Beyond minimax optimization: Learning Dynamics in adversarial learning

Gauthier Gidel, Mila and University of Montreal

Adversarial Approaches in ML workshop, Simons Institute, Feb 23rd 2022

Outline

- Part 1: Beyond minimax optimization by considering implicit biaises.

    The implicit biais of Adam for GANs training.

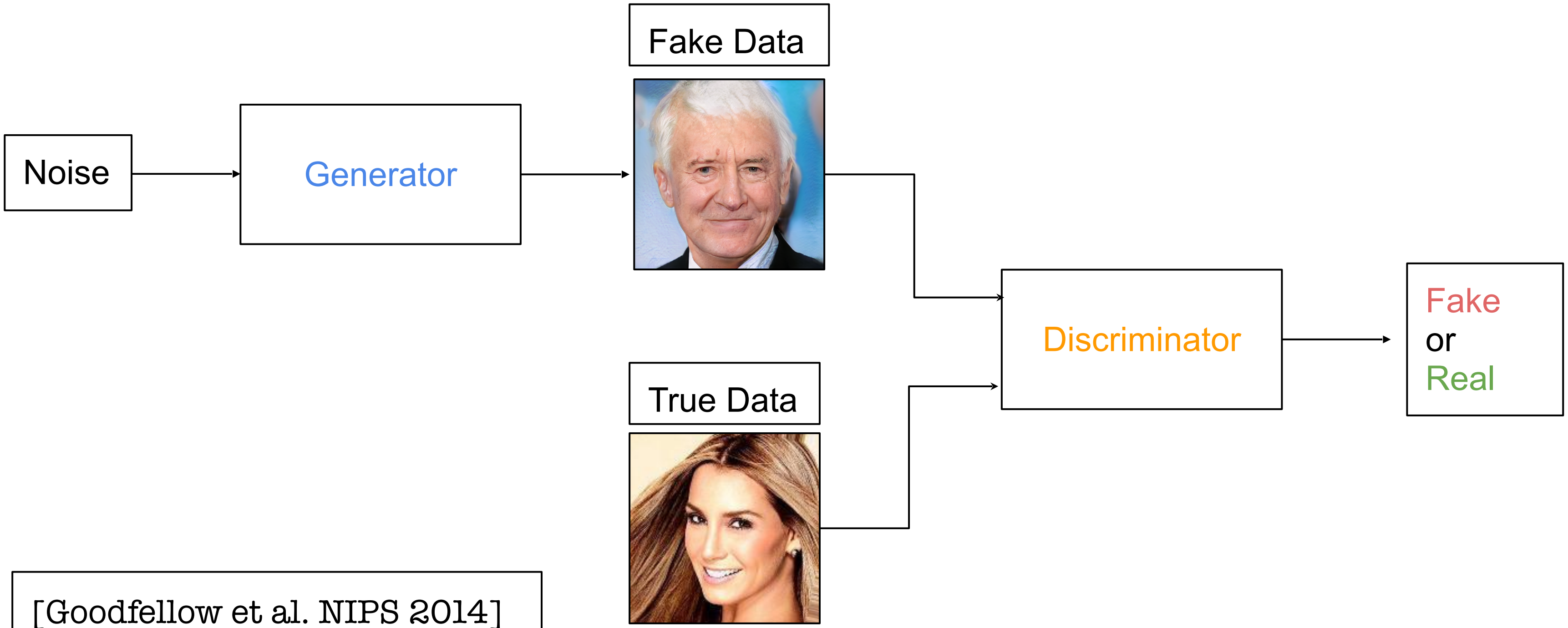
- Part 2: Beyond minimax optimization by replacing optimization by sampling.

    A distributional robustness perspective for adversarial training

The implicit biais of Adam for GANs training

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

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

Generator

Discriminator

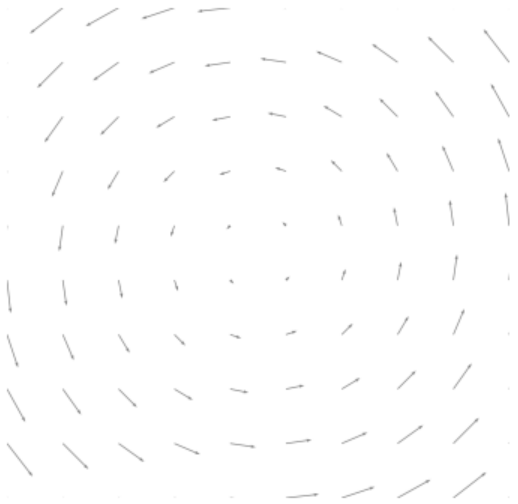
$$\min_{\theta} \max_{w} \varphi(\theta, w)$$

<u>Training:</u> 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!



<u>Principled minimax methods for GANs:</u>

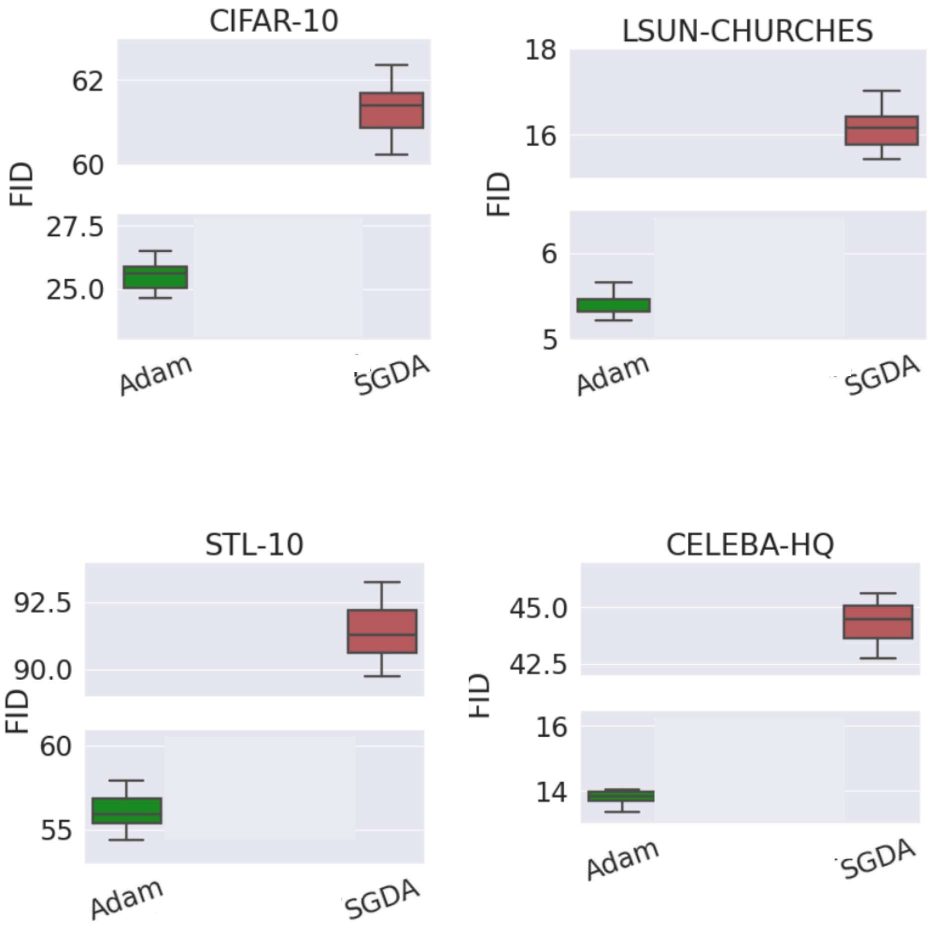
- Two timescale updates [Heusel et al. 2017]
- Optimistic method [Daskalakis et al. 2018]
- Hamiltonian Gradient Descent [Balduzzi et al. 2018]
- Extra-Gradient [G. et al 2019],
- Negative momentum [G. et al. 2019]
- Anchoring [Ryu et al 2019]

An inconvenient truth

Principled methods I tried for GANs:

- Extra-Gradient [G. et al 2019],
- Negative momentum [G. et al. 2019]

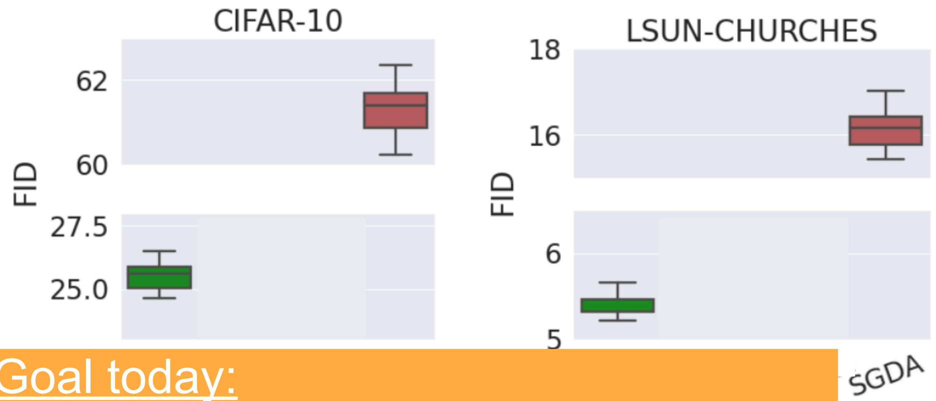
These principled methods alone could not close this gap! (At least for me)



An inconvenient truth

Principled methods I tried for GANs:

- Extra-
- Nega



My Goal today:

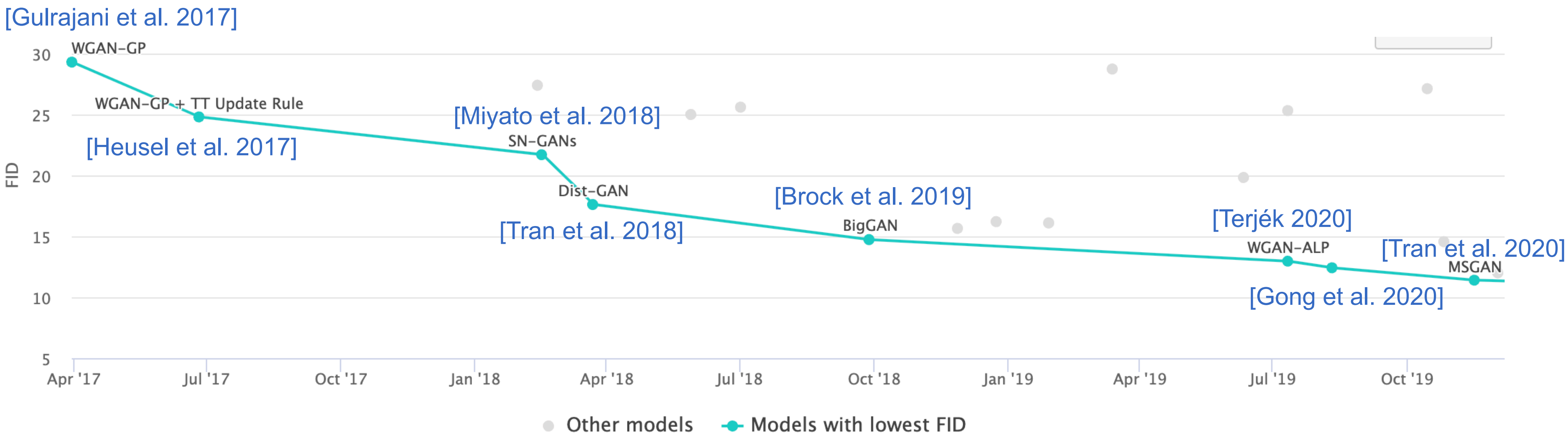
I want to address this **practical** gap due to the **choice of the optimizer**

Then: principled methods for minimax optimization can be used (extragradient/Optimism,…)

These principled methods alone could not close this gap! (At least for me)

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?

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

Has a **direction** and a **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

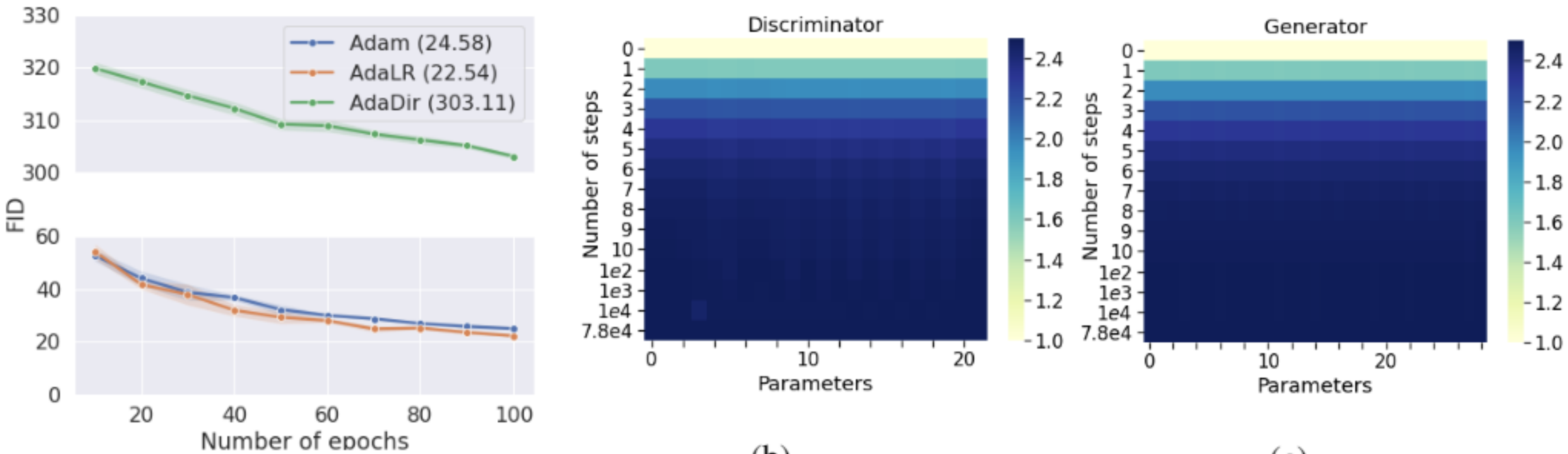
<u>AdaLR:</u>

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

<u>AdaDir:</u>

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

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:

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

If constant (verified in practice)

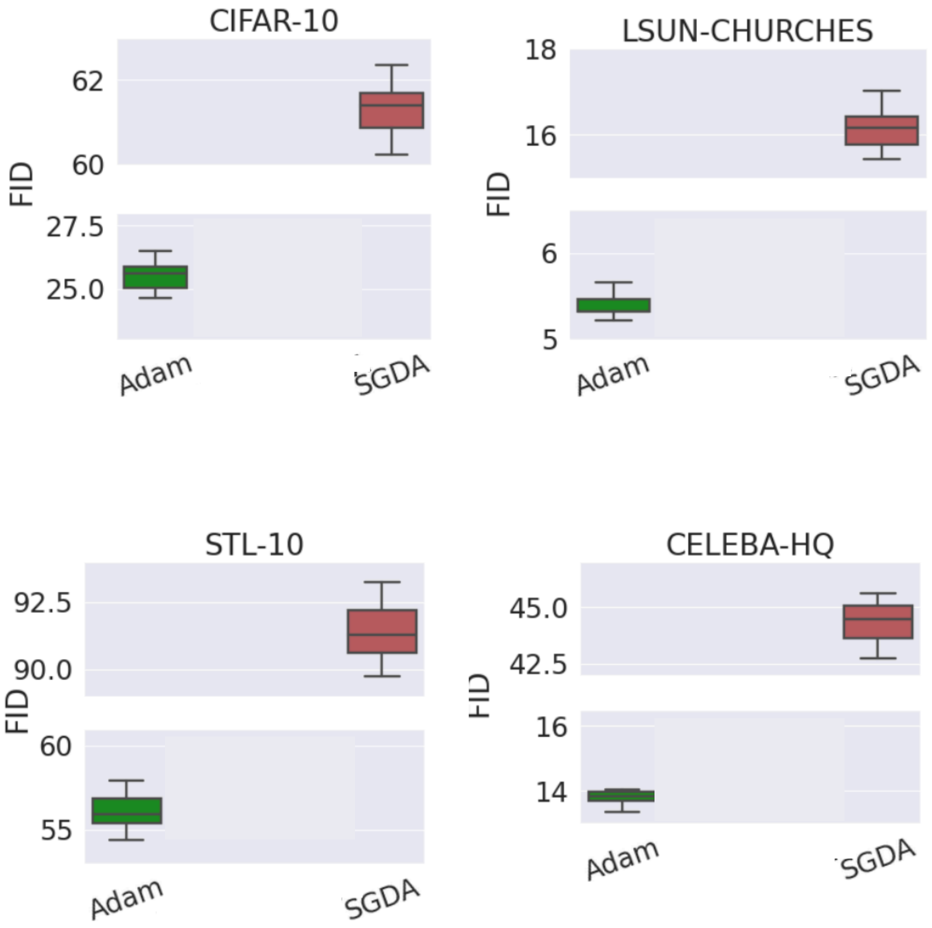
Hypothesis:

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

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!

Why is it great:

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 $u_1 + u_2$).

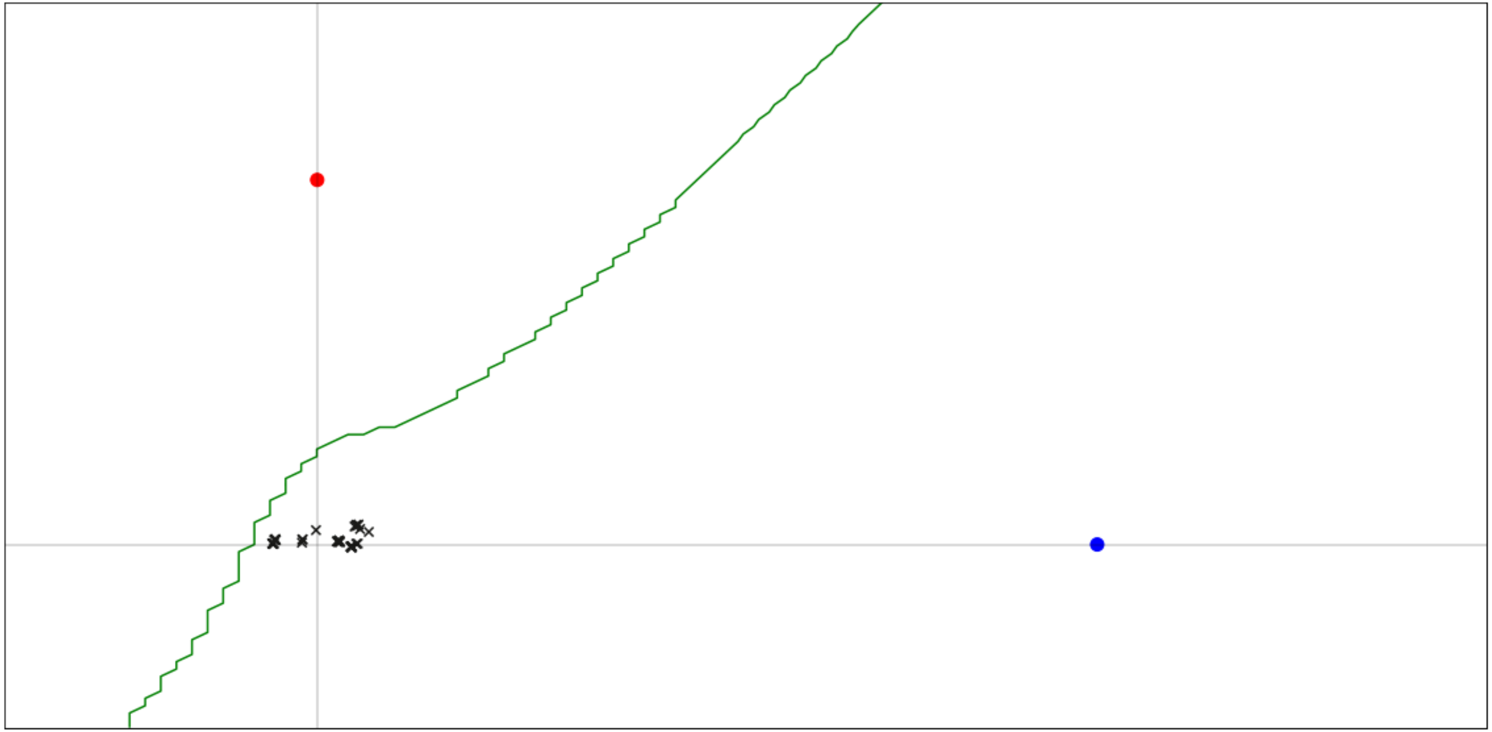
Theorem 4.1 (SGDA suffers from mode collapse). *Let T_0, η_G, η_D and the initialization as defined in Parametrization 4.1. Let t be such that $t \leq T_0$. Run SGDA for t iterations with step-sizes η_G, η_D. Then, with probability at least $1 - o(1)$, for all $z \in \{0,1\}^{m_G}$, we have:*

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

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. $\|\xi^{(T_0)}(z)\|_2 = o(\alpha^{(T_0)}(z))$.

Illustration: SGDA mode collapse (learn the average)



Our Results: normalized SGDA

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

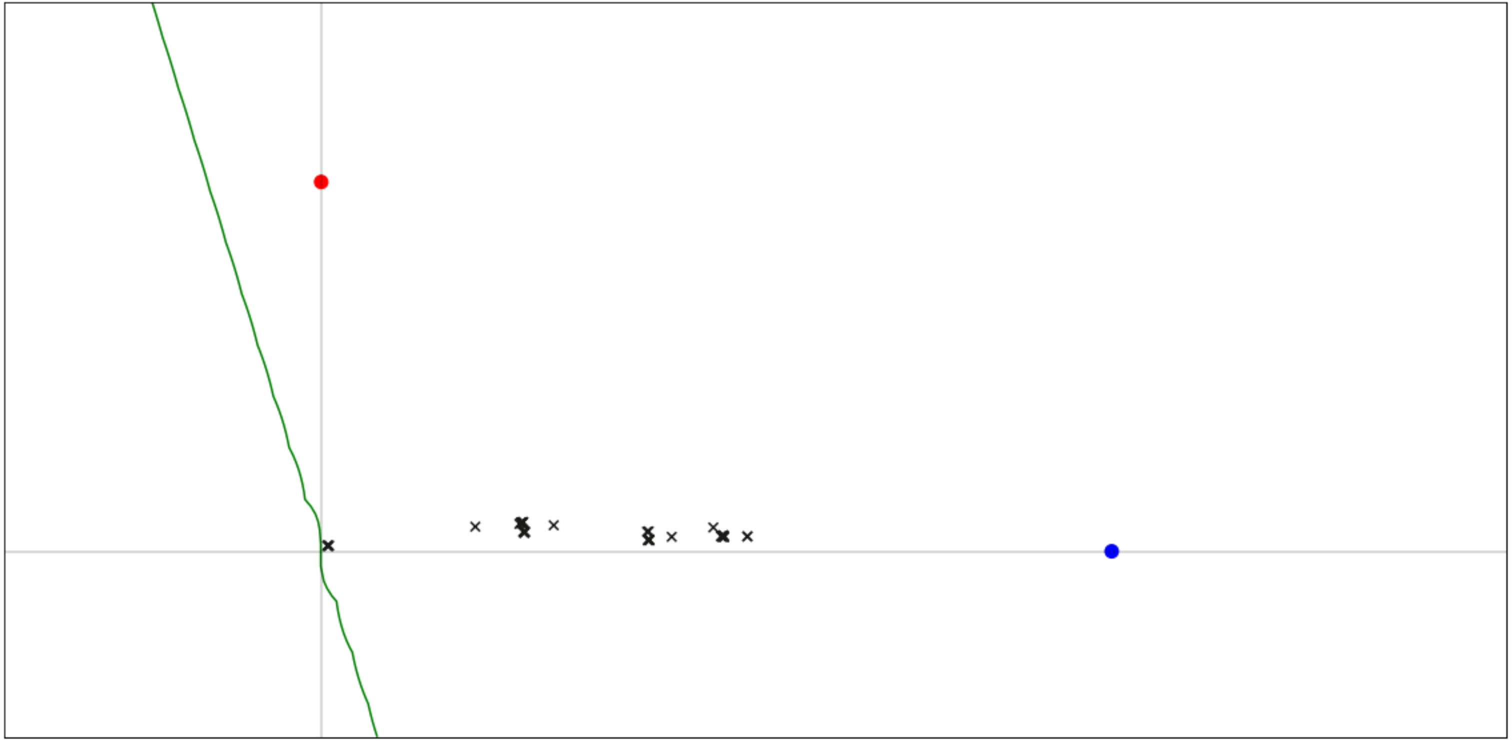
Theorem 4.2 (nSGDA recovers modes separately). *Let T_1, η_G, η_D and the initialization as defined in Parametrization 4.1. Run nSGDA for T_1 iterations with step-sizes η_G, η_D. Then, the generator learns both modes u_1, u_2 i.e.,*

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

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 $X = s_1u_1 + s_2u_2$.

Illustration normalized SGD learns both modes:



Conclusion (Part 1)

- GANs has some very bad (approximate) stationary points/local equilibrium.
- **Normalized gradients** seems to be key for training GANs.
- Converge guarantees are not enough!
- The **implicit biases** of each learning algorithms is significant!
- Need **fined-grained understanding of the dynamics** (depends on the game.)
- General nonconvex-nonconcave minimax optimization too general.

<u>Next Question:</u>

1. Importance of the minibatch-size (not too large mini batches seems critical)
2. Can we extend this to more complex generators and discriminators?
3. Does this idea extend to other setting? Multi-Agent RL?

Outline

- Part 1: Beyond minimax optimization by considering implicit biaises.

  The implicit biais of Adam for GANs training.

- Part 2: Beyond minimax optimization by replacing optimization by sampling.

  A distributional robustness perspective for adversarial training

A distributional robustness perspective for adversarial training

Joint work with David Dobre, Chiara Regniez, and Hugo Berard

Adversarial examples

“Machine Learning can make pigs fly” [Mądry and Kolter, 2018]

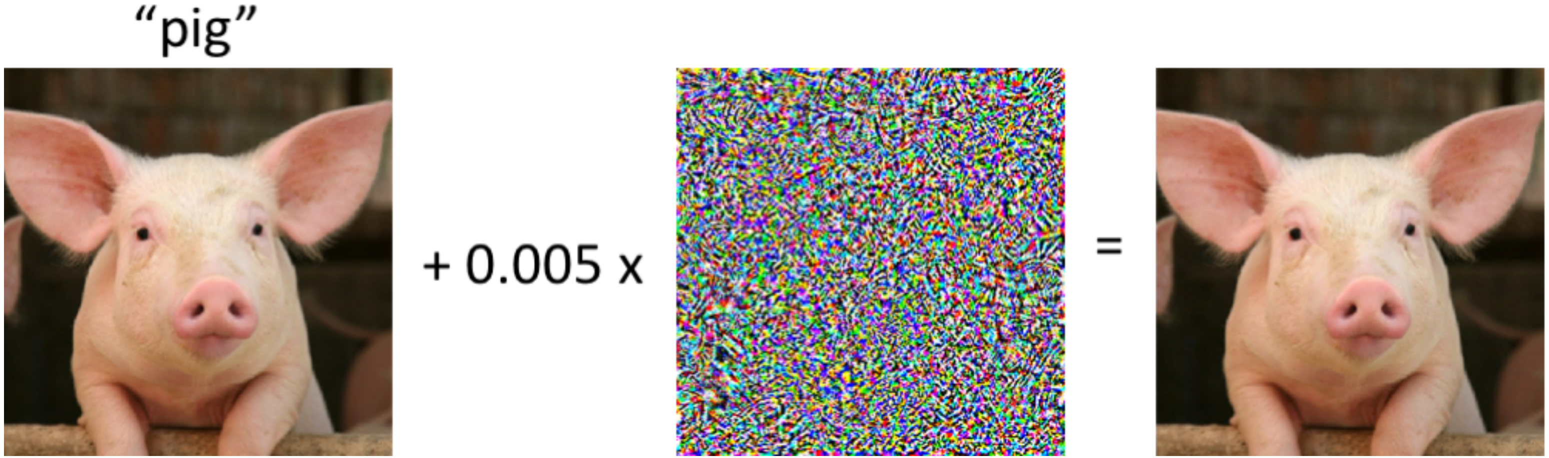


Image Source: https://gradientscience.org/intro_adversarial/

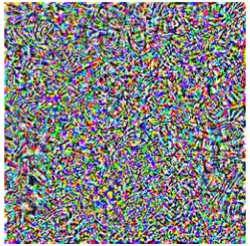
Training robust models

<u>Goal:</u> Robust classifier
i.e. **not** having flying pigs.

<u>Idea:</u> Train your model against adversarial examples

Not from p_{data} anymore !

$$\min_{\theta\in\mathbb{R}^d} \mathbb{E}_{(x,y)\sim p_{data}}\Big[\max_{\|\tilde{x}-x\|_\infty\le\epsilon} \ell(f_\theta(\tilde{x},y))\Big]$$


+ 0.005 x

=


Distributional robustness perspective

$$\tilde{x} \sim p_{adv}$$

$$\min_{\theta \in \mathbb{R}^d} \mathbb{E}_{(x,y)\sim p_{data}} \Big[\max_{\|\tilde{x}-x\|_\infty \leq \epsilon} \ell(f_\theta(\tilde{x}, y)) \Big] = \min_{\theta \in \mathbb{R}^d} \max_{p_{adv} \in \boxed{\mathcal{P}}} \mathbb{E}_{\tilde{x}\sim p_{adv}} [\ell(f_\theta(\tilde{x}, y))]$$

???

Distributional robustness perspective:

- [Delage and Ye, 2010] [Ben-Tal and Nemirovski, 1998]

Connection with Adversarial examples:

- [Gao et al., 2017] [Sinha et al., 2018]

Novelty here:

- Distributional robustness formulation of adversarial training (i.e. the right $\mathscr{P}$)
- Connection with optimal transport and $\infty - \infty -$ Wasserstein Ball constraint.

Distributional robustness perspective

$$\min_{\theta\in\mathbb{R}^d} \mathbb{E}_{(x,y)\sim p_{data}}\Big[\max_{\boxed{\|\tilde{x}-x\|_\infty\leq\epsilon}} \ell(f_\theta(\tilde{x},y))\Big] = \min_{\theta\in\mathbb{R}^d}\max_{p_{adv}\in\boxed{\mathcal{P}}} \mathbb{E}_{\tilde{x}\sim p_{adv}}[\ell(f_\theta(\tilde{x},y))]$$

Definition 3.1 (Transport plan). *Let p_{adv} and p_{data} be two distributions on $\mathcal{X}$. $\pi \in \mathcal{B}(\mathcal{X})\otimes\mathcal{B}(\mathcal{X}) \to [0,1]$ is a transport plan between p_{adv} and p_{data} if:*

$$\forall A \in \mathcal{B}(\mathcal{X}),\ \pi(A\times\mathcal{X}) = p_{data}(A) \quad \textit{and} \quad \forall B \in \mathcal{B}(\mathcal{X}),\ \pi(\mathcal{X}\times B) = p_{adv}(B) \tag{7}$$

The space of transport plan between p_{data} and p_{adv} is denoted $\Pi(p_{data}, p_{adv})$.

We want to transport x to $\tilde{x}$ with the proximity constraint:

$$\mathcal{P} = \{p_{adv} \,:\, \exists\pi \in \Pi(p_{data}|p_{adv}) \ \textit{with} \ \operatorname{supp}(\pi) \subset \{(x,y,\tilde{x},y) \mid \|\tilde{x}-x\|_\infty \leq \epsilon,\ y\in\mathcal{Y}\}\}$$

Distributional robustness perspective

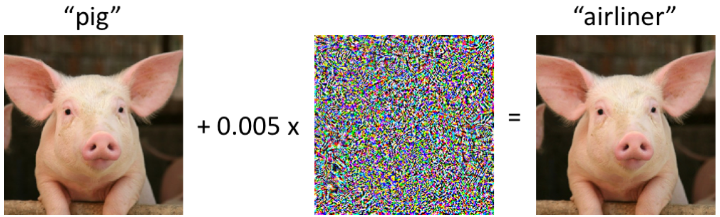
$$\min_{\theta\in\mathbb{R}^d} \mathbb{E}_{(x,y)\sim p_{data}}\Big[\max_{\|\tilde{x}-x\|_\infty\le\epsilon} \ell(f_\theta(\tilde{x},y))\Big] = \min_{\theta\in\mathbb{R}^d} \max_{p_{adv}\in\mathcal{P}} \mathbb{E}_{\tilde{x}\sim p_{adv}}[\ell(f_\theta(\tilde{x},y))]$$

$$\mathcal{P} = \{p_{adv} \,:\, \exists\pi \in \Pi(p_{data}|p_{adv}) \;\; \textit{with} \;\; \mathrm{supp}(\pi) \subset \{(x,y,\tilde{x},y) \mid \|\tilde{x}-x\|_\infty \le \epsilon,\, y\in\mathcal{Y}\}\}$$

<u>p-∞-Wasserstein distance:</u> $W_{p,\infty}(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \sup_{(z,\tilde{z})\in\mathrm{supp}(\pi)} \|z-\tilde{z}\|_p$

$$\min_{\theta\in\mathbb{R}^d} \mathbb{E}_{(x,y)\sim p_{data}}\Big[\max_{\|\tilde{x}-x\|_\infty\le\epsilon} \ell(f_\theta(\tilde{x},y))\Big] = \min_{\theta\in\mathbb{R}^d} \max_{p_{adv}\in\underbrace{\mathcal{B}_\epsilon(p_{data})}_{\boxed{\infty-\infty-\textbf{Wasserstein Ball}}}} \mathbb{E}_{(\tilde{x},y)\sim p_{adv}}[\ell(f_\theta(\tilde{x},y))]$$

Distributional robustness perspective



$$\min_{\theta\in\mathbb{R}^d} \mathbb{E}_{(x,y)\sim p_{data}}\Big[\max_{\|\tilde{x}-x\|_\infty\le\epsilon} \ell(f_\theta(\tilde{x},y))\Big] = \min_{\theta\in\mathbb{R}^d} \max_{p_{adv}\in\underbrace{\mathcal{B}_\epsilon(p_{data})}_{\boxed{\infty-\infty-\textbf{Wasserstein Ball}}}} \mathbb{E}_{(\tilde{x},y)\sim p_{adv}}[\ell(f_\theta(\tilde{x},y))]$$

<u>Distributional robustness perspective:</u>

- [Delage and Ye, 2010] [Ben-Tal and Nemirovski, 1998]

<u>Connection with Adversarial examples:</u>

- [Gao et al., 2017] [Sinha et al., 2018]

<u>Novelty here:</u>

- Distributional robustness formulation of adversarial training (i.e. equality above)
- Connection with the $\infty-\infty-$ Wasserstein Ball constraint.
- Connection with optimal transport.

Insight from optimal transport

- We need to (entropy) **regularize** the transport plan between p_{data} and p_{adv}.
- By using duality we have a **closed form** for the optimal p^*_{adv} (depends on θ):

$$p^*_{adv}(\tilde{x}, y) = \mathbb{E}_{x\sim p_{x,data}}\left[p_{data}(y|x) \underbrace{\frac{\exp(\lambda\ell(f_\theta(\tilde{x}),y))\mathbf{1}_{\|x-\tilde{x}\|_\infty\leq\epsilon}}{\int_{\tilde{x}\in\mathcal{B}_\epsilon(x)}\exp(\lambda\ell(f_\theta(\tilde{x}),y))}}_{\boxed{\text{Not easy to compute this integral}}}\right]$$

- Can **simulate** p_{adv} using Langevin Monte Carlo sampling.
- Inner problem of adversarial training: ~~optimization~~ $\rightarrow$ sampling

Practical implementation

Key insight: the adversarial dataset is a continuous function of $\boldsymbol{\theta}$ (because of the regularization).
Only need one inner step of Langevin

Algorithm 1 Adversarial Transport with Langevin

---

input: dataset $(x_i, y_i)_{i=1}^n$, step-size: η, number of adversarial examples: K, noise variance σ.
Initialization: $\tilde{x}_{i,k} = x_i$, $i \in [n]$, $k \in [K]$
for n : nb of steps **do**
    Sample a minibatch: $(x_i, y_i)_{i \in B}$, $B \subset [n]$
    Load the attacks: $\tilde{x}_{i,k}^{(0)} \leftarrow \tilde{x}_{i,k}$, $i \in B, k \in [K]$
    for T : nb langevin iter **do**
        for $i \in B, k \in [K]$ **do**
            $\tilde{x}_{i,k}^{(t+1)} \leftarrow \tilde{x}_{i,k}^{(t)} + \eta\, \mathrm{sign}(\nabla_x \ell(f_\theta(\tilde{x}_{i,k}^{(t+1)}, y_i))$
            $\tilde{x}_{i,k}^{(t+1)} \leftarrow P_{\mathcal{B}_i(\epsilon)}[\tilde{x}_{i,k}^{(t+1)} + \sigma\xi]$
            $\tilde{x}_{i,k}^{(t+1)} \leftarrow P_{\mathcal{X}}[\tilde{x}_{i,k}^{(t+1)}]$
        $\nabla_\theta g = \frac{1}{|B|K} \sum_{i \in B, k \in [K]} \nabla_\theta \ell(f_\theta(\tilde{x}_{i,k}^{(t+1)}, y_i))$
        $\theta \leftarrow \theta - \eta \nabla_\theta g$
    Save the attacks: $\tilde{x}_{i,k} \leftarrow \tilde{x}_{i,k}^{(T)}$, $i \in B, k \in [K]$
Return: θ

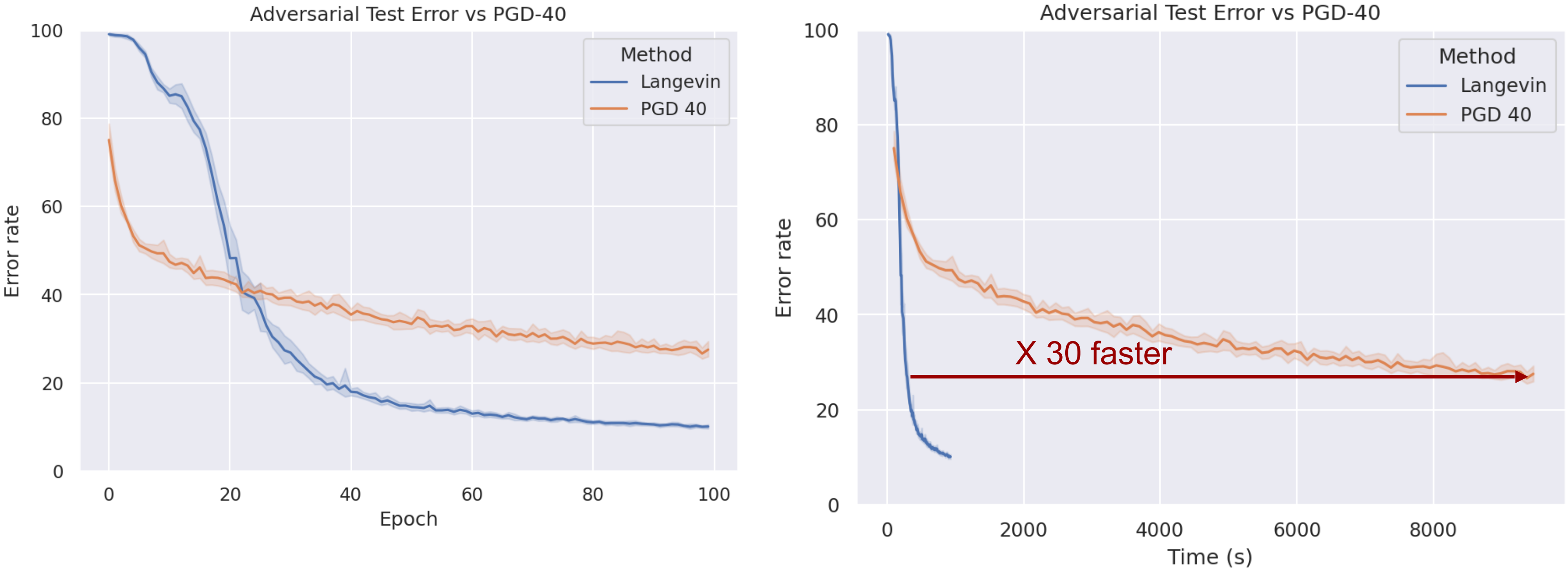
---

maintain an adversarial dataset!

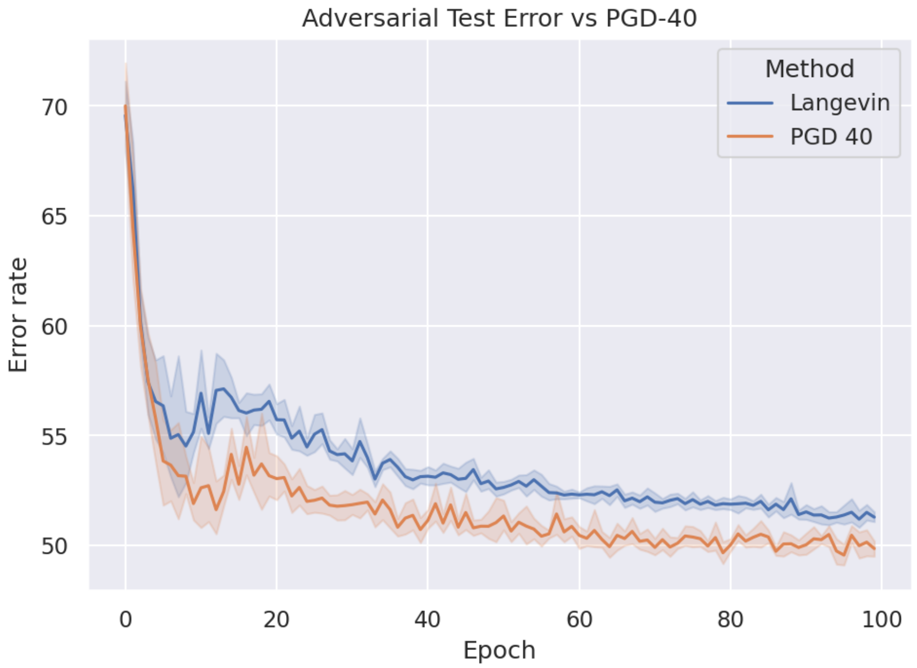
Update the adversarial dataset (depends on $\boldsymbol{\theta}$) via Langevin

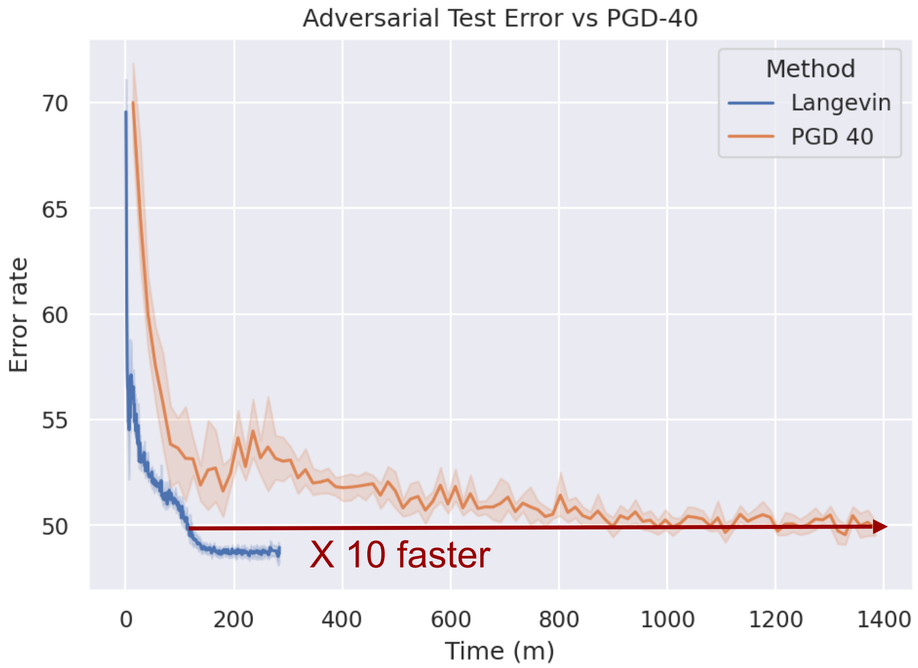
Update the model (to be more robust)

Results: MNIST



Results: CIFAR





Conclusion (part 2)

- Distributional robustness perspective on Adversarial Training
- Closed form for the (regularized) optimal adversarial distribution.
- Replace inner optimization by sampling
- By maintaining an adversarial dataset $\rightarrow$ warm start the sampling.
- No inner problem anymore (can alternate step of Langevin and step of SGD)
- Huge speed-up
- Caveat 1: need to keep an adversarial dataset in memory.
- Caveat 2: hard to deal with dataset augmentation.

Thanks, Questions?