# Optimization, Training, and Stabilization

Generative Adversarial Networks, Part II

Slides based on lectures by Ben Striner

1. Optimization Issues
2. Training and Stabilization
3. Take Aways

# GANs

- Powerful tool for generative modeling
- Has lots of potential
- Limited by pragmatic issues

1. Optimization Issues
2. Training and Stabilization
3. Take Aways

# Understanding Optimization Issues

Failures can be hard to diagnose

- “Mode collapse”
- Unclear errors

- Why? Many potential reasons

# Understanding Optimization Issues

1. Simultaneous updates
    a. Why can’t we just train an optimal discriminator?

2. Not guaranteed to converge to a global optimum
    a. Even a game of Rock-Paper-Scissors may never converge

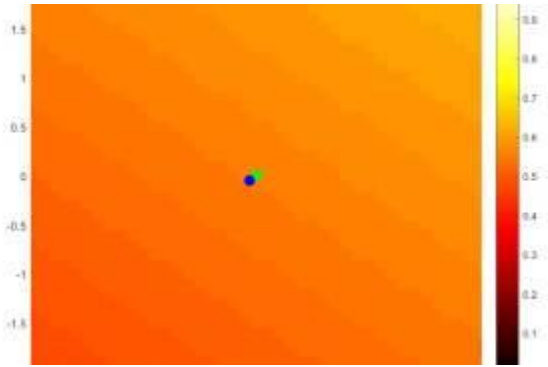
3. Not guaranteed to reach a stationary point
    a. Even if there is a stationary point, you might not converge toward it

# Simultaneous Updates

- Why can’t we just train an optimal discriminator?

- The optimal discriminator is conditional on the current generator, and you can’t train the generator without training the discriminator.

- Adversarial balance



# Factors Affecting Balance

- Optimizers and learning rates
- Architectures, number of parameters, depths
- Regularization
- Repeat training D for k_D iterations, then train G for k_G iterations
- Make the iterations of k_D and k_G dynamic, depending on metrics

# Not Guaranteed to Converge using Gradient Descent

Rock-Paper-Scissors example

- Player A randomly plays Rock
- Player B then should only play Paper
- Player A then should only play Scissors
- Player B then should only play Rock
- Player A then should only play Paper
- …


$$\mathbb{E}[W_A] = A_R B_S + A_P B_R + A_S B_P$$

# Not Guaranteed to Converge using Gradient Descent

Rock-Paper-Scissors example

- Global Optimum:
  - both players choose with probability (0.33, 0.33, 0.33)

- Local Optimum for (rock, paper, scissors):
  - If player A plays with probability (0.36, 0.32, 0.32)
  - Then player B should play with probability (0,1,0)..

# Not Guaranteed to reach a Stationary Point

- Gradients can circle or point away from the minima, so there might not be a local “well” around the stationary point.

- Even if a generated point is 0, it doesn’t mean that the points around that area area also 0; likewise, for 1.

1. Optimization Issues
2. Training and Stabilization
3. Take Aways

# GAN Training Techniques

## Foundations & Stability Fixes

- Gradient descent is locally stable
- Numerics of GANs
- Instance Noise
- Least-Squares GAN (LSGAN)
- DRAGAN
- Unrolled GAN
- Improved Techniques for GAN

## Divergence Replacements

- Wasserstein GAN (WGAN)
- WGAN Gradient Penalty
- Spectral Normalized GAN

# Gradient Descent GAN Optimization is Locally Stable

- The generator tries to minimize its original loss and also tries to minimize the squared norm of the discriminator's gradient.

- If the generator makes an improvement but the discriminator gradient is large, the discriminator could undo that improvement.

- If the generator improves in a way that nears the discriminator gradient towards zero, the discriminator could not undo the improvement.

$$\mathcal{L}_G = \mathcal{L}_{G,0} + \eta \|\nabla \mathcal{L}_D\|^2$$

# Numerics of GANs

- Considers the joint field of generator and discriminator parameters

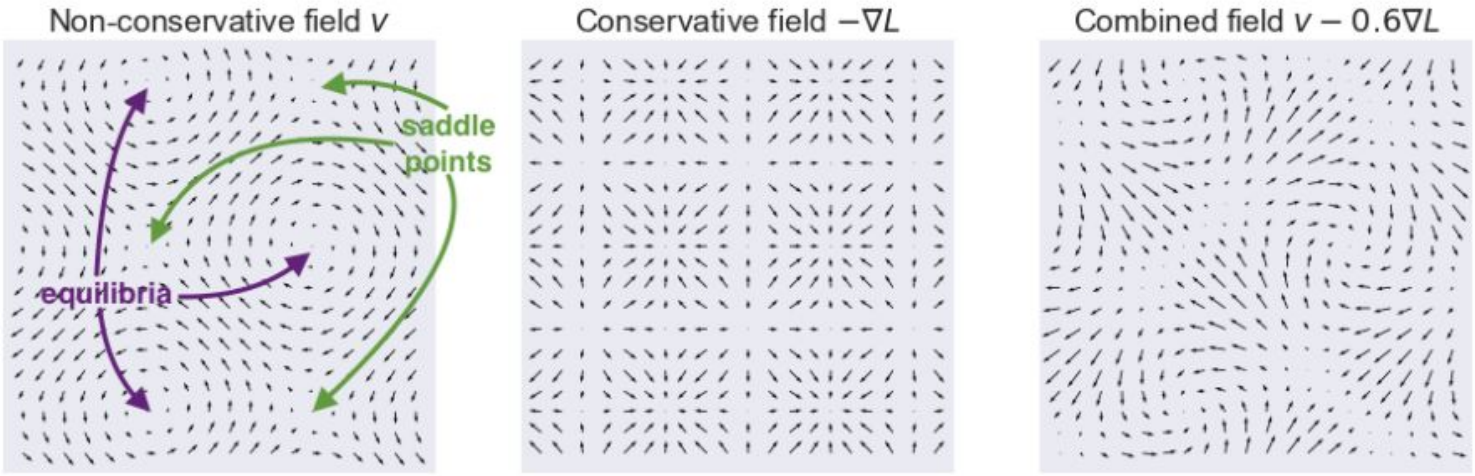
- Finds stationary points by minimizing the norm of the gradient of each player; both the generator and the discriminator try to minimize the other's gradients as well as their own.

- The regularization parameter balances between adversarial objective and consensus objective

$$L = L_0 + \lambda \|\nabla L_0\|_2^2$$

# Numerics of GANs

https://www.inference.vc/my-notes-on-the-numerics-of-gans/




- Since the 0 gradient can include minima, maxima, and saddlepoints, the consensus gives a small regularization parameter such that there is enough stability when it's near a minima so that your GAN goes down to it.

# Instance Noise

- Add noise to generated and real images

- Smooth the function learned by the discriminator

# Least-Squares GAN

- Instead of binary cross entropy, uses an L2 loss

- The generator tries to minimize L2 loss $\|a - D(G(Z))\|_2^2$

- The discriminator tries to minimize L2 loss

$$\|b - D(G(Z))\|_2^2 + \|c - D(G(Z))\|_2^2$$

- Simplifies the loss function over time

# DRAGAN

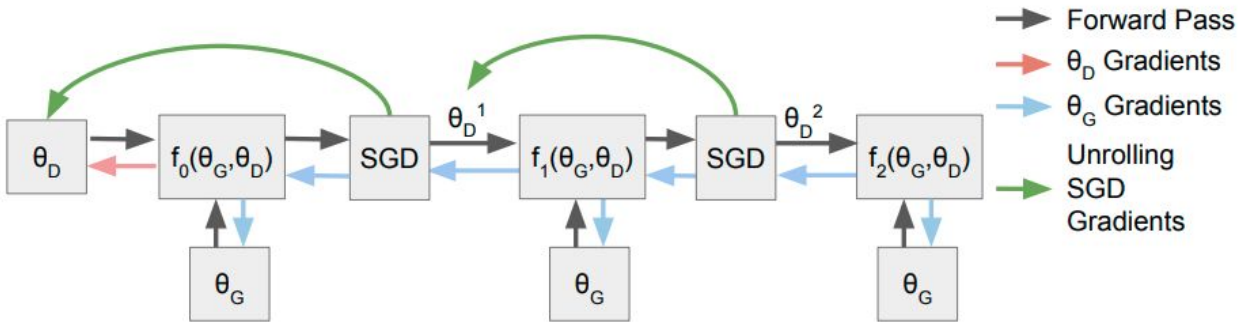
- Minimize the norm of the gradient in a region around real data to make a function smoother

- Smooth in a random region around real data to smooth the discriminator

$$\lambda \mathbb{E}_{X,\epsilon} \max\left(0, \|\nabla D(X+\epsilon)\|^2 - k\right)$$

# Unrolled GAN

- Calculate the discriminator after a few SGD steps
- Find the generator that has the best loss on the future discriminator
- Differentiate through gradient descent



# Unrolled GAN

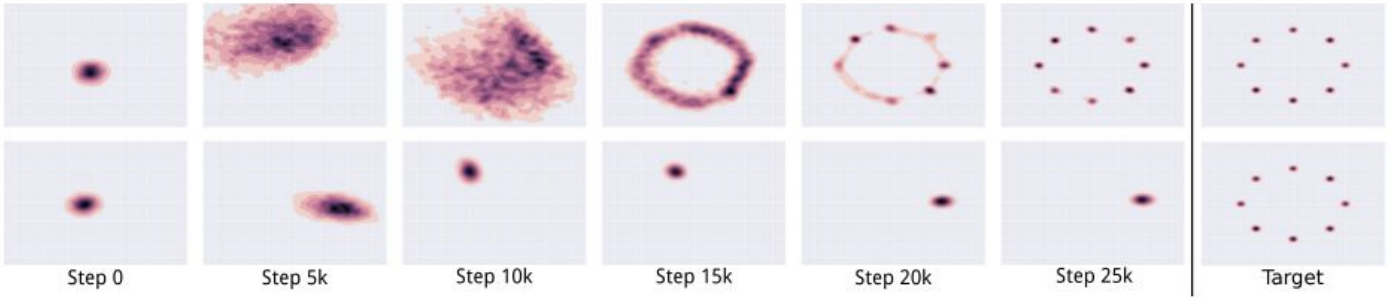



Figure 2: Unrolling the discriminator stabilizes GAN training on a toy 2D mixture of Gaussians dataset. Columns show a heatmap of the generator distribution after increasing numbers of training steps. The final column shows the data distribution. The top row shows training for a GAN with 10 unrolling steps. Its generator quickly spreads out and converges to the target distribution. The bottom row shows standard GAN training. The generator rotates through the modes of the data distribution. It never converges to a fixed distribution, and only ever assigns significant probability mass to a single data mode at once.

# Previously…

Jensen-Shannon Divergence in GANs:

- Theoretical foundation for GANS, symmetrizing KL divergence; defined as the average of two KL divergences.

$$KL(p\|q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

Problems with KL:

- If p(x) > 0 but q(x) = 0, then the result is infinity.
- If p(x) = 0, then the contribution to KL is zero.

# Wasserstein GAN

- **Wasserstein Distance**: Mass times the distance required to transform one distribution to another.

- **Intuitive and Differentiable**: Provides smoother gradients compared to other metrics.

- **Hard to Calculate**: Involves optimal transport, which is computationally intensive.

# Wasserstein GAN

$$W_c(\mu, \nu) = \sup_{\phi, \psi} \left( \int_X \phi(x)\, d\mu(x) + \int_Y \psi(y)\, d\nu(y) \right)$$

$$\phi(x) - \psi(y) \leq c(x, y)$$

$$\phi(x) = f(x), \quad \psi(y) = -f(y) \qquad\qquad c(x, y) = \|x - y\|$$

$$|f(x) - f(y)| \leq \|x - y\|$$

# Wasserstein GAN

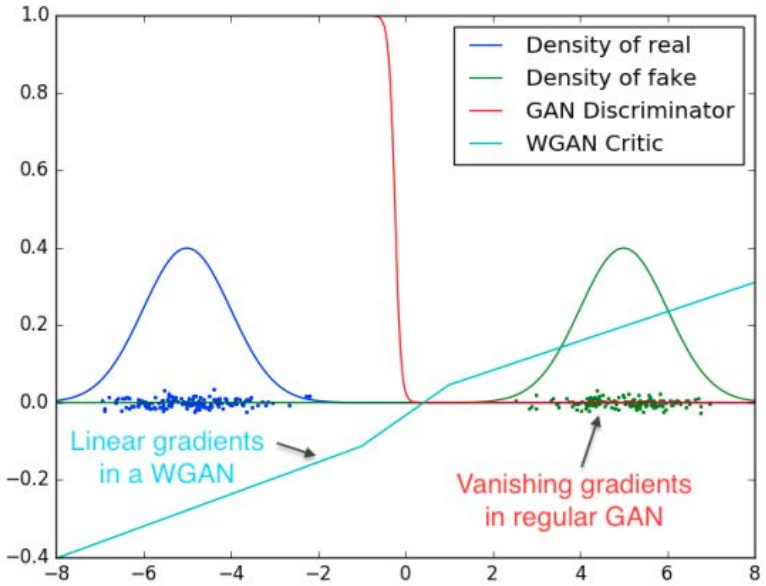
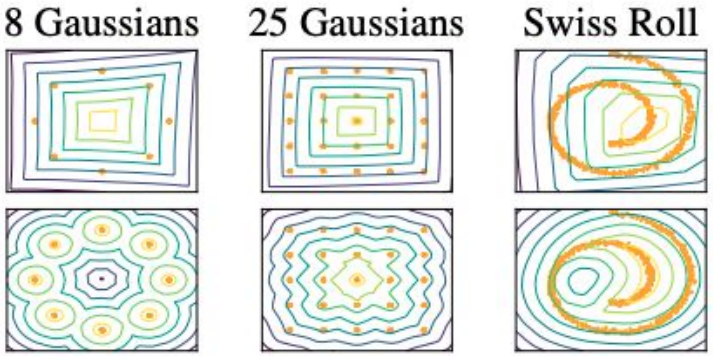



*Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.*

# Wasserstein GAN - Gradient Penalty (GP)

An alternative to gradient clipping

- Lipsch should be 1 everywhere
- Add penalty of mean squared distance between gradient and 1
- Use samples that are random linear interpolations between real and fake data
- Calculate gradient of D at random samples that are between points




(a) Value surfaces of WGAN critics trained to optimality on toy datasets using (top) weight clipping and (bottom) gradient penalty. Critics trained with weight clipping fail to capture higher moments of the data distribution. The 'generator' is held fixed at the real data plus Gaussian noise.

$$L = \mathbb{E}_X[D(X)] - \mathbb{E}_Z[D(G(Z))] + \lambda \mathbb{E}_{X'}\left(\|\nabla D(X')\|_2 - 1\right)^2$$

# Spectral Norm

- Max amount a matrix can stretch input vectors

$$\sup_x \frac{\|xA\|_2}{\|x\|_2} = \|A\|_2^*$$

- The Lipschitz is bound on how fast a function can change
- Divides weights by spectral norm to keep each layer stable

$$\hat{W} = \frac{W}{\|W\|_2^*}$$

- Helps keep discriminator smooth and well-behaved

1. Optimization Issues
2. Training and Stabilization
3. Take Aways

# Effective Trends

- Regularize and smooth the discriminator around real and generated points.
- Update players towards consensus or stable regions, not just greedy updates.
- Simplify networks and losses to eliminate nonlinearity.
- Constrain discriminator complexity to avoid overtraining.
- Grow or chain GANs to reduce complexity and learn a curriculum.

# Common Problems

- Unrolling is computationally expensive
- Sampling is unreliable
- Bound methods learn poor local optima

# Where to?

- How can we architect a neural network to learn a meaningful loss function?