Introduction to Generative Adversarial Networks

Luke de Oliveira
Vai Technologies
Lawrence Berkeley National Laboratory

@lukede0
@lukedeo
lukedeo@vaitech.io
https://ldo.io
Outline

• Why Generative Modeling?
• Taxonomy of Generative Models
• Generative Adversarial Networks
• Pitfalls
• Modifications
• Questions
Generative Models
Generative Modeling

- Asks question - can we build a model to approximate a *data distribution*?

- Formally we are given $x \sim p_{\text{data}}(x)$ and a finite sample from this distribution

$$ X = \{x | x \sim p_{\text{data}}(x)\}, \ |X| = n $$

- Problem: can we find a model such that

$$ p_{\text{model}}(x; \theta) \approx p_{\text{data}}(x) $$

- **Why** might this be useful?
Why care about Generative Models?

Oft over-used quote:

“What I cannot create, I do not understand”

-R. Feynman
Why care about Generative Models?

• Classic uses:
  • Through maximum likelihood, can fit to some interpretable parameters for a hand-designed $p_{\text{model}}(x; \theta)$
  • Learn a joint distribution with labels $p_{\text{data}}(x, y; \theta) \approx p_{\text{data}}(x, y)$ and transform to $p(y|x; \theta)$

• More interesting uses:
  • Fast-simulation of compute-heavy tasks
  • Interpolation between distributions
Traditional MLE Approach

• We are given a finite sample from a data distribution

\[ X = \{x | x \sim p_{\text{data}}(x)\}, |X| = n \]

• We construct a parametric model for the distribution, and build a likelihood

\[ \mathcal{L}(\theta; X) = \prod_{x \in X} p_{\text{model}}(x; \theta) \]

• In practice, we optimize through MCMC or other means, and obtain

\[ \theta_{\text{opt}} = \arg\min_{\theta} \{-\ln \mathcal{L}(\theta; X)\} \]
Generative Model Taxonomy

- Maximum Likelihood
  - Explicit density
    - Tractable density
      - Fully visible belief nets:
        - NADE
        - MADE
        - PixelRNN
        - Change of variables models (nonlinear ICA)
  - Implicit density
    - Approximate density
      - Variational
        - VAE
    - Markov Chain
      - GSN
      - Boltzmann machine
- Direct
  - GAN

From I. Goodfellow
Generative Adversarial Networks
Generative Adversarial Networks

• As before, we have a data distribution \( x \sim p_{\text{data}}(x), x \in \mathcal{X} \)

• We cast the process of building a model of the data distribution as a two-player game between a generator and a discriminator

• Our generator has a latent prior \( z \sim p_z(z), z \in \mathcal{Z} \) and maps this to sample space \( G : \mathcal{Z} \rightarrow \mathcal{X} \)

• \( G(\cdot; \theta_G) \) Implicitly defines a distribution \( p_{\text{model}}(x; \theta_G) \)

• Our discriminator \( D(\cdot; \theta_D) \) tells how fake or real a sample looks via a score \( D : \mathcal{X} \rightarrow \mathbb{R} \) (in practice, \( \text{Prob}[\text{Fake}] \))
Generative Adversarial Networks

- Distinguish real samples from fake samples
- Transform noise into a realistic sample

Real data
Vanilla GAN formulation

• How can we jointly optimize $G$ and $D$?

• Construct a two-person zero-sum minimax game with a value $V$

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x; \theta_D)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z; \theta_G); \theta_D))]$$

• We have an inner maximization by $D$ and an outer minimization by $G$

$$\min_G \max_D V(D, G)$$
Theoretical Guarantees

• Let’s step through the proof for equilibrium and implicit minimization of JSD
Theoretical Guarantees

- From original paper, know that $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x; \theta_G)}$

- Define generator solving for infinite capacity discriminator, $C(G) = V(D^*, G)$

- We can rewrite value as

$$C(G) = \mathbb{E}_{x \sim p_{\text{data}}}(\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x; \theta_G)}) + \mathbb{E}_{x \sim p_{\text{model}}(x; \theta_G)}(\log \frac{p_{\text{model}}(x; \theta_G)}{p_{\text{data}}(x) + p_{\text{model}}(x; \theta_G)})$$

- Simplifying notation, and applying some algebra

$$C(G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log \frac{p_{\text{data}}}{p_{\text{data}} + p_{\text{model}}}] + \mathbb{E}_{x \sim p_{\text{model}}}[\log \frac{p_{\text{model}}}{p_{\text{data}} + p_{\text{model}}}]$$

$$C(G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log \frac{p_{\text{data}}}{p_{\text{data}}/2 + p_{\text{model}}/2}] + \mathbb{E}_{x \sim p_{\text{model}}}[\log \frac{p_{\text{model}}}{p_{\text{data}}/2 + p_{\text{model}}/2}] - \log(4)$$

- But we recognize this as a summation of two KL-divergences

$$C(G) = D_{KL}(p_{\text{data}} \parallel \frac{p_{\text{data}} + p_{\text{model}}}{2}) + D_{KL}(p_{\text{model}} \parallel \frac{p_{\text{data}} + p_{\text{model}}}{2}) - \log(4)$$

- And can combine these into the Jensen-Shannon divergence

$$C(G) = 2 \cdot JSD(p_{\text{data}} \parallel p_{\text{model}}) - \log(4)$$

- This yields a unique global minimum precisely when

$$p_{\text{model}} = p_{\text{data}} \implies C(G) = -\log(4)$$
Theoretical Guarantees

• TL;DR from the previous proof is as follows

• If D and G are allowed to come from the space of all continuous functions, then we have:
  
  • Unique equilibrium \((\theta_{G}^{\text{opt}}, \theta_{D}^{\text{opt}})\)
  
  • The discriminator admits a flat posterior, i.e.,
    
    \[ D(x; \theta_{D}^{\text{opt}}) = 1/2 \quad \forall x \sim p_{\text{data}}(x) \]
    
    \[ D(G(z; \theta_{G}^{\text{opt}}); \theta_{D}^{\text{opt}}) = 1/2 \quad \forall z \sim p_{z}(z) \]
  
  • The implicit distribution defined by the generator exactly recovers the data distribution
    
    \[ p_{\text{model}}(x; \theta_{G}^{\text{opt}}) = p_{\text{data}}(x) \]
Pitfalls
GANs in Practice

• This minimax formulation saturates quickly, causing gradients propagating from the discriminator to vanish when the generator does poorly. Non saturating formulation:

  • Before: minimize $\theta_G \left\{ \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z; \theta_G); \theta_D))] \right\}$

  • After: maximize $\theta_G \left\{ \mathbb{E}_{z \sim p_z(z)} [\log D(G(z; \theta_G); \theta_D)] \right\}$
Failure Modes (ways in which GANs fail)

• Mode Collapse — i.e., learn to produce one mode of data distribution, stop there

• Vanishing/Exploding Gradients from discriminator to generator

• Generator produces garbage that fools discriminator
Introspection

• GANs do not naturally have a metric for convergence

• Ideally, all losses go to $\approx 0.69 = -\log(1/2)$

  • Often does not happen in practice
Modifications
GANs in Practice

- Even when using “non-saturating” heuristic, convergence still difficult
- “Tricks” needed to make things work on real data
- Two major (pre-2017) categories
  - Architectural Guidelines
  - Side information / Information theoretic
• Deep Convolutional Generative Adversarial Networks provide a set of ad-hoc guidelines for building architectures for images

• Enabled a lot of current progress in GAN research
Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-stride convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.
Side Information

• Conditional GAN, Auxiliary Classifier GAN, InfoGAN etc.

• Key idea: can we leverage side information (a label, description, trait, etc.) to produce either better quality or conditional samples?

• The discriminator can either be shown the side information or tasked with reconstructing it
Conditional GAN (CGAN)

• Latent variable is passed to the generator and the discriminator.

• The generator learns side-information conditional distributions, as it is able to disentangle this from the overall latent space.
Auxiliary Classifier GAN (ACGAN)

- Similar to CGAN, latent variable is passed to the generator
- Discriminator is tasked with jointly learning real-vs-fake and the ability to reconstruct the latent variable being passed in
InfoGAN

• Instead of the latent variables being known a priori from a dataset, make parts of latent space randomly drawn from different distributions
  • Bernoulli, Normal, multiclass, etc.
• Make the discriminator reconstruct these arbitrary elements of latent space that are passed into generator
• Learns disentangled features (maximizes mutual information)
Conclusion

• Showed theoretical guarantees of GANs (in unrealistic settings) and convergence properties

• Discussed pitfalls of GANS

• Explored a few basic methods with ~no theory that try to improve GANs
  • Architecture improvements, side information
  • I didn’t talk about Minibatch Discrimination or Feature Matching
Questions?
Thanks!

(2) A. Radford, L. Metz and S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.


