Reading group: Latent Optimized GANs (Game theory brings guns to GANs)

Michal Šustr

Dept. of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University

michal.sustr@aic.fel.cvut.cz

January 9th, 2020

Reading group VKR

The reading group is based on two papers:

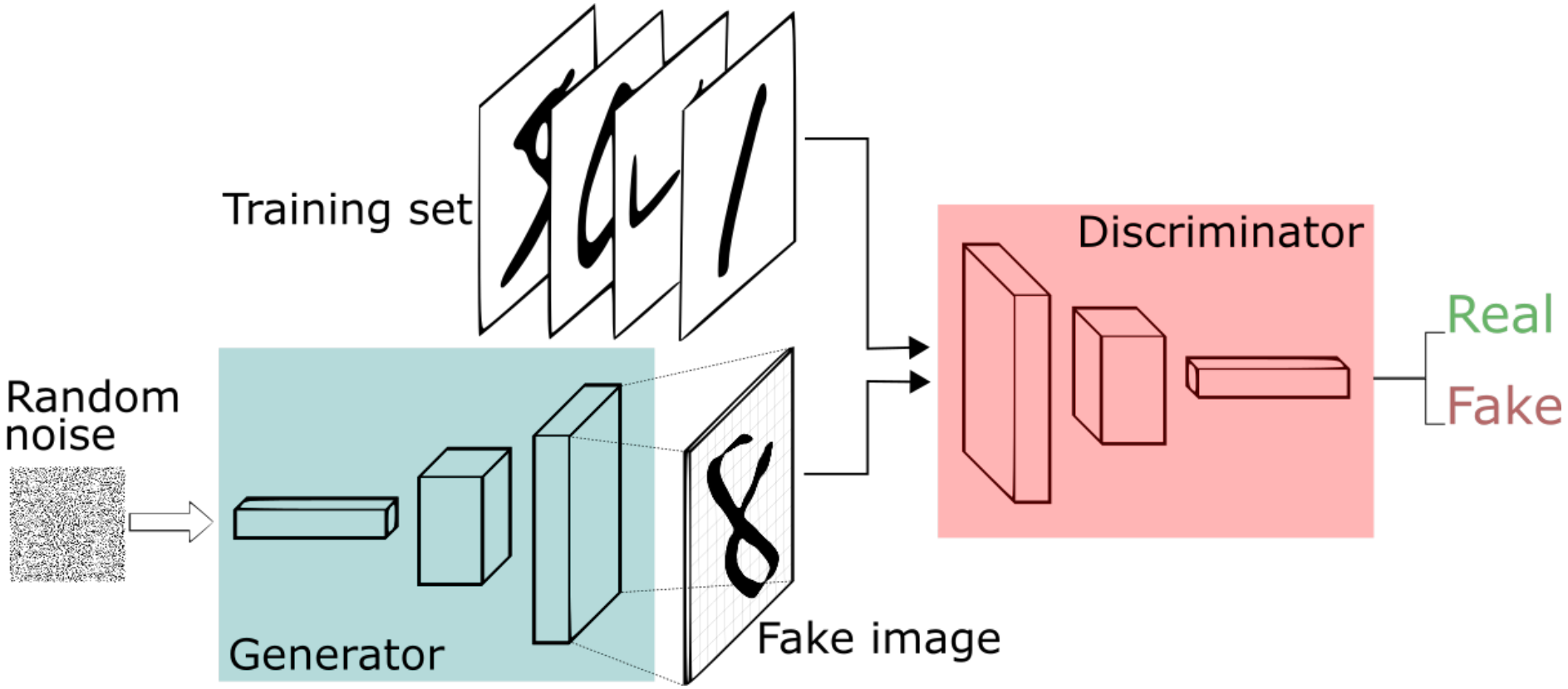
- **Differentiable Game Mechanics**
  A. Letcher, D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, T. Graepel
  Journal of Machine Learning Research (JMLR), v20 (84) 1-40, 2019
  Arxiv, Reading group
- **LOGAN: Latent Optimisation for Generative Adversarial Networks**
  Y. Wu, J. Donahue, D. Balduzzi, K. Simonyan, T. Lillicrap
  Submission to ICLR2020 conference (rejected)
  Openreview, Arxiv

Other relevant literature:

- **Stable Opponent Shaping in Differentiable Games**
  A. Letcher, J. Foerster, D. Balduzzi, T. Rocktaschel, S. Whiteson
  ICLR 2019 poster, Openreview, Arxiv
- **Uncoupled Dynamics Do Not Lead to Nash Equilibrium**
  Andreu Mas-Colell and Sergiu Hart, Paper

NeurIPS 2019 workshop: Bridging Game Theory and Deep Learning

Motivation


Training set
Discriminator
Real
Fake
Random noise
Generator
Fake image

Motivation

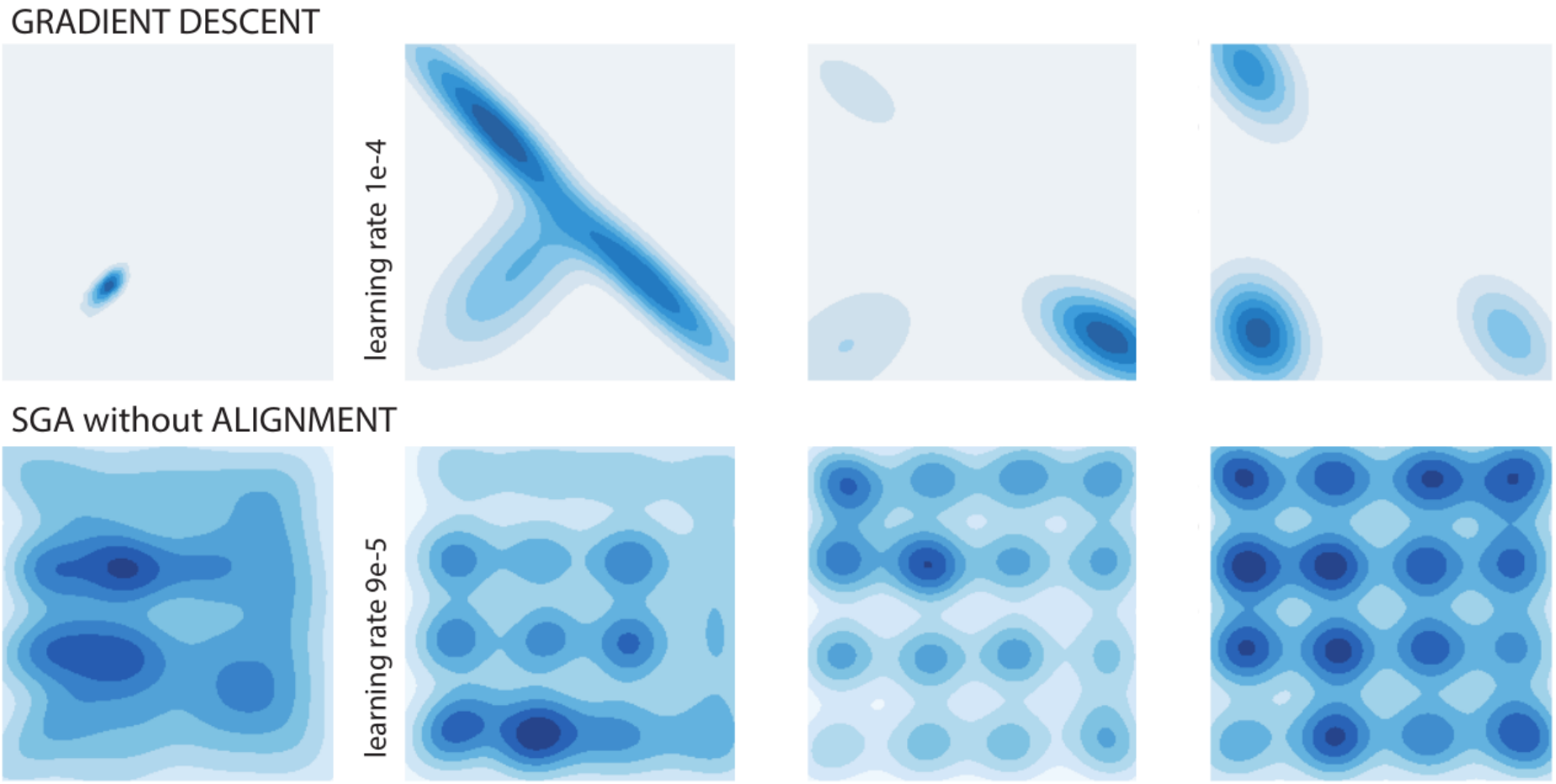
GANs involve interplay of multiple objectives:

$$\min_{\theta_D} \max_{\theta_G} \mathbb{E}_{x\sim p(x)} \left[h_D(D\left(x;\theta_D\right))\right] + \mathbb{E}_{z\sim p(z)} \left[h_G(D\left(G\left(z;\theta_G\right);\theta_D\right))\right]$$

The field of GT deals with such min-max optimization problems.

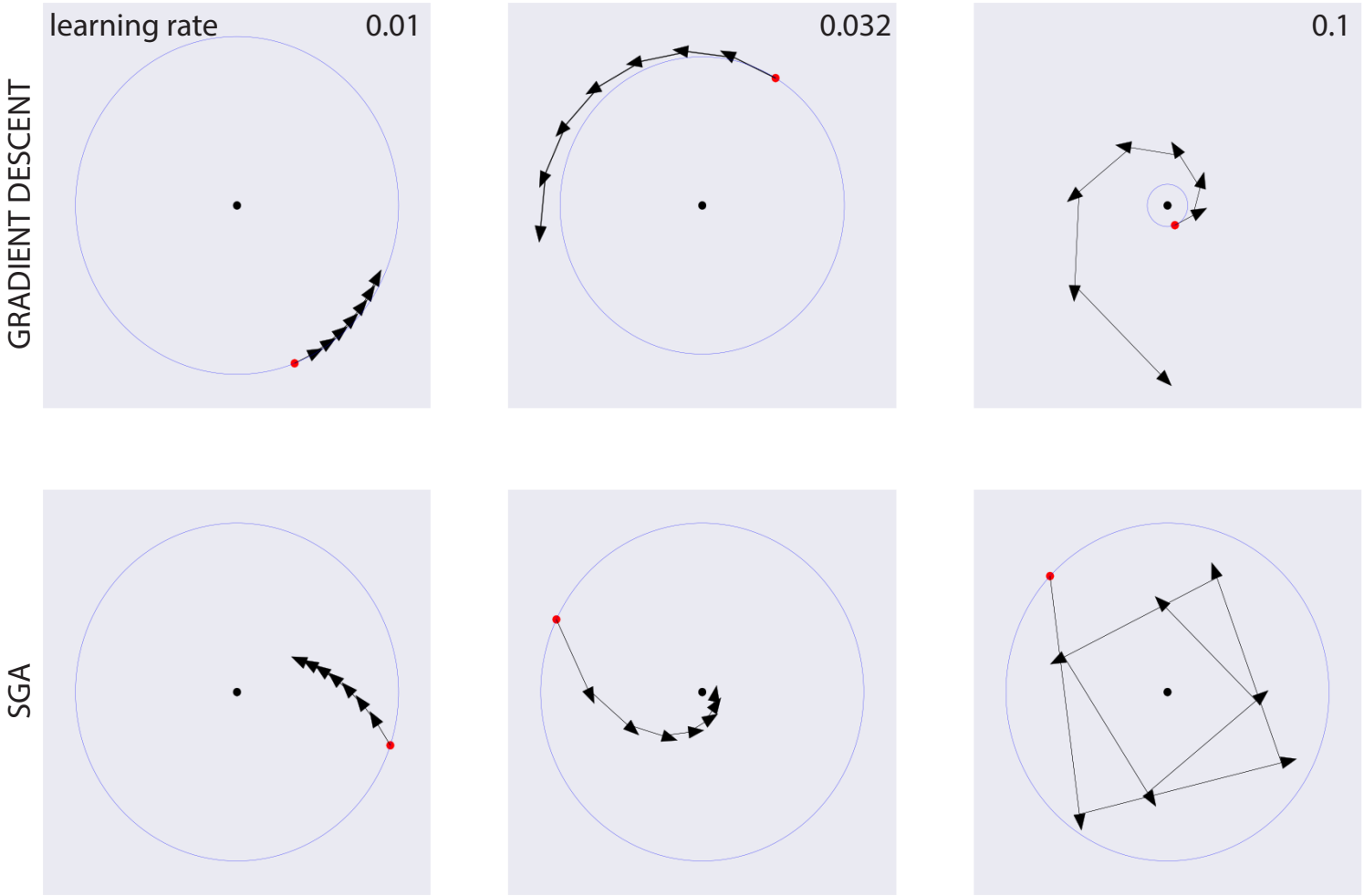
Motivation

We can use GT-inspired algorithms to improve the results! (mode collapse)



Motivation

Higher learning rates and convergence to local minima



Motivation

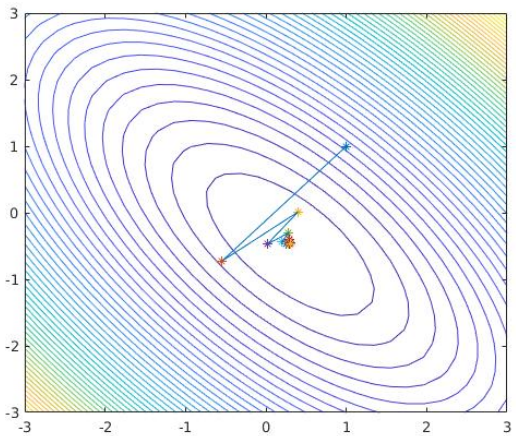
Applying latent optimization leads to more diverse image generation.

(**a**) (**b**)

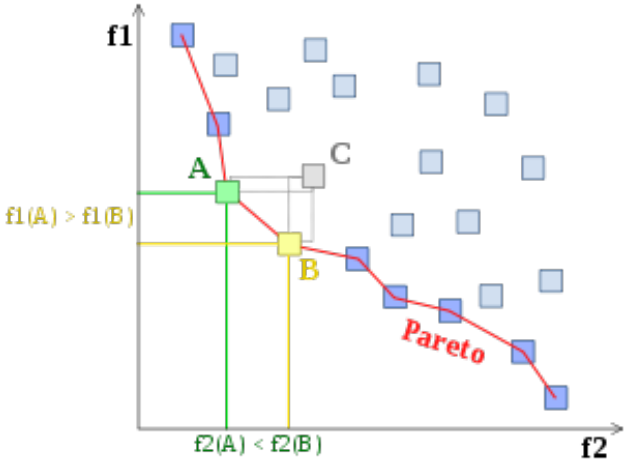
Figure: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with similarly high IS.

Broader application than just GANs

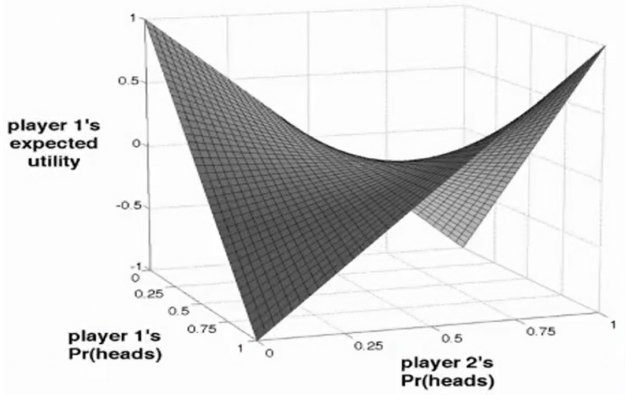
Normal optimization: single objective



Multiple objectives: pareto-optimality



Multiple objectives: equilibrium outcomes



Possible use-cases other than GANs:

Proximal gradient TD learning, multi-level optimization, synthetic gradients, hierarchical reinforcement learning, intrinsic curiosity, and imaginative agents.

Outline

Background

- GT basic concepts
- Computation of NE
- (Un)constrained games
- (Un)stable fixed points
- Dynamics in games
- Fixed points and equilibria

Differentiable Game Mechanics

- Game decomposition
- Optimization on decomposed games
- Symplectic gradient adjustment
- Experiments

LOGAN

- GANs in general
- Latent optimization
- Comparison to SGA
- Computing the hessian update
- Experiments

Normal form games

Normal form games:

Rock, Paper, Scissors	Rock	Paper	Scissors
Rock	0, 0	-1, 1	1, -1
Paper	1, -1	0, 0	-1, 1
Scissors	-1, 1	1, -1	0, 0

Solution concepts: Pareto efficient, (course) correlated eq., Stackelberg eq., **Nash eq.**, *Stable fixed points*, ...

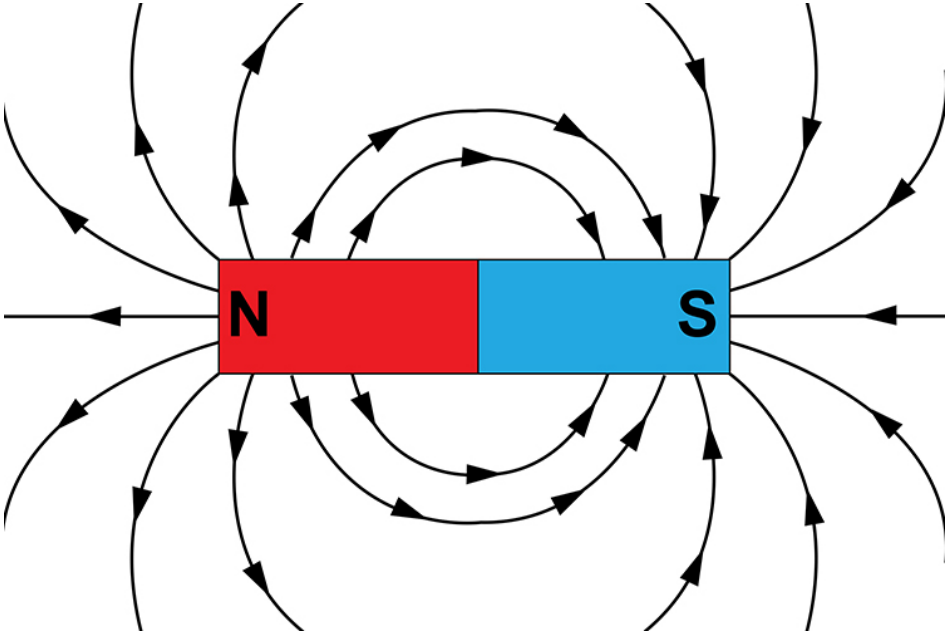
Nash equilibria

- Strategy profile where no player has incentive to deviate from their current strategy.

Nash equilibria

- Strategy profile where no player has incentive to deviate from their current strategy.
- Strategy profile that consists of strategies which are mutual best responses.

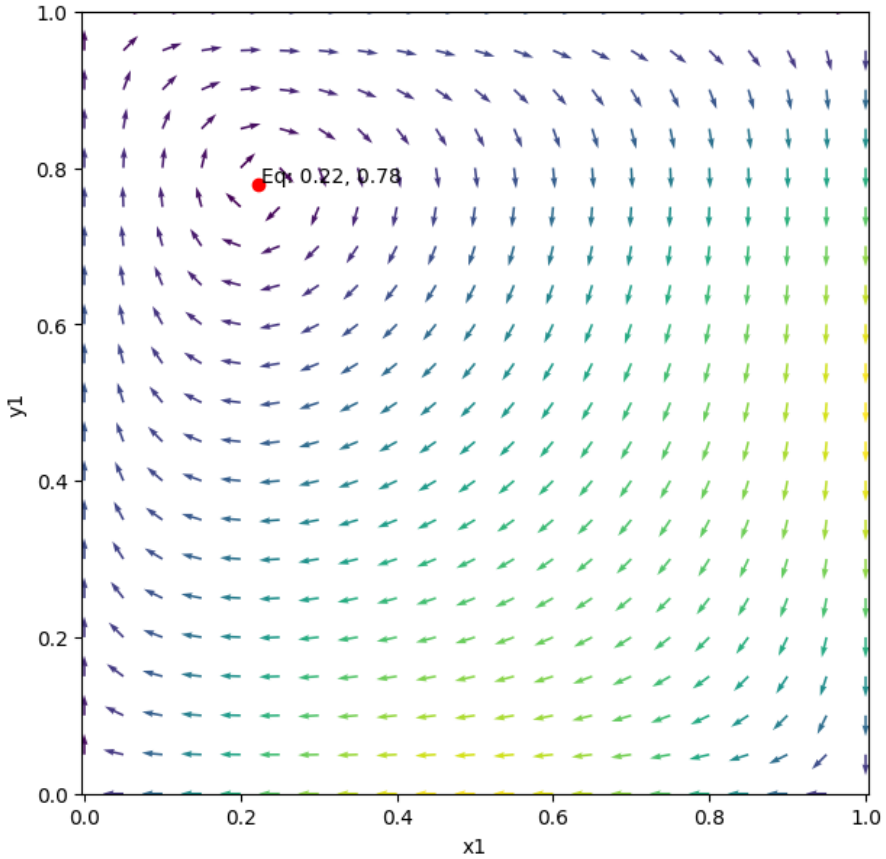
Dynamics, i.e. vector fields



What about learning in games?

Learning dynamics in games

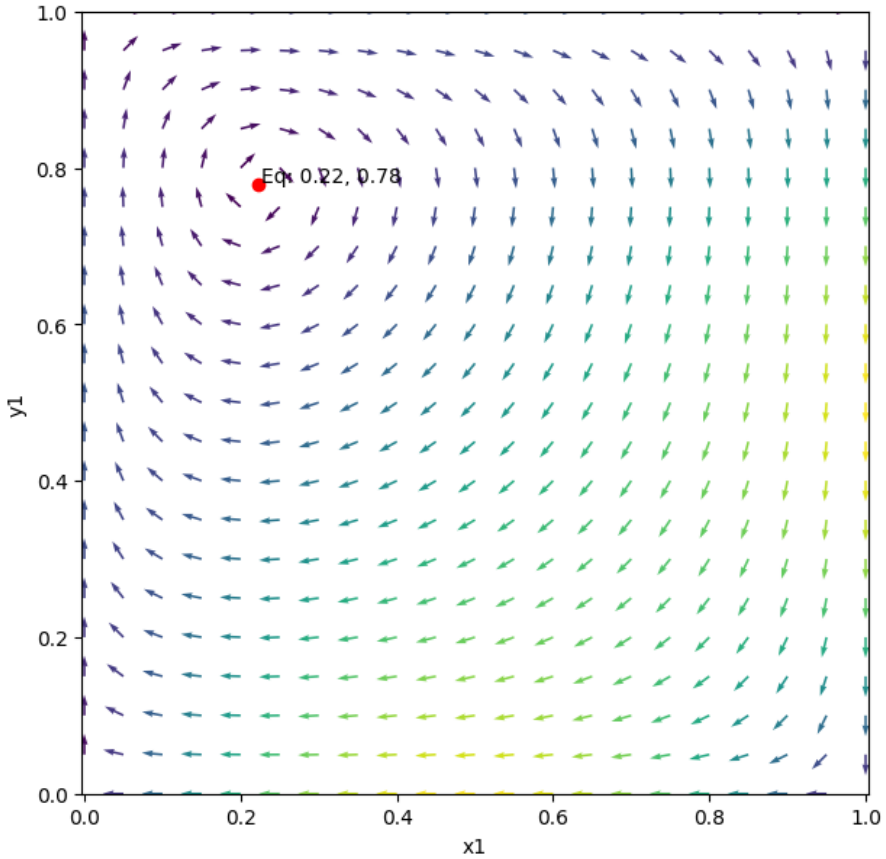
.



Iterative algorithms can be described with game dynamics:

Learning dynamics in games

.



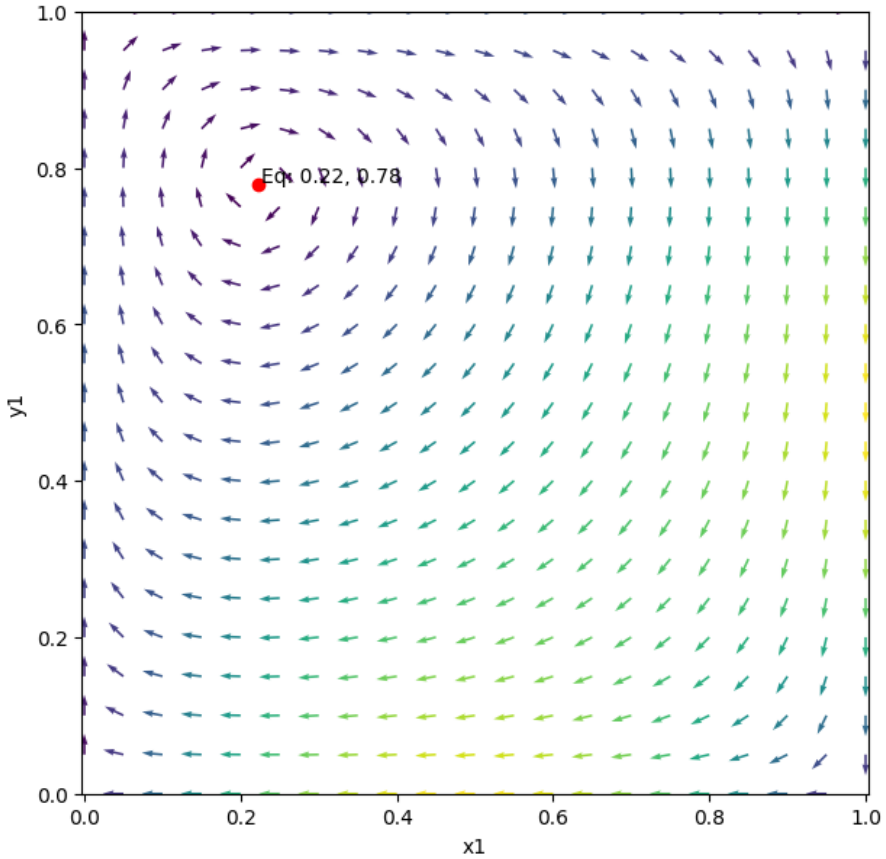
Iterative algorithms can be described with game dynamics:

Also called evolutionary / population / replicator dynamics

(successful strategies are rewarded by high reproductive rates, so become more likely to participate in subsequent playings of the game.)

Learning dynamics in games

.



Iterative algorithms can be described with game dynamics:

Also called evolutionary / population / replicator dynamics

(successful strategies are rewarded by high reproductive rates, so become more likely to participate in subsequent playings of the game.)

Can be described with differential equations:

$$\frac{dx_a}{dt} = x_a \left[(Ay)_a - x^T Ay \right]$$
$$\frac{dy_a}{dt} = y_a \left[(Bx)_a - x^T By \right]$$

Examples of dynamics

Prisoners' Dilemma

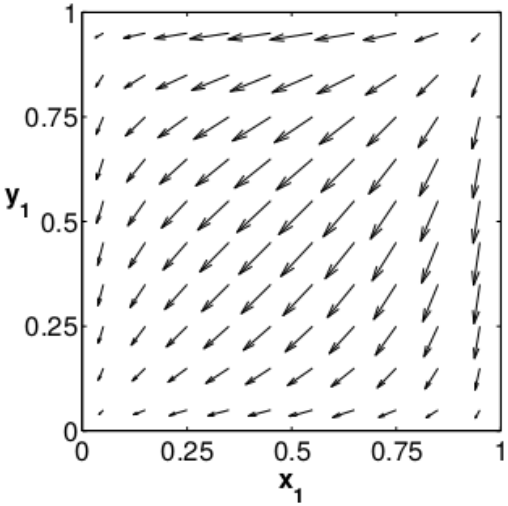
$$\begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} C \\ D \end{array} & \begin{pmatrix} 3,3 & 0,5 \\ 5,0 & 1,1 \end{pmatrix} \end{array}$$

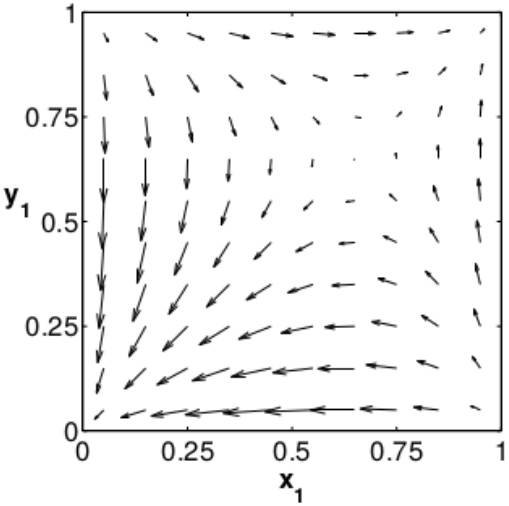
Stag Hunt

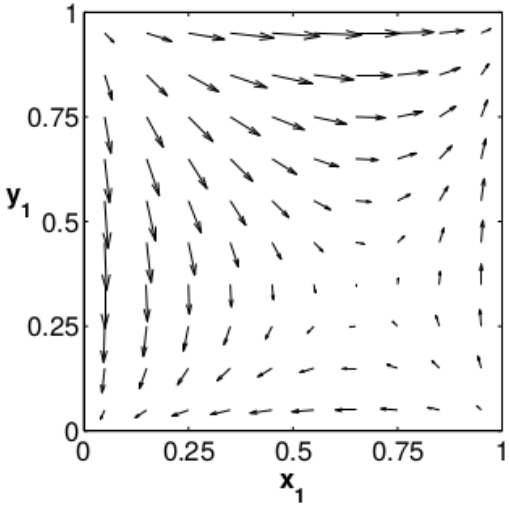
$$\begin{array}{cc} & \begin{array}{cc} S & H \end{array} \\ \begin{array}{c} S \\ H \end{array} & \begin{pmatrix} 4,4 & 1,3 \\ 3,1 & 3,3 \end{pmatrix} \end{array}$$

Battle of the Sexes

$$\begin{array}{cc} & \begin{array}{cc} O & F \end{array} \\ \begin{array}{c} O \\ F \end{array} & \begin{pmatrix} 2,1 & 0,0 \\ 0,0 & 1,2 \end{pmatrix} \end{array}$$







(From Daan Bloembergen thesis: Analyzing reinforcement learning algorithms using evolutionary game theory)

Uncoupled dynamics do not lead to NE

From Hart and Mas-Colell (2003):

Coupled dynamics

$$\dot{x}_i = F_i(x; (u_j)_{j \in N})$$

Uncoupled dynamics

$$\dot{x}_i = F_i(x; u_i)$$

Uncoupled dynamics do not lead to NE

From Hart and Mas-Colell (2003):

Coupled dynamics

$$\dot{x}_i = F_i(x; (u_j)_{j \in N})$$

Uncoupled dynamics

$$\dot{x}_i = F_i(x; u_i)$$

There exist no uncoupled dynamics that guarantee convergence to NE.

Uncoupled dynamics do not lead to NE

From Hart and Mas-Colell (2003):

Coupled dynamics

$$\dot{x}_i = F_i(x; (u_j)_{j \in N})$$

Uncoupled dynamics

$$\dot{x}_i = F_i(x; u_i)$$

There exist no uncoupled dynamics that guarantee convergence to NE.

- Exceptions (special families of games): two-player zero-sum games, two-player potential games, and others.

Uncoupled dynamics do not lead to NE

From Hart and Mas-Colell (2003):

Coupled dynamics

$$\dot{x}_i = F_i(x; (u_j)_{j \in N})$$

Uncoupled dynamics

$$\dot{x}_i = F_i(x; u_i)$$

There exist no uncoupled dynamics that guarantee convergence to NE.

- Exceptions (special families of games): two-player zero-sum games, two-player potential games, and others.
- There exist uncoupled dynamics converging to correlated equilibria.

Uncoupled dynamics do not lead to NE

From Hart and Mas-Colell (2003):

Coupled dynamics

$$\dot{x}_i = F_i(x; (u_j)_{j \in N})$$

Uncoupled dynamics

$$\dot{x}_i = F_i(x; u_i)$$

There exist no uncoupled dynamics that guarantee convergence to NE.

- Exceptions (special families of games): two-player zero-sum games, two-player potential games, and others.
- There exist uncoupled dynamics converging to correlated equilibria.
- There exist uncoupled dynamics that are most of the time in ϵ-NE, but do not converge to NE.

Uncoupled dynamics do not lead to NE

From Hart and Mas-Colell (2003):

Coupled dynamics

$$\dot{x}_i = F_i(x; (u_j)_{j \in N})$$

Uncoupled dynamics

$$\dot{x}_i = F_i(x; u_i)$$

There exist no uncoupled dynamics that guarantee convergence to NE.

- Exceptions (special families of games): two-player zero-sum games, two-player potential games, and others.
- There exist uncoupled dynamics converging to correlated equilibria.
- There exist uncoupled dynamics that are most of the time in ϵ-NE, but do not converge to NE.
- Therefore: *GANs should have coupled dynamics to converge*.

(Un)stable fixed points (1/2)

A fixed point satisfies:

$$f(x_0) = x_0$$

Stable (attractive) fixed point

$$x, f(x), f(f(x)), f(f(f(x))), \ldots$$

converges to x_0 (Banach FP)

Unstable fixed point

ϵ-neighbourhood of x_0 does not contract to x_0 but diverges.

(Un)stable fixed points (1/2)

A fixed point satisfies:

$$f(x_0) = x_0$$

Stable (attractive) fixed point

$$x, f(x), f(f(x)), f(f(f(x))), \ldots$$

converges to x_0 (Banach FP)

Unstable fixed point

ϵ-neighbourhood of x_0 does not contract to x_0 but diverges.

Example: golden ratio $\phi = 1.61809339887\ldots$
(solution to $\frac{a+b}{a} = \frac{a}{b} = \phi \rightarrow \phi^2 - \phi - 1 = 0$)

(Un)stable fixed points (1/2)

A fixed point satisfies:

$$f(x_0) = x_0$$

Stable (attractive) fixed point

$$x, f(x), f(f(x)), f(f(f(x))), \ldots$$

converges to x_0 (Banach FP)

Unstable fixed point

ϵ-neighbourhood of x_0 does not contract to x_0 but diverges.

Example: golden ratio $\phi = 1.61809339887...$
(solution to $\frac{a+b}{a} = \frac{a}{b} = \phi \rightarrow \phi^2 - \phi - 1 = 0$)

Other solution $\phi' = -0.61809339887...$

(Un)stable fixed points (1/2)

A fixed point satisfies:

$$f(x_0) = x_0$$

Stable (attractive) fixed point

$$x, f(x), f(f(x)), f(f(f(x))), \ldots$$

converges to x_0 (Banach FP)

Unstable fixed point

ϵ-neighbourhood of x_0 does not contract to x_0 but diverges.

Example: golden ratio $\phi = 1.61809339887...$
(solution to $\frac{a+b}{a} = \frac{a}{b} = \phi \rightarrow \phi^2 - \phi - 1 = 0$)

Other solution $\phi' = -0.61809339887...$

Iterative method to find ϕ:

$$\phi_{t+1} = 1 + \frac{1}{\phi_t}$$

(Un)stable fixed points (1/2)

A fixed point satisfies:

$$f(x_0) = x_0$$

Stable (attractive) fixed point

$$x, f(x), f(f(x)), f(f(f(x))), \ldots$$

converges to x_0 (Banach FP)

Unstable fixed point

ϵ-neighbourhood of x_0 does not contract to x_0 but diverges.

Example: golden ratio $\phi = 1.61809339887...$
(solution to $\frac{a+b}{a} = \frac{a}{b} = \phi \rightarrow \phi^2 - \phi - 1 = 0$)

Other solution $\phi' = -0.61809339887...$

Iterative method to find ϕ:

$$\phi_{t+1} = 1 + \frac{1}{\phi_t}$$

(Un)stable fixed points (2/2)

Stable:

```
In [1]: x = 1

In [2]: x = 1 + 1.0 / x; print(x)
2.0

In [3]: x = 1 + 1.0 / x; print(x)
1.5

In [4]: x = 1 + 1.0 / x; print(x)
1.6666666666666665

In [5]: x = 1 + 1.0 / x; print(x)
1.6

In [6]: x = 1 + 1.0 / x; print(x)
1.625

In [7]: x = 1 + 1.0 / x; print(x)
1.6153846153846154

In [8]: x = 1 + 1.0 / x; print(x)
1.619047619047619

In [9]: x = 1 + 1.0 / x; print(x)
1.6176470588235294

In [10]: x = 1 + 1.0 / x; print(x)
1.6181818181818182
```

Unstable:

```
In [1]: x = -0.61803

In [2]: x = 1 + 1.0 / x; print(x)
-0.618044431500089

In [3]: x = 1 + 1.0 / x; print(x)
-0.6180066497368901

In [4]: x = 1 + 1.0 / x; print(x)
-0.618105566381429

In [5]: x = 1 + 1.0 / x; print(x)
-0.617846617778081

In [6]: x = 1 + 1.0 / x; print(x)
-0.6185246810870806

In [7]: x = 1 + 1.0 / x; print(x)
-0.6167503586800482

In [8]: x = 1 + 1.0 / x; print(x)
-0.6214015702239264

In [9]: x = 1 + 1.0 / x; print(x)
-0.6092653252222118

In [10]: x = 1 + 1.0 / x; print(x)
-0.6413210445470192
```

.

```
In [11]: x = 1 + 1.0 / x; print(x)
-0.5592814371253396

In [12]: x = 1 + 1.0 / x; print(x)
-0.7880085653117996

In [13]: x = 1 + 1.0 / x; print(x)
-0.26902173912832983

In [14]: x = 1 + 1.0 / x; print(x)
-2.7171717172008023

In [15]: x = 1 + 1.0 / x; print(x)
0.6319702602269878

In [16]: x = 1 + 1.0 / x; print(x)
2.582352941166607

In [17]: x = 1 + 1.0 / x; print(x)
1.387243735764577

In [18]: x = 1 + 1.0 / x; print(x)
1.7208538587841247

In [19]: x = 1 + 1.0 / x; print(x)
1.5811068702292672

In [20]: x = 1 + 1.0 / x; print(x)
1.6324683162340543
```

Takeaways

Things to consider:

- What is the "right" solution concept?
- Is it robust (stable)?
- Can we compute it efficiently?
- On which family of games?
- Can we approximate it using iterative algorithms?

Motivation (1/2)

- In single objective problems gradient descent (GD) converges to local minima.
- Even escapes saddle points almost surely! Lee et. al (2017)
- However, simultaneous/alternating GD is not guaranteed to converge even to local minima!
- Cyclic behaviour!

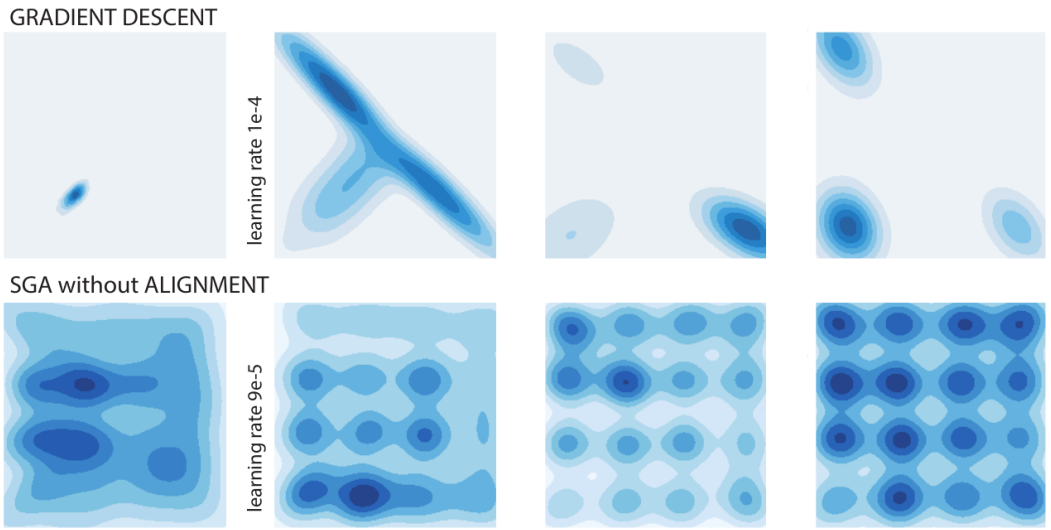
Motivation (1/2)

- In single objective problems gradient descent (GD) converges to local minima.
- Even escapes saddle points almost surely! Lee et. al (2017)
- However, simultaneous/alternating GD is not guaranteed to converge even to local minima!
- Cyclic behaviour!

Therefore: convergence to local minima is difficult!

Motivation (2/2)

Simultaneous GD vs SGA



Differentiable games

The normal form games we talked about so far were constrained to probability simplex

Differentiable games

The normal form games we talked about so far were constrained to probability simplex

This does not make sense in settings such as GANs - weights are not on simplex!

Differentiable games

The normal form games we talked about so far were constrained to probability simplex

This does not make sense in settings such as GANs - weights are not on simplex!

Introduce differentiable games:

- Set of players $N = \{1, \ldots, n\}$
- Continuously twice differentiable losses $\{\ell_i : \mathbb{R}^d \to \mathbb{R}\}_{i=1}^n$
- NN parameters $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_n) \in \mathbb{R}^d$ where $\sum_{i=1}^n d_i = d$
- Player i controls $\mathbf{w}_i \in \mathbb{R}^{d_i}$, and aims to minimize its loss

Dynamics, Jacobian

Simultaneous gradient is the gradient of the losses w/ respect to the parameters of the respective players:

$$\boldsymbol{\xi}(\mathbf{w}) = (\nabla_{\mathbf{w}_1}\ell_1, \ldots, \nabla_{\mathbf{w}_n}\ell_n) \in \mathbb{R}^d.$$

Dynamics of the game mean following $-\boldsymbol{\xi}$ with infinitesimal steps.

Dynamics, Jacobian

Simultaneous gradient is the gradient of the losses w/ respect to the parameters of the respective players:

$$\boldsymbol{\xi}(\mathbf{w}) = (\nabla_{\mathbf{w}_1}\ell_1, \ldots, \nabla_{\mathbf{w}_n}\ell_n) \in \mathbb{R}^d.$$

Dynamics of the game mean following $-\boldsymbol{\xi}$ with infinitesimal steps.

The **Jacobian** of a game with dynamics $\boldsymbol{\xi}$ is the $(d \times d)$-matrix of second-derivatives $\mathbf{J}(\mathbf{w}) := \nabla_{\mathbf{w}} \cdot \boldsymbol{\xi}(\mathbf{w})^\mathsf{T}$. Concretely:

$$\mathbf{J}(\mathbf{w}) = \begin{pmatrix} \nabla^2_{\mathbf{w}_1}\ell_1 & \nabla^2_{\mathbf{w}_1,\mathbf{w}_2}\ell_1 & \cdots & \nabla^2_{\mathbf{w}_1,\mathbf{w}_n}\ell_1 \\ \nabla^2_{\mathbf{w}_2,\mathbf{w}_1}\ell_2 & \nabla^2_{\mathbf{w}_2}\ell_2 & \cdots & \nabla^2_{\mathbf{w}_2,\mathbf{w}_n}\ell_2 \\ \vdots & & & \vdots \\ \nabla^2_{\mathbf{w}_n,\mathbf{w}_1}\ell_n & \nabla^2_{\mathbf{w}_n,\mathbf{w}_2}\ell_n & \cdots & \nabla^2_{\mathbf{w}_n}\ell_n \end{pmatrix}$$

where $\nabla^2_{\mathbf{w}_i,\mathbf{w}_j}\ell_k$ is the $(d_i \times d_j)$-block of 2nd-order derivatives.

Fixed points and equilibria

local Nash equilibrium: NE only within a neighbourhood:
if, for all i, there exists a neighborhood U_i of $\mathbf{w}_i^*$
such that $\ell_i(\mathbf{w}_i', \mathbf{w}_{-i}^*) \geq \ell_i(\mathbf{w}_i^*, \mathbf{w}_{-i}^*)$ for $\mathbf{w}_i' \in U_i$.

Fixed points and equilibria

local Nash equilibrium: NE only within a neighbourhood:
if, for all i, there exists a neighborhood U_i of $\mathbf{w}_i^*$
such that $\ell_i(\mathbf{w}_i', \mathbf{w}_{-i}^*) \geq \ell_i(\mathbf{w}_i^*, \mathbf{w}_{-i}^*)$ for $\mathbf{w}_i' \in U_i$.

A fixed point $\mathbf{w}^*$ with $\boldsymbol{\xi}(\mathbf{w}^*) = 0$ is

- **stable** if $\mathbf{J}(\mathbf{w}^*) \succeq 0$ and $\mathbf{J}(\mathbf{w}^*)$ is invertible
- **unstable** if $\mathbf{J}(\mathbf{w}^*) \prec 0$
- **strict saddle** if $\mathbf{J}(\mathbf{w}^*)$ has an eigenvalue with negative real part

(Strict saddles are a subset of unstable fixed points.)

Fixed points and equilibria

local Nash equilibrium: NE only within a neighbourhood:
if, for all i, there exists a neighborhood U_i of $\mathbf{w}_i^*$
such that $\ell_i(\mathbf{w}_i', \mathbf{w}_{-i}^*) \geq \ell_i(\mathbf{w}_i^*, \mathbf{w}_{-i}^*)$ for $\mathbf{w}_i' \in U_i$.

A fixed point $\mathbf{w}^*$ with $\boldsymbol{\xi}(\mathbf{w}^*) = 0$ is

- **stable** if $\mathbf{J}(\mathbf{w}^*) \succeq 0$ and $\mathbf{J}(\mathbf{w}^*)$ is invertible
- **unstable** if $\mathbf{J}(\mathbf{w}^*) \prec 0$
- **strict saddle** if $\mathbf{J}(\mathbf{w}^*)$ has an eigenvalue with negative real part

(Strict saddles are a subset of unstable fixed points.)

Properties

- if general-sum: local NE $\not\Longleftrightarrow$ stable
- if zero-sum: local NE $\Longleftrightarrow$ stable

Decomposition of games (1/2)

Any matrix decomposes uniquely as $\mathbf{M} = \mathbf{S} + \mathbf{A}$ where

$\mathbf{S} = \frac{1}{2}(\mathbf{M} + \mathbf{M}^\intercal)$ and $\mathbf{A} = \frac{1}{2}(\mathbf{M} - \mathbf{M}^\intercal)$

$\mathbf{S} - \mathbf{S}^\intercal \equiv 0$ is symmetric and $\mathbf{A} + \mathbf{A}^\intercal \equiv 0$ is antisymmetric.

Decomposition of games (2/2)

We can apply this decomposition to the jacobian $\mathbf{J}(\mathbf{w}) = \mathbf{S}(\mathbf{w}) + \mathbf{A}(\mathbf{w})$

Let's define:

- **potential game**: the jacobian is symmetric, i.e. $\mathbf{A}(\mathbf{w}) \equiv 0$.
- **hamiltonian game**: the jacobian is antisymmetric, i.e. $\mathbf{S}(\mathbf{w}) \equiv 0$.

Optimization on the respective game types

If game is *potential*:

- Well studied games in literature.
- Simultaneous gradient descent on the losses corresponds to gradient descent on a single function.
- GD on ξ converges to a fixed point that is a local minimum or a saddle.

Optimization on the respective game types

If game is *potential*:

- Well studied games in literature.
- Simultaneous gradient descent on the losses corresponds to gradient descent on a single function.
- GD on $\boldsymbol{\xi}$ converges to a fixed point that is a local minimum or a saddle.

If game is *hamiltonian*:

- Novel contribution of the paper, analyzes properties of hamiltonian games
- Let $\mathcal{H}(\mathbf{w}) := \frac{1}{2}\|\boldsymbol{\xi}(\mathbf{w})\|_2^2$
- (by theorem 4) GD on $\mathcal{H}$ converges to stable fixed point
- Further gradient can be computed as $\nabla\mathcal{H} = \mathbf{A}^\intercal\boldsymbol{\xi}$

Symplectic gradient adjustment

We'd like to solve general games, i.e. $\mathbf{J}(\mathbf{w}) = \mathbf{S}(\mathbf{w}) + \mathbf{A}(\mathbf{w})$

This paper introduces symplectic gradient adjustment (SGA):

$$\boldsymbol{\xi}_\lambda := \boldsymbol{\xi} + \lambda \cdot \mathbf{A}^\mathsf{T} \boldsymbol{\xi}.$$

SGA desiderata

SGA satisfies following desiderata:

1. *compatible*[1] *with game dynamics:* $\langle \boldsymbol{\xi}_\lambda, \boldsymbol{\xi} \rangle = \alpha_1 \cdot \|\boldsymbol{\xi}\|^2$;
2. *compatible with potential dynamics:*
   if the game is a potential game then $\langle \boldsymbol{\xi}_\lambda, \nabla\phi \rangle = \alpha_2 \cdot \|\nabla\phi\|^2$;
3. *compatible with Hamiltonian dynamics:*
   If the game is Hamiltonian then $\langle \boldsymbol{\xi}_\lambda, \nabla\mathcal{H} \rangle = \alpha_3 \cdot \|\nabla\mathcal{H}\|^2$;
4. *attracted to stable equilibria:*
   in neighborhoods where $\mathbf{S} \succ 0$, require $\theta(\boldsymbol{\xi}_\lambda, \nabla\mathcal{H}) \leq \theta(\boldsymbol{\xi}, \nabla\mathcal{H})$;
5. *repelled by unstable equilibria:*
   in neighborhoods where $\mathbf{S} \prec 0$, require $\theta(\boldsymbol{\xi}_\lambda, \nabla\mathcal{H}) \geq \theta(\boldsymbol{\xi}, \nabla\mathcal{H})$.

for some $\alpha_1, \alpha_2, \alpha_3 > 0$.

[1] Two nonzero vectors are compatible if they have positive inner product.

SGA pseudocode

Input: losses $\mathcal{L} = \{\ell_i\}_{i=1}^n$, weights $\mathcal{W} = \{\mathbf{w}_i\}_{i=1}^n$

$\boldsymbol{\xi} \leftarrow \left[\texttt{gradient}(\ell_i, \mathbf{w}_i) \ \texttt{for} \ (\ell_i, \mathbf{w}_i) \in (\mathcal{L}, \mathcal{W})\right]$

$\mathbf{A}^\intercal \boldsymbol{\xi} \leftarrow \texttt{get\_sym\_adj}(\mathcal{L}, \mathcal{W})$ // appendix A

if `align` **then**

$\nabla \mathcal{H} \leftarrow \left[\texttt{gradient}(\frac{1}{2}\|\boldsymbol{\xi}\|^2, \mathbf{w}) \ \texttt{for} \ \mathbf{w} \in \mathcal{W})\right]$

$\lambda \leftarrow \texttt{sign}\left(\frac{1}{d}\langle \boldsymbol{\xi}, \nabla \mathcal{H}\rangle \langle \mathbf{A}^\intercal \boldsymbol{\xi}, \nabla \mathcal{H}\rangle + \epsilon\right)$ // $\epsilon = \frac{1}{10}$

else

$\lambda \leftarrow 1$

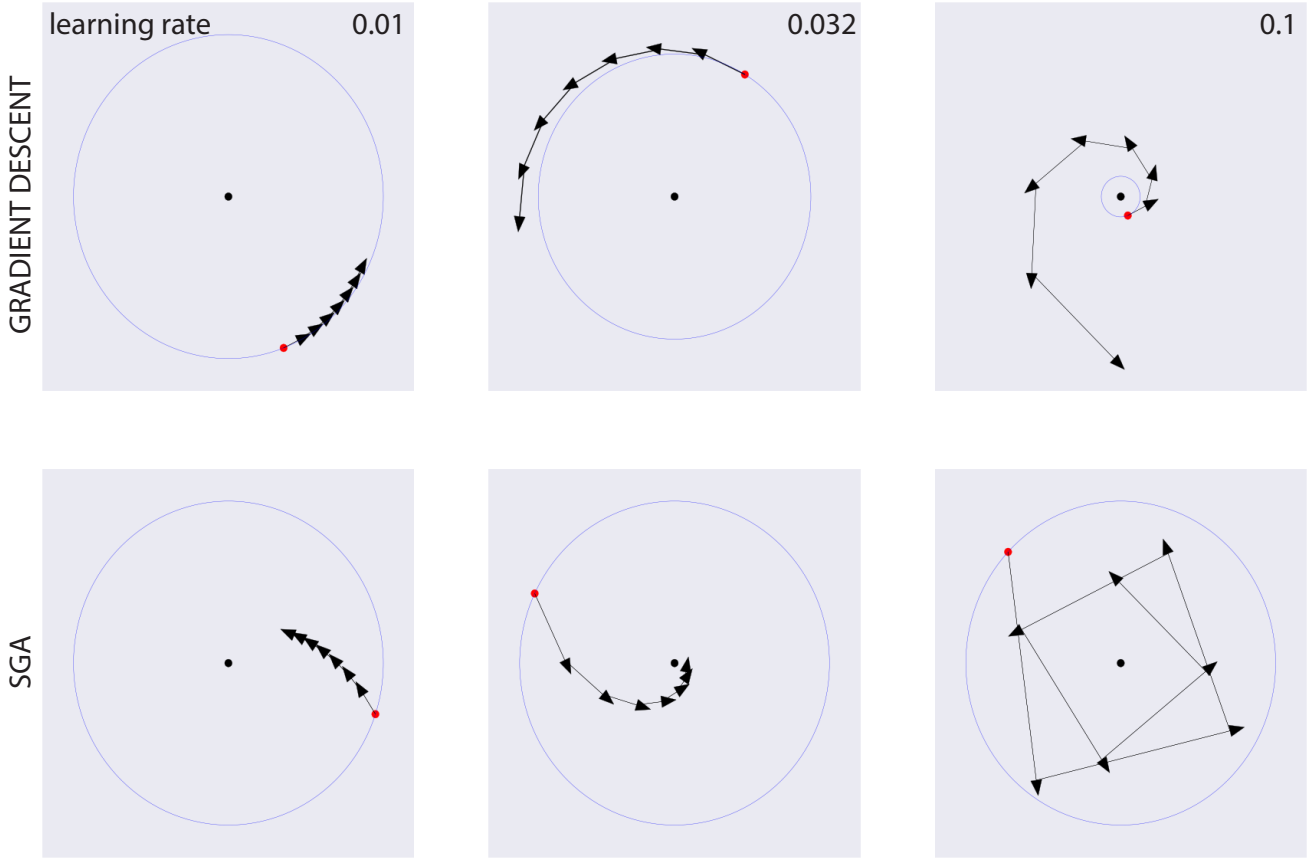
end if

Output: $\boldsymbol{\xi} + \lambda \cdot \mathbf{A}^\intercal \boldsymbol{\xi}$ // plug into any optimizer

SGA (pytorch)

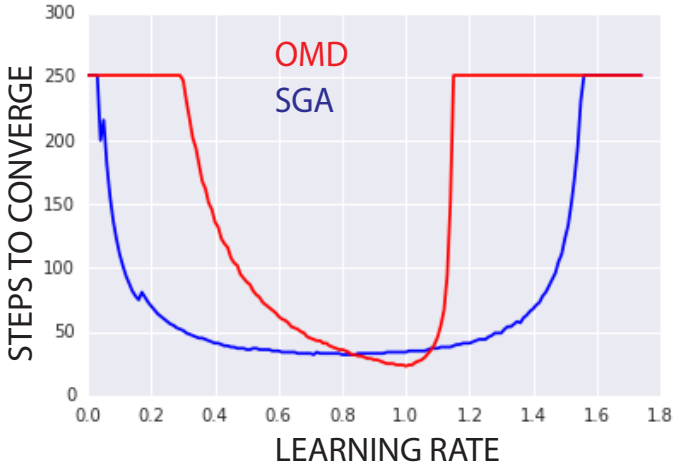
```
xi = torch.cat([grad_L[i][i] for i in range(n)])
ham = torch.dot(xi, xi.detach())
H_t_xi = [get_gradient(ham, th[i]) for i in range(n)]
H_xi = [get_gradient(sum([torch.dot(grad_L[j][i], grad_L[j][j].detach())
                          for j in range(n)]), th[i]) for i in range(n)]
A_t_xi = [H_t_xi[i] / 2 - H_xi[i] / 2 for i in range(n)]
# Compute lambda (sga with alignment)
dot_xi = torch.dot(xi, torch.cat(H_t_xi))
dot_A = torch.dot(torch.cat(A_t_xi), torch.cat(H_t_xi))
d = sum([len(th[i]) for i in range(n)])
lam = torch.sign(dot_xi * dot_A / d + ep)
grads = [grad_L[i][i] + lam * A_t_xi[i] for i in range(n)]
```

Experiment: higher learning rates



SGA allows faster and more robust convergence to stable fixed points than vanilla gradient descent in the presence of 'rotational forces', by bending the direction of descent towards the fixed point.

Experiment: convergence with various learning rates



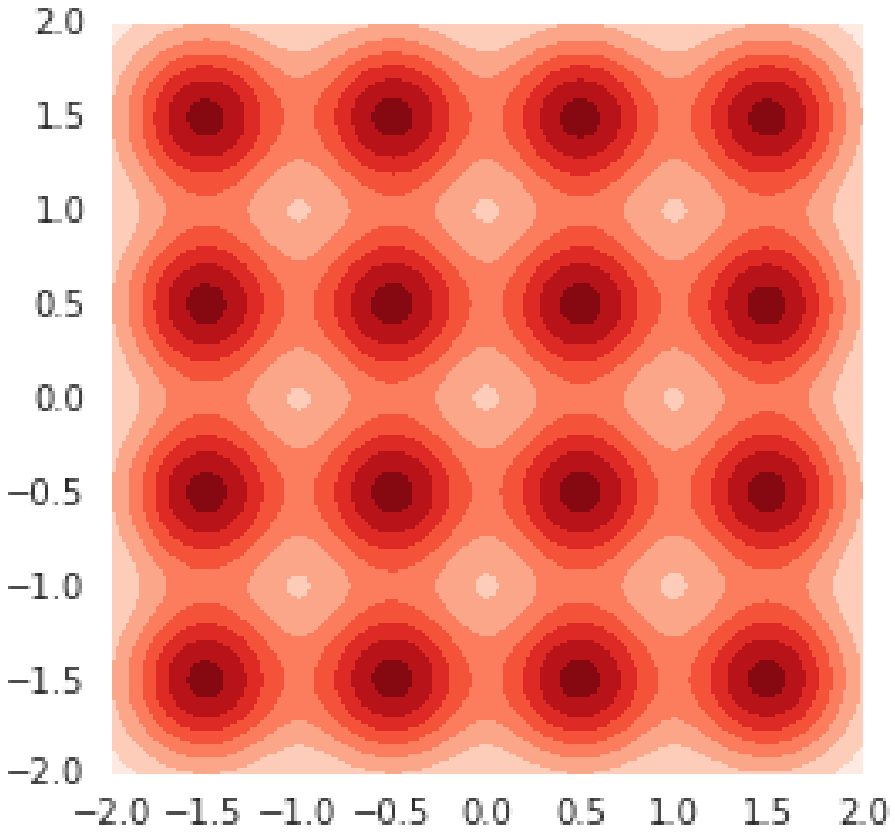
Comparison of SGA with optimistic mirror descent. SGA with $\lambda = 1$

(Left): iterations to convergence, with maximum 250 iters.

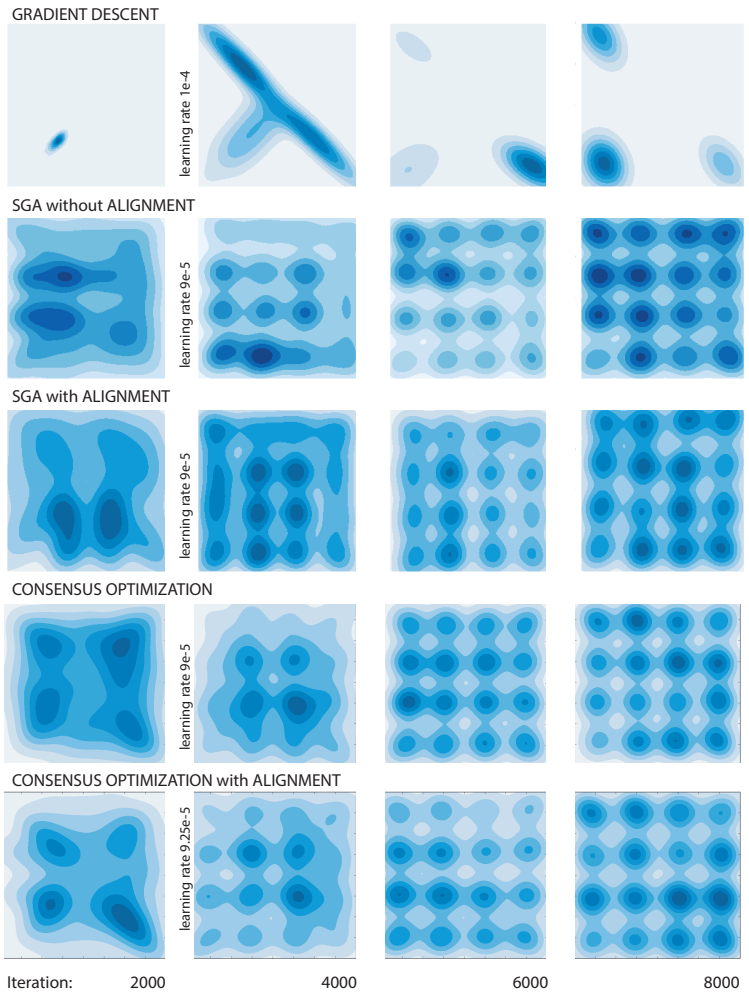
(Right): average absolute value of losses over the last 10 iterations, i.e. iterations 240-250

Experiment: convergence to all modes (1/2)

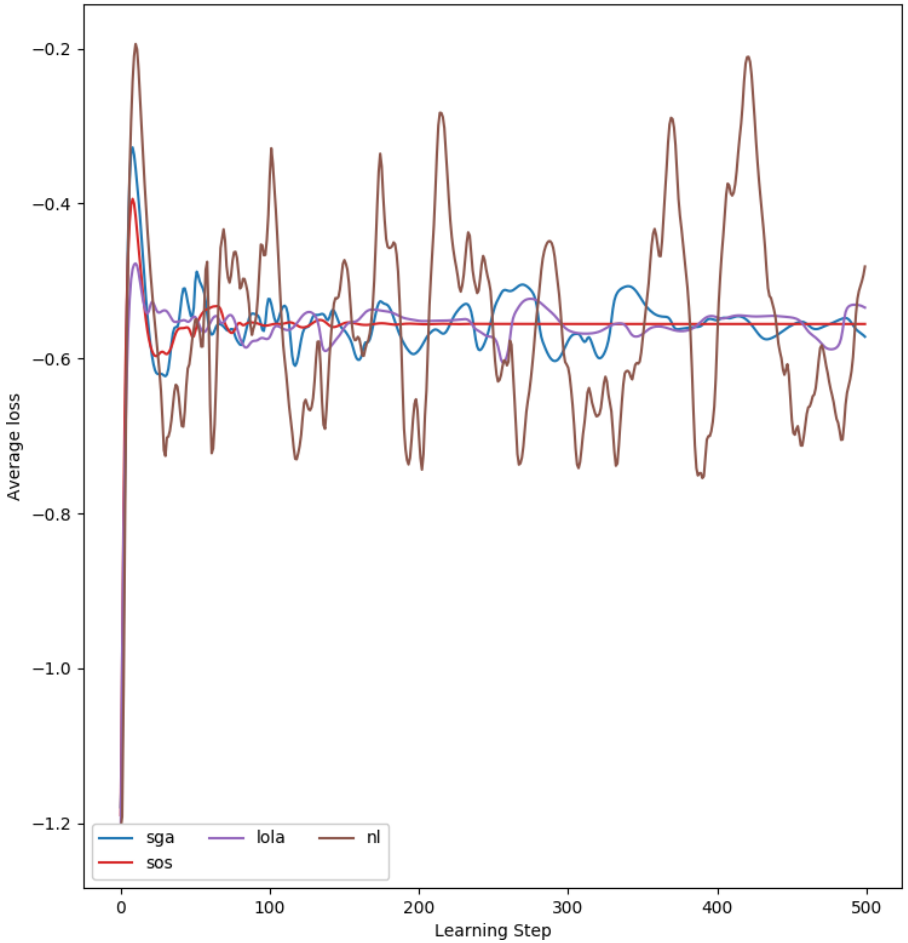
Ground truth:

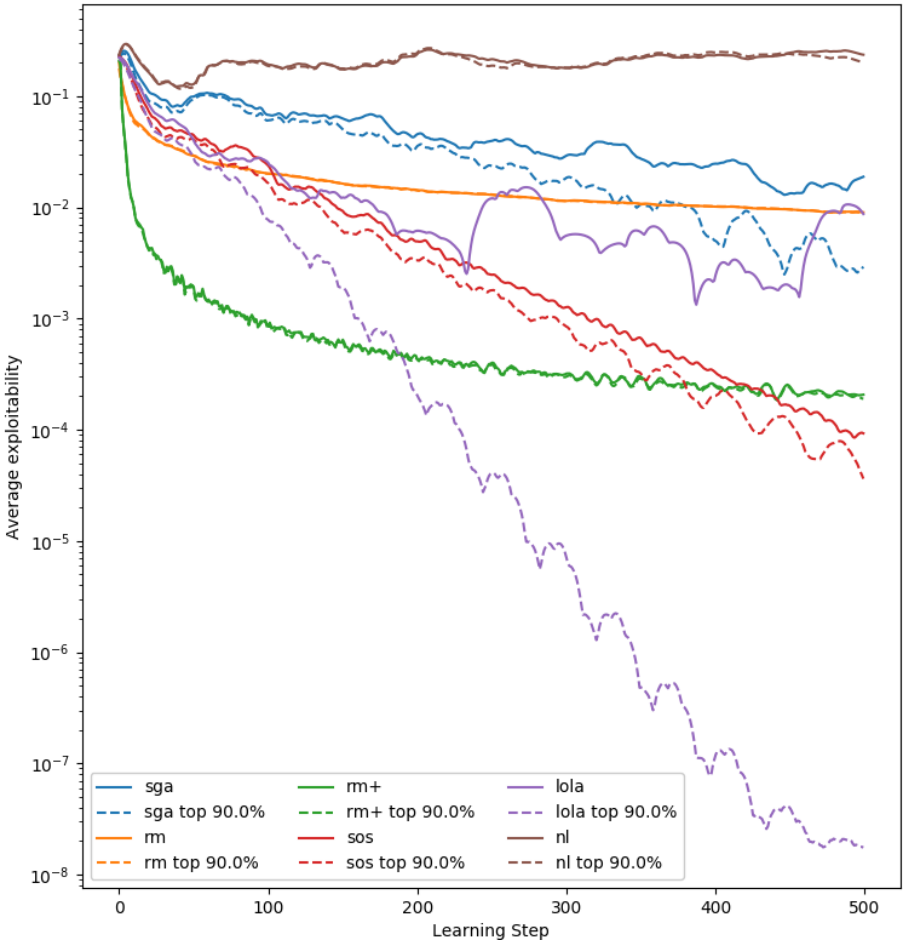


Experiment: convergence to all modes (2/2)

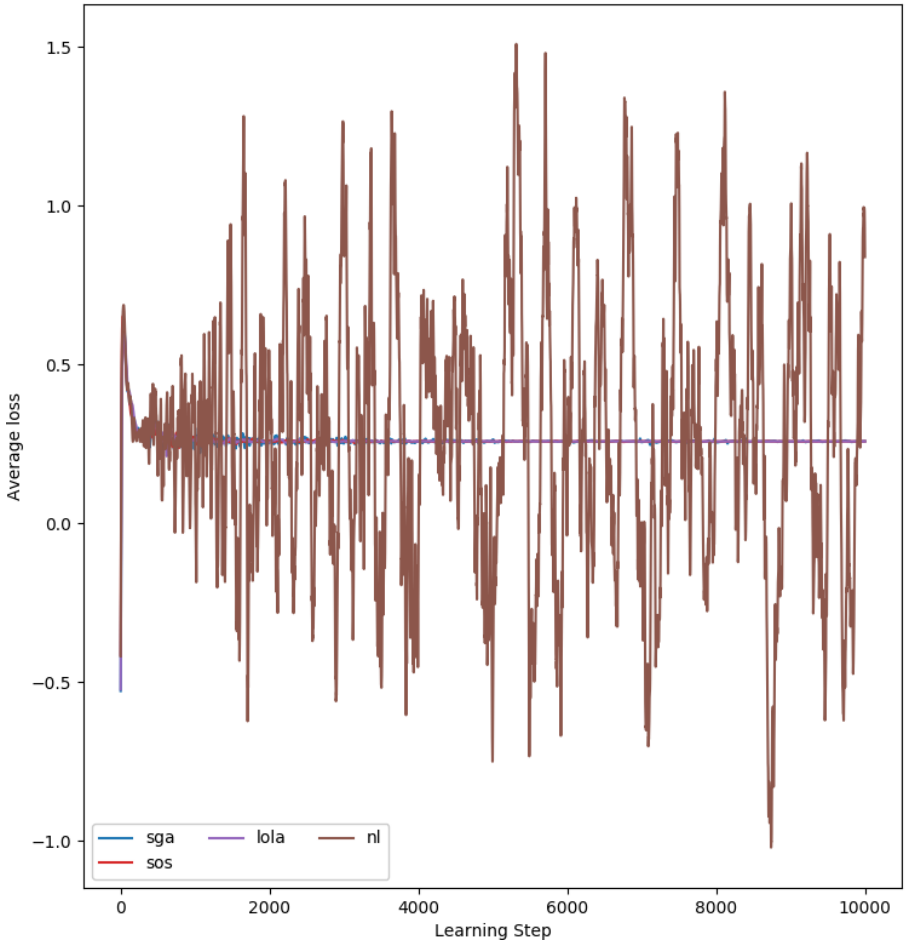


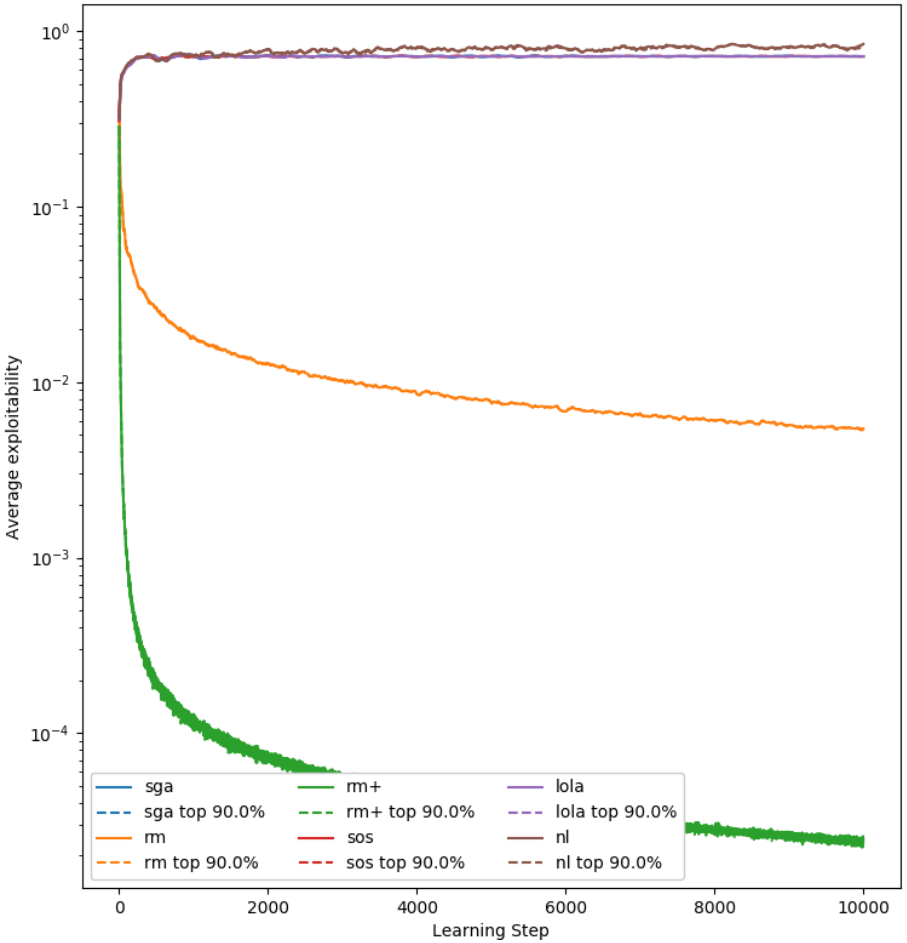
My experiments: learning in NFGs - 2x2 game





My experiments: learning in NFGs - 10x10 game





Notes

Adjustment update independently discovered in *Global Convergence to the Equilibrium of GAS using Variational Inequalities*, I. Gemp, S. Mahadevan Arxiv
(called Crossing-the-Curl)

Things I've left out:

Notes

Adjustment update independently discovered in *Global Convergence to the Equilibrium of GAS using Variational Inequalities*, I. Gemp, S. Mahadevan Arxiv
(called Crossing-the-Curl)

Things I've left out:

- Sign and magnitude of the adjustment λ

Notes

Adjustment update independently discovered in *Global Convergence to the Equilibrium of GAS using Variational Inequalities*, I. Gemp, S. Mahadevan Arxiv
(called Crossing-the-Curl)

Things I've left out:

- Sign and magnitude of the adjustment λ
- Properties between Hamiltonian and zero-sum games

Notes

Adjustment update independently discovered in *Global Convergence to the Equilibrium of GAS using Variational Inequalities*, I. Gemp, S. Mahadevan Arxiv
(called Crossing-the-Curl)

Things I've left out:

- Sign and magnitude of the adjustment λ
- Properties between Hamiltonian and zero-sum games
- Jacobian-vector product can be computed efficiently

Notes

Adjustment update independently discovered in *Global Convergence to the Equilibrium of GAS using Variational Inequalities*, I. Gemp, S. Mahadevan Arxiv
(called Crossing-the-Curl)

Things I've left out:

- Sign and magnitude of the adjustment λ
- Properties between Hamiltonian and zero-sum games
- Jacobian-vector product can be computed efficiently
- Type consistency check of the game reveals if gradient adjustment is needed

Notes

Adjustment update independently discovered in *Global Convergence to the Equilibrium of GAS using Variational Inequalities*, I. Gemp, S. Mahadevan Arxiv
(called Crossing-the-Curl)

Things I've left out:

- Sign and magnitude of the adjustment λ
- Properties between Hamiltonian and zero-sum games
- Jacobian-vector product can be computed efficiently
- Type consistency check of the game reveals if gradient adjustment is needed
- Relation to differential and symplectic geometry and Hodge decomposition

Takeaways

Multiobjective optimization:

- Simultaneous gradient is not guaranteed to converge to local minima in general games (only in potential games).
- Causes mode collapse and mode-hopping

Symplectic gradient adjustment

- Simple plug-n-play adjustment for optimization
- Supports multiplayer settings $n \geq 2$
- Implementation for Tensorflow/Pytorch available
- Computes stable fixed points which are not necessarily local NE

Motivation

Applying latent optimization leads to more diverse image generation.

(**a**) (**b**)

Figure: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with similarly high IS.

Gradient update in GANs

$$\min_{\theta_D} \max_{\theta_G} \mathbb{E}_{x\sim p(x)} \left[h_D(D\left(x;\theta_D\right))\right] + \mathbb{E}_{z\sim p(z)} \left[h_G(D\left(G\left(z;\theta_G\right);\theta_D\right))\right]$$

Gradient update in GANs

$$\min_{\theta_D} \max_{\theta_G} \mathbb{E}_{x\sim p(x)} \left[h_D(D\left(x;\theta_D\right))\right] + \mathbb{E}_{z\sim p(z)} \left[h_G(D\left(G\left(z;\theta_G\right);\theta_D\right))\right]$$

LOGAN settings $h_D(t) = -t, h_G(t) = t$:

$$\min_{\theta_D} \max_{\theta_G} \mathbb{E}_{x\sim p(x)} \left[-D\left(x;\theta_D\right)\right] + \mathbb{E}_{z\sim p(z)} \left[D\left(G\left(z;\theta_G\right);\theta_D\right)\right]$$

Notation: $f(z) := D\left(G\left(z;\theta_G\right);\theta_D\right)$

Gradient update for discriminator and generator:

$$\boldsymbol{\xi} = \left[\frac{\partial f(z)}{\partial \theta_D}, -\frac{\partial f(z)}{\partial \theta_G}\right]^T \tag{1}$$

SGA for GANs

SGA has extra term:

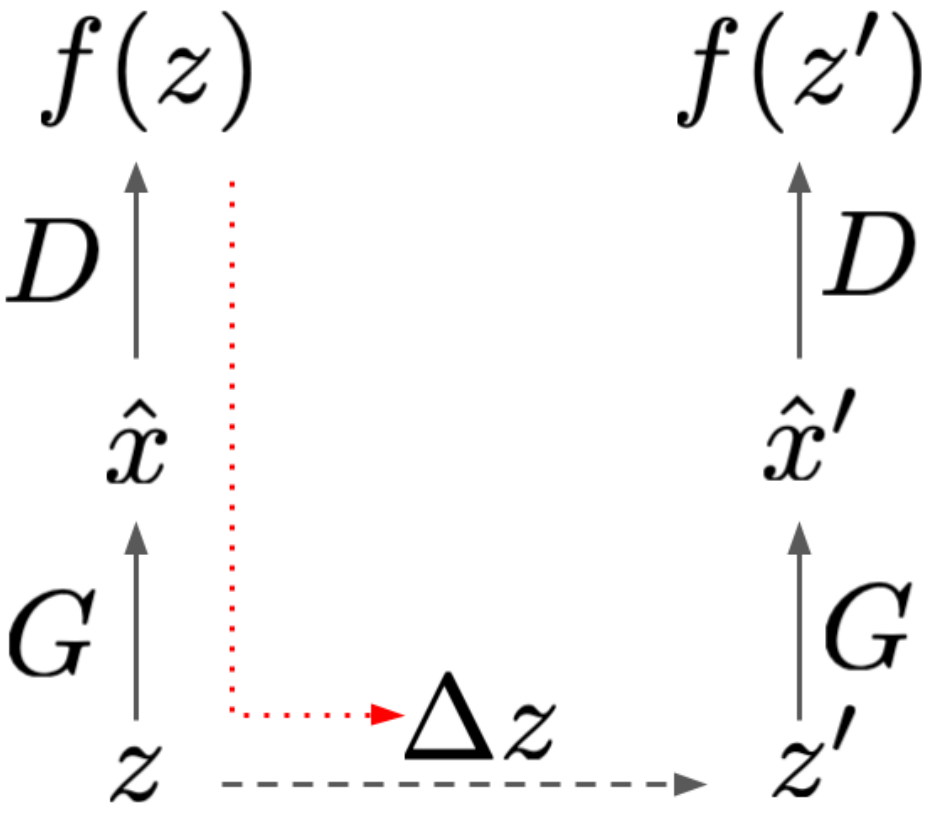
$$\boldsymbol{\xi}_\lambda := \boldsymbol{\xi} + \lambda \cdot \mathbf{A}^\intercal \boldsymbol{\xi}$$

Applying SGA to GANs:

$$\boldsymbol{\xi}_\lambda := \begin{bmatrix} \dfrac{\partial f(z)}{\partial \theta_D} + \lambda \left(\dfrac{\partial^2 f(z)}{\partial \theta_G \, \partial \theta_D} \right)^T \dfrac{\partial f(z)}{\partial \theta_G} \\ -\dfrac{\partial f(z)}{\partial \theta_G} + \lambda \left(\dfrac{\partial^2 f(z)}{\partial \theta_D \, \partial \theta_G} \right)^T \dfrac{\partial f(z)}{\partial \theta_D} \end{bmatrix}$$

Latent optimized sources

Idea: instead of using z, use **latent optimized** $z' := z + \Delta z$



Computing Δz

Two approaches to compute Δz:

- Gradient Descent:

$$\Delta z = \alpha\, g$$

- Natural Gradient Descent (approximation of 2nd order method):

$$\Delta z = \frac{\alpha}{\beta + \|g\|^2}\, g$$

with $g = \frac{\partial f(z)}{\partial z}$.

Latent Optimised GANs with Automatic Differentiation

Input:
data distribution $p(x)$
latent distribution $p(z)$
$D(\cdot;\theta_D)$, $G(\cdot;\theta_G)$
learning rate α,
batch size N
repeat
Initialise discriminator and generator parameters θ_D, θ_G
for $i = 1$ **to** N **do**
Sample $z \sim p(z)$, $x \sim p(x)$
Compute the gradient $\frac{\partial D(G(z))}{\partial z}$ and obtain Δz using GD or NGD
Compute the optimized latent $z' \leftarrow [z + \Delta z]$ [a]
Compute generator loss $L_G^{(i)} = -D(G(z'))$
Compute discriminator loss $L_D^{(i)} = D(G(z')) - D(x)$
end for
Compute batch losses $L_G = \frac{1}{N}\sum_{i=1}^{N} L_G^{(i)}$ and $L_D = \frac{1}{N}\sum_{i=1}^{N} L_D^{(i)}$
Update θ_D and θ_G with the gradients $\frac{\partial L_D}{\partial \theta_D}$, $\frac{\partial L_G}{\partial \theta_G}$
until reaches the maximum training steps

[a] $[\cdot]$ indicates clipping the value between -1 and 1

Latent optimized dynamics

$$\text{How does } z' = z + \Delta z = z + \alpha \frac{\partial f(z)}{\partial z}$$

$$\text{change the original dynamics } \boldsymbol{\xi} = \left[\frac{\partial f(z)}{\partial \theta_D}, -\frac{\partial f(z)}{\partial \theta_G} \right]^T ?$$

Latent optimized dynamics

How does $z' = z + \Delta z = z + \alpha \frac{\partial f(z)}{\partial z}$

change the original dynamics $\boldsymbol{\xi} = \left[\frac{\partial f(z)}{\partial \theta_D}, -\frac{\partial f(z)}{\partial \theta_G}\right]^T$?

$$\boldsymbol{\xi}' = \begin{bmatrix} \frac{\partial f(z')}{\partial \theta_D} + \left(\frac{\partial \Delta z}{\partial \theta_D}\right)^T \frac{\partial f(z')}{\partial \Delta z} \\ -\frac{\partial f(z')}{\partial \theta_G} - \left(\frac{\partial \Delta z}{\partial \theta_G}\right)^T \frac{\partial f(z')}{\partial \Delta z} \end{bmatrix} = \begin{bmatrix} \frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial z'} \\ -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial z'} \end{bmatrix}$$

Comparison of the two dynamics

SGA:

$$\boldsymbol{\xi}_\lambda = \begin{bmatrix} \frac{\partial f(z)}{\partial \theta_D} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_G \, \partial \theta_D} \right)^T \frac{\partial f(z)}{\partial \theta_G} \\ -\frac{\partial f(z)}{\partial \theta_G} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_D \, \partial \theta_G} \right)^T \frac{\partial f(z)}{\partial \theta_D} \end{bmatrix}$$

Latent optimization:

$$\boldsymbol{\xi}' = \begin{bmatrix} \frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D} \right)^T \frac{\partial f(z')}{\partial z'} \\ -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G} \right)^T \frac{\partial f(z')}{\partial z'} \end{bmatrix}$$

Comparison of the two dynamics

SGA:

$$\xi_\lambda = \begin{bmatrix} \frac{\partial f(z)}{\partial \theta_D} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_G \, \partial \theta_D} \right)^T \frac{\partial f(z)}{\partial \theta_G} \\ -\frac{\partial f(z)}{\partial \theta_G} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_D \, \partial \theta_G} \right)^T \frac{\partial f(z)}{\partial \theta_D} \end{bmatrix}$$

Latent optimization:

$$\xi' = \begin{bmatrix} \frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D} \right)^T \frac{\partial f(z')}{\partial z'} \\ -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G} \right)^T \frac{\partial f(z')}{\partial z'} \end{bmatrix}$$

- Approximates SGA using only second-order derivatives with respect to the latent z and parameters of the discriminator and generator *separately*.

Comparison of the two dynamics

SGA:

$$\xi_\lambda = \begin{bmatrix} \frac{\partial f(z)}{\partial \theta_D} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_G \, \partial \theta_D} \right)^T \frac{\partial f(z)}{\partial \theta_G} \\ -\frac{\partial f(z)}{\partial \theta_G} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_D \, \partial \theta_G} \right)^T \frac{\partial f(z)}{\partial \theta_D} \end{bmatrix}$$

Latent optimization:

$$\xi' = \begin{bmatrix} \frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D} \right)^T \frac{\partial f(z')}{\partial z'} \\ -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G} \right)^T \frac{\partial f(z')}{\partial z'} \end{bmatrix}$$

- Approximates SGA using only second-order derivatives with respect to the latent z and parameters of the discriminator and generator *separately*.
- The second order terms involving parameters of both the discriminator and the generator – which are extremely expensive to compute – are not used.

Comparison of the two dynamics

SGA:

$$\xi_\lambda = \begin{bmatrix} \frac{\partial f(z)}{\partial \theta_D} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_G \, \partial \theta_D} \right)^T \frac{\partial f(z)}{\partial \theta_G} \\ -\frac{\partial f(z)}{\partial \theta_G} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_D \, \partial \theta_G} \right)^T \frac{\partial f(z)}{\partial \theta_D} \end{bmatrix}$$

Latent optimization:

$$\xi' = \begin{bmatrix} \frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D} \right)^T \frac{\partial f(z')}{\partial z'} \\ -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G} \right)^T \frac{\partial f(z')}{\partial z'} \end{bmatrix}$$

- Approximates SGA using only second-order derivatives with respect to the latent z and parameters of the discriminator and generator *separately*.
- The second order terms involving parameters of both the discriminator and the generator – which are extremely expensive to compute – are not used.
- For latent z's with dimensions typically used in GANs, the second order terms can be computed efficiently.

Comparison of the two dynamics

SGA:

$$\xi_\lambda = \begin{bmatrix} \frac{\partial f(z)}{\partial \theta_D} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_G \, \partial \theta_D} \right)^T \frac{\partial f(z)}{\partial \theta_G} \\ -\frac{\partial f(z)}{\partial \theta_G} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_D \, \partial \theta_G} \right)^T \frac{\partial f(z)}{\partial \theta_D} \end{bmatrix}$$

Latent optimization:

$$\xi' = \begin{bmatrix} \frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D} \right)^T \frac{\partial f(z')}{\partial z'} \\ -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G} \right)^T \frac{\partial f(z')}{\partial z'} \end{bmatrix}$$

- Approximates SGA using only second-order derivatives with respect to the latent z and parameters of the discriminator and generator *separately*.
- The second order terms involving parameters of both the discriminator and the generator – which are extremely expensive to compute – are not used.
- For latent z's with dimensions typically used in GANs, the second order terms can be computed efficiently.
- In short, latent optimisation *efficiently* couples the gradients of the D and G, as prescribed by SGA, but using the much lower-dimensional latent source z.

Evaluation

Evaluation of GANs is hard

Evaluation

Evaluation of GANs is hard

If we could specify it well enough, it could become the objective!

Evaluation

Evaluation of GANs is hard

If we could specify it well enough, it could become the objective!

Best response against a fixed discriminator may not be what we want.

Evaluation

Evaluation of GANs is hard

If we could specify it well enough, it could become the objective!

Best response against a fixed discriminator may not be what we want.

Scores used in LOGAN:

- Inception Score (IS)
  a measure the "objectness" of a generated image, computed by model with Inception architecture (Salimans 2016)

Evaluation

Evaluation of GANs is hard

If we could specify it well enough, it could become the objective!

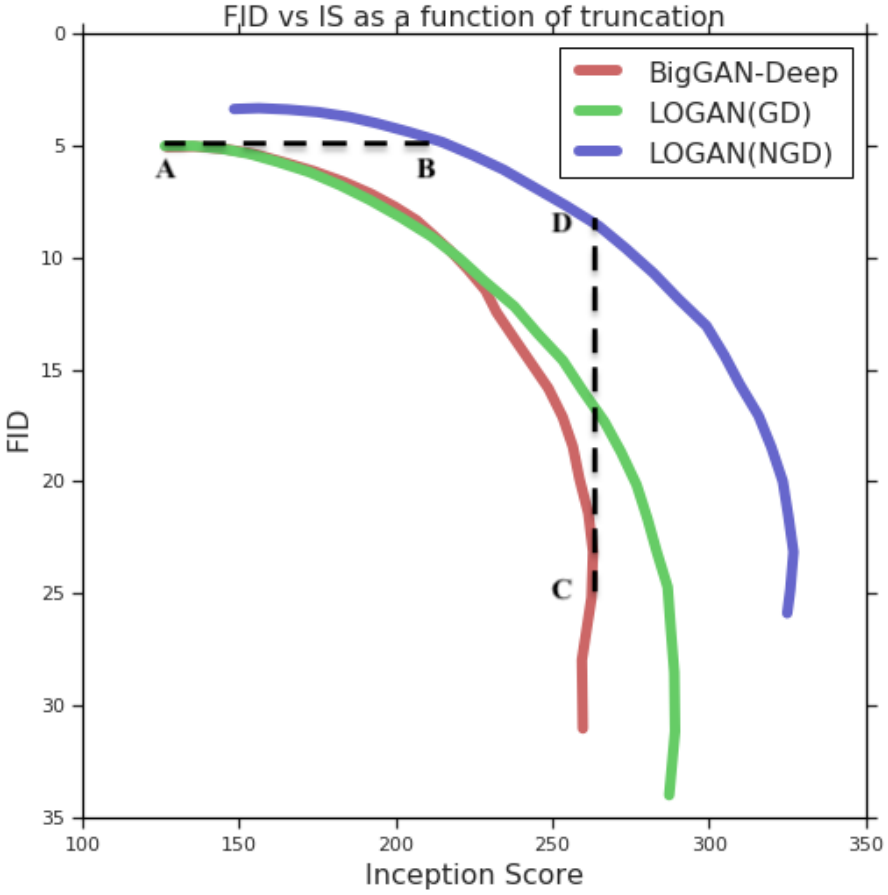
Best response against a fixed discriminator may not be what we want.

Scores used in LOGAN:

- Inception Score (IS)
  a measure the "objectness" of a generated image, computed by model with Inception architecture (Salimans 2016)
- Fréchet Inception Distance (FID) todo
  evaluate the similarity between two dataset of images (Heusel 2017)

Experiments: IS vs FID

.

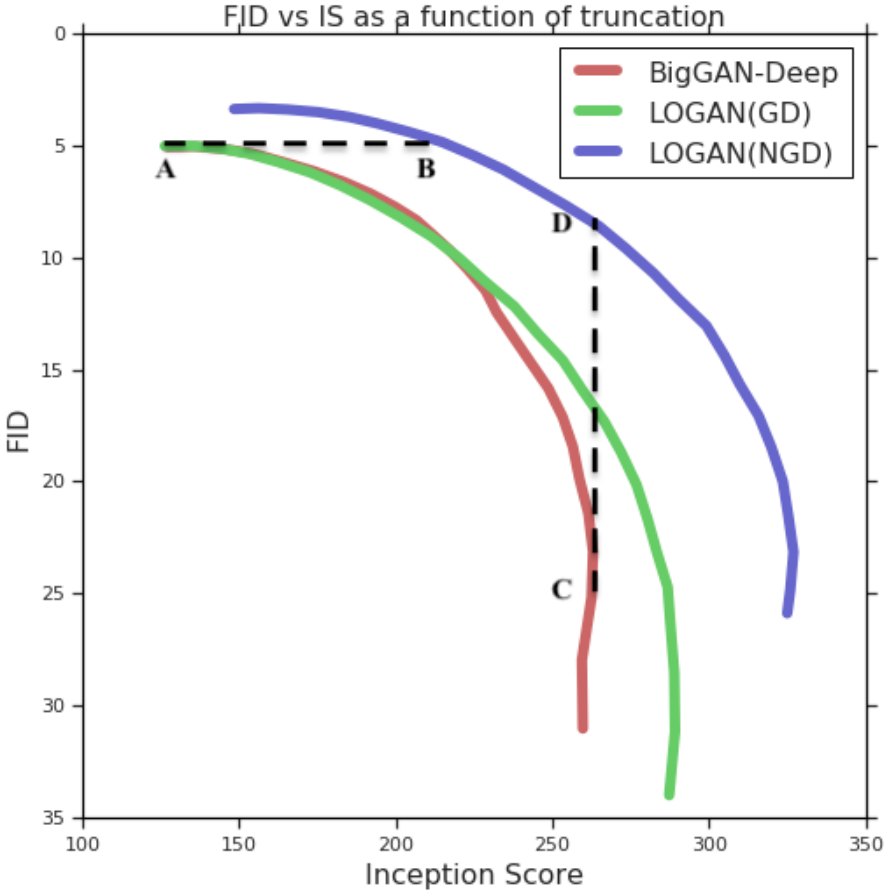


BigGAN:

- architecture based on residual blocks
- regularisation mechanisms and self-attention

Experiments: IS vs FID

.



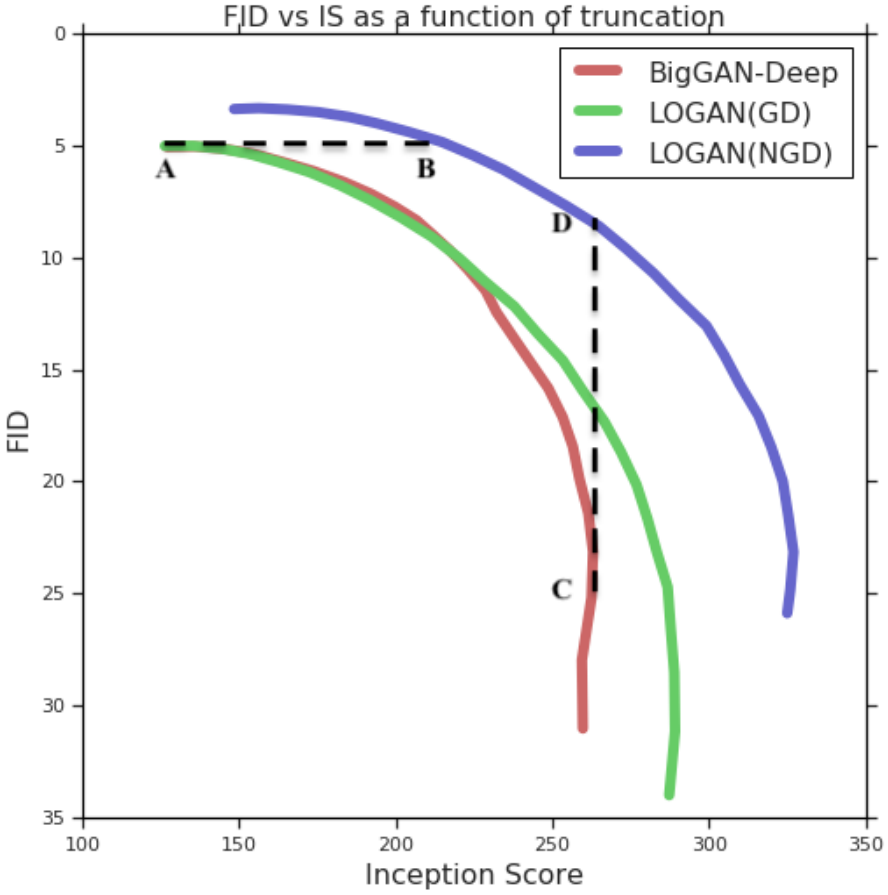
BigGAN:

- architecture based on residual blocks
- regularisation mechanisms and self-attention

LOGAN improves the adversarial dynamics during training.

Experiments: IS vs FID

.

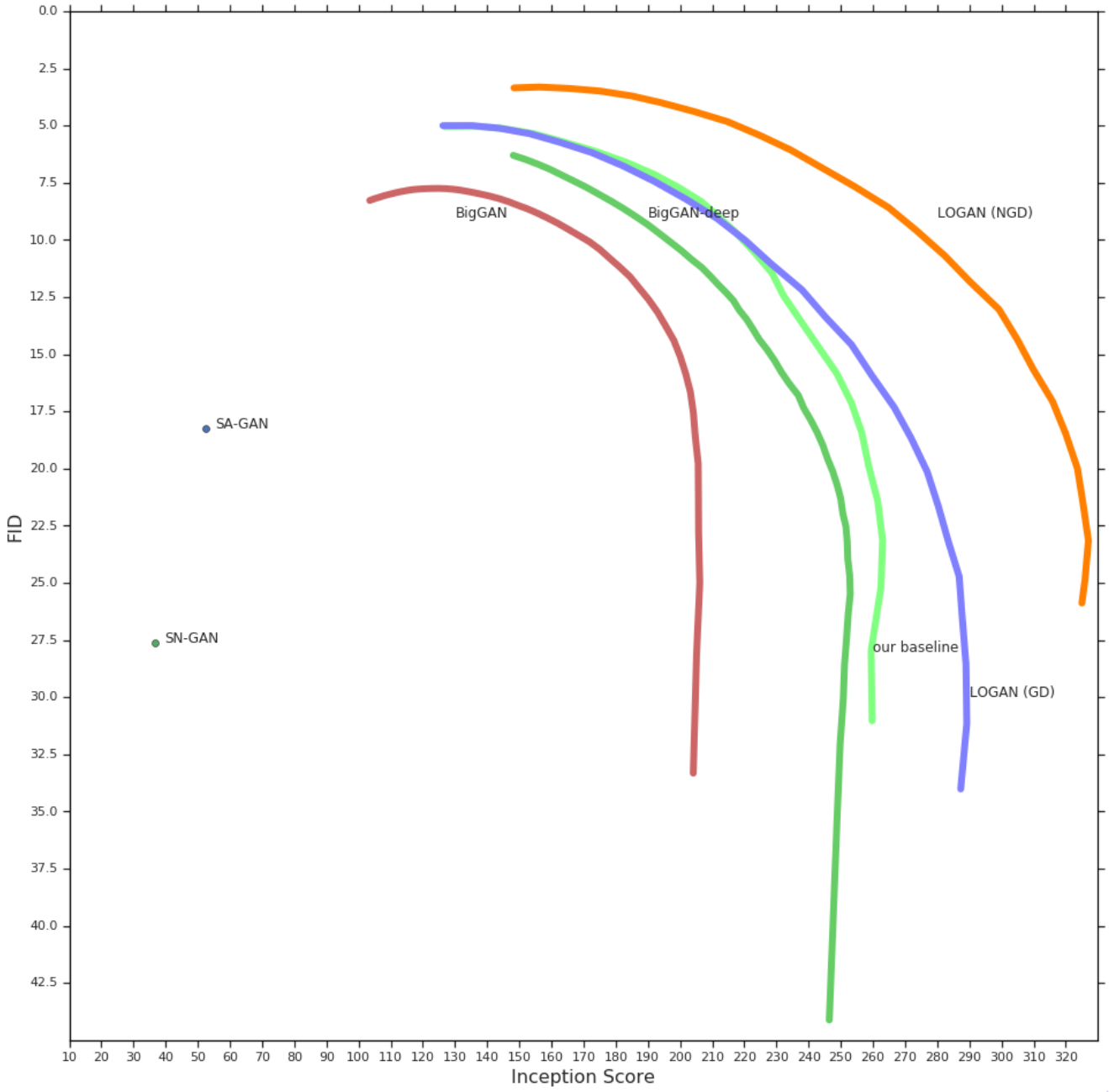


BigGAN:

- architecture based on residual blocks
- regularisation mechanisms and self-attention

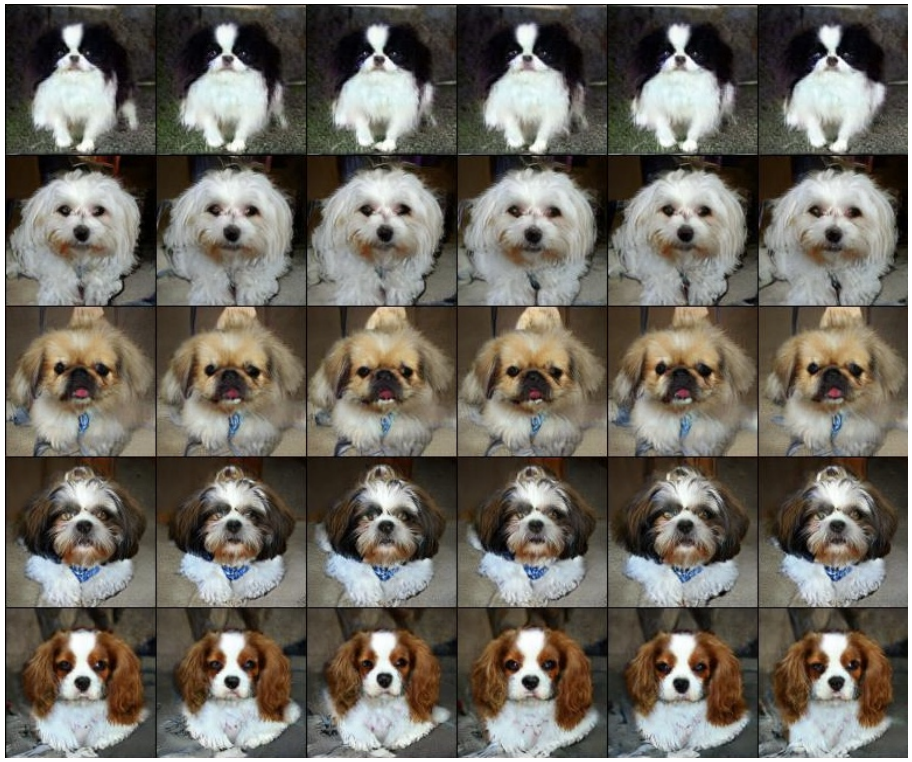
LOGAN improves the adversarial dynamics during training.

No need to optimize latents for evaluation.

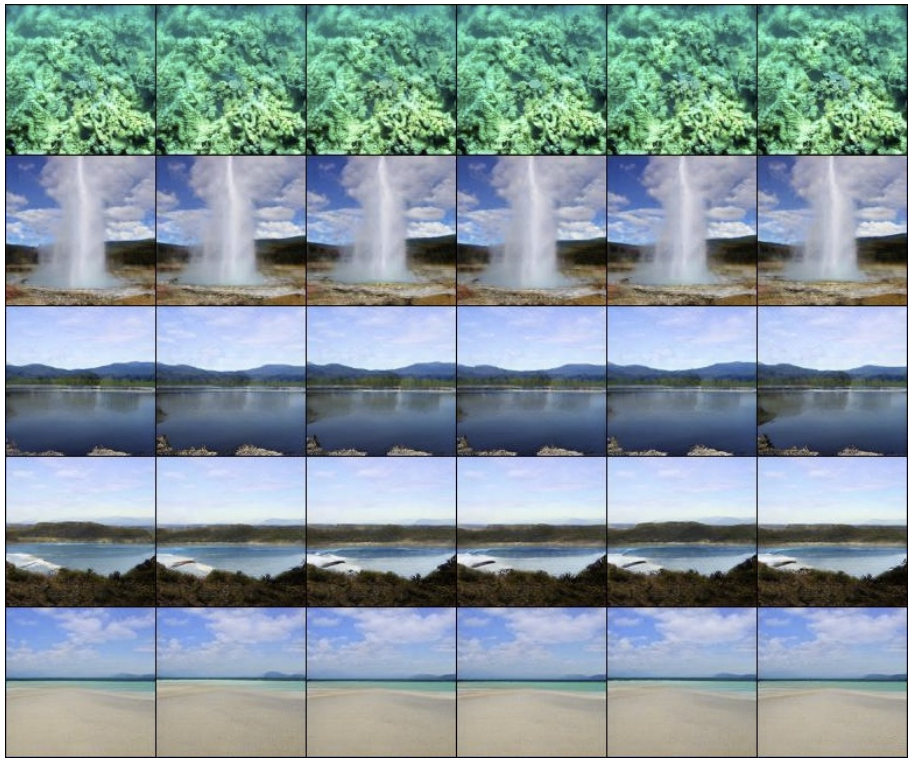

FID
Inception Score
BigGAN
BigGAN-deep
LOGAN (NGD)
SA-GAN
SN-GAN
our baseline
LOGAN (GD)

Samples - High IS (points C,D)

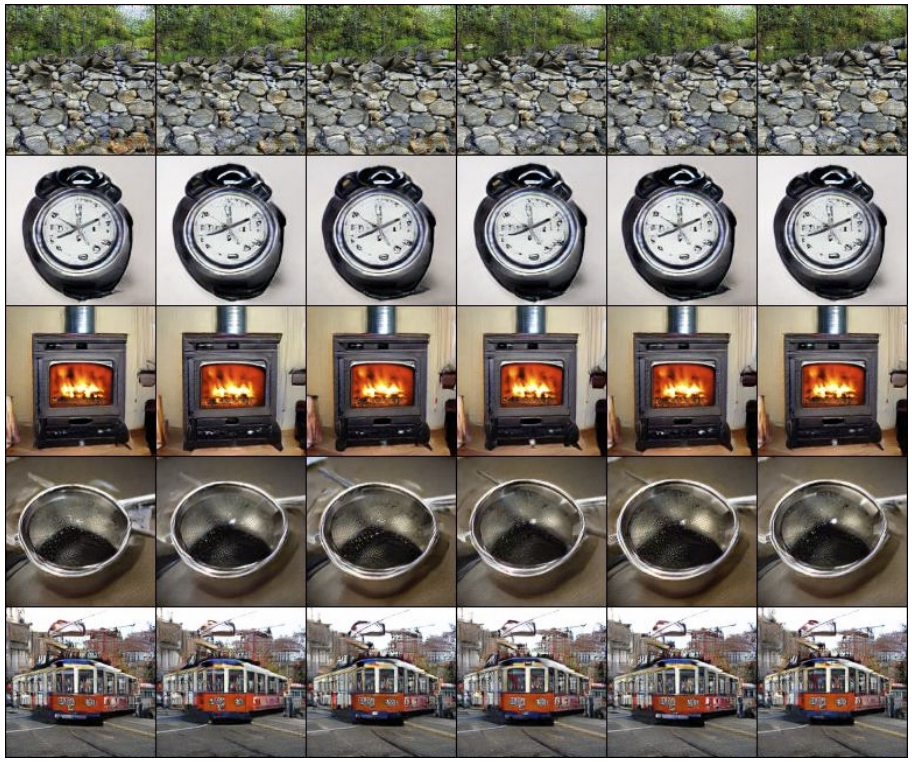
Samples - High IS (points C,D)



Samples - High IS (points C,D)



Samples - High IS (points C,D)



Samples - Low FID (points A,B)

Samples - Low FID (points A,B)

Samples - Low FID (points A,B)

Samples - Low FID (points A,B)

Negative review

The LOGAN paper was not accepted to ICLR 2020.

- Main criticism: authors do not compare to SGA. The reviewers claim that computing the Hessian vector product is not that expensive and they should've done it.

Negative review

The LOGAN paper was not accepted to ICLR 2020.

- Main criticism: authors do not compare to SGA. The reviewers claim that computing the Hessian vector product is not that expensive and they should've done it.
- Authors did not cite concurrent submission to ICLR 2020 :)

Notes

I did not talk about:

- Relation with Unrolled GANs, and that SGA can be seen as approximating Unrolled GANs
- Relation with stochastic approximation with two timescales.

Takeaways

- Mainly experimental paper.
- Introduces a simplfication of the SGA update.

\+ Coupling the generator/discriminator via latent optimization improves sample quality.

\- Latent optimization has a higher cost per iteration (they claim about 3x slower).

Conclusion

- Multiobjective optimization is an area of game theory.
- Game theory offers various solution concepts.
- It's not clear what is the "best" solution concept.
- It's hard to converge to Nash equilibria.
- We might be satisfied with stable fixed points.
- It's possible to update simultaneous gradient descent to converge to stable FP.
- This is called symplectic gradient adjustment (SGA).
- SGA can be applied to GANs of any architecture.
- LOGAN is an approximation of SGA to GANs.
- LOGAN generates higher quality and more diverse images.

Questions?

- Hannan consistency of SGA? Convergence to correlated equilibria?
- Relation to algorithms like exploitability descent?
- Find a simple failure case: finds local NE but not a global NE
- What GANs (do not) use coupling and how it influences performance?

Strategy profile $\sigma = (\sigma_i, \sigma_{-i})$ is an ϵ-*Nash equilibrium* (ϵ-NE) if

$$(\forall i \in N) \ : \ u_i(\sigma) \geq \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i}) - \epsilon.$$

Computation of NE (1/2)

For two-player zero-sum normal-form games

Computation of NE (1/2)

For two-player zero-sum normal-form games

Exact NE: linear programming

Computation of NE (1/2)

For two-player zero-sum normal-form games

Exact NE: linear programming

Approximate $\epsilon-$NE (iterative algorithms):

- Regret-matching
- Regret matching+
- Hedge
- ...

Computation of NE (1/2)

For two-player zero-sum normal-form games

Exact NE: linear programming

Approximate $\epsilon-$NE (iterative algorithms):

- Regret-matching
- Regret matching+
- Hedge
- ...

Goal of regret minimization: ensure that average regret approaches zero, regardless of the opponents' strategies.

Computation of NE (2/2)

Any algorithm that is external regret minimizing, or ϵ-Hannan consistent (HC) converges to 2ϵ-Nash eq.

Computation of NE (2/2)

Any algorithm that is external regret minimizing, or ϵ-Hannan consistent (HC) converges to 2ϵ-Nash eq.

- Player's strategy is ϵ-HC in repeated NFG if against any opponent's strategy it's external regret is $\leq \epsilon$ in the limit.
- I.e. cumulative regret grows sublinearly, so if we take average strategy it's regret approaches zero.

Computation of NE (2/2)

Any algorithm that is external regret minimizing, or ϵ-Hannan consistent (HC) converges to 2ϵ-Nash eq.

- Player's strategy is ϵ-HC in repeated NFG if against any opponent's strategy it's external regret is $\leq \epsilon$ in the limit.
- I.e. cumulative regret grows sublinearly, so if we take average strategy it's regret approaches zero.

Most of the work we do in our group is based on extensions of regret-matching to sequential games (CFR).