GAN Tutorial
CSC413/2516 Winter 2020

Jonathan Lorraine
March 24th, 2020

Associated Colaboratory Notebook

https://colab.research.google.com/drive/1F9qO1bDekuHqbq-s6eI-MuyxtSQazvAI

This implements a simple GAN on MNIST with Jax (or Autograd).

Overview

1. GAN Case Study
2. GAN Training Difficulties

GAN Case Study - BigGAN

Brock, Andrew, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis." *International Conference on Learning Representations*. 2018.

(a) 128×128 (b) 256×256 (c) 512×512 (d)

Figure 4: Samples from our BigGAN model with truncation threshold 0.5 (a-c) and an example of class leakage in a partially trained model (d).

BigGAN Tricks

1. Self-Attention module (see [14]) for global structure
2. Hinge loss
3. Class-conditional information
4. Spectral Normalization of weight matrices
5. Update Discriminator more than Generator
6. Model weight averaging

See https://machinelearningmastery.com/a-gentle-introduction-to-the-biggan/ for a nice summary of each point

BigGAN Tricks

6. Orthogonal initialization and regularization of weight matrices

7. Large batch size

8. More capacity

9. Skip-connections

10. Truncation trick - use a truncated Gaussian for latent space during inference.

11. Non-standard ADAM settings - set momentum low

See https://machinelearningmastery.com/a-gentle-introduction-to-the-biggan/ for a nice summary of each point

GAN Training

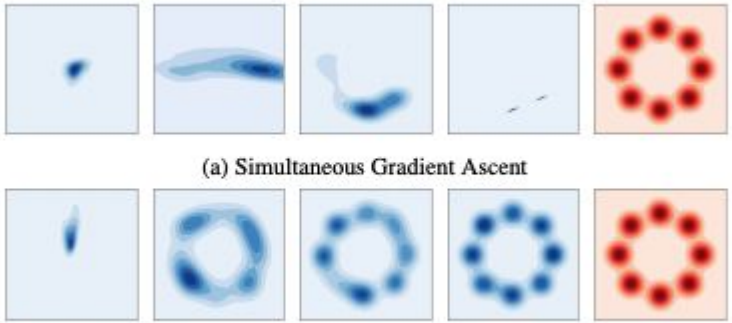
At least 3 main training problems with GANs

(1) How do we assess how well the GAN did?
(2) Are the gradients for the Generator (and Discriminator) sensible? I.e., not infinite / 0
(3) Does our optimization algorithm converge with correct gradients?

Assessing GAN performance

Having people look at samples doesn't always work

From [4]



(a) Simultaneous Gradient Ascent

(b) Consensus optimization

From [1]

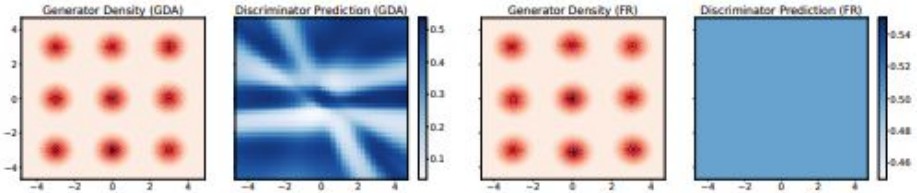


Figure 5: Comparison between FR and GDA on 2D mixture of Gaussians. **Left:** GDA; **Right:** FR.

If we have a downstream task, could use the performance on that

Assessing GAN performance

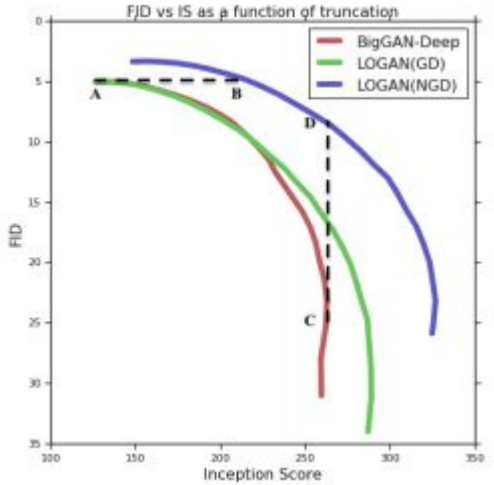
Inception Score: Estimates performance by how well Inception v3 classifies it as a known object. “Human judgement of quality”

Frechet Inception Distance: Calculate statistics on real and generated images with inception v3. Then estimate the difference between these distributions. Proposed by [3]

Good Reading on these scores:
https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/

$$\mathrm{IS}(G) = \exp\left(\mathbf{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right)$$

$$\mathrm{FID} = ||M_t - M_g||_2^2 + Tr(C_t + C_g - 2(C_t C_g)^{\frac{1}{2}})$$



From [2], Showing truncation curves. Illustrating quality/diversity tradeoff

Sensible GAN Gradients

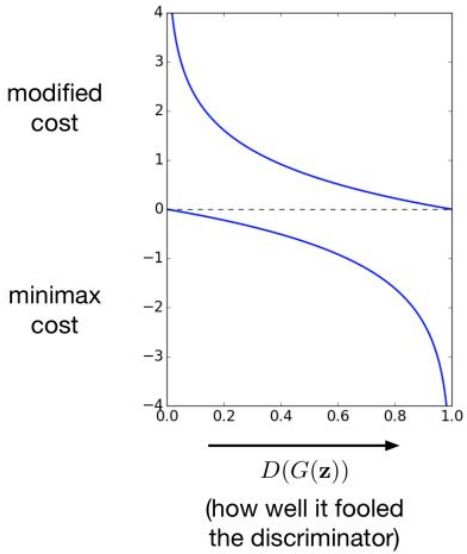
- Original minimax cost:

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

- Modified generator cost:

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

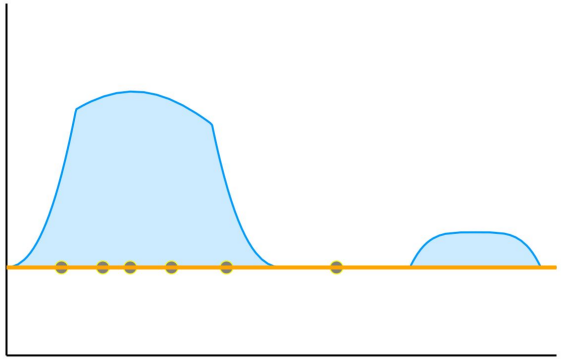
- This fixes the saturation problem.



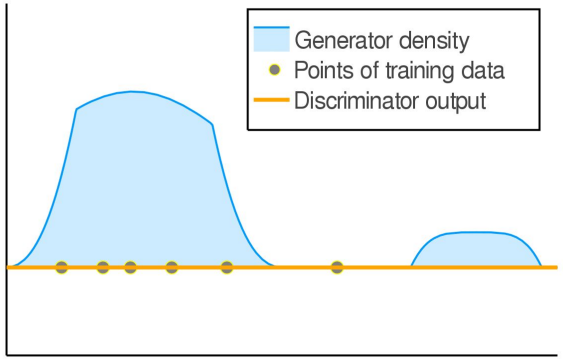
Sensible GAN Gradients

From https://f-t-s.github.io/projects/icr/ with associated paper [14]

Left - what we want.



Right - what happens



Sensible GAN Gradients

Problem: Not-saturating loss doesn't minimize a divergence.

$$E_{x\sim p_g}\left[-\log\left(D_G^*\left(x\right)\right)\right]$$
$$= KL(p_g \| p_{data}) - 2JS(p_{data} \| p_g) +$$
$$E_{x\sim p_{data}}\left[\log D_G^*\left(x\right)\right] + 2\log 2.$$

[13] provides a derivation of this, and the picture

Sensible GAN Gradients

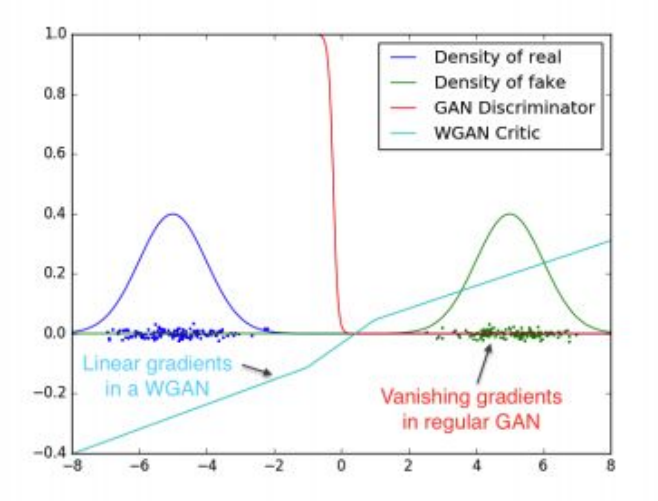


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

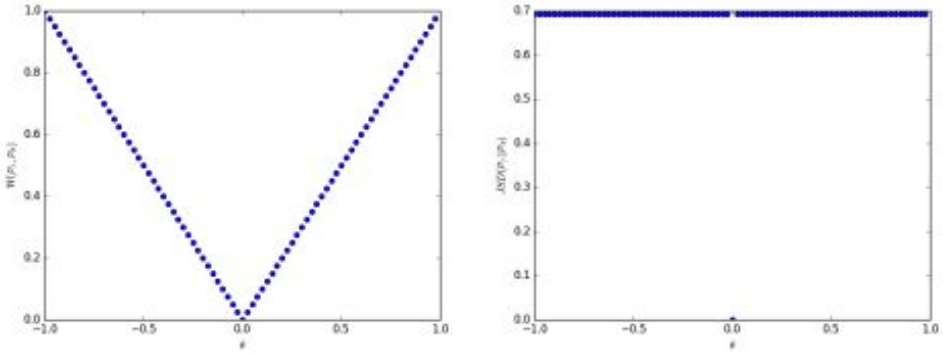


Figure 1: These plots show $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$ as a function of θ when ρ is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.

Images from [5]

WGAN

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

Theorem 3. *Let $\mathbb{P}_r$ be any distribution. Let $\mathbb{P}_\theta$ be the distribution of $g_\theta(Z)$ with Z a random variable with density p and g_θ a function satisfying assumption 1. Then, there is a solution $f : \mathcal{X} \to \mathbb{R}$ to the problem*

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

and we have

$$\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\mathbb{E}_{z \sim p(z)}[\nabla_\theta f(g_\theta(z))]$$

$$L_D^{WGAN} = E[D(x)] - E[D(G(z))]$$

$$L_G^{WGAN} = E[D(G(z))]$$

$$W_D \leftarrow clip\_by\_value(W_D, -0.01, 0.01)$$

Other GAN Losses

Check out https://github.com/znxlwm/pytorch-generative-model-collections for some different GAN losses in Pytorch.

There have been many different proposed solutions to this since WGAN came oout

Sensible GAN Optimization

Eq. from [6] and [7]

Solve for D*

$$\begin{aligned}\theta_G^* &= \underset{\theta_G}{\operatorname{argmin}} \max_{\theta_D} f\left(\theta_G, \theta_D\right) \\ &= \underset{\theta_G}{\operatorname{argmin}} f\left(\theta_G, \theta_D^*\left(\theta_G\right)\right) \\ \theta_D^*\left(\theta_G\right) &= \underset{\theta_D}{\operatorname{argmax}} f\left(\theta_G, \theta_D\right),\end{aligned}$$

$$f\left(\theta_G, \theta_D\right) = \mathbb{E}_{x \sim p_{data}}\left[\log\left(D\left(x; \theta_D\right)\right)\right] + \mathbb{E}_{z \sim \mathcal{N}(0,I)}\left[\log\left(1 - D\left(G\left(z; \theta_G\right); \theta_D\right)\right)\right]$$

$$D^*\left(x\right) = \frac{p_{data}\left(x\right)}{p_{data}\left(x\right) + p_G\left(x\right)}.$$

Sensible GAN Optimization

Substitute D* into the loss for G and get a divergence

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}.$$

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \quad = \quad C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)$$

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

Sensible GAN Optimization

Problem! Optimal D* is a function of G. The response function D*(G) is difficult to differentiate (or even evaluate).

$$\begin{aligned}\theta_G^* &= \underset{\theta_G}{\operatorname{argmin}} \max_{\theta_D} f\left(\theta_G, \theta_D\right) \\ &= \underset{\theta_G}{\operatorname{argmin}} f\left(\theta_G, \theta_D^*\left(\theta_G\right)\right) \\ \theta_D^*\left(\theta_G\right) &= \underset{\theta_D}{\operatorname{argmax}} f\left(\theta_G, \theta_D\right),\end{aligned}$$

Sensible GAN Optimization

Idea 1: Approximate D*(G) in a way with an easy derivative

Idea 2: Use a smarter optimization algorithm that can work with only first (or second) order information about the losses.

Sensible GAN Optimization

Idea 1: Approximate D*(G) in a way with an easy derivative

Unrolled differentiation as in [7]

Only needs second order information

$$\theta_D^0 = \theta_D$$

$$\theta_D^{k+1} = \theta_D^k + \eta^k \frac{\mathrm{d}f\left(\theta_G, \theta_D^k\right)}{\mathrm{d}\theta_D^k}$$

$$\theta_D^*\left(\theta_G\right) = \lim_{k \to \infty} \theta_D^k,$$

$$\frac{\mathrm{d}f_K\left(\theta_G, \theta_D\right)}{\mathrm{d}\theta_G} = \frac{\partial f\left(\theta_G, \theta_D^K\left(\theta_G, \theta_D\right)\right)}{\partial \theta_G} + \frac{\partial f\left(\theta_G, \theta_D^K\left(\theta_G, \theta_D\right)\right)}{\partial \theta_D^K\left(\theta_G, \theta_D\right)} \frac{\mathrm{d}\theta_D^K\left(\theta_G, \theta_D\right)}{\mathrm{d}\theta_G}.$$

Sensible GAN Optimization

Idea 2: Design a smarter optimization alg - remember, our goal is to find fixed points in a vector field, subject to curvature constraints

2.3 VARIATIONAL INEQUALITY PROBLEM FORMULATION

We first consider the local necessary conditions that characterize the solution of the *smooth* two-player game (3), defining *stationary points*, which will motivate the definition of a variational inequality. In the unconstrained setting, a *stationary point* is a couple $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ with zero gradient:

$$\|\nabla_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)\| = \|\nabla_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)\| = 0\,. \tag{4}$$

When constraints are present,[3] a *stationary point* $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ is such that the directional derivative of each cost function is non-negative in any feasible direction (i.e. there is no feasible descent direction):

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^*) \geq 0 \quad \text{and} \quad \nabla_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)^\top (\boldsymbol{\varphi} - \boldsymbol{\varphi}^*) \geq 0\,, \quad \forall (\boldsymbol{\theta}, \boldsymbol{\varphi}) \in \Theta \times \Phi. \tag{5}$$

Defining $\boldsymbol{\omega} \stackrel{\text{def}}{=} (\boldsymbol{\theta}, \boldsymbol{\varphi})$, $\boldsymbol{\omega}^* \stackrel{\text{def}}{=} (\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$, $\Omega \stackrel{\text{def}}{=} \Theta \times \Phi$, Eq. (5) can be compactly formulated as:

$$F(\boldsymbol{\omega}^*)^\top (\boldsymbol{\omega} - \boldsymbol{\omega}^*) \geq 0\,, \quad \forall \boldsymbol{\omega} \in \Omega \quad \text{where} \quad F(\boldsymbol{\omega}) \stackrel{\text{def}}{=} \left[\nabla_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\theta}, \boldsymbol{\varphi}) \quad \nabla_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\theta}, \boldsymbol{\varphi})\right]^\top . \tag{6}$$

From [8]

A More General Problem

A special case of learning in differentiable games - see [11] for more

Have n players, each with a loss L_i.

Let w = (w_1, …, w_n) where w_i is the parameters of the ith player.

The dynamics of simultaneous gradient descent given by v(w) = [grad_{w_1} L_1, …, grad_{w_n} L_n]

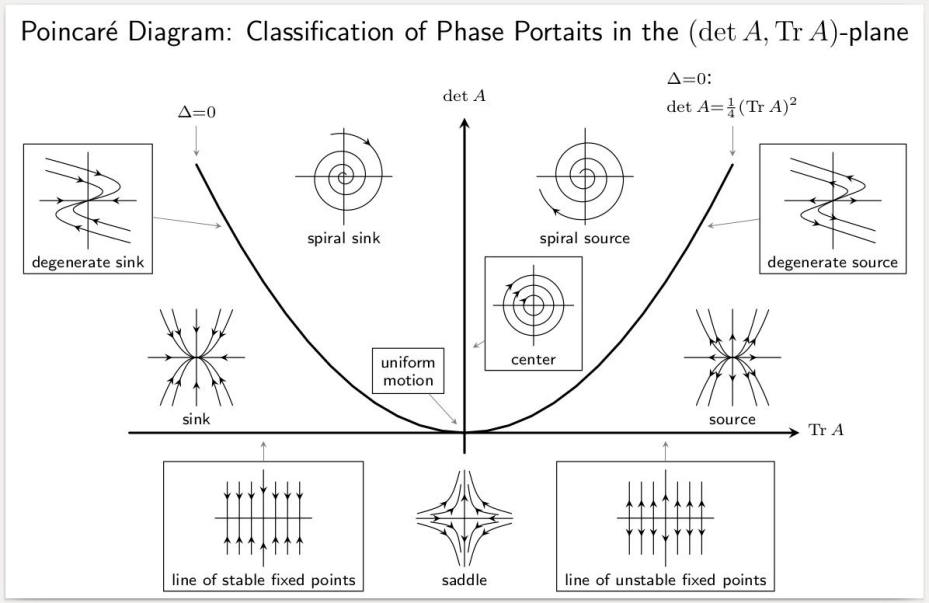
Jacobian of dynamics given by

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

Classification of fixed points for 2-D Dynamics

Eigenvalues of 2x2 matrix A satisfy:

$$\frac{\mathrm{tr}(A) \pm \sqrt{\mathrm{tr}(A)^2 - 4\det(A)}}{2}$$



Sensible GAN optimization

How about minimizing the joint-gradient norm? Proposed in [4].

Could do a weighted combination of this and original loss - called Consensus Optimization (CO)

$$v(\varphi, \theta) := \left[\nabla_\varphi \mathcal{L}_D(\varphi, \theta) \quad \nabla_\theta \mathcal{L}_G(\varphi, \theta)\right]^\top \qquad L(x) = \tfrac{1}{2}\|v(x)\|^2$$

Problem: New fixed points could violate curvature constraints

Sensible Optimization

Lets view optimization as a dynamical system.

The optimization algorithm gives our dynamics:

$$v(\phi, \theta) = \begin{pmatrix} \nabla_\phi f(\phi, \theta) \\ \nabla_\theta g(\phi, \theta) \end{pmatrix}$$

We can analyze the fixed points of our dynamics system by looking an eigenvalues of the Jacobian of dynamics:

$$v'(\phi, \theta) = \begin{pmatrix} \nabla^2_\phi f(\phi, \theta) & \nabla_{\phi,\theta} f(\phi, \theta) \\ -\nabla_{\phi,\theta} f(\phi, \theta) & -\nabla^2_\theta f(\phi, \theta) \end{pmatrix}$$

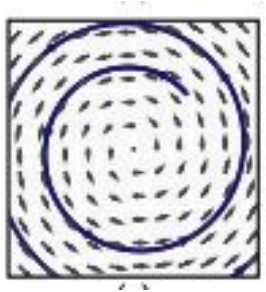
Sensible GAN Optimization

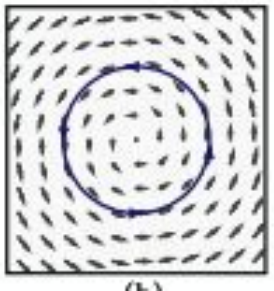
A toy example: Bilinear problems $\min_{\theta \in \mathbb{R}^d} \max_{\varphi \in \mathbb{R}^p} \theta^\top A \varphi + \theta^\top b + c^\top \varphi, \quad A \in \mathbb{R}^{d \times p}.$

An even more toy example: min_x max_y x*y

Simultaneous Gradient descent on this objectives gives dynamics v = [x, -y], and v' = [[0, 1], [-1, 0]]

The eigenvalues of v' are +- i!





Sensible GAN Optimization

A solution for Bilinear games [8]:

$$\omega \stackrel{\text{def}}{=} (\theta, \varphi)$$

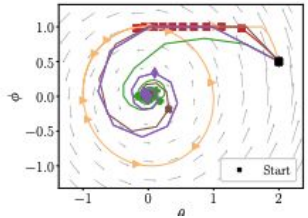


Figure 1: Comparison of the basic gradient method (as well as Adam) with the techniques presented in §3 on the optimization of (9). Only the algorithms advocated in this paper (Averaging, Extrapolation and Extrapolation from the past) converge quickly to the solution. Each marker represents 20 iterations. We compare these algorithms on a non-convex objective in §G.1.

Averaging:

$$\omega_T \stackrel{\text{def}}{=} \frac{\sum_{t=0}^{T-1} \rho_t \omega_t}{S_T}, \quad S_T \stackrel{\text{def}}{=} \sum_{t=0}^{T-1} \rho_t$$

Extrapolation:

$$\text{Compute extrapolated point: } \omega_{t+1/2} = P_\Omega[\omega_t - \eta F(\omega_t)],$$
$$\text{Perform update step: } \omega_{t+1} = P_\Omega[\omega_t - \eta F(\omega_{t+1/2})].$$

Sensible GAN Optimization

Order of updates?

$$\textbf{Simultaneous update:} \begin{cases} \theta_{t+1} = \theta_t - \eta\phi_t \\ \phi_{t+1} = \phi_t + \eta\theta_t \end{cases}, \quad \textbf{Alternating update:} \begin{cases} \theta_{t+1} = \theta_t - \eta\phi_t \\ \phi_{t+1} = \phi_t + \eta\theta_{t+1} \end{cases}.$$

Alternating updates are better! Use more information

Sensible GAN Optimization

Method	β	Bounded	Converges
Simult.	>0	✗	✗
Thm. 5	0	✗	✗
	<0	✗	✗
Altern.	>0	✗	✗
Thm. 6	0	✓	✗
	<0	✓	✓

Extrapolation generalizes gradient descent linear convergence results to bilinear games.

What about generalizing momentum? Well, in some cases we want negative momentum due to complex eigenvalues in the dynamics [9]. Acceleration like in single-objective case is still open question

$$F_{\eta,\beta}(\boldsymbol{\omega}_t, \boldsymbol{\omega}_{t-1}) = (\boldsymbol{\omega}_{t+1}, \boldsymbol{\omega}_t)$$
where $\boldsymbol{\omega}_{t+1} := \boldsymbol{\omega}_t - \eta \boldsymbol{v}(\boldsymbol{\omega}_t) + \beta(\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t-1})$,

Theorem 3. *The eigenvalues of* $\nabla F_{\eta,\beta}(\boldsymbol{\omega}^*)$ *are*

$$\mu_\pm(\beta, \eta, \lambda) := (1 - \eta\lambda + \beta)\frac{1 \pm \Delta^{\frac{1}{2}}}{2}, \qquad (14)$$

where $\Delta := 1 - \frac{4\beta}{(1-\eta\lambda+\beta)^2}$, $\lambda \in \mathrm{Sp}(\nabla \boldsymbol{v}(\boldsymbol{\omega}^*))$ *and* $\Delta^{\frac{1}{2}}$ *is the* complex *square root of* Δ *with positive real part**. *Moreover we have the following Taylor approximation,*

$$\rho_{\lambda,\eta}(\beta) := \max\left\{|\mu_+(\beta, \eta, \lambda)|^2, |\mu_-(\beta, \eta, \lambda)|^2\right\}$$

Sensible GAN Optimization

What about generalizing optimization algorithms that correct for curvature like Newton method?

Follow-the-ridge [1]:

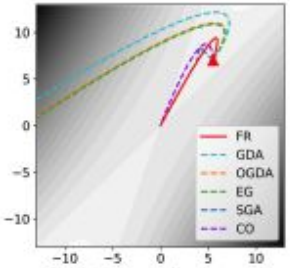
Algorithm 2 Follow-the-Ridge (FR) for general-sum Stackelberg games.

Require: Learning rate $\eta_{\mathbf{x}}$ and $\eta_{\mathbf{y}}$; number of iterations T.

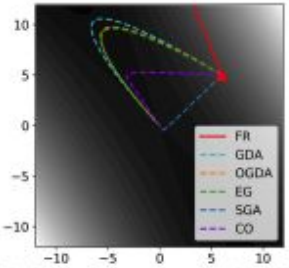
1: **for** $t = 1, ..., T$ **do**

2: $\quad \mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta_{\mathbf{x}} D_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t)$ $\quad \triangleright$ total derivative $D_{\mathbf{x}} f = \nabla_{\mathbf{x}} f - \nabla^2_{\mathbf{xy}} g (\nabla^2_{\mathbf{yy}} g)^{-1} \nabla_{\mathbf{y}} f$

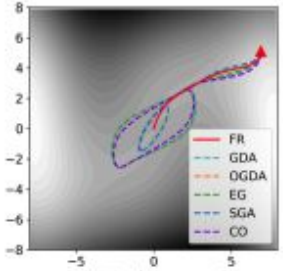
3: $\quad \mathbf{y}_{t+1} \leftarrow \mathbf{y}_t - \eta_{\mathbf{y}} \nabla_{\mathbf{y}} g(\mathbf{x}_t, \mathbf{y}_t) + \eta_{\mathbf{x}} (\nabla^2_{\mathbf{yy}} g)^{-1} \nabla^2_{\mathbf{yx}} g D_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t)$



(a) GDA diverges



(b) GDA converges to a bad fixed point that is non local minimax



(c) Limiting cycle

Sensible GAN Optimization

Another method:

Competitive Gradient Descent [10]

Algorithm 1: Competitive Gradient Descent (CGD)

for $0 \leq k \leq N-1$ **do**

$$x_{k+1} = x_k - \eta \left(\mathrm{Id} - \eta^2 D^2_{xy} f D^2_{yx} g\right)^{-1} \left(\nabla_x f - \eta D^2_{xy} f \nabla_y g\right);$$

$$y_{k+1} = y_k - \eta \left(\mathrm{Id} - \eta^2 D^2_{yx} g D^2_{xy} f\right)^{-1} \left(\nabla_y g - \eta D^2_{yx} g \nabla_x f\right);$$

return (x_N, y_N);

What I think that they think that I think ... that they do: Another game-theoretic interpretation of CGD follows from the observation that its update rule can be written as

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} \mathrm{Id} & \eta D^2_{xy} f \\ \eta D^2_{yx} g & \mathrm{Id} \end{pmatrix}^{-1} \begin{pmatrix} \nabla_x f \\ \nabla_y g \end{pmatrix}. \tag{4}$$

Some GAN Visualizations

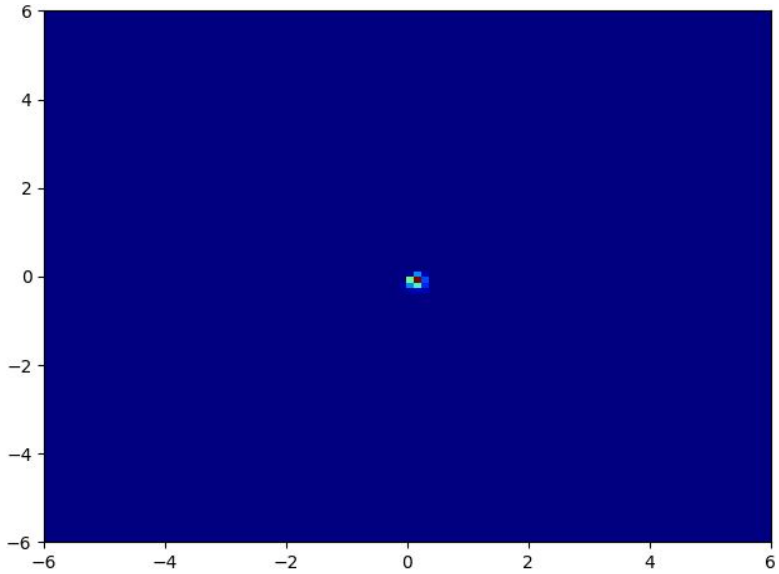
Some visualizations for training a simple GAN

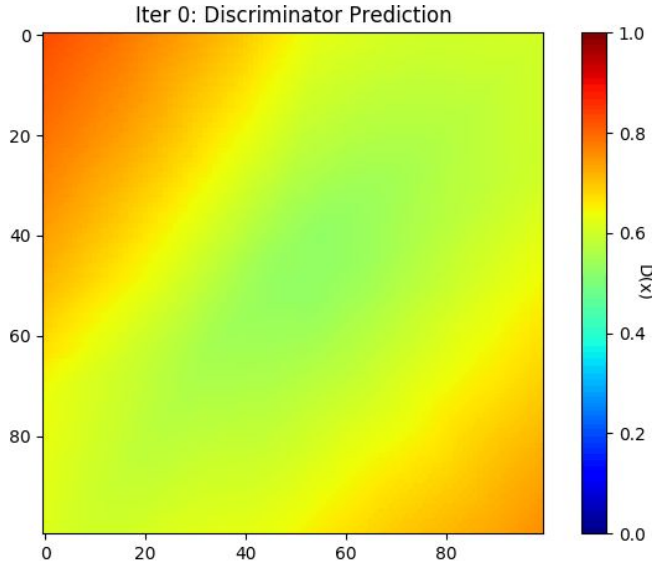
Generator: 1 Hidden layer with 64 units. 4 dimension noise

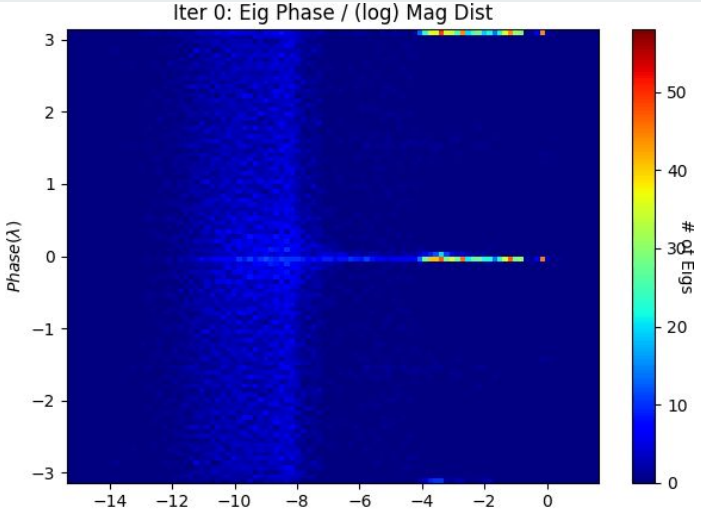
Discriminator: 1 Hidden layer with 64 hidden units

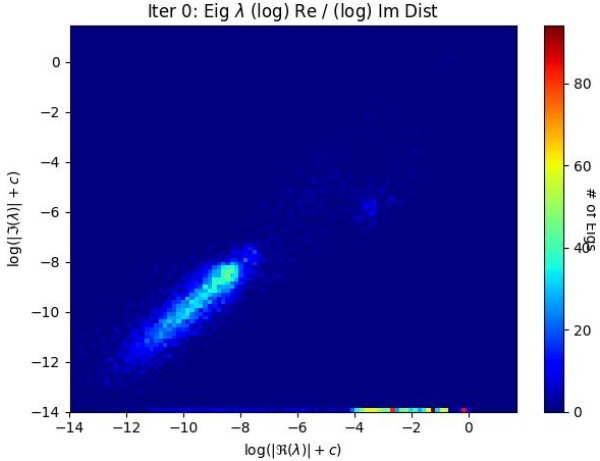
Optimizer: Adam with settings from BigGAN

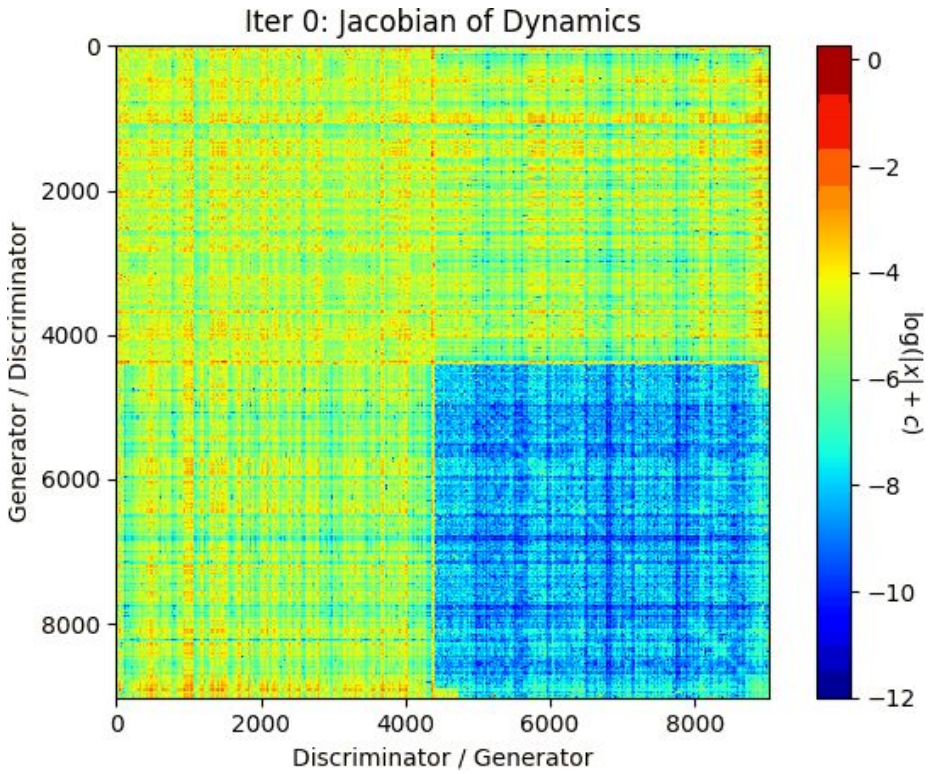
GAN Samples and Discriminator Prediction



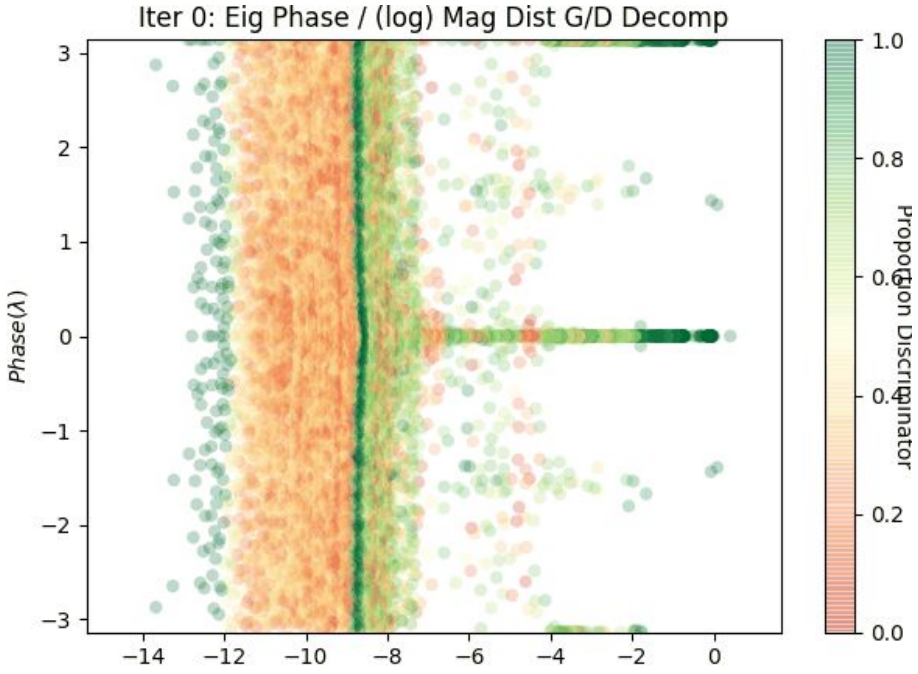


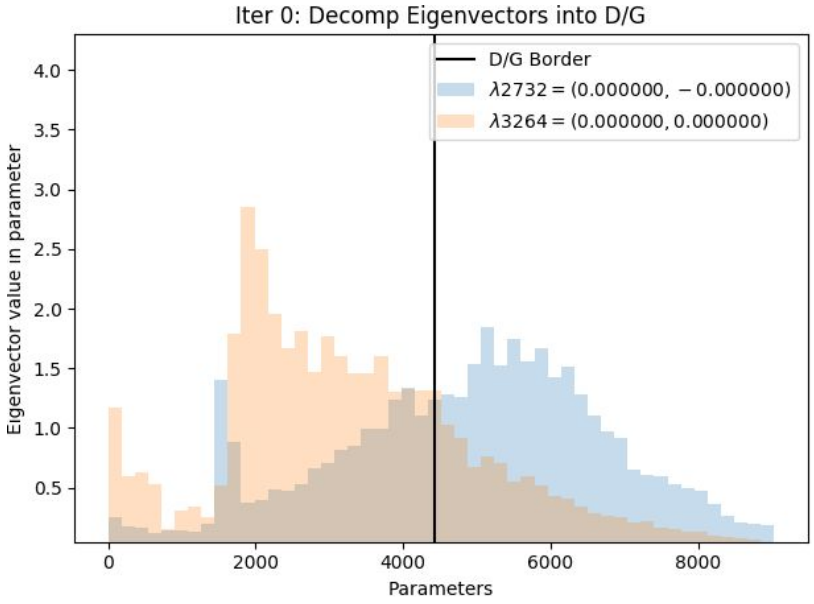

Iter 0: Eig Phase / (log) Mag Dist
Phase(λ)
# of Eigs


Iter 0: Eig λ (log) Re / (log) Im Dist
log(|ℑ(λ)| + c)
log(|ℜ(λ)| + c)
# of Eigs

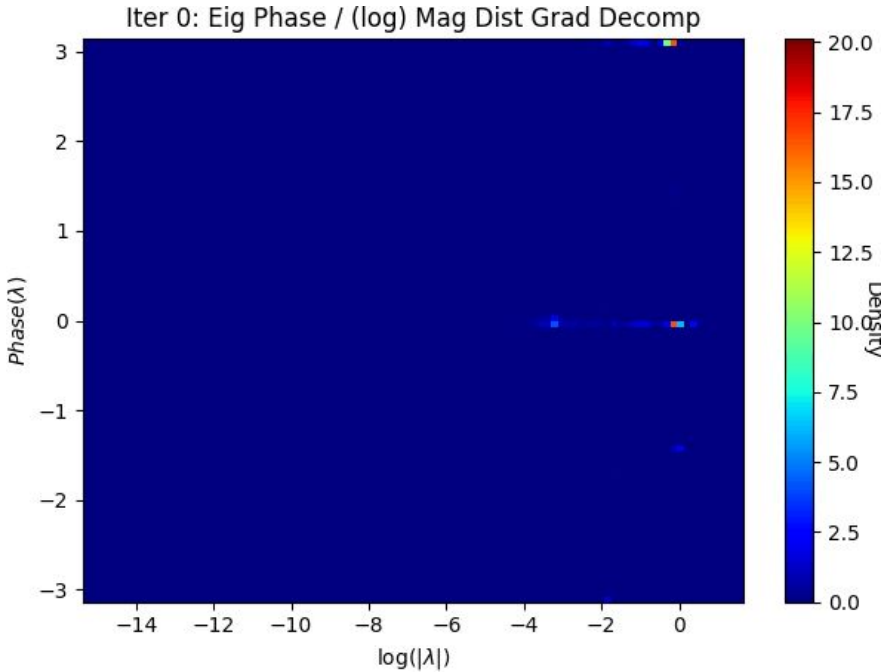

Iter 0: Jacobian of Dynamics
Generator / Discriminator
Discriminator / Generator
log(|x| + c)

Decomposing “who has each Eigenvalue”





Eigenspaces the Joint-Gradient Lies in



BigGAN Tricks Again

1. Self-Attention module and hinge loss
2. Class-conditional information
3. Spectral Normalization of weight matrices
4. **Update Discriminator more than Generator**
5. **Model weight averaging**
6. Orthogonal initialization and regularization of weight matrices
7. Large batch size
8. More capacity
9. Skip-connections
10. Truncation trick - use a truncated Gaussian for latent space during inference.
11. **Non-standard ADAM settings**

See https://machinelearningmastery.com/a-gentle-introduction-to-the-biggan/ for a nice summary of each point

References

[1] Wang, Yuanhao, Guodong Zhang, and Jimmy Ba. "On Solving Minimax Optimization Locally: A Follow-the-Ridge Approach." *arXiv preprint arXiv:1910.07512* (2019).

[2] Wu, Yan, et al. "LOGAN: Latent Optimisation for Generative Adversarial Networks." *arXiv preprint arXiv:1912.00953* (2019).

[3] Heusel, Martin, et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." *Advances in neural information processing systems*. 2017.

[4] Mescheder, Lars, Sebastian Nowozin, and Andreas Geiger. "The numerics of gans." *Advances in Neural Information Processing Systems*. 2017.

[5] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein gan." *arXiv preprint arXiv:1701.07875* (2017).

References

[6] Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

[7] Metz, Luke, et al. "Unrolled generative adversarial networks." *arXiv preprint arXiv:1611.02163* (2016).

[8] Gidel, Gauthier, et al. "A variational inequality perspective on generative adversarial networks." *arXiv preprint arXiv:1802.10551* (2018).

[9] Gidel, Gauthier, et al. "Negative momentum for improved game dynamics." *arXiv preprint arXiv:1807.04740* (2018).

[10] Schäfer, Florian, and Anima Anandkumar. "Competitive gradient descent." *Advances in Neural Information Processing Systems*. 2019.

[11] Letcher, Alistair, et al. "Differentiable Game Mechanics." *Journal of Machine Learning Research* 20.84 (2019): 1-40.

References

[12] Azizian, Waïss, et al. "A tight and unified analysis of extragradient for a whole spectrum of differentiable games." *arXiv preprint arXiv:1906.05945* (2019).

[13] Gui, Jie, et al. "A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications." *arXiv preprint arXiv:2001.06937* (2020).

[14] Zhang, Han, et al. "Self-attention generative adversarial networks." *arXiv preprint arXiv:1805.08318* (2018).

[15] Schäfer, Florian, Hongkai Zheng, and Anima Anandkumar. "Implicit competitive regularization in GANs." *arXiv preprint arXiv:1910.05852* (2019).