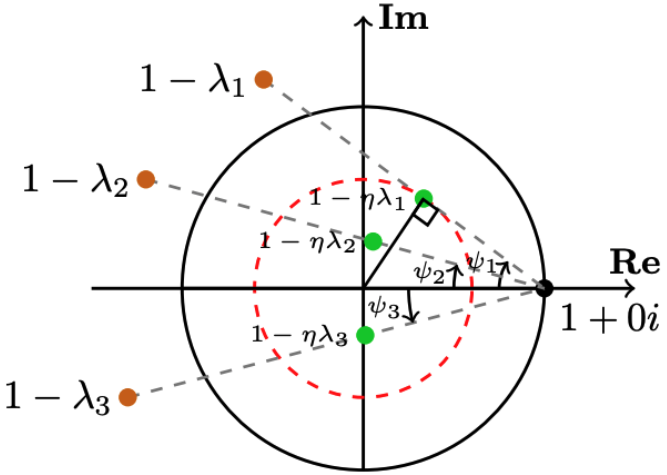
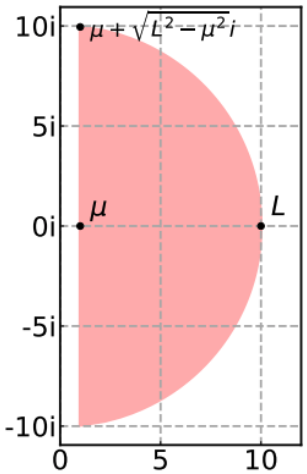



Image Credit: Negative Momentum for Improved Game Dynamics




Eigenvalues of $\mathbf{H}$

- The best convergence rate is limited by the eigenvalue $\lambda = \mu + \sqrt{L^2 - \mu^2}i$.
- The optimal convergence rate is $1 - \frac{1}{\kappa^2}$, which implies that Sim-GDA takes roughly $\mathcal{O}(\kappa^2)$ steps to converge. Recall that gradient descent only takes $\mathcal{O}(\kappa)$ steps to converge in minimizing a strongly-convex function!

Can we accelerate the convergence of Sim-GDA?

# Rotational Dynamics: Games and Momentum

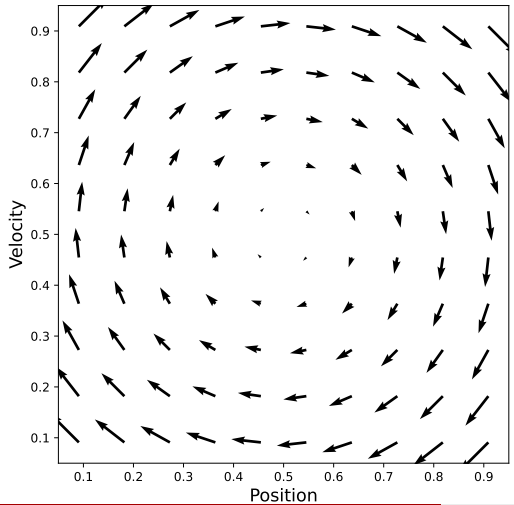
- The culprit for slow convergence is the complex eigenvalues, which cause rotational behavior.
- This is closely analogous to the momentum dynamics discussed last week:

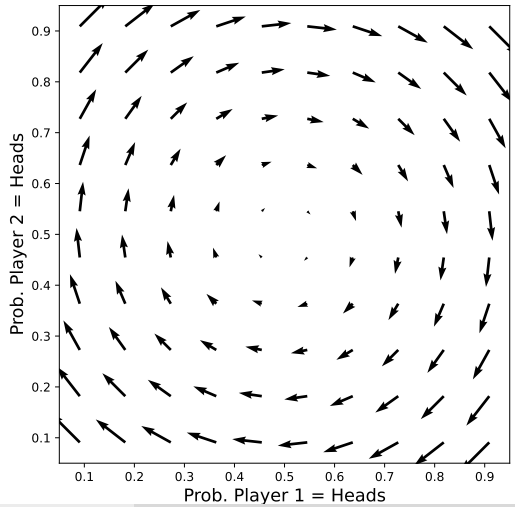
**Momentum**

$$\mathcal{J}(w) = \tfrac{1}{2}w^2$$
$$\dot{v} = -\nabla\mathcal{J}(w)$$
$$\dot{w} = v$$

**Matching Pennies game**

$$f(x,y) = -xy - (1-x)(1-y)$$
$$\dot{x} = -\nabla_x f(x,y)$$
$$\dot{y} = \nabla_y f(x,y)$$

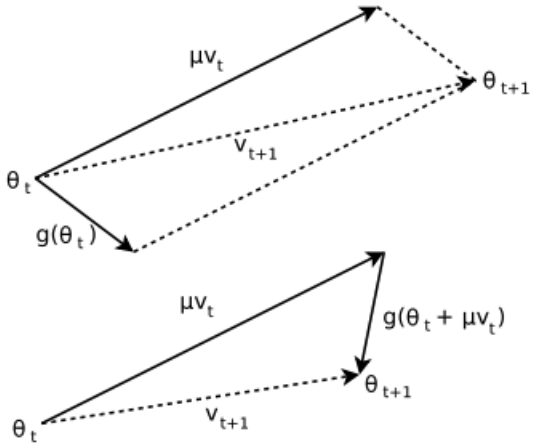


# Rotational Dynamics: Games and Momentum

Two ideas motivated by this momentum analogy:

1. If momentum creates oscillations, then can we remove oscillations by doing the opposite of momentum?
2. Nesterov Accelerated Gradient dampens the oscillations by computing the gradient at an extrapolated point. Can we do the same thing for differentiable games?




(Sutskever et al., 2013)

# Negative Momentum

- Negative momentum is basically Heavy-ball momentum with a negative damping value:

$$\mathbf{z}^{(k+1)} \leftarrow (1+\beta)\mathbf{z}^{(k)} - \beta\mathbf{z}^{(k-1)} - \eta F(\mathbf{z}^{(k)})$$

- Intuition: negative momentum reduces the imaginary parts of complex eigenvalues, and hence suppresses the rotational behaviour. (recall the rate of Sim-GDA was limited by the eigevalue $\lambda = \mu + \sqrt{L^2 - \mu^2}i$)
- Negative momentum converges in $\mathcal{O}(\kappa^{1.5})$ steps, which is slightly faster than Sim-GDA (recall the complexity of $\mathcal{O}(\kappa^2)$). However, this rate is suboptimal as some other algorithms converge in $\mathcal{O}(\kappa)$ steps.
- Proving this convergence rate is hard. Need to leverage the connection between Chebyshev polynomial and Heavy-ball momentum. See "*On the suboptimality of negative momentum for minimax optimization*".

# Negative Momentum

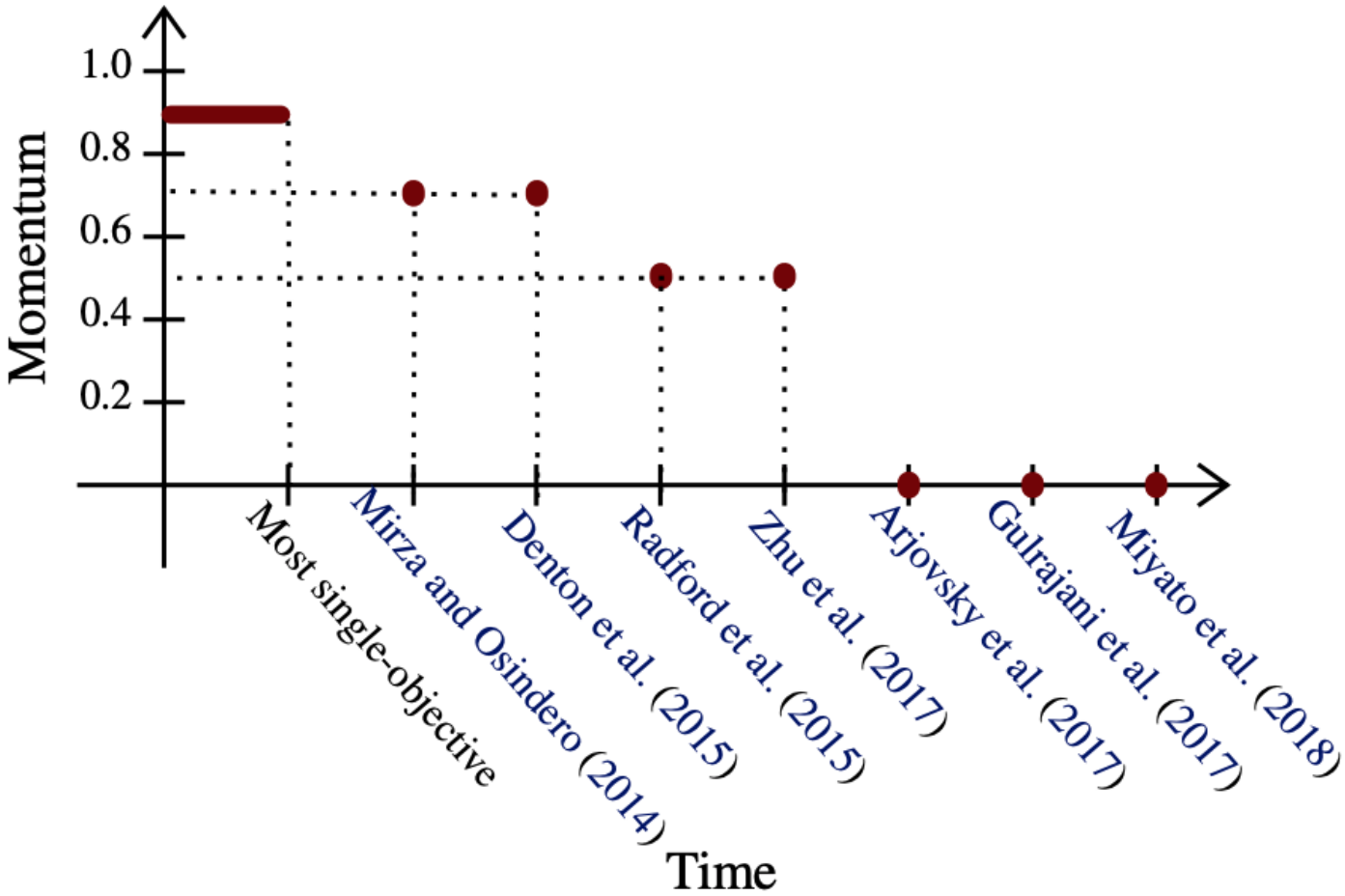



Image Credit: Negative Momentum for Improved Game Dynamics

# Extra-Gradient Method

- The Extra-gradient method computes the gradient with one-step lookahead (extrapolated gradient):

$$\mathbf{z}^{(k+1/2)} \leftarrow \mathbf{z}^{(k)} - \eta F(\mathbf{z}^{(k)})$$
$$\mathbf{z}^{(k+1)} \leftarrow \mathbf{z}^{(k)} - \eta F(\mathbf{z}^{(k+1/2)})$$

- It was first proposed by Korpelevich in 70's to solve monotone variational inequality problem.
- It was recently re-introduced by Gidel, et.al (2019) and Mokhtari, et.al (2019) in the context of differentiable games and minimax optimization.
- Over the last three years, more than 10 papers discussed the extra-gradient method in different settings.

# Extra-Gradient Method

- Extra-gradient can be motivated as an approximation to the proximal point method (Rockafeller, 1976), which is an implicit method:

$$\mathbf{z}^{(k+1)} \leftarrow \mathbf{z}^{(k)} - \eta F(\mathbf{z}^{(k+1)})$$

- Intuition: compute gradient at a future point, but it is not implementable in many cases (chicken and egg situtation).
- In optimization, the proximal point method is largely regarded as a "conceptual" guiding principle for accelerating optimization algorithms. NAG can be derived from the proximal point method (see "*From Proximal Point Method to Nesterov's Acceleration*" paper).
- It can be shown that for smooth and strongly monotone games, the proximal point method converges linearly for any $\eta$:

$$\|\mathbf{z}^{(k)} - \mathbf{z}^*\|^2 \leq \left(\frac{1}{1+2\eta\mu}\right)^k \|\mathbf{z}^{(0)} - \mathbf{z}^*\|^2$$

check out the proof in "A Unified Analysis of First-Order Methods for Smooth Games via Integral Quadratic Constraints"

# Extra-Gradient Method

- The extra-gradient method computes the gradient with one-step lookahead:

$$\mathbf{z}^{(k+1/2)} \leftarrow \mathbf{z}^{(k)} - \eta F(\mathbf{z}^{(k)})$$
$$\mathbf{z}^{(k+1)} \leftarrow \mathbf{z}^{(k)} - \eta F(\mathbf{z}^{(k+1/2)})$$

- Intuition: approximate $F(\mathbf{z}^{(k+1)})$ with $F(\mathbf{z}^{(k+1/2)})$, hoping to inherit the convergence properties of proximal point method.
- Formally, it can shown that starting with the same $\mathbf{z}^{(k)}$, the solution of extra-gradient $\mathbf{z}_{\text{eg}}^{(k+1)}$ after one step is relatively close to the solution of proximal point method $\mathbf{z}_{\text{ppm}}^{(k+1)}$:

$$\|\mathbf{z}_{\text{eg}}^{(k+1)} - \mathbf{z}_{\text{ppm}}^{(k+1)}\| \leq o(\eta^2)$$

- Under the same set of assumptions, the extra-gradient method converges linearly

$$\|\mathbf{z}^{(k)} - \mathbf{z}^*\|^2 \leq \left(1 - \frac{1}{2\kappa}\right)^k \|\mathbf{z}^{(0)} - \mathbf{z}^*\|^2$$

see more details in "A Unified Analysis of Extra-gradient and Optimistic Gradient Methods for Saddle Point Problems: Proximal Point Approach"

# Extra-Gradient Method

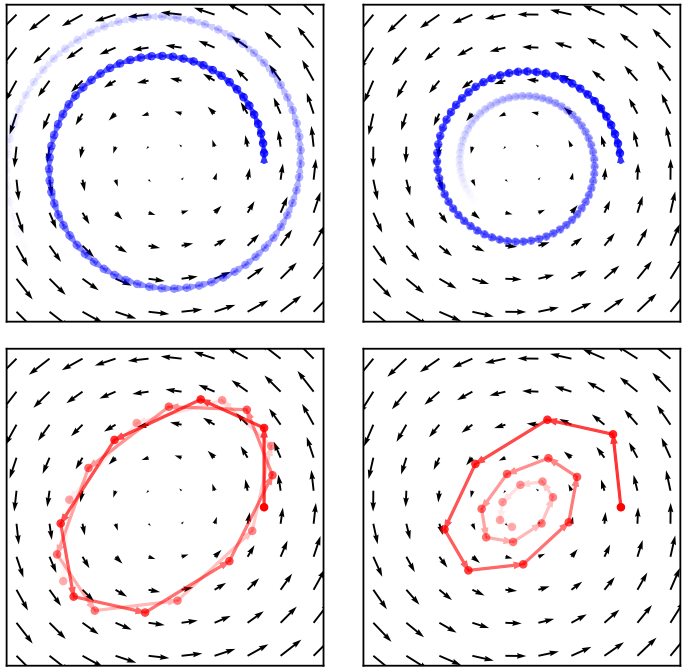
- Negative momentum, Extra-gradient, and related methods are easy to implement and have provably faster convergence rates than Sim-GDA in the strongly-convex-strongly-concave setting.
- Still, they're not used much in practice for models like GANs.
- Observation: models are typically trained with alternating, rather than simultaneous, GDA updates.
  - Maybe alternating GDA is more efficient?

# Alternating Gradient Descent-Ascent

- Alt-GDA updates multiple players sequentially:

$$\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \eta \nabla_{\mathbf{x}} f(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$$
$$\mathbf{y}^{(k+1)} \leftarrow \mathbf{y}^{(k)} + \eta \nabla_{\mathbf{y}} f(\mathbf{x}^{(k+1)}, \mathbf{y}^{(k)})$$

- Alt-GDA converges with $\mathcal{O}(\kappa)$ steps (which matches the coarse lower-bound).
- The result could be extended to n-player setting (ongoing work).
- In the bilinear case, Alt-GDA is a symplectic integrator applied on the continuous dynamic.



**Left**: $f(x, y) = 10xy$;
**Right**: $0.5x^2 + 10xy - 0.5y^2$;
**Top**: Sim-GDA;
**Bottem**: Alt-GDA.

- The discussion of simultaneous and alternating updates dates back to the Jacobi and Gauss-Seidel methods in numerical linear algebra, see the celebrated Stein-Rosenberg theorem.

see more details in "Don't fix what ain't broke: near-optimal local convergence of alternating gradient descent-ascent for minimax optimization"

# Alternating Gradient Descent-Ascent

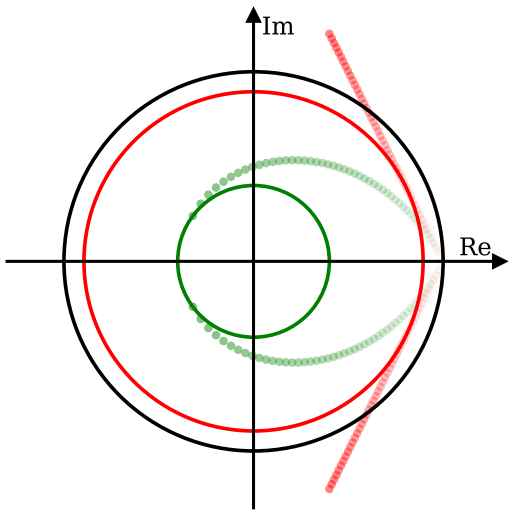
- Consider the quadratic problem $f(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{x}^\top \mathbf{B}\mathbf{y} - \frac{1}{2}\mathbf{y}^\top \mathbf{C}\mathbf{y}$.
- We have Alt-GDA as the following form:

$$\begin{bmatrix} \mathbf{x}^{(k+1)} \\ \mathbf{y}^{(k+1)} \end{bmatrix} \leftarrow \underbrace{\begin{bmatrix} \mathbf{I} - \eta\mathbf{A} & -\eta\mathbf{B} \\ \eta\mathbf{B}^\top(\mathbf{I} - \eta\mathbf{A}) & \mathbf{I} - \eta\mathbf{C} - \eta^2\mathbf{B}^\top\mathbf{B} \end{bmatrix}}_{\mathbf{J}_{\text{Alt}}} \begin{bmatrix} \mathbf{x}^{(k)} \\ \mathbf{y}^{(k)} \end{bmatrix}$$

- Recall Sim-GDA dynamcis for the quadratic case:

$$\begin{bmatrix} \mathbf{x}^{(k+1)} \\ \mathbf{y}^{(k+1)} \end{bmatrix} \leftarrow \underbrace{\begin{bmatrix} \mathbf{I} - \eta\mathbf{A} & -\eta\mathbf{B} \\ \eta\mathbf{B}^\top & \mathbf{I} - \eta\mathbf{C} \end{bmatrix}}_{\mathbf{J}_{\text{Sim}}} \begin{bmatrix} \mathbf{x}^{(k)} \\ \mathbf{y}^{(k)} \end{bmatrix}$$

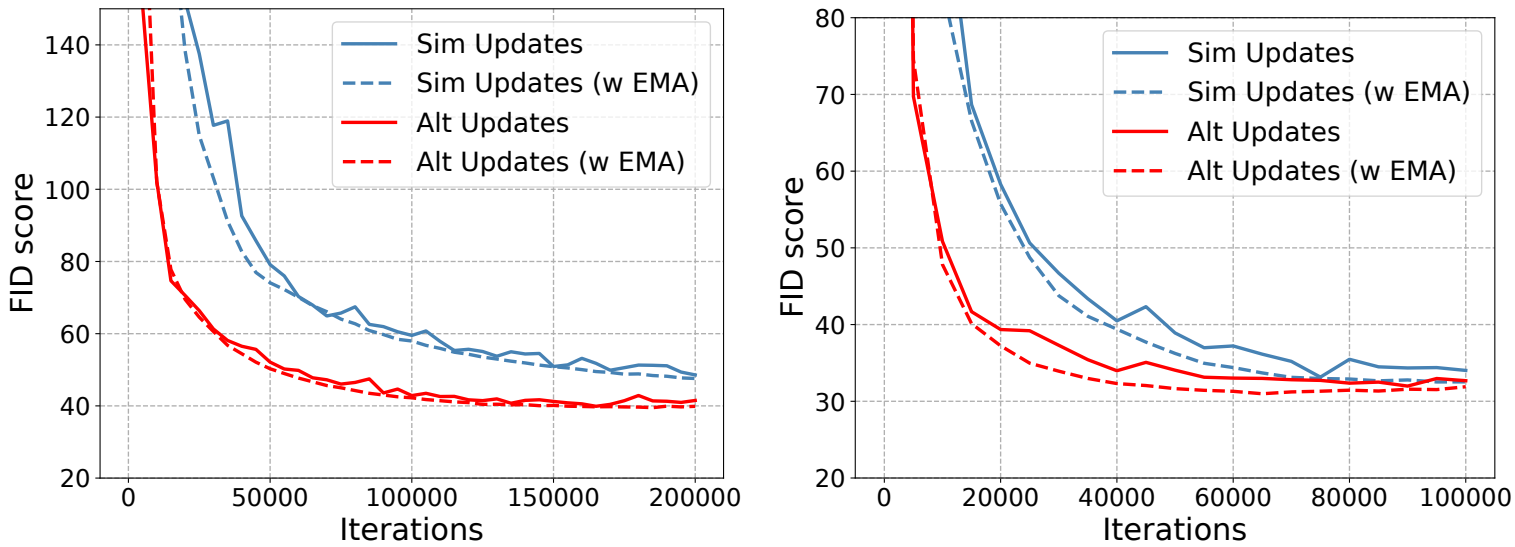
- Alt-GDA allows us to use **larger** step sizes. The optimal step size for Sim-GDA is $\frac{\mu}{L^2}$ while the optimal one for Alt-GDA is roughly $\frac{1}{L}$.




Eigenvalues of $\mathbf{J}_{\text{Alt}}$ (green dots) and $\mathbf{J}_{\text{Sim}}$ (red dots) for the minimax problem $f(x, y) = 0.3x^2 + 1.2xy - 0.3y^2$. Their trajectories as $\eta$ sweeps in $[0, 1]$ are shown from light colors to dark colors

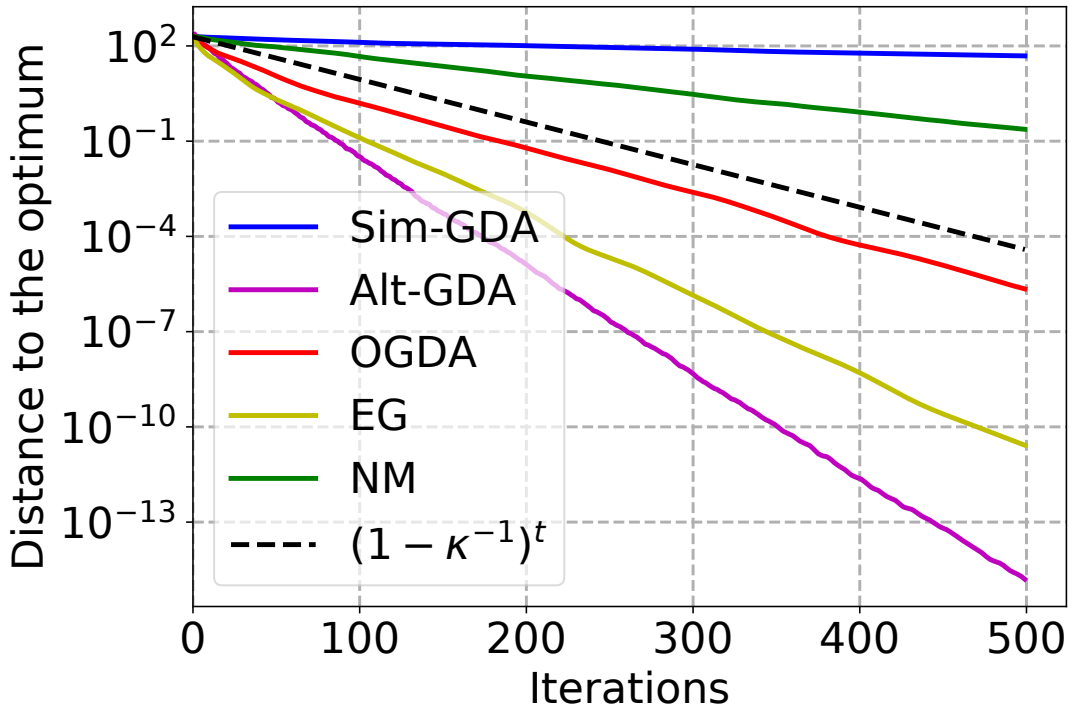
# Alternating Gradient Descent-Ascent

- We are implicitly using alternating updates in GAN training.




DCGAN on CIFAR-10. **Left:** SGD as base optimizer; **Right:** AMSGrad as base optimizer.

# Comparison between different algorithms




Distances to the optimum as a function of iterations on a quadratic minimax problem.

# Opponent Shaping

# Opponent Shaping

- So far, our updates have all treated the opponents' strategies as fixed. Our only concern was faster convergence.
- However, some equilibria may be better than others. It can be advantageous to consider the opponent's learning process in order to land in a more advantageous equilibrium.
- Recall: in the (single-shot) Prisoner's Dilemma, the unique Nash equilibrium is (Defect, Defect), which is suboptimal for both players.
- In the Iterated Prisoner's Dilemma (IPD), the players play each other for multiple rounds, and can condition their action on the opponent's previous action.
- Unlike the single-shot case, it's possible for cooperation to emerge in the IPD.

# Opponent Shaping

- Consider the Tit-for-Tat (TFT) strategy:
  - Cooperate in the first round.
  - In subsequent rounds, copy the opponent's action from the previous round.
- What is your optimal strategy if your opponent is playing TFT?
- Two players playing TFT will cooperate with each other, and hence improve the social welfare (average payoff). This is a Nash equilibrium.
  - This is an instance of reciprocity: agents settle on an equilibrium where cooperation is individually beneficial.

# Opponent Shaping

- To make the IPD into a differentiable game, consider the following tabular policy representation:
  - There are 4 possible observations from the previous time step, depending if each player cooperated or defected.
  - The policy is represented as a vector in $\mathbb{R}^4$, with the probability of cooperation for each of the 4 possible observations.
- GDA usually lands in the bad Nash equilibrium.
  - Defection is usually the best response to a fixed opponent policy, unless that policy specifically rewards cooperation. Most randomly chosen policies don't do this.
  - Need to account for the opponent's learning behavior!

# Opponent Shaping

- Opponent shaping algorithms consider the *opponent's* learning rule when determining an agent's update.
- Ordinary gradient dynamics:

$$\mathbf{z}_1^{(k+1)} \leftarrow \mathbf{z}_1^{(k)} - \eta \nabla_{\mathbf{z}_1} f_1(\mathbf{z}_1^{(k)}, \mathbf{z}_2^{(k)}) \qquad \text{for all } i.$$

- Learning with Opponent Learning Awareness (LOLA) differentiates through one gradient descent update of the opponent:

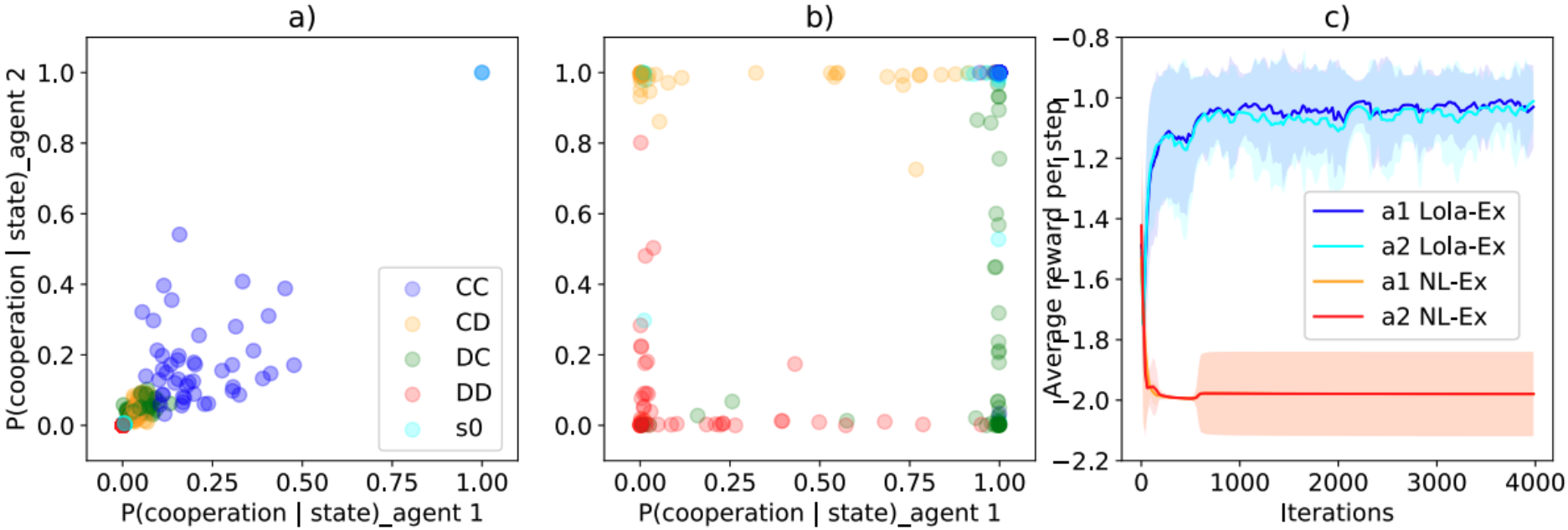
$$\mathbf{z}_1^{(k+1)} \leftarrow \mathbf{z}_1^{(k)} - \eta \nabla_{\mathbf{z}_1} \left[ f_1(\mathbf{z}_1, \mathbf{z}_2^{(k)} + \Delta \mathbf{z}_2(\mathbf{z}_1)) \right]$$

$$\Delta \mathbf{z}_2(\mathbf{z}_1) \triangleq -\eta' \nabla_{\mathbf{z}_2} f_2(\mathbf{z}_1, \mathbf{z}_2^{(k)})$$

- **Intuition:** the opponent's learning gradient update will tend to move towards its best response. So LOLA favors parameters for which the opponent's best response is to cooperate.

# Opponent Shaping

- In the Iterated Prisoner's Dilemma, gradient dynamics ("naive learning") converges to defection, while LOLA converges to Tit-for-Tat.
- **Left:** conditional probabilities of cooperation for naive learner.
- **Middle:** conditional probabilities of cooperation for LOLA.
- **Right:** average returns when playing against a LOLA agent.




Foerster et al., "Learning with opponent learning awareness"

# Opponent Shaping

- Most of this lecture has focused on simultaneous games, where the relevant solution concept is a Nash equilibrium.
- In a Nash equilibrium, all players choose their best response assuming the opponents' parameters are *fixed*.
- LOLA doesn't find Nash equilibria in general, since each player is considering how the opponents will respond to their policy. It's hard to characterize the solutions it actually converges to.
- It sort of hints at sequential games, which we'll cover in the next 2 weeks.
  - One player (the leader) goes first, and has to consider how the other player will react to its choice of parameters.