Can we optimize **and** learn in nonconvex-nonconcave game?

Gauthier Gidel, Mila and University of Montreal

All-Russian Opt Seminar, March 30th 2022

Outline

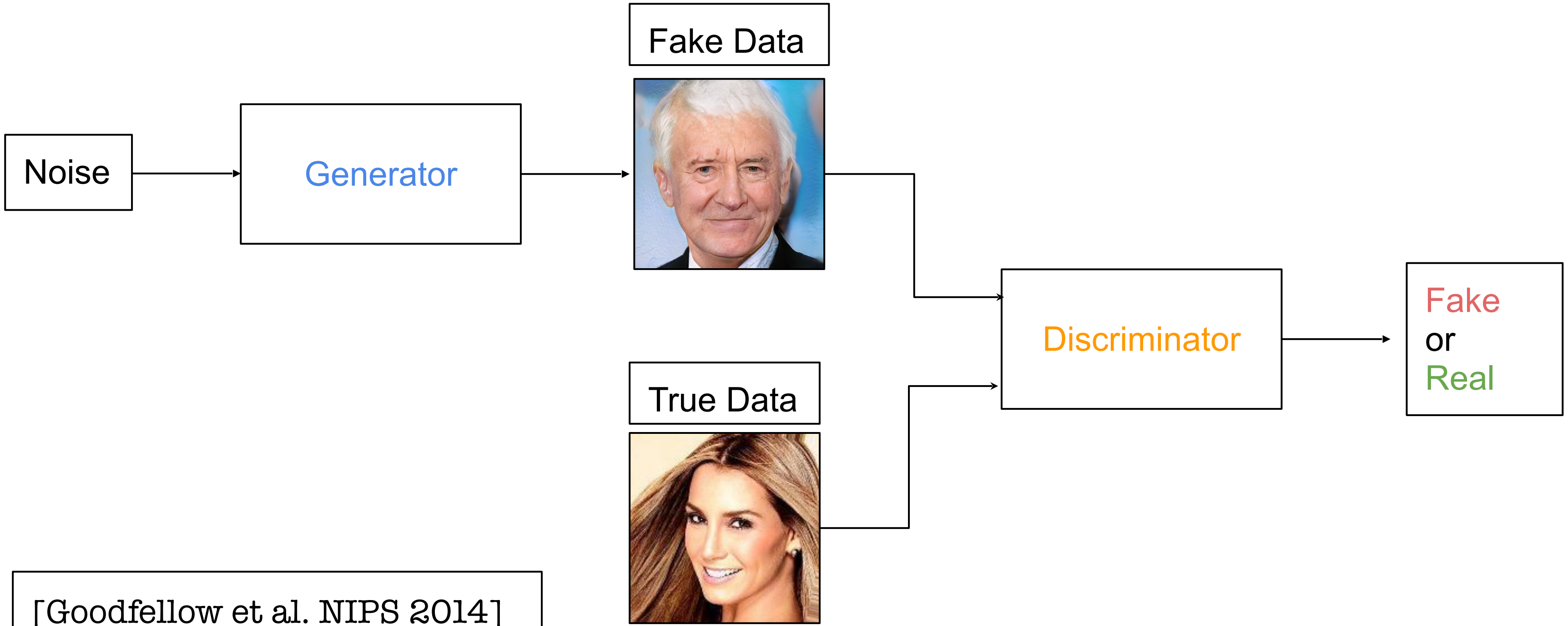
- Part 1: Can we optimize **and** learn in nonconvex-nonconcave games?
- Part 2: a idea; hidden convex-concave games.

The implicit biais of Adam for GANs training

Gauthier Gidel, Mila and University of Montreal

Joint work with Samy Jelassi, Arthur Mensch and Yuanzhi Li

GANs



[Goodfellow et al. NIPS 2014]

Payoff of GANs

Generator

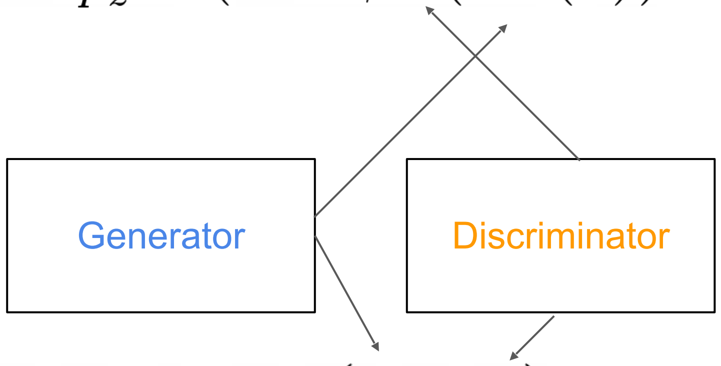
$$\varphi(p_G, \psi) := \underbrace{\mathbb{E}_{x' \sim data}[\ln \psi(x')]}_{\text{How well the } \textbf{dataset} \text{ is classified as } \textbf{“real”}} + \underbrace{\mathbb{E}_{x \sim p_G}[\ln(1 - \psi(x))]}_{\text{How well the } \textbf{fake image} \text{ is classified as } \textbf{“fake”}}$$

Discriminator

. . .

Training Dynamics

$$\varphi(\theta, w) = \mathbb{E}_{x' \sim p_{data}} \ln \psi_w(x') + \mathbb{E}_{z \sim p_z} \ln(1 - \psi_w(G_\theta(z)))$$



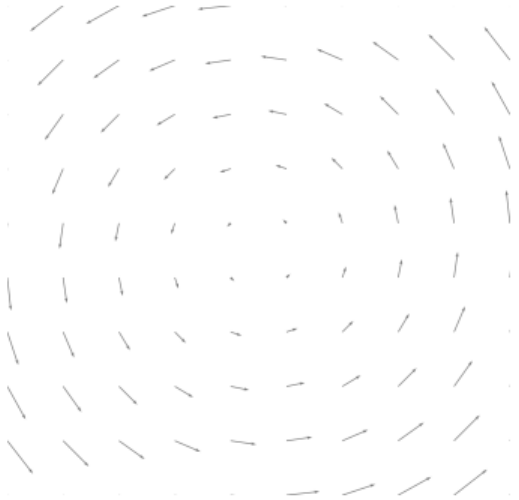
$$\min_\theta \max_w \varphi(\theta, w)$$

Training: The Deep Learning way!

1. Compute (stochastic) gradient for theta
2. Update theta with your **favorite optimizer**
3. Compute gradient for w
4. Update w
5. Repeat

Principled optimization for minimax games

BUT Gradient descent ascent does (should) not work!



Principled minimax methods for GANs:

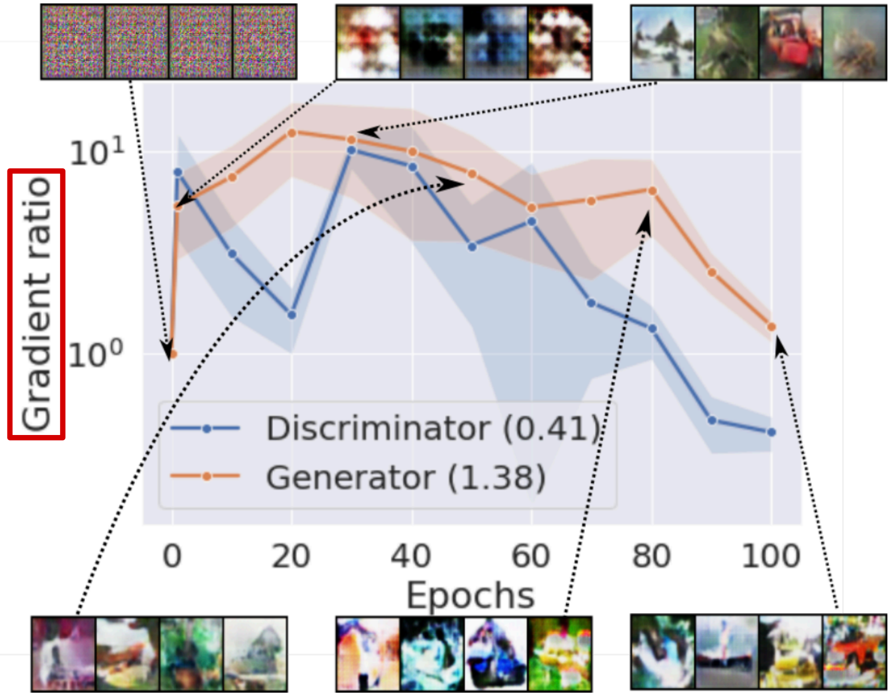
- Two timescale updates [Heusel et al. 2017]
- Optimistic method [Daskalakis et al. 2018]
- Hamiltonian Gradient Descent [Balduzzi et al. 2018]
- Extra-Gradient [G. et al 2019],
- Negative momentum [G. et al. 2019]
- Anchoring [Ryu et al 2019]

An inconvenient truth

Gradient descent ascent should not work?

- It doesn't (from an **optimization** point of view)

- It does (from a **learning** point of view)



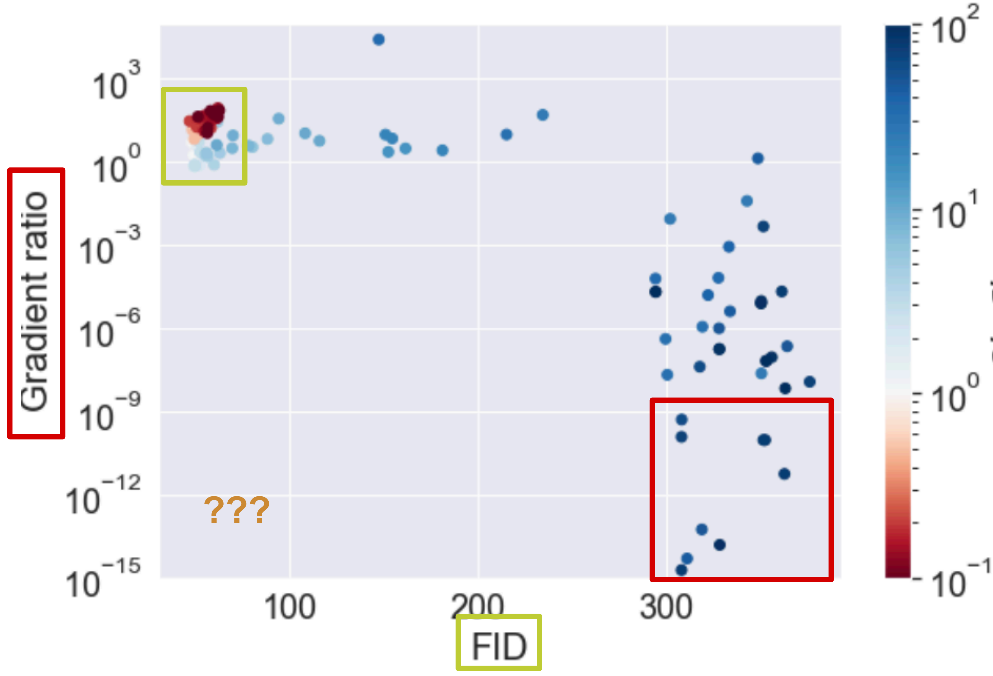
An inconvenient truth

Gradient descent ascent should not work?

- It doesn't (from an **optimization** point of view)

- It does (from a **learning** point of view)

We are not able to learn properly AND solve the optimization problem.



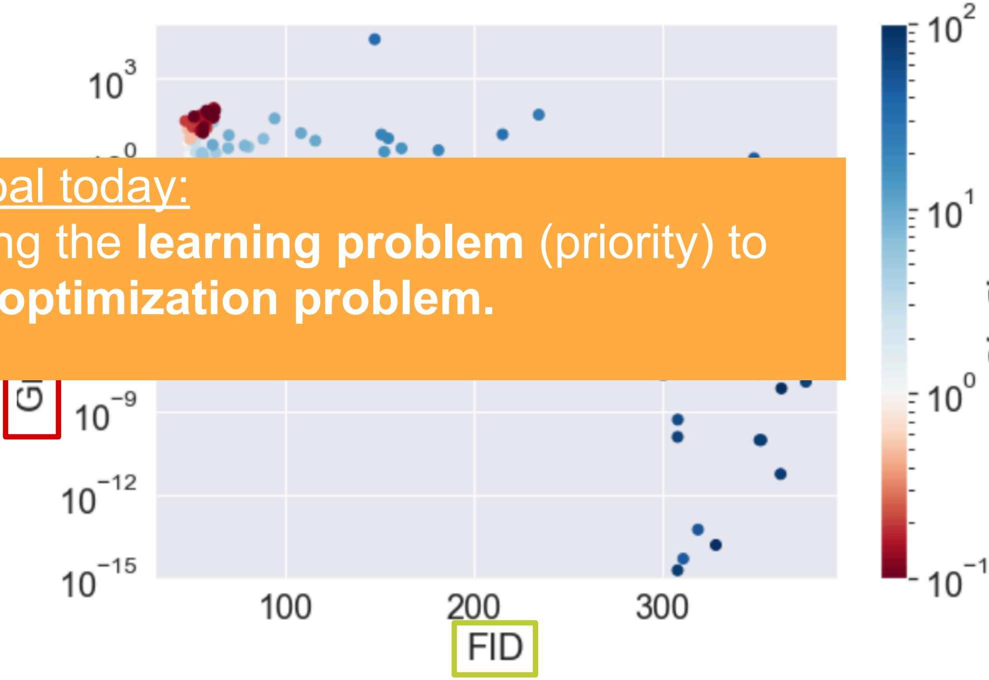
An inconvenient truth

Gradient descent ascent should not work?

- It doesn't (from an **optimization** point of view)

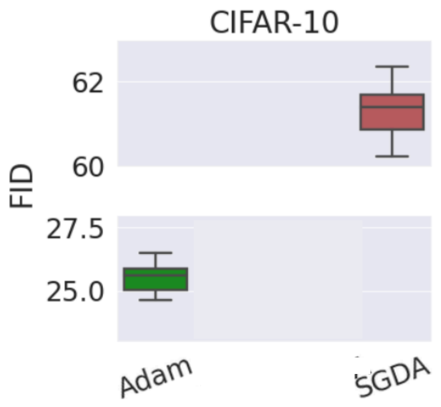
- It do

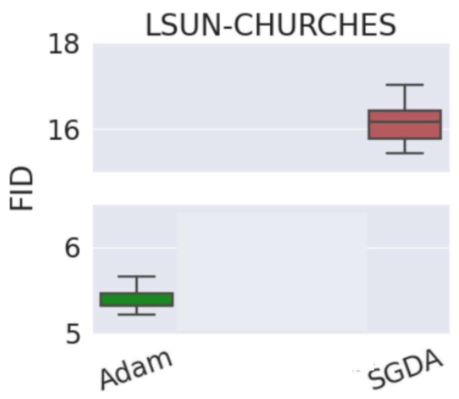
We are not able to learn properly AND solve the optimization problem.



My Goal today:
Find out which methods are solving the **learning problem** (priority) to eventually solve the **optimization problem.**

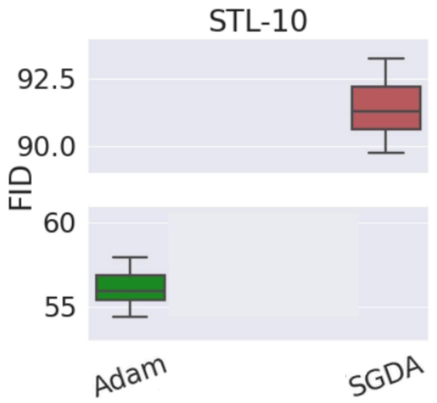
An inconvenient truth

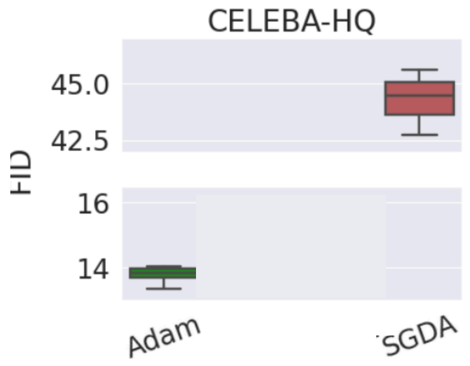




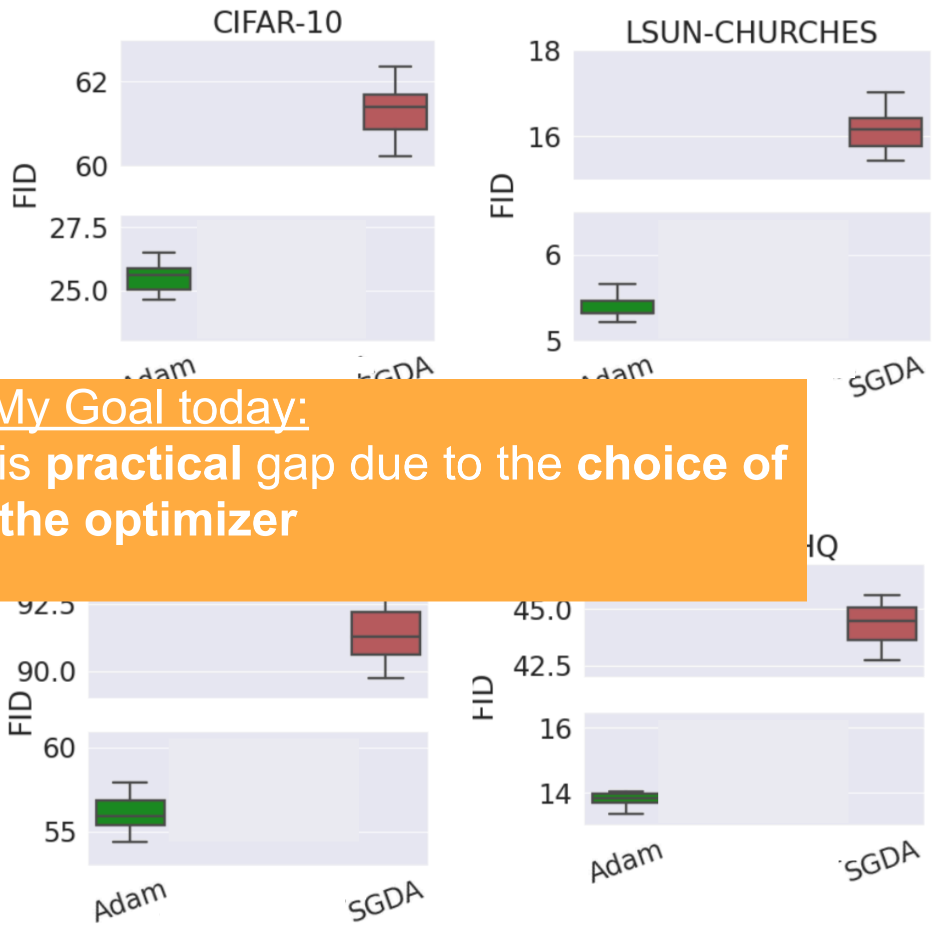
It seems that we need to use Adam to "solve" the **learning problem.**

BUT WHY?





An inconvenient truth



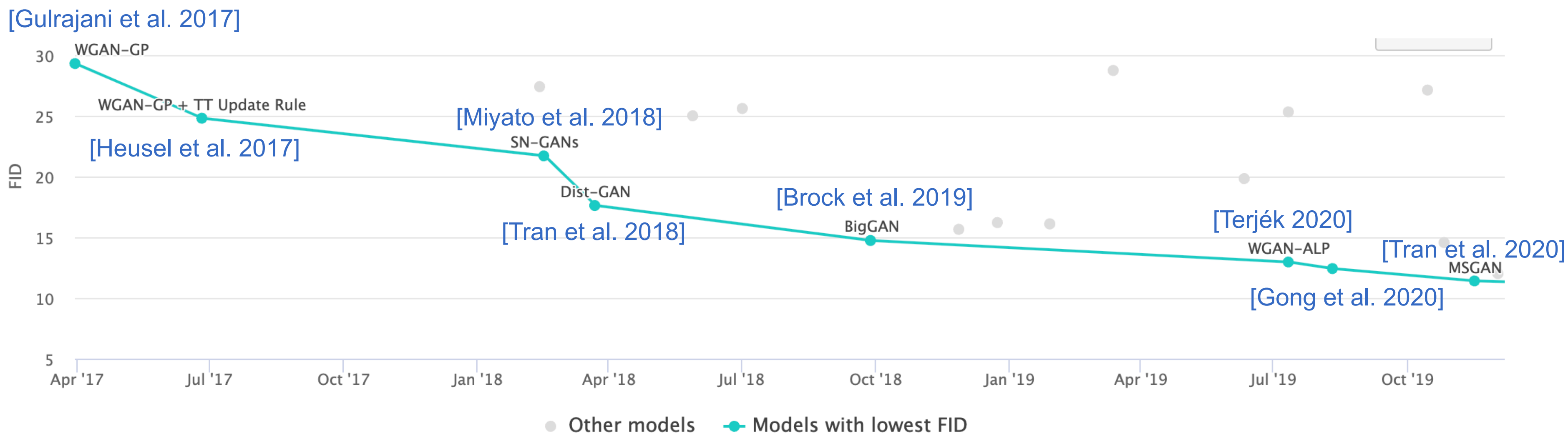
It seems that we n
“solve” the **learnin**

BUT WHY?

My Goal today:
I want to understand this **practical** gap due to the **choice of the optimizer**

Some Facts; beyond the anecdote

- SGD is **outperformed (significantly) by Adam.**
- Adam is the optimizer of choice of **all the latest SOTA results on GANs**:



Source: https://paperswithcode.com/sota/image-generation-on-cifar-10

Some Facts; beyond the anecdote

- SGD is **outperformed (significantly) by Adam.**
- Adam is the optimizer of choice of **all the latest SOTA results on GANs**:

- Adam (not that principled) is added on top of principled methods:
  - Two timescale updates [Heusel et al. 2017]
  - Optimistic Adam [Daskalakis et al. 2018]
  - Extra-Adam [G. et al 2019],
  - Negative momentum with Adam [G. et al. 2019]
  - Adam with anchoring [Ryu et al 2019]

Question: Why does Adam do that SGD does not?

What does Adam do?

- It’s an adaptive method.
- Many justifications to explain why it works well in practice:
  - Rescale “each coordinate” of the gradient. (so moves globally faster)
  - Avoid saddle points [Ovieto et al. 2021]
- But:
  - Oral presentation at Neurips 2017: [Wilson et al. 2017] Outperformed by SGD for image classification
  - Best paper award at ICLR 2018 [Reddi et al. 2018] showed it **does not converge!**

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t.$$

Coordinate-wise renormalization

Momentum

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

What does Adam do?

$$\theta_{t+1} = \theta_t - \underbrace{\frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t}_{\boxed{\text{Has a \textbf{direction} and a \textbf{norm} that differs from SGD}}}.$$

What we did

Check whether it is the **direction** or the **norm** (or both?) of the Adam update that implies superior performance

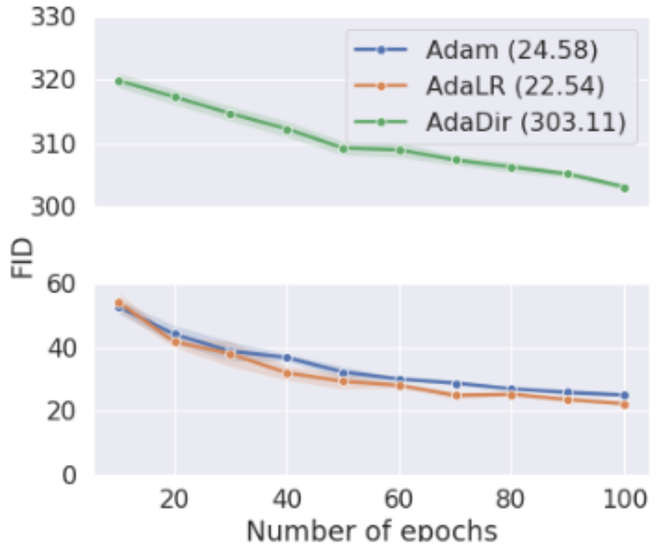
AdaLR:

$$\mathcal{W}^{(t+1)} = \mathcal{W}^{(t)} + \eta_D^{(t)} \overbrace{\|\mathbf{A}_{\mathcal{W}}^{(t)}\|_2}^{\textbf{norm} \text{ of Adam}} \underbrace{\frac{\mathbf{M}_{\mathcal{W},1}^{(t)}}{\|\mathbf{M}_{\mathcal{W},1}^{(t)}\|_2 + \varepsilon}}_{\textbf{Direction} \text{ of SGD}}, \quad \mathcal{V}^{(t+1)} = \mathcal{V}^{(t)} - \eta_G^{(t)} \|\mathbf{A}_{\mathcal{V}}^{(t)}\|_2 \frac{\mathbf{M}_{\mathcal{V},1}^{(t)}}{\|\mathbf{M}_{\mathcal{V},1}^{(t)}\|_2 + \varepsilon},$$

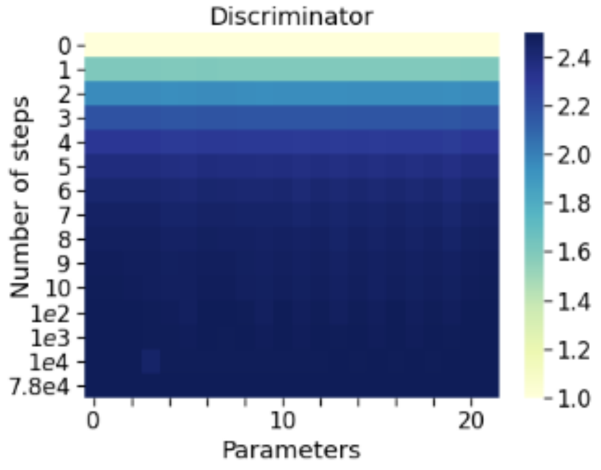
AdaDir:

$$\mathcal{W}^{(t+1)} = \mathcal{W}^{(t)} + \underset{\text{(constant step-size)}}{\eta_D^{(t)}} \overbrace{\|\mathbf{M}_{\mathcal{W},1}^{(t)}\|_2}^{\textbf{norm} \text{ of SGD}} \underbrace{\frac{\mathbf{A}_{\mathcal{W}}^{(t)}}{\|\mathbf{A}_{\mathcal{W}}^{(t)}\|_2 + \varepsilon}}_{\textbf{Direction} \text{ of Adam}}, \quad \mathcal{V}^{(t+1)} = \mathcal{V}^{(t)} - \eta_G^{(t)} \|\mathbf{M}_{\mathcal{V},1}^{(t)}\|_2 \frac{\mathbf{A}_{\mathcal{V}}^{(t)}}{\|\mathbf{A}_{\mathcal{V}}^{(t)}\|_2 + \varepsilon}.$$

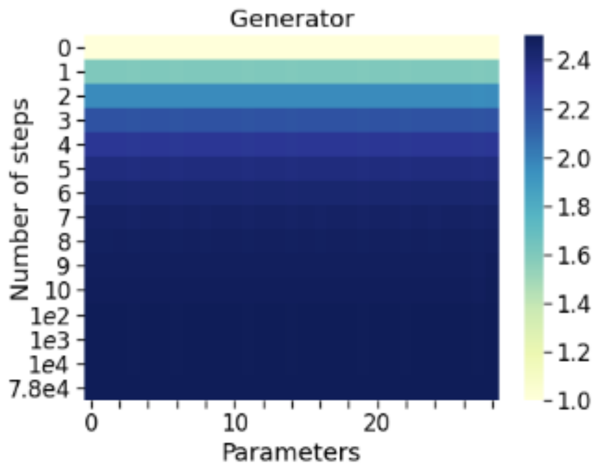
Results



(a)



(b)



(c)

1. AdaLR is as good as Adam (i.e. the direction does **not** matter)
2. The norm of the Adam direction is **constant across time.**

The method that worked

AdaLR:

$$\mathcal{W}^{(t+1)} = \mathcal{W}^{(t)} + \eta_D \overbrace{\|\mathbf{A}_{\mathcal{W}}^{(t)}\|_2}^{\textbf{norm}\text{ of Adam}} \underbrace{\frac{\mathbf{M}_{\mathcal{W},1}^{(t)}}{\|\mathbf{M}_{\mathcal{W},1}^{(t)}\|_2 + \varepsilon}}_{\textbf{Direction}\text{ of SGD}}, \quad \mathcal{V}^{(t+1)} = \mathcal{V}^{(t)} - \eta_G \|\mathbf{A}_{\mathcal{V}}^{(t)}\|_2 \frac{\mathbf{M}_{\mathcal{V},1}^{(t)}}{\|\mathbf{M}_{\mathcal{V},1}^{(t)}\|_2 + \varepsilon},$$

If constant (verified in practice)

Hypothesis:

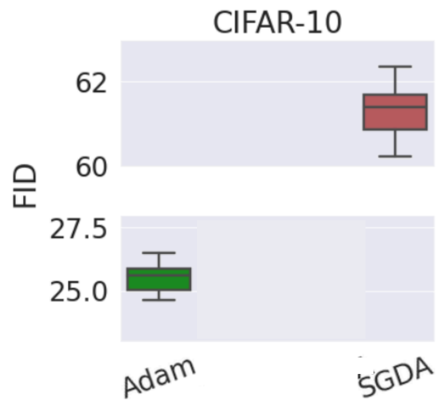
$$\mathcal{W}^{(t+1)} = \mathcal{W}^{(t)} + \eta_D \frac{\mathbf{M}_{\mathcal{W},1}^{(t)}}{\|\mathbf{M}_{\mathcal{W},1}^{(t)}\|_2 + \varepsilon}, \qquad \mathcal{V}^{(t+1)} = \mathcal{V}^{(t)} - \eta_G \frac{\mathbf{M}_{\mathcal{V},1}^{(t)}}{\|\mathbf{M}_{\mathcal{V},1}^{(t)}\|_2 + \varepsilon}.$$

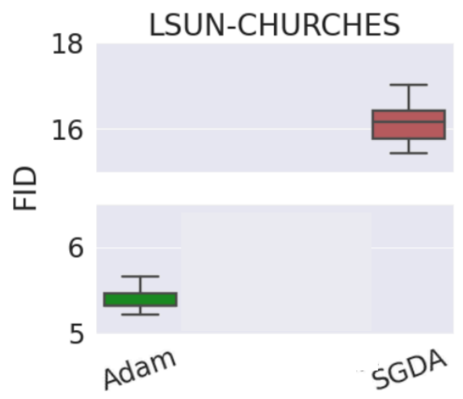
Should work as well as Adam!

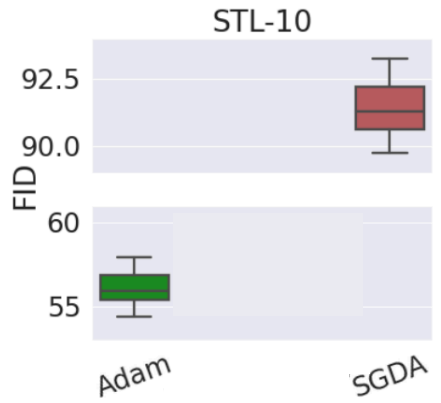
NormSGDA vs. Adam

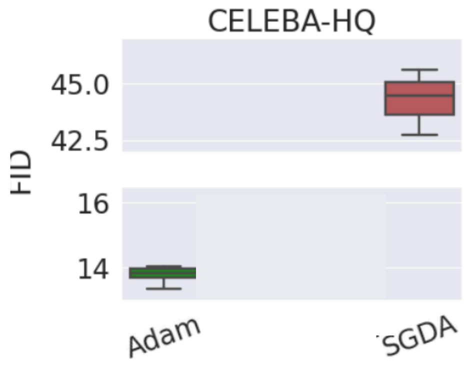
l-nSGDA: layer normalized SGDA

g-nSGDA: normalized SGDA









Experimental Conclusion

- ~~(In our GAN setting) normalized SGD is a proxy for Adam~~
- Adam is a proxy for normalized SGD!

<u>Why is it great:</u>

1. Way easier to analyze and understand normalized SGD and its implicit biaises.
2. Way easier to tune (significantly less hyperparameters)
3. Room for improvement (easier to build on top of Normalized SGD than Adam)

Analysis of Normalized SGDA vs SGDA

Linear Generator:

$$G_{\mathcal{V}}(z) = Vz = \sum_{i=1}^{m_G} v_i z_i,$$

Distribution with two modes u_1, u_2:

data-point $X = s_1 u_1 + s_2 u_2 \in \mathbb{R}^d$.

with $\langle u_1, u_2 \rangle > 0$ and $X = u_1$ or $X = u_2$

Binary Latent random variable: $\Pr[z_i = 1] = \Theta\left(\frac{1}{m_G}\right), \quad \Pr[z_i = z_j = 1] = \frac{1}{m_G^2 \mathrm{polylog}(d)}$

[Allen-Zhu & Li (2021)]: z_i can be seen at the distribution of the hidden layers of a deeper NN: they are sparse, non-negative, and non-positively correlated.

Analysis of Normalized SGDA vs SGDA

Cubic Discriminator:

$$D_{\mathcal{W}}(X) = \text{sigmoid}\left(a \sum_{i \in [m_D]} \sigma(\langle w_i, X \rangle) + \lambda b \right), \quad \sigma(z) = \begin{cases} z^3 & \text{if } |z| \leq \Lambda \\ 3\Lambda^2 z - 2\Lambda^3 & \text{if } z > \Lambda \\ 3\Lambda^2 z + 2\Lambda^3 & \text{otherwise} \end{cases},$$

Objective:

$$\min_{\mathcal{V}} \max_{\mathcal{W}} \quad \mathbb{E}_{X \sim \mathcal{D}}[\log(D_{\mathcal{W}}(X))] + \mathbb{E}_{z \sim \mathcal{D}_z}[\log(1 - D_{\mathcal{W}}(G_{\mathcal{V}}(z)))].$$

Our Results: SGDA

Informal: For any stepwise choice w.h.p. SGD suffer from **mode collapse**.
(Precisely, can only learn the direction of $u_1 + u_2$).

Theorem 4.1 (SGDA suffers from mode collapse)**.** *Let T_0, η_G, η_D and the initialization as defined in Parametrization 4.1. Let t be such that $t \leq T_0$. Run SGDA for t iterations with step-sizes η_G, η_D. Then, with probability at least $1 - o(1)$, for all $z \in \{0,1\}^{m_G}$, we have:*

$$G_{\mathcal{V}}^{(t)}(z) = \alpha^{(t)}(z)(u_1 + u_2) + \xi^{(t)}(z), \quad \textit{where } \alpha^{(t)}(z) \in \mathbb{R} \textit{ and } \xi^{(t)}(z) \in \mathbb{R}^d,$$

such that for all $\ell \in [2]$, $|\langle \xi^{(t)}(z), u_\ell \rangle| = o(1)\|\xi^{(t)}(z)\|_2$ for every $z \in \{0,1\}^{m_G}$.

In the specific case where $\eta_G = \frac{\sqrt{d}\eta_D}{\mathrm{polylog}(d)}$, the model mode collapses i.e. $\|\xi^{(T_0)}(z)\|_2 = o(\alpha^{(T_0)}(z))$.

Our Results: normalized SGDA

Informal: For a correct choice of step-size nSGDA **learns the direction of both modes**.

Theorem 4.2 (nSGDA recovers modes separately). *Let T_1, η_G, η_D and the initialization as defined in Parametrization 4.1. Run nSGDA for T_1 iterations with step-sizes η_G, η_D. Then, the generator learns both modes u_1, u_2 i.e.,*

$$\Pr_{z \sim \mathcal{D}_z}\left(\left\| \frac{G_{\mathcal{V}}^{(T_1)}(z)}{\|G_{\mathcal{V}}^{(T_1)}(z)\|_2} - u_\ell \right\|_2 = o(1) \right) = \tilde{\Omega}(1), \quad \textit{for} \quad \ell = 1, 2\,. \tag{10}$$

Remarks:

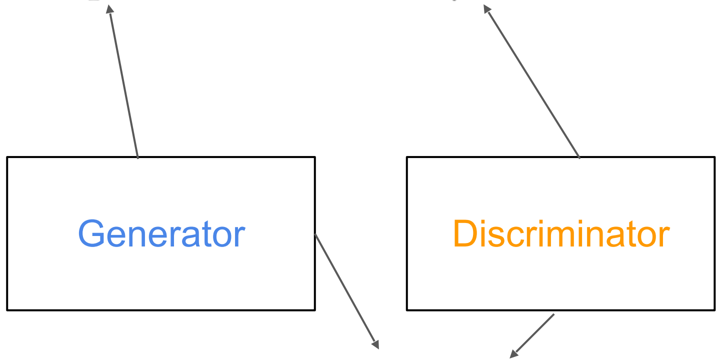
1. For specific initialization and choice of Generator/Discriminator.
2. Result only about learned **directions** (not the norm).
3. No guarantees that nSGD learns the correct weighting for $X = s_1 u_1 + s_2 u_2$.

Conclusion (part 1)

- GANs has some very bad (approximate) stationary points/local equilibrium.
- We can find them if we focus on the ‘optimization problem’
- Harder the solve the **learning problem** and the **optimization**. (Is it even possible?)
- **Normalized gradients** seems to be promising for training GANs.
- But no convergence for this!
- Can we have an algorithm that **learns and optimize properly?**
- Can we have something in between **convex-concave and nonconvex-nonconcave? (YES)**

Hidden convex concave objectives

$$\Psi(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\mathbf{x} \sim p_{data}} [\log D_{\boldsymbol{\phi}}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{\theta}}} [\log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{x}))] \,.$$



$$\min_{\boldsymbol{\theta} \in \mathbb{R}^M} \max_{\boldsymbol{\phi} \in \mathbb{R}^N} \Psi(\boldsymbol{\theta}, \boldsymbol{\phi})$$

- Nonconvex-nonconcave objective: (w.r.t θ and w)
- But **HIDDEN** Convex concave !!!

Generalized Natural Gradient Flows in Hidden Convex-Concave Games and GANs

Andjela Mladenovic, Iosif Sakos, Georgios Piliouras

Hidden convex concave objectives

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^M} \max_{\boldsymbol{\phi}\in\mathbb{R}^N} \Psi(\boldsymbol{\theta},\boldsymbol{\phi}) \quad where \quad \Psi(\boldsymbol{\theta},\boldsymbol{\phi}) = L(F_{\boldsymbol{\theta}}, G_{\boldsymbol{\phi}}) := \mathbb{E}_{(\boldsymbol{x},\boldsymbol{x}')\sim p}[L_{\boldsymbol{x},\boldsymbol{x}'}(F_{\boldsymbol{\theta}}(\boldsymbol{x}), G_{\boldsymbol{\phi}}(\boldsymbol{x}'))]\,.$$

L is convex-concave
Because $L_{x,x'}(\,\cdot\,,\,\cdot\,)$ is convex concave

$\theta \mapsto F_\theta(x)$ and $w \mapsto G_w(x')$
are usually neural networks!

Example:

$$\Psi(\theta,\phi) = \mathbb{E}_{x\sim p_{data}}[\log D_\phi(x)] + \mathbb{E}_{x\sim p_\theta}[\log(1 - D_\phi(x))]$$

$$= \mathbb{E}_{x\sim p_{data}}[\log D_\phi(x) + \frac{p_\theta(x)}{p_{data}(x)} \log(1 - D_\phi(x))]$$

$$= \mathbb{E}_{x\sim p_{data}}[L_x(p_\theta(x), D_\phi(x))]$$

where $\quad L_x(p, D) := \log D + \frac{p}{p_{data}(x)} \log(1 - D)$

Why are hidden convex-concave games interesting

- Simpler than general nonconvex-nonconcave games where impossibility results are know [Hsieh, Mertikopoulos, Cevher, 2021],[Letcher 2021]

- Capture the complexity of our current application (i.e., *the non convexity comes for the Neural Networks).*

- Proposed by *[Flokas et al. 2019]* for hidden bilinear games then by *[Flokas et al. (2021)] and [Gidel et al. (2021)]* in the general convex-concave setting.

- Just like non convex optimization, we can show that we have a “global target” (solution of the convex-concave minimax game in the function space *[Gidel et al 2021]*)
- In this setting: we have tools to differentiate “good” from “bad” local stationery points (how close they are from the target).

A global target

- There exists a Nash (F^*, G^*):

$$\min_{F \in \mathcal{F}} \max_{G \in \mathcal{G}} L(F, G) = \max_{G \in \mathcal{G}} \min_{F \in \mathcal{F}} L(F, G) = L(F^*, G^*)$$

- $\mathcal{F} = hull\{F_\theta \ : \ \theta \in \Theta\}, \qquad \mathcal{G} = hull\{G_\phi \ : \ \phi \in \Phi\}$

- Does exists if $\mathcal{F}$ and $\mathcal{G}$ are convex compact sets!
- Exists under mild assumptions (continuous mappings and compacts sets of parameters).
- May not be achievable (because of the convex hull).
- For Neural networks almost no need to take the hull *[Gidel et al. 2021]*.

How do we find this target?

- First method: gradient flow

$$\dot{\theta}(t) = -\nabla_\theta \Psi(\theta(t), \phi(t)) = -\nabla_\theta L(F_\theta(t), G_\phi(t))$$
$$\dot{\phi}(t) = \nabla_\phi \Psi(\theta(t), \phi(t)) = \nabla_\phi L(F_\theta(t), G_\phi(t))$$

- Here we are **not** leveraging the hidden convex-concavity!

- **<u>Intuition</u>:** in the space of mapping the payoff is convex-concave.

- We want to update **the parameters** with a gradient update that correspond to a gradient step in the space of F_θ and G_ϕ.

Natural Gradients

- Geometry induced in the mapping space

$$\|F_{\theta+\delta\theta} - F_\theta\|^2 = \langle \delta\theta, A_\theta \delta\theta \rangle + o(\|\delta\theta\|^2)$$

- Natural gradient (steepest descent direction with this "new inner product")

$$\dot{\theta}(t) = -A_\theta^{-1} \nabla_\theta \Psi(\theta(t), \phi(t))$$
$$\dot{\phi}(t) = B_\phi^{-1} \nabla_\phi \Psi(\theta(t), \phi(t))$$

- Where $A_\theta := \mathbb{E}_{x \sim p_x}[\nabla_\theta F_\theta(x) \nabla_\theta F_\theta(x)^\top]$

Natural Gradient

- Idea dates from (Amari, 1985; 1998)
- Revived in the context of classification (KL divergence) [Martens and Grosse 2015].
- Dynamics independent of the classifier’s parametrization!
- Theoretically studied by (Ollivier et al., 2017)
- Not really looked at (to our knowledge) in the case of the ℓ_2 distance between mappings.

Natural Gradients

- Natural gradient

$$\dot{\theta}(t) = -A_\theta^{-1} \nabla_\theta \Psi(\theta(t), \phi(t))$$
$$\dot{\phi}(t) = B_\phi^{-1} \nabla_\phi \Psi(\theta(t), \phi(t))$$

- **Intuition:** Now we are flowing gradient descent-ascent in the **mapping** space (convex-concave payoff!!!)

- We can hope for **global convergence results!!!!**

- **Need assumptions** (no free lunch :-( )

Lyapunov Function

- Distance to the optimum (Nash):

$$V(\theta, \phi) = \frac{1}{2}\|F_\theta - F^*\|^2 + \frac{1}{2}\|G_\phi - G^*\|^2$$

- May not be achievable but could still decrease.
- Distance in the function space $\|F_1 - F_2\| := \mathbb{E}[(F_1(x) - F_2(x))^2]$

Warm-up

- Simple case $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
- Equivalent to a single datapoint

$$\Psi(\theta, \phi) = L(f(\theta), g(\phi))$$

- Already non-trivial! (Nonconvex-nonconcave)

- Example: Matching penny game with non-linear parametrization of the strategy:

$$\Psi(\theta, \phi) = (1 - 2f(\theta))(1 - 2g(\phi)), \quad f(\theta), g(\phi) \in [0,1]$$

Warm-up

- Objective function: $\Psi(\theta, \phi) = L(f(\theta), g(\phi))$
- Dynamics: $\dot{\theta} = -\dfrac{\nabla_\theta L(f(\theta), g(\phi))}{\|\nabla_\theta f(\theta)\|^2}, \qquad \dot{\phi} = \dfrac{\nabla_\phi L(f(\theta), g(\phi))}{\|\nabla_\phi g(\phi)\|^2}$
- Lyapunov function: $V(\theta, \phi) = \dfrac{1}{2}(f^* - f(\theta))^2 + \dfrac{1}{2}(g^* - g(\phi))^2$
- Theorem: If L is strictly convex-concave, then $V(\theta, \phi) \to 0$.

Warm up

Proof:

$$
\begin{aligned}
\dot{V}(\theta, \phi) &= \langle \dot{\theta}, \nabla_\theta f(\theta) \rangle (f_\theta - f^*) + \langle \dot{\phi}, \nabla_\phi g(\phi) \rangle (g_\phi - g^*) \\
&= -\langle \frac{\nabla_\theta L(f(\theta), g(\phi))}{\|\nabla_\theta f(\theta)\|^2}, \nabla_\theta f(\theta) \rangle (f_\theta - f^*) + \langle \frac{\nabla_\phi L(f(\theta), g(\phi))}{\|\nabla_\phi g(\phi)\|^2}, \nabla_\phi g(\phi) \rangle (g_\phi - g^*) \\
&= -\langle \frac{\nabla_\theta f(\theta) \partial_f L(f(\theta), g(\phi))}{\|\nabla_\theta f(\theta)\|^2}, \nabla_\theta f(\theta) \rangle (f_\theta - f^*) + \langle \frac{\nabla_\phi g(\phi) \partial_g L(f(\theta), g(\phi))}{\|\nabla_\phi g(\phi)\|^2}, \nabla_\phi g(\phi) \rangle (g_\phi - g^*) \\
&= -\partial_f L(f(\theta), g(\phi))(f_\theta - f^*) + \partial_g L(f(\theta), g(\phi))(g_\phi - g^*)
\end{aligned}
$$

This is the optimality conditions for a convex-concave game!!!!!

Finite-Sum

- Finite-sum case

$$\Psi(\theta, \phi) = \frac{1}{nm} \sum_{(i,j)\in[m]\times[b]} L_{i,j}(F_\theta(x_i), G_\phi(x'_j))$$

- Assumption: Along the trajectory, $(\nabla_\theta F_\theta(x_i))_{i\in[m]}$ and $(\nabla_\phi G_\phi(x'_j))_{j\in[n]}$ are linearly independent.

- Theorem: If L is strictly convex-concave, the following Lyapunov function converges to 0.

$$V(\theta, \phi) = \frac{1}{2n} \sum_{i=1}^{n} (F_\theta(x_i) - F^*(x_i))^2 + \frac{1}{2m} \sum_{j=1}^{m} (G_\phi(x'_j) - G^*(x'_j))^2$$

Assumption

- Assumption: Along the trajectory, $(\nabla_\theta F_\theta(x_i))_{i\in[m]}$ and $(\nabla_\phi G_\phi(x'_j))_{j\in[n]}$ are linearly independent.

- We absolutely need **overparametrized** models (mode dimension than data-points).
- Besides that we observed it was true in practice for large enough neural nets.
- Kinda the equivalent of "*almost surely, the trajectory never exactly pass by a saddle point*" in minimization.

Discussion

- Here we get **global convergence But….**
- **Continuous dynamics**
- **Need matrix inversion and full batch gradients….**
- **Still global!!!!**

Experiments (why it is not easy to implement)

- Main problem: matrix inversion.

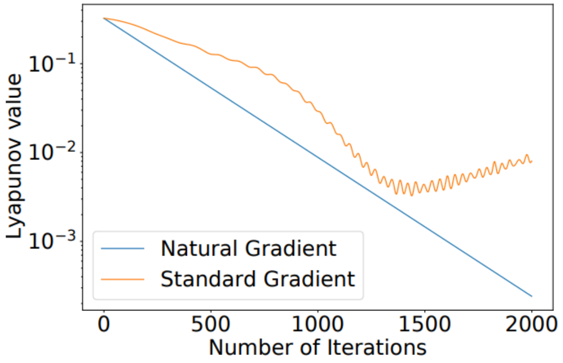
- Already addressed in the standard natural gradient literature:
  - “Standard” natural gradient [Martens and Grosse, 2015; Martens, 2020],
  - Wasserstein natural gradient [Li & Montufar, 2018; Arbel et al., 2020]

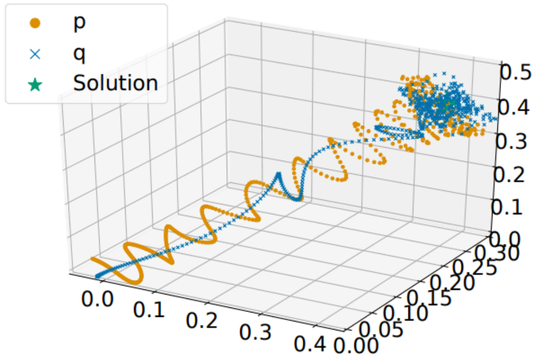
- K-FAC algorithm [Martens and Grosse, 2015] (approximate the Fisher information matrix with Kronecker products)

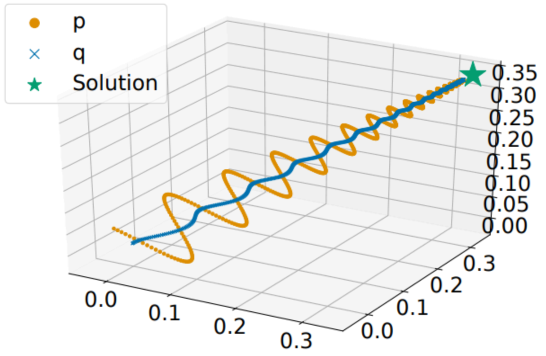
Experimental results

- Rock-Paper Scissors:

$$\Psi(\theta,\phi) = F_\theta^\top M G_\phi + \frac{\lambda}{2}(\|F_\theta - \tfrac{1}{3}\|^2 - \|G_\phi - \tfrac{1}{3}\|^2), \quad M = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}$$



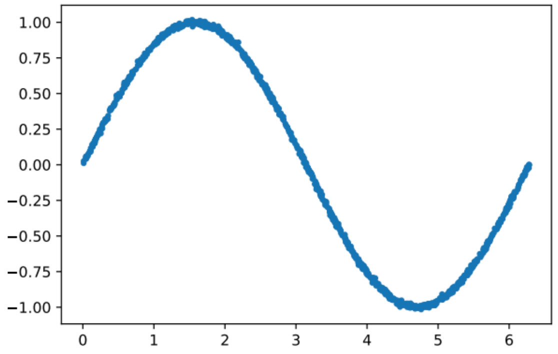




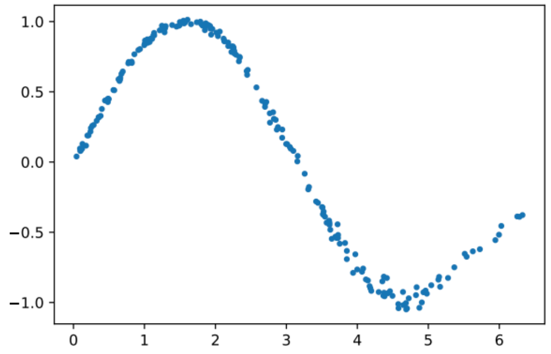
Experimental results

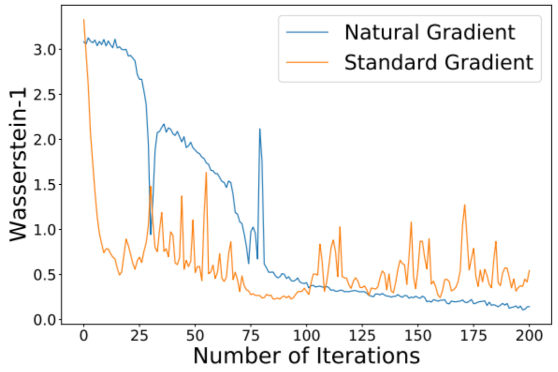
- Simple GAN (discriminators and Generators are neural nets)

True Data



Generated data





Conclusion

- Hidden convex-concave games are very exciting!!!!
- Propose a well defined target (like the global minima in optimization).
- Can leverage the hidden convex-concavity for the optimization algorithms.
- Lots a open questions/directions:
    - Discrete case.
    - What about vanilla GDA (or Extragradient) in such games?
    - Try this for larger GANs

Thanks, Questions?