



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



JOHN HOPCROFT
CENTER FOR
COMPUTER SCIENCE



Generative Adversarial Nets meet Reinforcement Learning

Weinan Zhang

Shanghai Jiao Tong University

<http://wnzhang.net>

Tutorial at NLPCC 2021

Content

1. Introduction to Generative Adversarial Nets
2. GANs on Continuous Data
3. GAN and RL
4. GANs on Discrete Data
5. Summary

Problem Definition of Data Generation

- Given a dataset $D = \{x\}$, build a model $q_\theta(x)$ of the data distribution that fits the real one $p(x)$
- Traditional objective: maximum likelihood estimation (MLE)

$$\max_{\theta} \frac{1}{|D|} \sum_{x \in D} [\log q_\theta(x)] \simeq \max_{\theta} \mathbb{E}_{x \sim p(x)} [\log q_\theta(x)]$$

- Check whether a real data is with a high mass density of the learned model

Inconsistency of Evaluation and Use

- Given a generator q with a certain generalization ability

$$\max_{\theta} \mathbb{E}_{x \sim p(x)} [\log q_{\theta}(x)]$$

Training/evaluation

- Check whether a real data is with a high mass density of the learned model
- Approximated by

$$\max_{\theta} \frac{1}{|D|} \sum_{x \in D} [\log q_{\theta}(x)]$$

$$\max_{\theta} \mathbb{E}_{x \sim q_{\theta}(x)} [\log p(x)]$$

Use

- Check whether a model-generated data is considered as real as possible
- More straightforward but it is hard or impossible to directly calculate $p(x)$

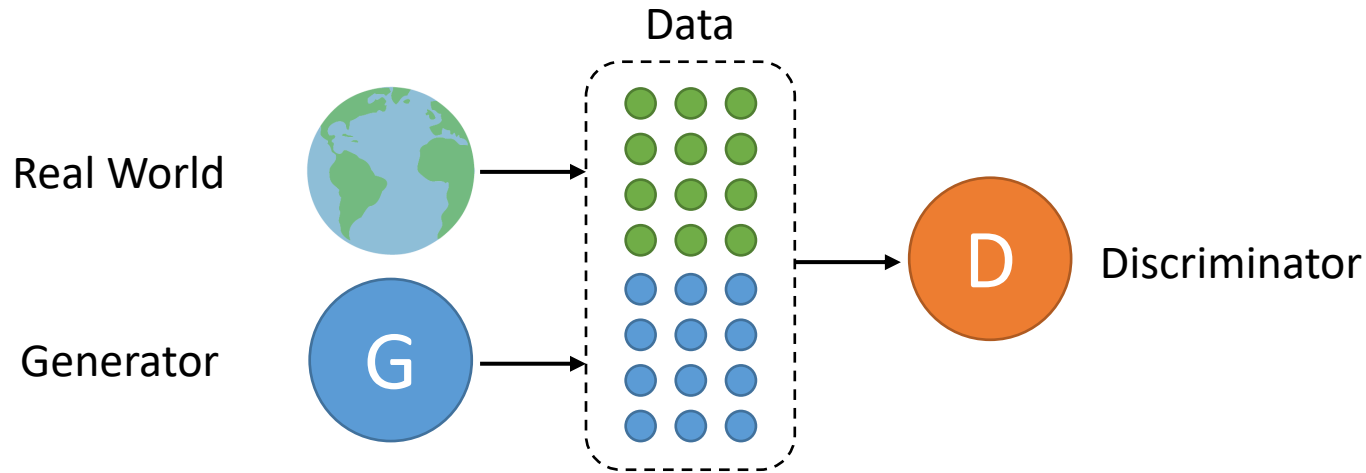
Generative Adversarial Nets (GANs)

- What we really want is

$$\max_{\theta} \mathbb{E}_{x \sim q_{\theta}(x)} [\log p(x)]$$

- But we cannot directly calculate $p(x)$
- Idea: what if we build a discriminator to judge whether a data instance is real or fake (artificially generated)?
 - Leverage the strong power of deep learning based discriminative models

Generative Adversarial Nets (GANs)



- Discriminator tries to correctly distinguish the real data and the fake model-generated data
- Generator tries to generate high-quality data to fool discriminator
- G & D can be implemented via neural networks
- Ideally, when D cannot distinguish the real and generated data, G nicely fits the real underlying data distribution

Generator and Discriminator Nets

- Generator network

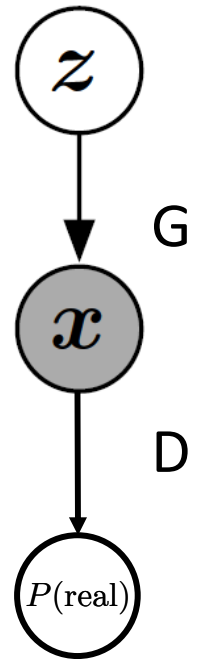
$$\mathbf{x} = G(\mathbf{z}; \boldsymbol{\theta})$$

- Must be differentiable
- No invertibility requirement
- Popular implementation: multi-layer perceptron

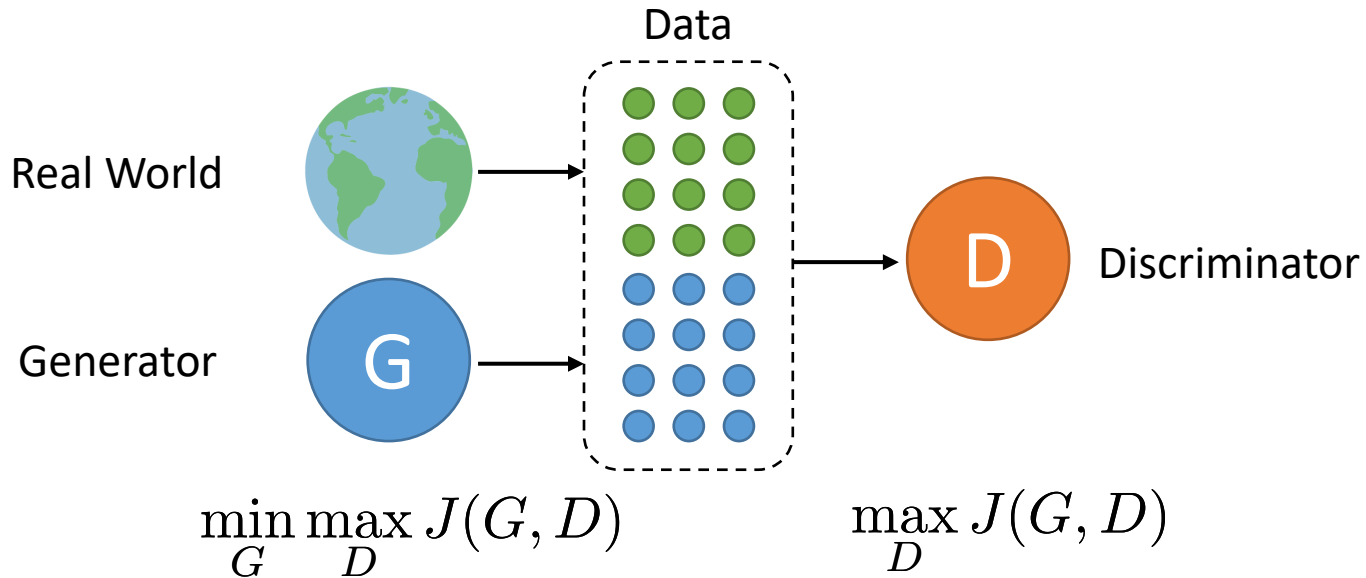
- Discriminator network

$$P(\text{real}|\mathbf{x}) = D(\mathbf{x}; \boldsymbol{\phi})$$

- Can be implemented by any neural networks with a probabilistic prediction
- For example
 - Multi-layer perceptron with logistic output
 - AlexNet etc.



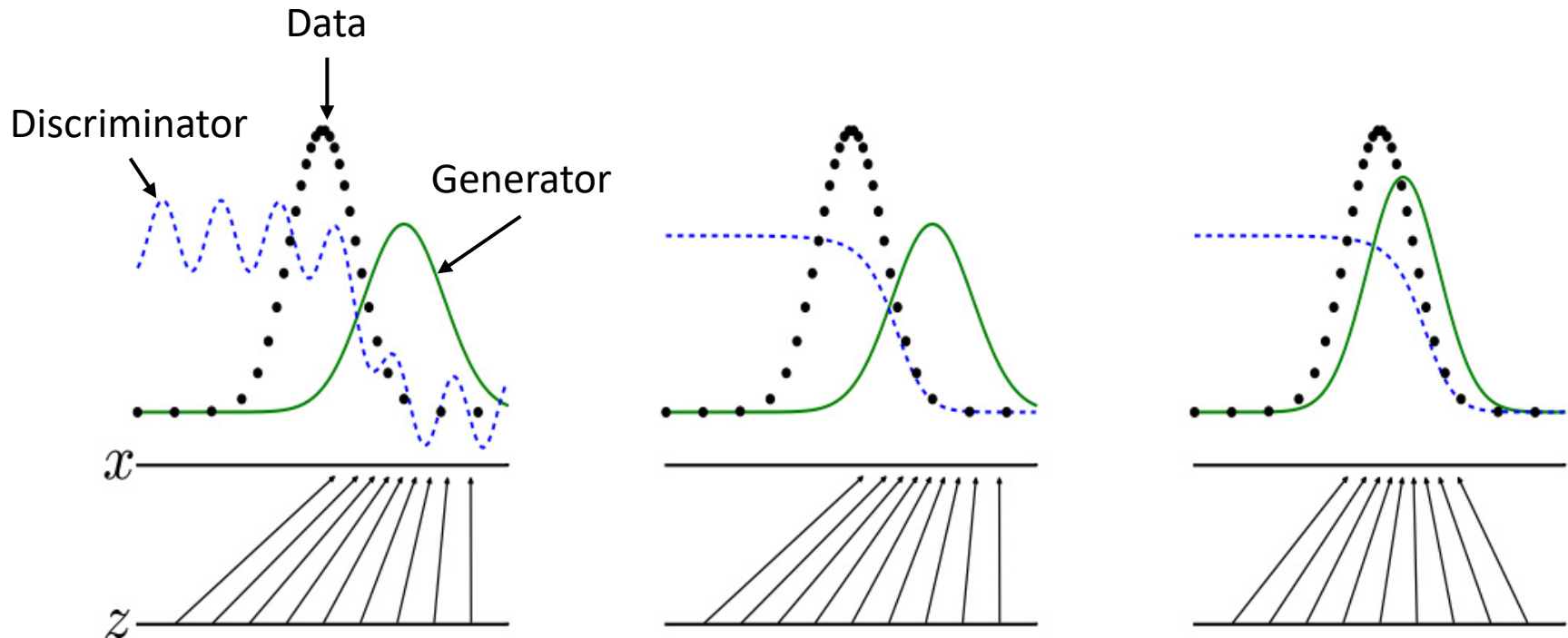
GAN: A Minimax Game



The joint objective function

$$J(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Illustration of GANs



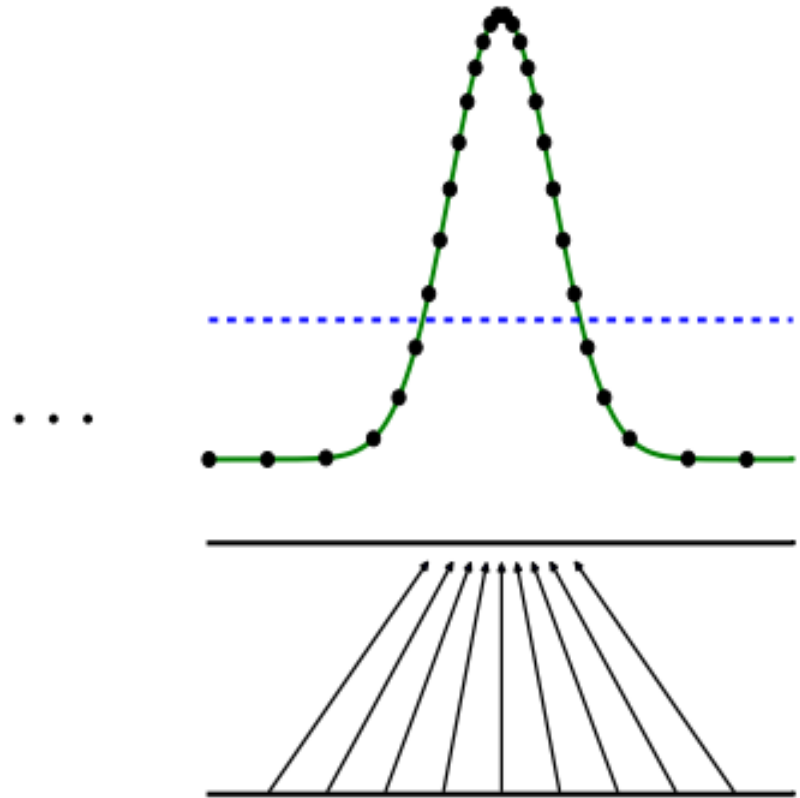
Generator $\min_G \max_D J^{(D)}$

Discriminator $\max_D J^{(D)}$

$$J^{(D)} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Ideal Final Equilibrium

- Generator generates perfect data distribution
- Discriminator cannot distinguish the real and generated data

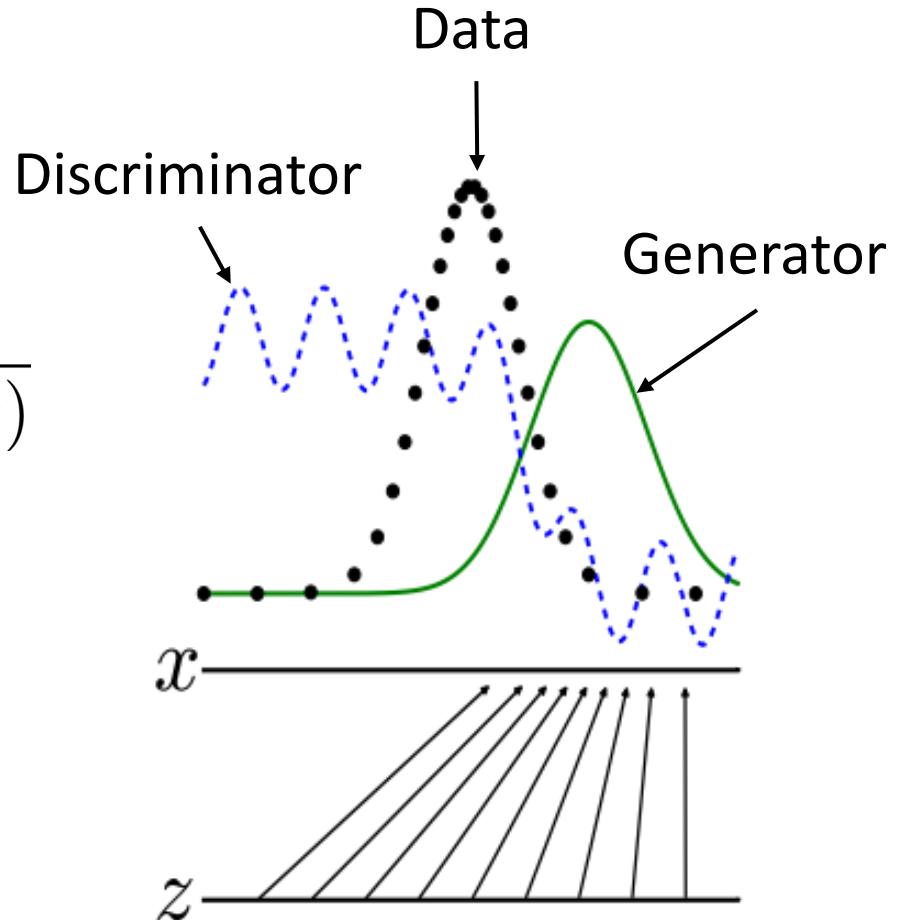


Optimal Strategy for Discriminator

- Optimal $D(\mathbf{x})$ for any $p_{\text{data}}(\mathbf{x})$ and $p_G(\mathbf{x})$ is always

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

- If this optimum is allowed to reach, then we have an ideal equilibrium for GAN.



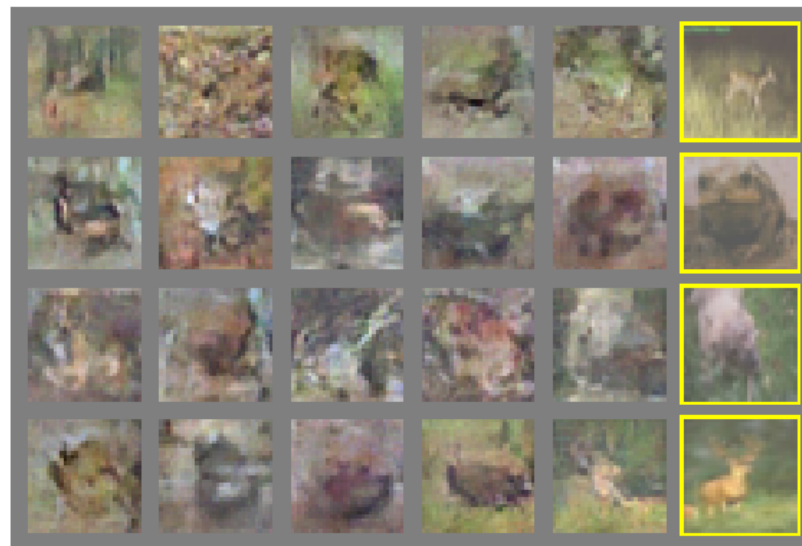
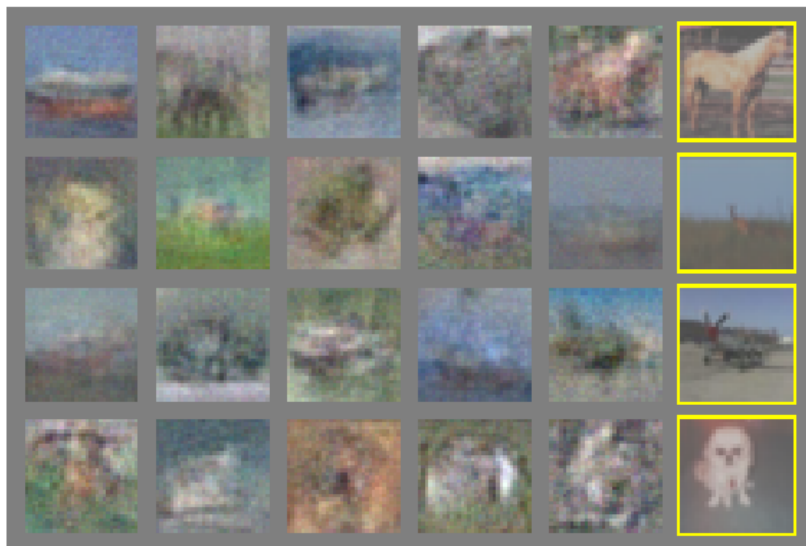
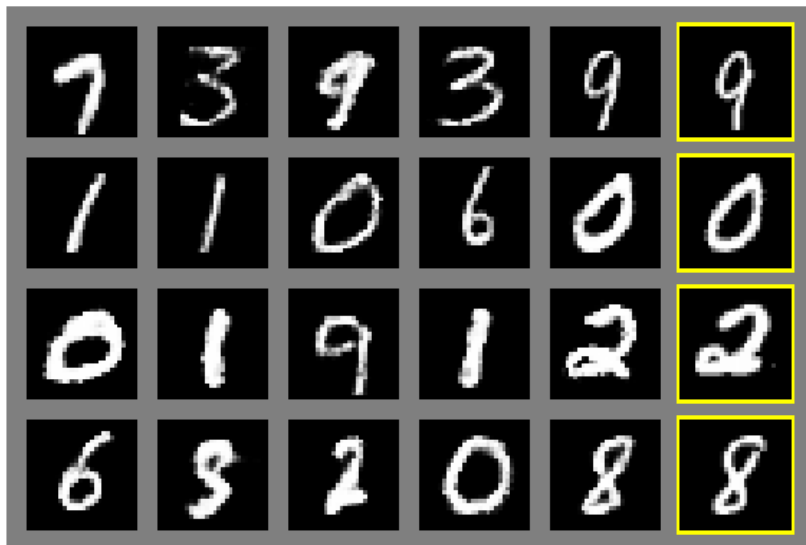
Equilibrium for the Minimax Game

$$\mathbf{G}: \min_G \max_D J(G, D) \quad \mathbf{D}: \max_D J(G, D)$$

$$\begin{aligned} J(G, D) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_G(\mathbf{x})} [\log(1 - D(\mathbf{x}))] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} \right] \\ &\quad + \mathbb{E}_{\mathbf{x} \sim p_G(\mathbf{x})} \left[\log \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} \right] \\ &= -\log(4) + \underbrace{\text{KL} \left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_G}{2} \right\| \right)}_{\geq 0} + \underbrace{\text{KL} \left(p_G \left\| \frac{p_{\text{data}} + p_G}{2} \right\| \right)}_{\geq 0} \end{aligned}$$

- An equilibrium is $p_G(x) = p_{\text{data}}(x)$ and $D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)} = 0.5$

Case Study of GANs for Continuous Data

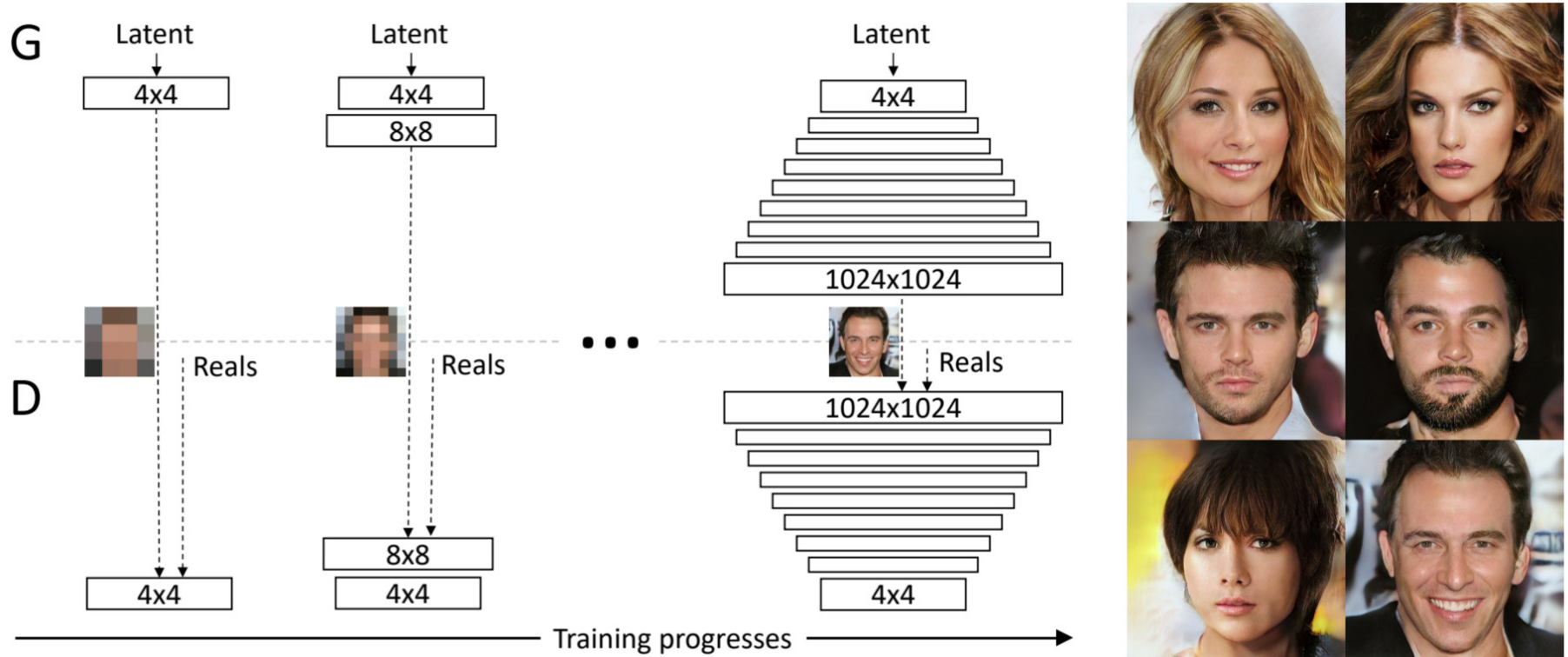


Why study generative models?

- Excellent test of our ability to use high-dimensional, complicated probability distributions
- Simulate possible futures for planning or simulated RL
- Missing data
 - Semi-supervised learning
- Multi-modal outputs
- Realistic generation tasks

High Resolution and Quality Images

- Progressive Growing of GANs from 4^2 to 1024^2



High Resolution and Quality Images

- BigGAN in ICLR 2019



Single Image Super-Resolution

bicubic
(21.59dB/0.6423)



SRResNet
(23.53dB/0.7832)



SRGAN
(21.15dB/0.6868)



original



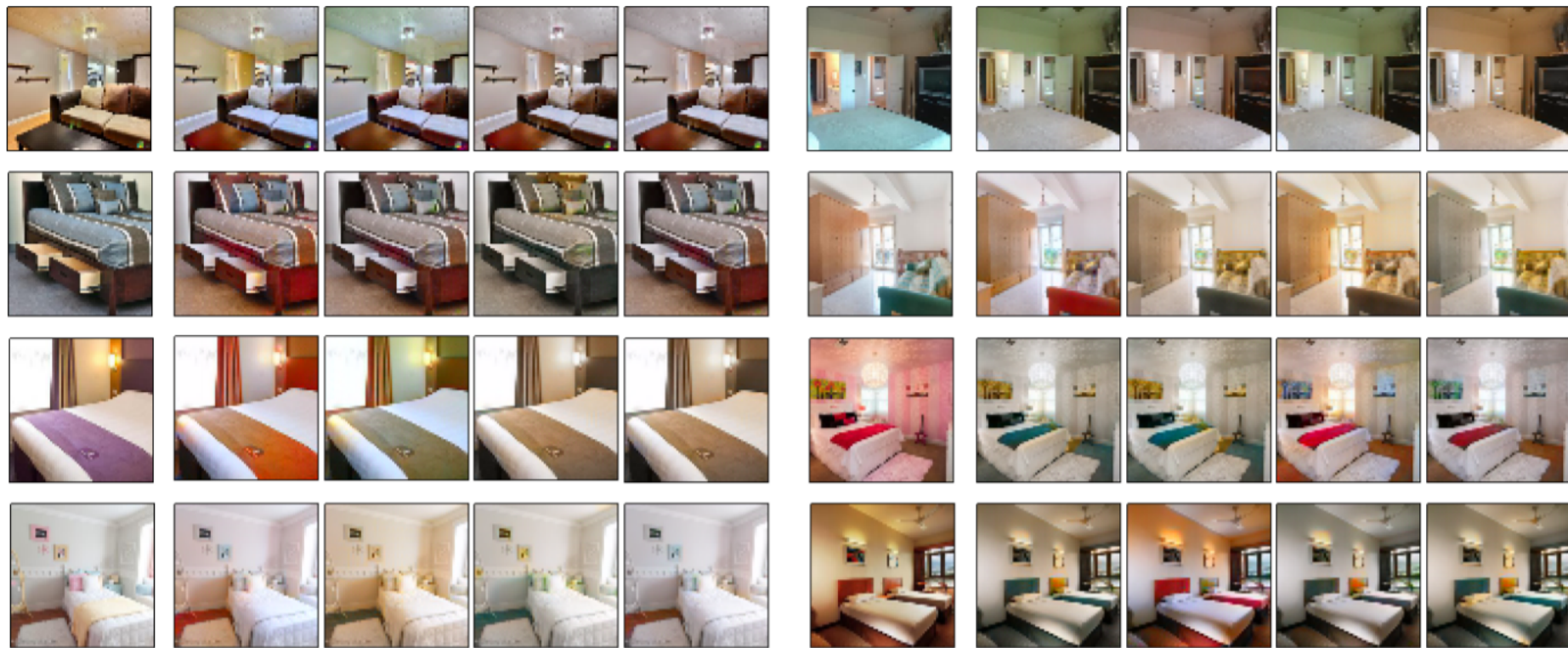
[4× upscaling]

deep residual generative adversarial
network optimized for a loss more
sensitive to human perception

Image to Image Translation



Grayscale Image Colorization



Ground
Truth

Generated Colorization
after Performing Grayscale

Ground
Truth

Generated Colorization
after Performing Grayscale

High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs

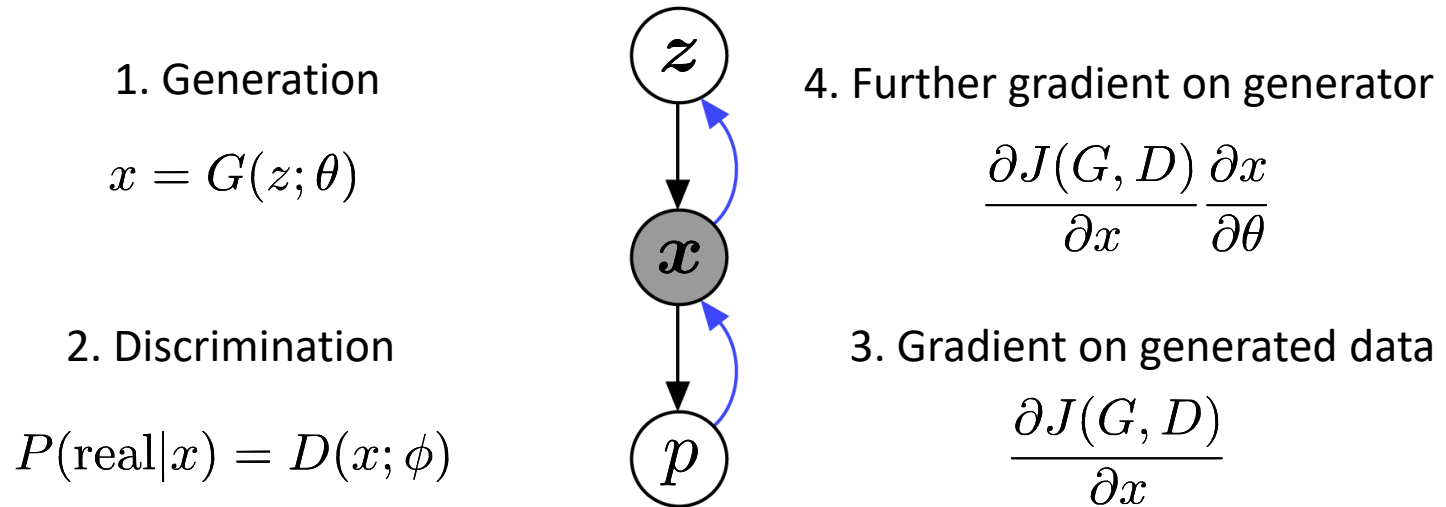


Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs", arXiv preprint arXiv:1711.11585.

Content

1. Introduction to Generative Adversarial Nets
2. GANs on Continuous Data
3. GAN and RL
4. GANs on Discrete Data
5. Summary

GANs for Continuous Data



- In order to take gradient on the generator parameter, x has to be continuous

$$J(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$\text{Generator } \min_G \max_D J(G, D) \quad \text{Discriminator } \max_D J(G, D)$$

Review GAN Objective

$$\min_G \max_D J(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$\begin{aligned} J(G) &= \max_D J(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_G(\mathbf{x})} [\log(1 - D(\mathbf{x}))] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} \right] \\ &\quad + \mathbb{E}_{\mathbf{x} \sim p_G(\mathbf{x})} \left[\log \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} \right] \\ &= -\log(4) + \text{KL} \left(p_{\text{data}} \parallel \frac{p_{\text{data}} + p_G}{2} \right) + \text{KL} \left(p_G \parallel \frac{p_{\text{data}} + p_G}{2} \right) \end{aligned}$$

\uparrow
 $2 \times JS(p_{\text{data}} \parallel p_G)$

General GAN form

Vanilla GAN form

$$J(G) = \max_D J(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

General GAN form

$$\min_{f \in \mathcal{F}} J_D \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\phi(f(g(z)))] + \mathbb{E}_{x \sim \mathcal{P}_r} [\varphi(f(x))],$$

$$\min_{g \in \mathcal{G}} J_G \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\psi(f(g(z)))],$$

$\phi, \psi, \varphi: \mathbb{R} \rightarrow \mathbb{R}$ are loss metrics.
--

$$\text{GAN} \quad : \quad \phi(x) = \psi(-x) = -\log(\sigma(-x))$$

$$\text{WGAN} \quad : \quad \phi(x) = \psi(-x) = x$$

$$\text{LSGAN} \quad : \quad \phi(x) = \psi(-x) = (x + \alpha)^2$$

Problems on Gradients

Vanilla GAN form

$$J(G) = \max_D J(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

General GAN form

$$\min_{f \in \mathcal{F}} J_D \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\phi(f(g(z)))] + \mathbb{E}_{x \sim \mathcal{P}_r} [\varphi(f(x))],$$

$$\min_{g \in \mathcal{G}} J_G \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\psi(f(g(z)))],$$

$$\frac{\partial J_G(z)}{\partial \theta} = \frac{\partial \psi(f(x))}{\partial f(x)} \cdot \frac{\partial f(x)}{\partial x} \cdot \frac{\partial g_\theta(z)}{\partial \theta}$$

$x = g_\theta(z)$

↑

Problem 1:
Gradient
vanishing

↑

Problem 2:
Gradient
uninformativeness

Towards Solving Gradient Vanishing Problem

Wassertein GAN

Arjovsky et al. Wasserstein GANs. ICML 2017.

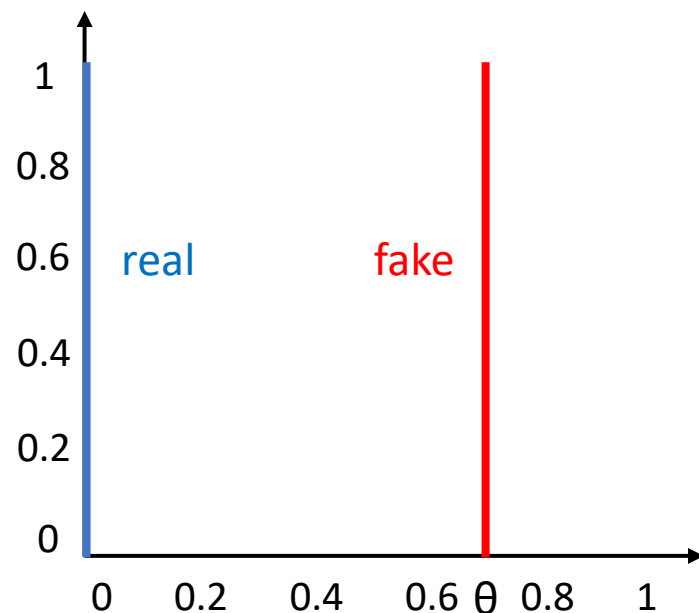
Different Divergences/Distances

$$W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$$

$$JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases}$$

$$(\text{TV}) \delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases}$$

$$KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$



KL, JS and TV require intersections.

Initially, p_{model} is far away from p_{data}

Illustration of EM Distance (W1) and JSD

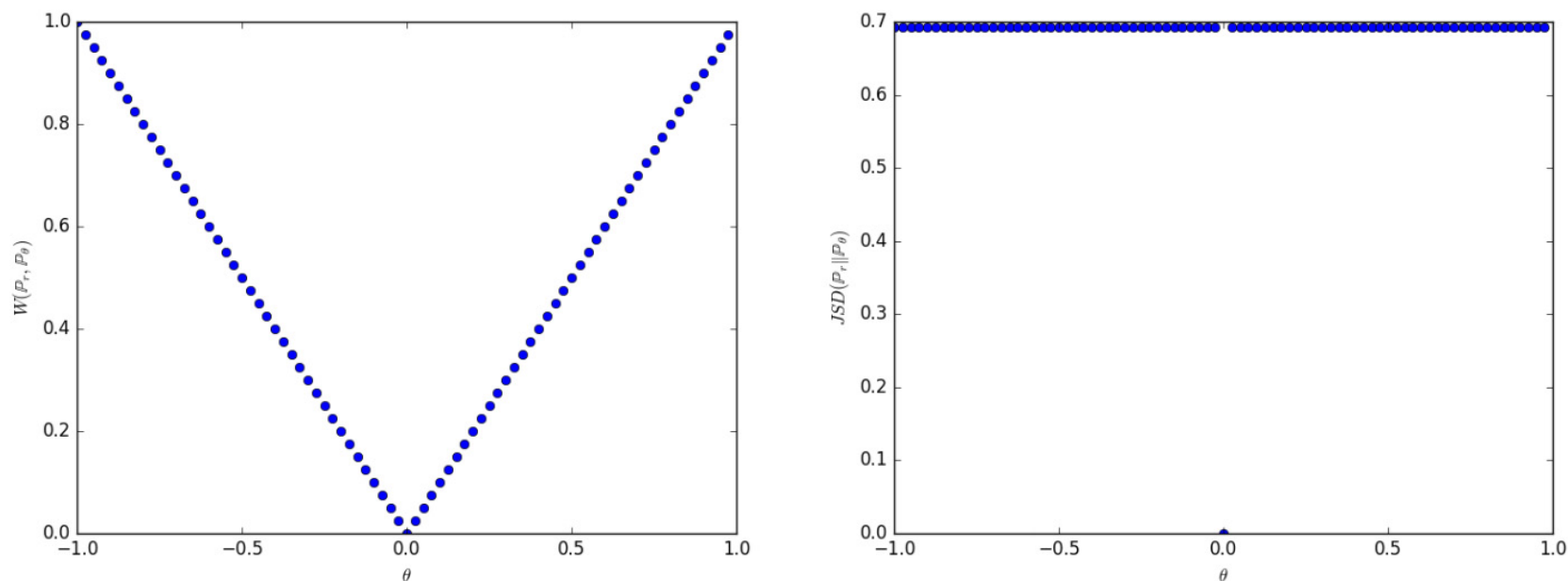


Figure 1: These plots show $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$ as a function of θ when ρ is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.

Wasserstein GAN

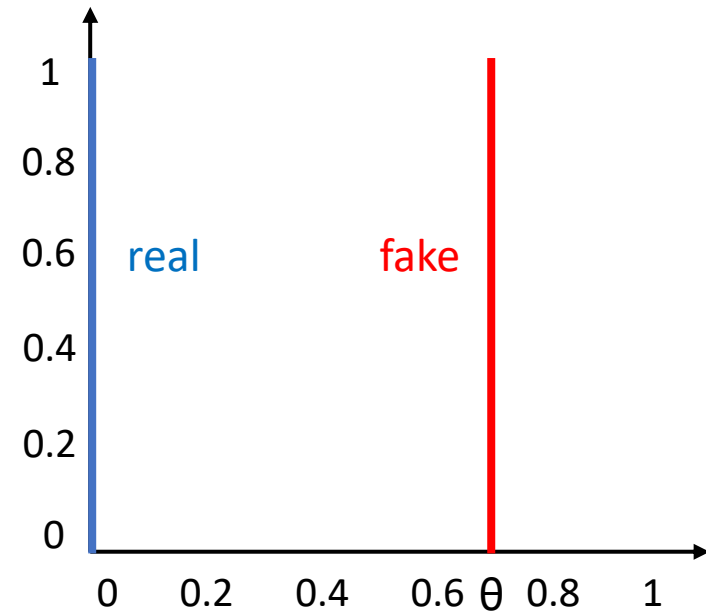
- The optimal D:
 - Approaching the f that maximizes the Wasserstein Distance

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_\theta)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$



Kantorovich-Rubinstein duality

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



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

Ensure K-Lipschitz Property

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

$$\|f\|_L \leq K \text{ (K-Lipschitz for some constant } K)$$

- Weight Clipping
- Remove the *log* function from the original objective

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] .$$

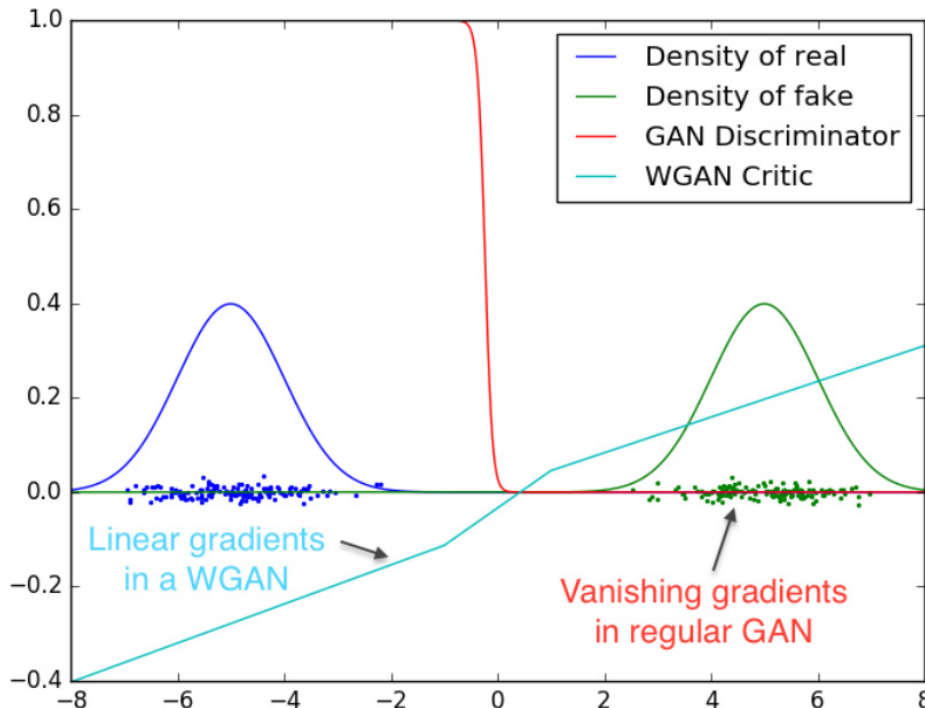
- Gradient penalty

$$L = \underbrace{\mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})]}_{\text{Original critic loss}} + \lambda \underbrace{\mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} [(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]}_{\text{Our gradient penalty}} .$$

$\hat{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}$ uniformly along straight lines between pairs of points

WGAN Advantages

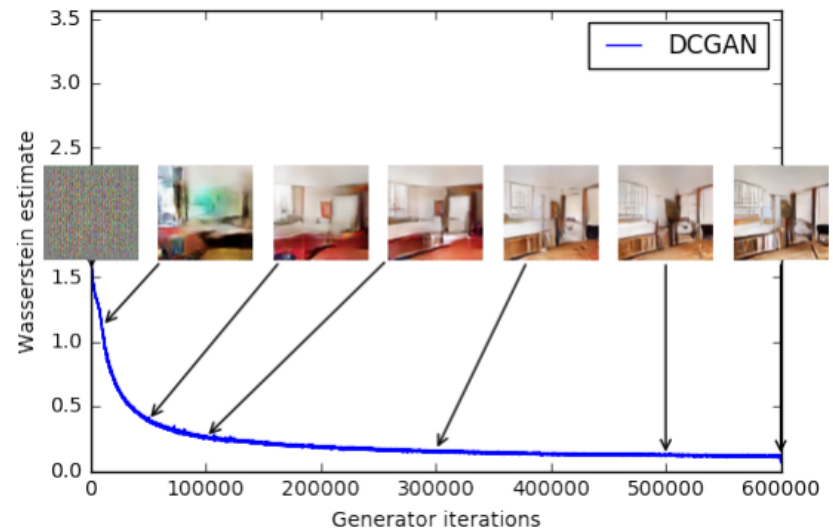
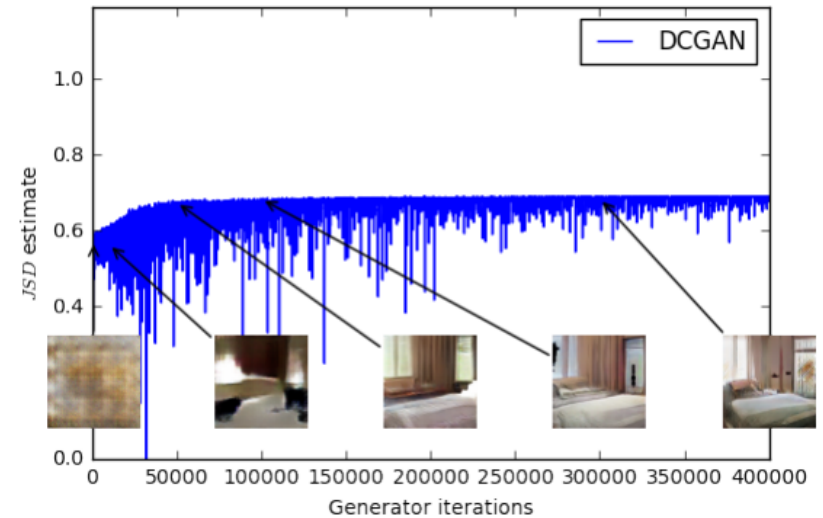
- Can train the D till optimality.
 - The original GAN needs delicate balance G and D to avoid gradient vanishing



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.

WGAN Advantages

- Can train the D till optimality
 - The original GAN need delicate balance G and D to avoid gradient vanishing
- Meaningful D loss
 - D is approximating the W distance

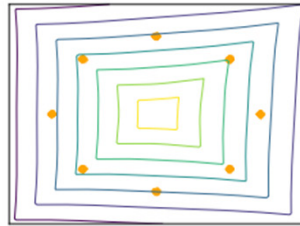


WGAN Improved Stability?

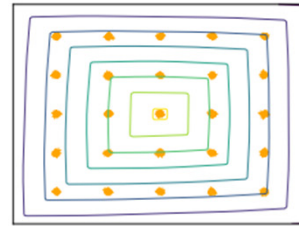
- Can work with MLP generator
- Can work without batch normalization
- No mode collapse observed
- Unstable
 - when uses momentum based optimizer such as Adam
 - when one uses high learning rates

Weight Clipping

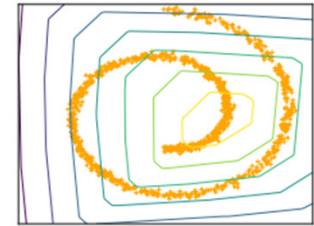
8 Gaussians



25 Gaussians

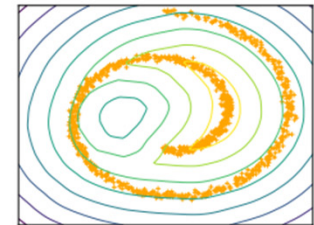
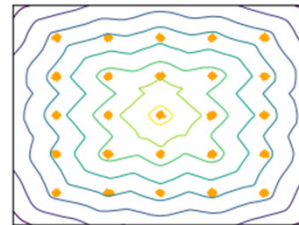
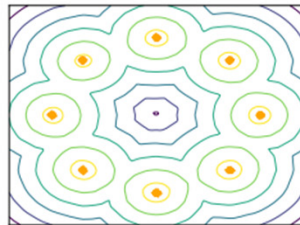


Swiss Roll



Learned patterns
with **weight clipping**

Learned patterns
with **gradient penalty**



Value surfaces of WGAN critics trained to optimality on toy datasets using (top) weight clipping and (bottom) gradient penalty.

Observations: 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.

Weight Clipping Trained D is not Optimal

Proposition 4. *Consider a (deep) NN with ReLU activation functions and linear output layer. A function generated by the NN under constraining each weight in absolute value by c_{\max} exhausts the common Lipschitz constraint if and only if*

- (a) The weight matrix of the first layer consists of constant columns with value c_{\max} or $-c_{\max}$.*
- (b) The weights of all other layers are given by a matrix C^{\max} with every entry equal to c_{\max} .*


The optimal f^* is not in the class of functions that can be generated by the network under the weight clipping constraint.

Spectral Normalization

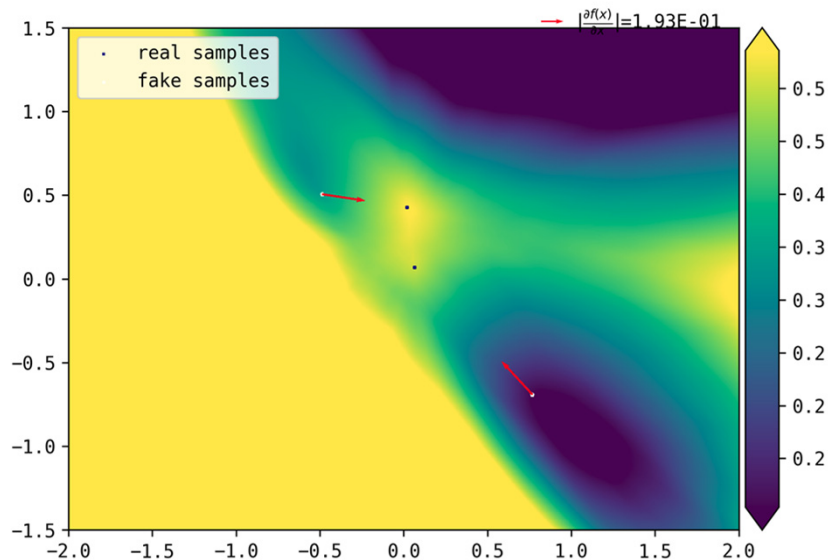
$$\begin{aligned}\|f\|_{\text{Lip}} &\leq \|(\mathbf{h}_L \mapsto W^{L+1}\mathbf{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\mathbf{h}_{L-1} \mapsto W^L\mathbf{h}_{L-1})\|_{\text{Lip}} \\ &\quad \cdots \|a_1\|_{\text{Lip}} \cdot \|(\mathbf{h}_0 \mapsto W^1\mathbf{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\mathbf{h}_{l-1} \mapsto W^l\mathbf{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l).\end{aligned}$$

Spectral normalization normalizes the spectral norm of the weight matrix W so that it satisfies the Lipschitz constraint

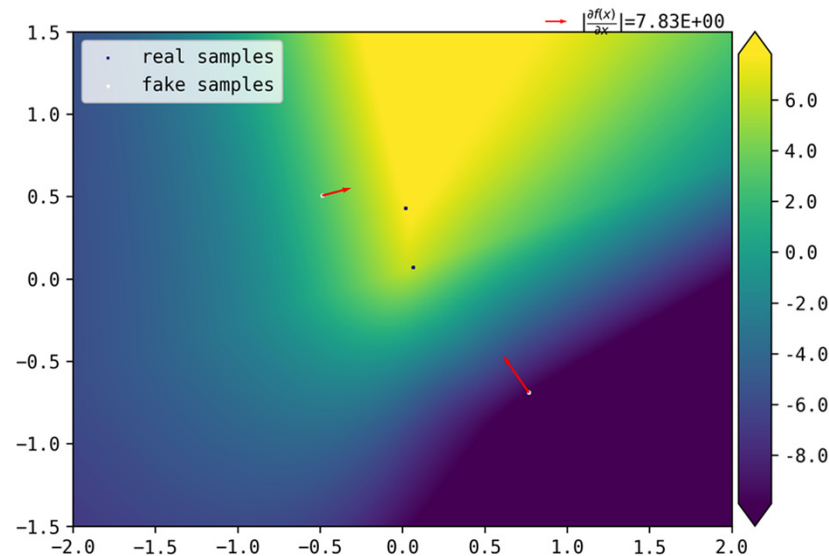
$$\bar{W}_{\text{SN}}(W) := W / \sigma(W). \quad \Longrightarrow \quad \sigma(\bar{W}_{\text{SN}}(W)) = 1$$


largest singular value of W

Spectral Normalization



Gradient Penalty



Spectral Normalization

In our preliminary analysis, spectral normalization failed to achieve the optimal discriminative function.

Towards Solving Gradient Uninformativeness Problem

Lipschitz GAN

Zhiming Zhou et al. Lipschitz Generative Adversarial Nets.
ICML 2019.

Problems on Gradients

Vanilla GAN form

$$J(G) = \max_D J(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

General GAN form

$$\min_{f \in \mathcal{F}} J_D \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\phi(f(g(z)))] + \mathbb{E}_{x \sim \mathcal{P}_r} [\varphi(f(x))],$$

$$\min_{g \in \mathcal{G}} J_G \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\psi(f(g(z)))],$$

$\phi, \psi, \varphi: \mathbb{R} \rightarrow \mathbb{R}$ are loss metrics.
--

$$\text{GAN} \quad : \quad \phi(x) = \psi(-x) = -\log(\sigma(-x))$$

$$\text{WGAN} \quad : \quad \phi(x) = \psi(-x) = x$$

$$\text{LSGAN} \quad : \quad \phi(x) = \psi(-x) = (x + \alpha)^2$$

Problems on Gradients

Vanilla GAN form

$$J(G) = \max_D J(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

General GAN form

$$\min_{f \in \mathcal{F}} J_D \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\phi(f(g(z)))] + \mathbb{E}_{x \sim \mathcal{P}_r} [\varphi(f(x))],$$

$$\min_{g \in \mathcal{G}} J_G \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\psi(f(g(z)))],$$

$$\frac{\partial J_G(z)}{\partial \theta} = \frac{\partial \psi(f(x))}{\partial f(x)} \cdot \frac{\partial f(x)}{\partial x} \cdot \frac{\partial g_\theta(z)}{\partial \theta}$$

\uparrow
Problem 1:
Gradient
vanishing

\uparrow
Problem 2:
Gradient
uninformativeness

$x = g_\theta(z)$

The Gradient Uninformativeness

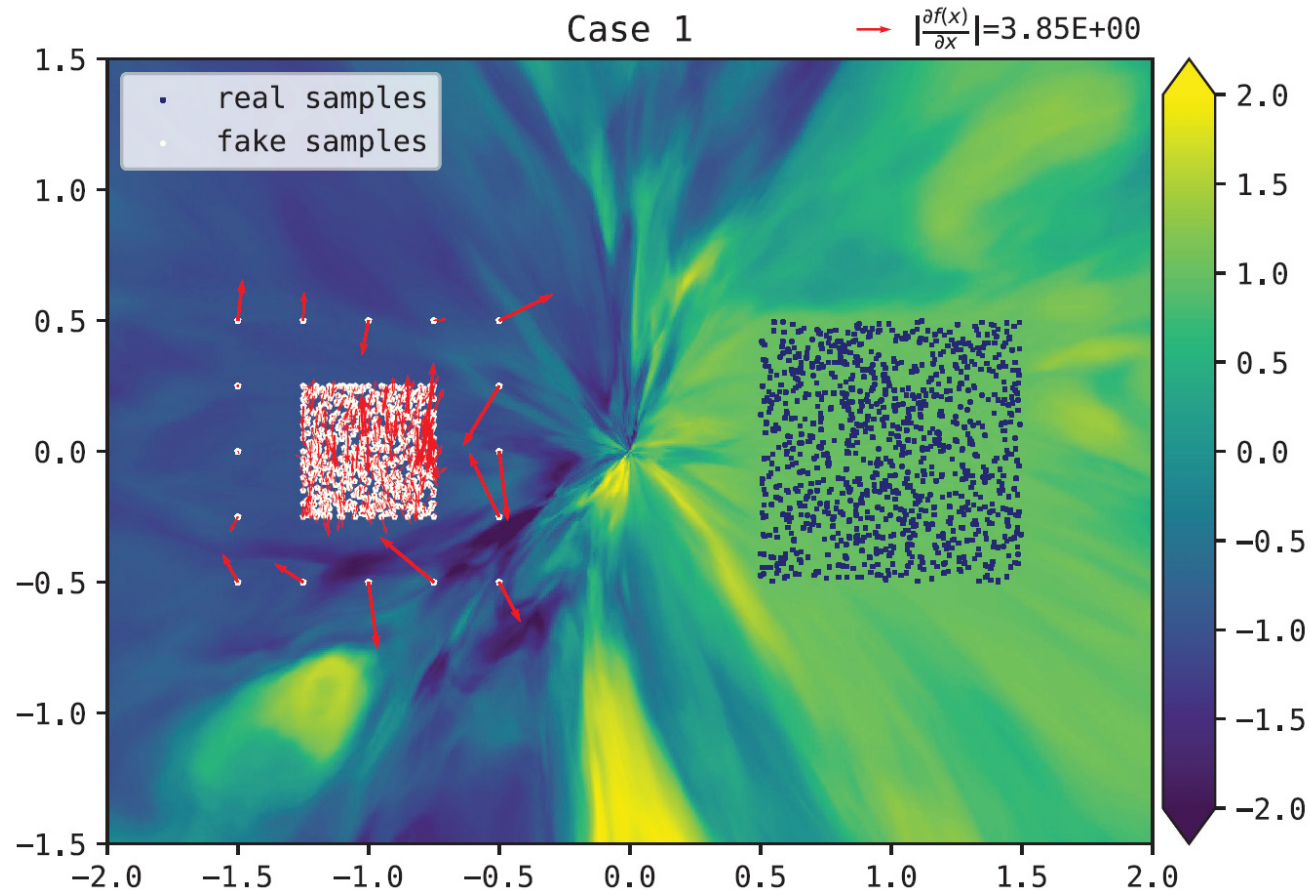
- The problem that the gradient from the discriminator does not contain any information about the real distribution.
- If there is no restriction on $f(x)$, then $f^*(x)$ is independently defined (universal approximation) and only reflects the local densities P_g and P_r

$$f^*(x) = \arg \min_{f(x) \in \mathbb{R}} \mathcal{P}_g(x)\phi(f(x)) + \mathcal{P}_r(x)\varphi(f(x)), \quad \forall x$$

- Thus gradient that x receives from $f^*(x)$ does not reflect any information about P_r

$$\frac{\partial J_G(z)}{\partial \theta} = \frac{\partial \psi(f(x))}{\partial f(x)} \cdot \frac{\partial f(x)}{\partial x} \cdot \frac{\partial g_\theta(z)}{\partial \theta}$$

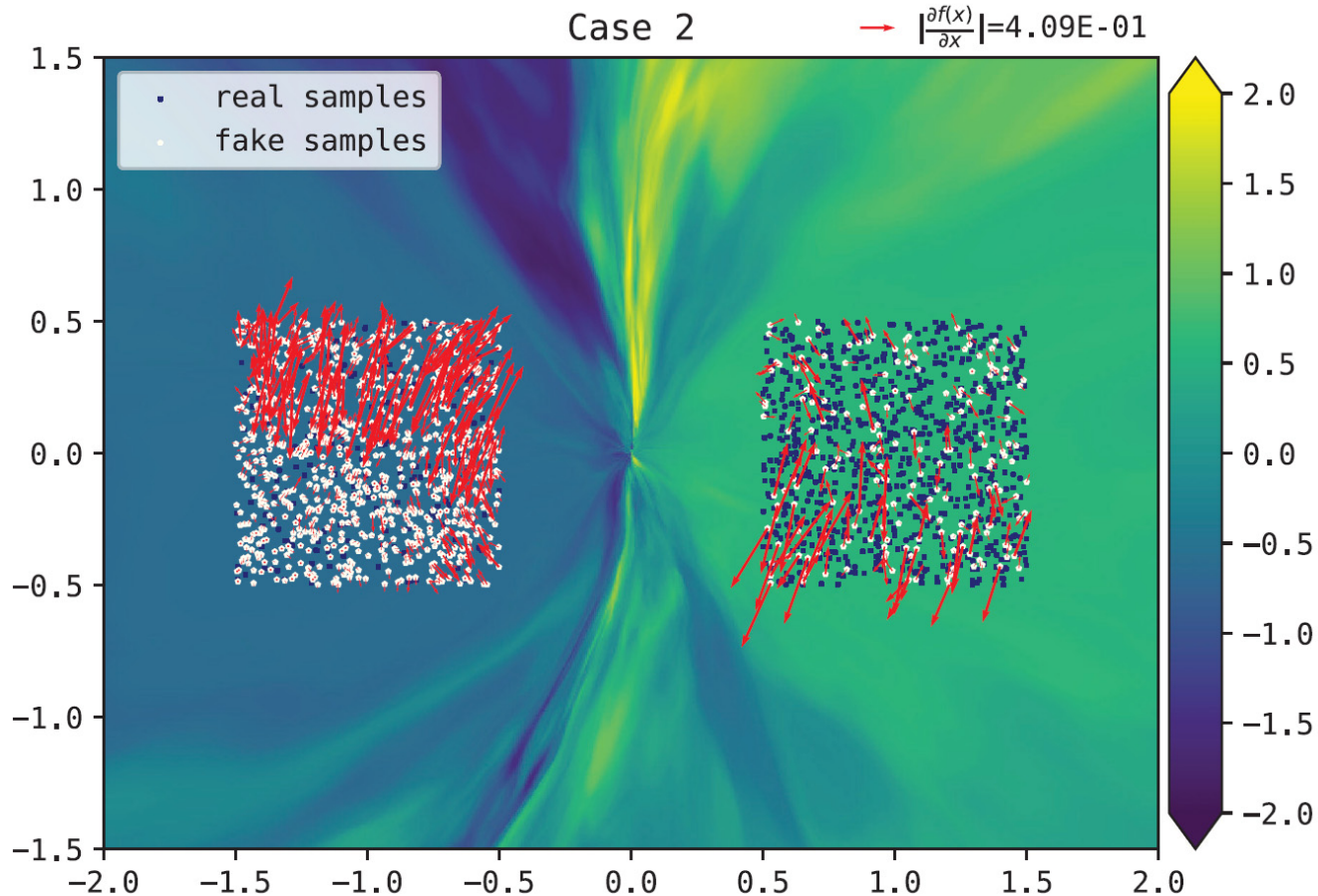
Gradient Uninformativeness



(a) Disjoint Case

Gradient uninformativeness practically behaviors as noisy gradient.

Gradient Uninformativeness



(b) Overlapping Case

Gradient uninformativeness practically behaviors as noisy gradient.

Lipschitz GANs (LGAN)

$$\min_{f \in \mathcal{F}} J_D \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\phi(f(g(z)))] + \mathbb{E}_{x \sim \mathcal{P}_r} [\varphi(f(x))] + \lambda k(f)^2$$

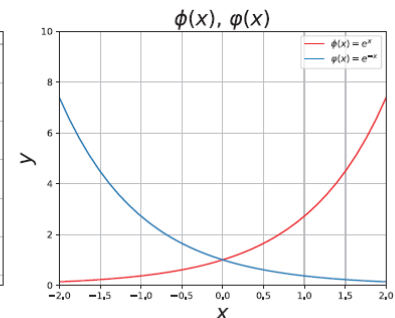
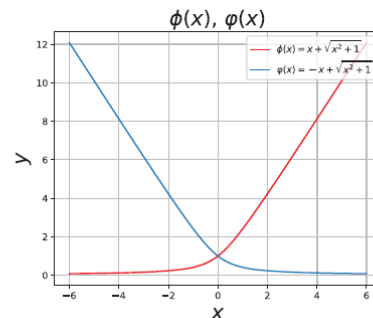
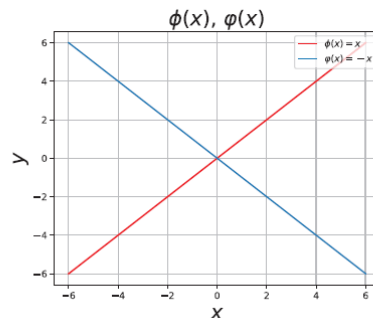
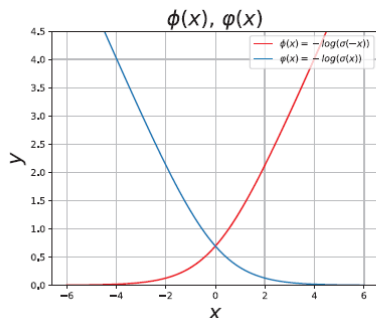
$$\min_{g \in \mathcal{G}} J_G \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\psi(f(g(z)))]$$

↑
Lipschitz constant of f ,
implemented by **max
gradient penalty** in a
minibatch

- LGAN requires ϕ and φ to satisfy:

$$\begin{cases} \phi'(x) > 0, \\ \phi''(x) \geq 0, \end{cases} \quad \text{and} \quad \phi(x) = \psi(-x).$$

Any increasing function with non-decreasing derivative.



Lipschitz GANs Properties

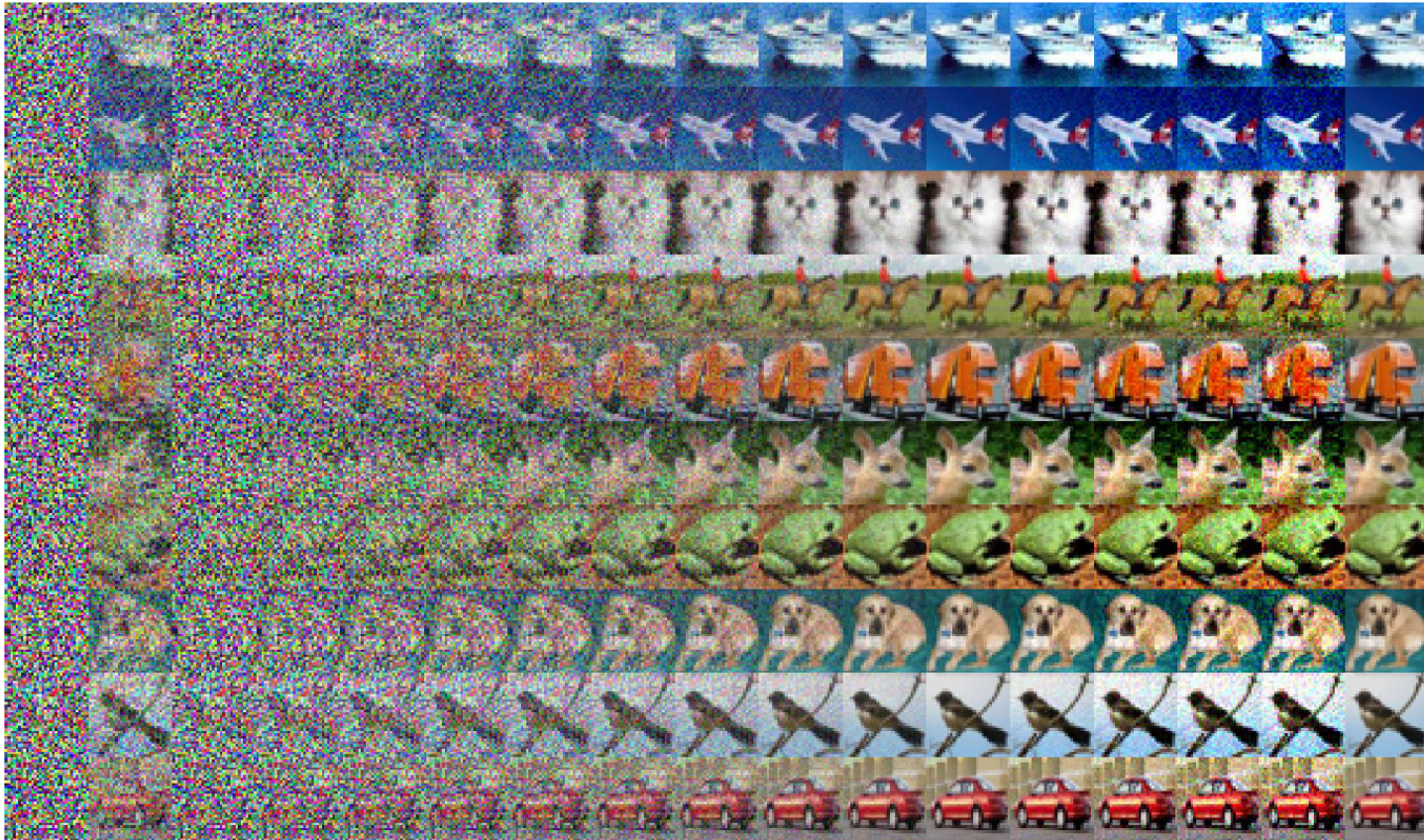
$$\min_{f \in \mathcal{F}} J_D \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\phi(f(g(z)))] + \mathbb{E}_{x \sim \mathcal{P}_r} [\varphi(f(x))] + \lambda k(f)^2$$

$$\min_{g \in \mathcal{G}} J_G \triangleq \mathbb{E}_{z \sim \mathcal{P}_z} [\psi(f(g(z)))]$$

- Theoretically guaranteed properties:
 - The optimal discriminative function f^* exists;
 - If ϕ is strictly convex, then f^* is unique;
 - There exists a unique Nash equilibrium where $P_r = P_g$ and $k(f^*) = 0$;
 - Do not suffer from gradient uninformativeness;
 - For each generated sample, the gradient directly points towards a real sample.

Lipschitz GANs

- $\frac{\partial f^*(x)}{\partial x}$ directly points towards real samples.



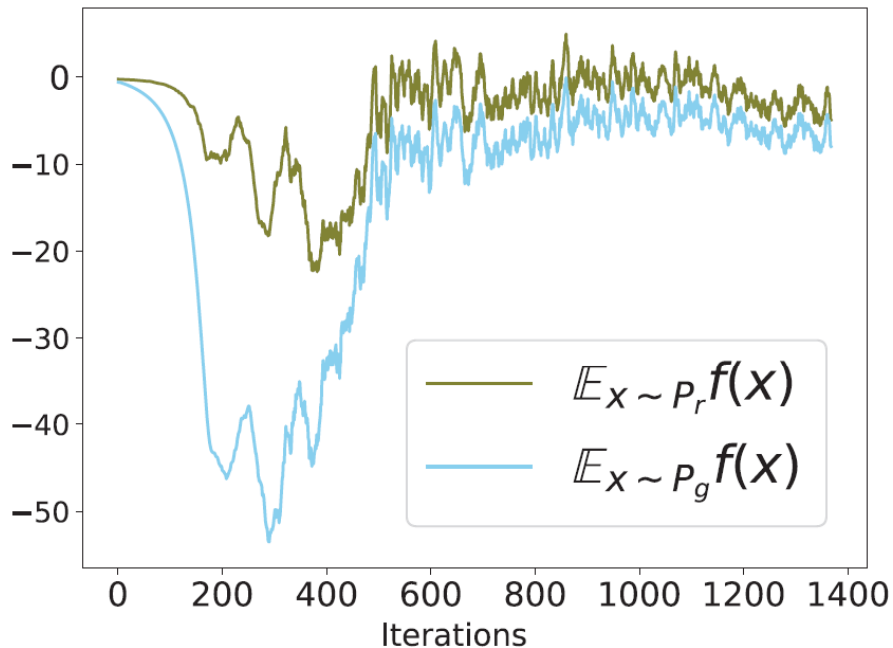
$$x \quad \frac{\partial f^*(x)}{\partial x}$$

$$x + \epsilon \frac{\partial f^*(x)}{\partial x}$$

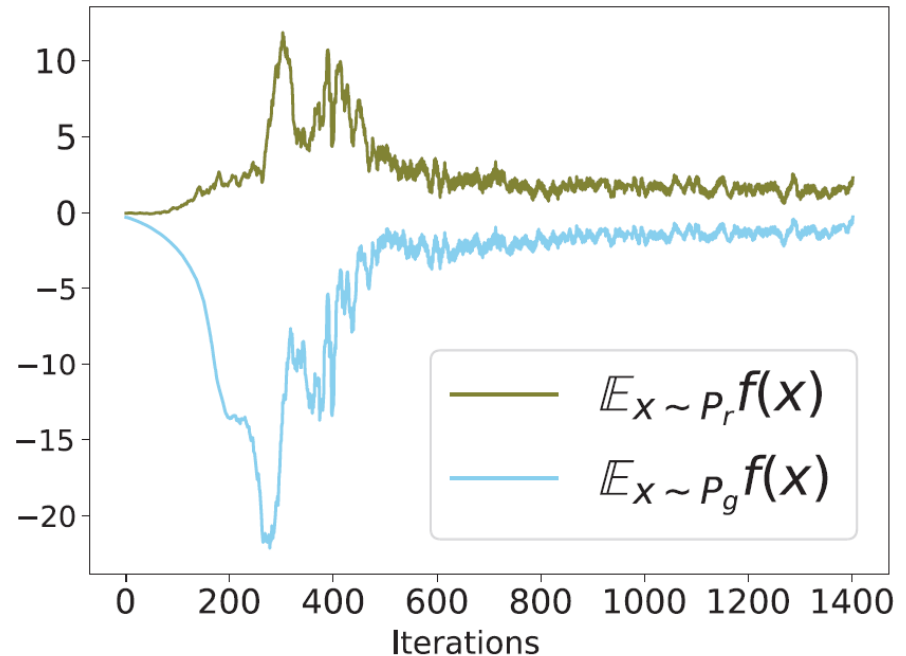
$$y \in \mathcal{S}_r$$

Lipschitz GANs

- The uniqueness of $f^*(x)$ leads to stabilized discriminative functions.



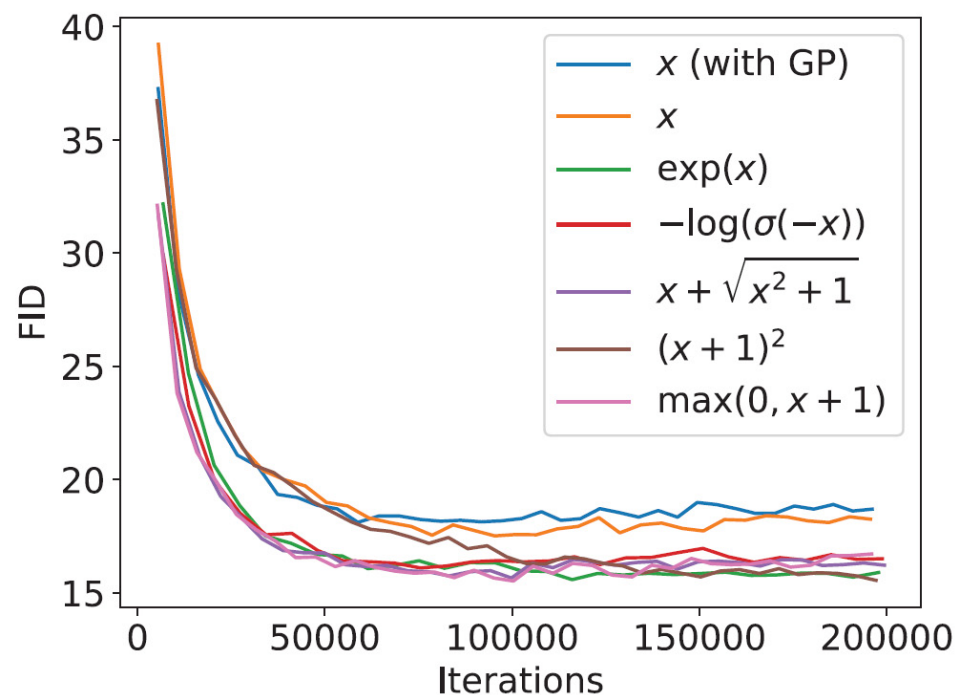
(a) $f(x)$ in WGAN.



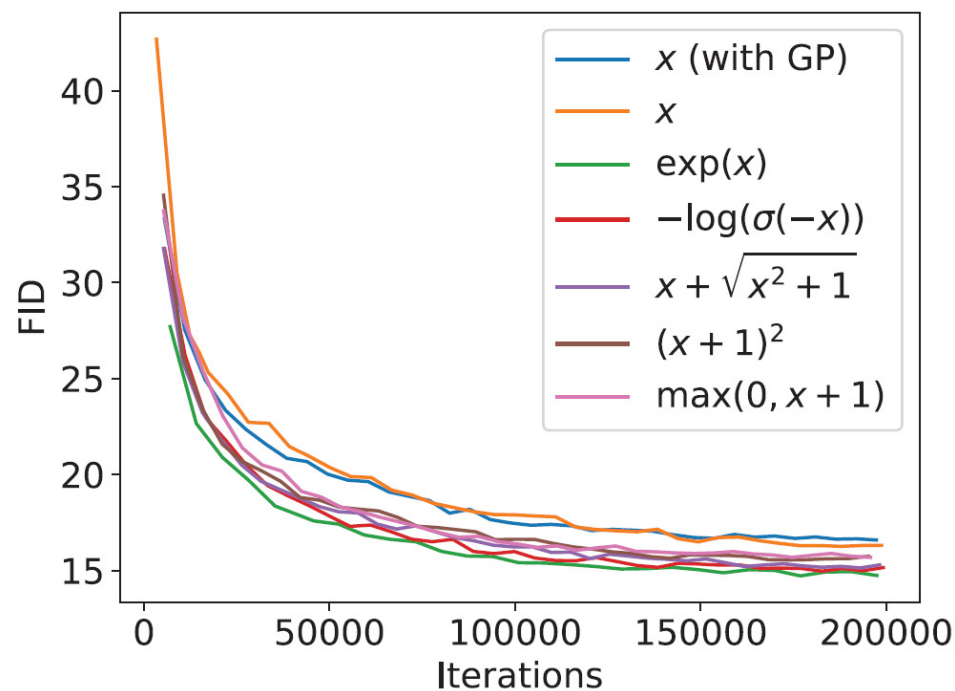
(b) $f(x)$ in LGANs.

Lipschitz GANs

- LGANs, with different choices of $\phi(x)$ consistently outperform WGAN.



(a) Training curves on CIFAR.



(b) Training curves on Tiny.

FID: Frechet Inception Distance

Content

1. Introduction to Generative Adversarial Nets
2. GANs on Continuous Data
3. GAN and RL
4. GANs on Discrete Data
5. Summary

REVIEW

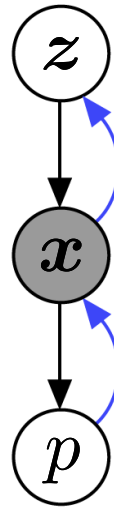
GANs for Continuous Data

1. Generation

$$x = G(z; \theta)$$

2. Discrimination

$$P(\text{real}|x) = D(x; \phi)$$



4. Further gradient on generator

$$\frac{\partial J(G, D)}{\partial x} \frac{\partial x}{\partial \theta}$$

3. Gradient on generated data

$$\frac{\partial J(G, D)}{\partial x}$$

- In order to take gradient on the generator parameter, x has to be continuous

$$J(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$\text{Generator } \min_G \max_D J(G, D) \quad \text{Discriminator } \max_D J(G, D)$$

Such GANs Fail on Discrete Data

- Single token

- Information retrieval



$$p(\text{doc}_n | \text{query}; \theta)$$

- Sequence

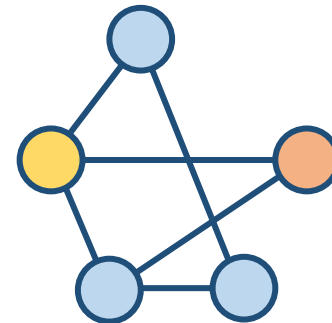
- Text
 - Music score
 - DNA/RNA pieces
 - ...



$$p(\text{word}_n | \text{word}_{1\dots n-1}; \theta)$$

- Graph

- Social network
 - User-item shopping behavior
 - Paper citations
 - ...



$$p(\text{node}_n | \text{node}_m, \text{neighbor}(m); \theta)$$

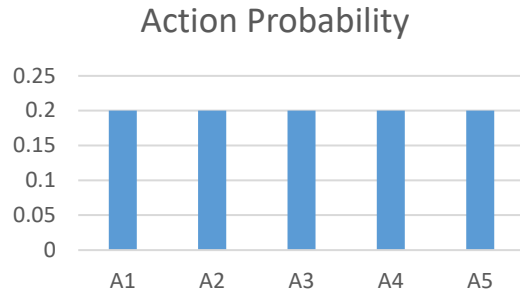
How to Optimize on Discrete Output

- Problem: we cannot take gradient on discrete data
- Borrow the idea from classification
 - Build a parameterized distribution on the discrete tokens
 - Optimize such a parameterized distribution with gradient
 - But GAN normally outputs an instance instead of a distribution
- Borrow the idea from reinforcement learning
 - Optimize the parameterized stochastic policy to higher/lower the probability of action that leads to positive/negative reward

Policy Gradient in RL

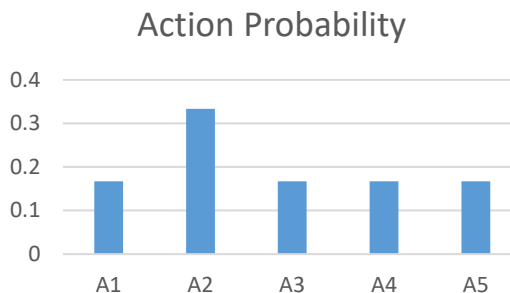
- For stochastic policy $\pi_{\theta}(a|s) = P(a|s; \theta)$
- Intuition
 - lower the probability of the action that leads to low value/reward
 - higher the probability of the action that leads to high value/reward
- A 5-action example

1. Initialize ϑ



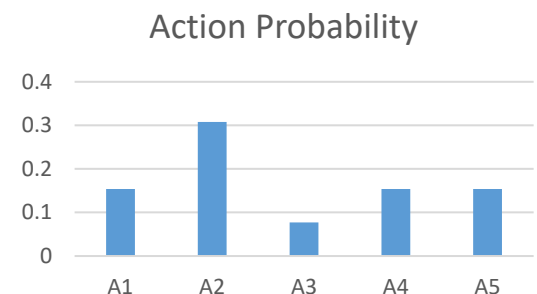
2. Take action A2
Observe positive reward

3. Update ϑ by policy gradient



4. Take action A3
Observe negative reward

5. Update ϑ by policy gradient



Policy Gradient in One-Step MDPs

- Consider a simple class of one-step MDPs
 - Starting in state $s \sim d(s)$
 - Terminating after one time-step with reward r_{sa}
- Policy expected value

$$J(\theta) = \mathbb{E}_{\pi_{\theta}}[r] = \sum_{s \in S} d(s) \sum_{a \in A} \pi_{\theta}(a|s) r_{sa}$$

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{s \in S} d(s) \sum_{a \in A} \frac{\partial \pi_{\theta}(a|s)}{\partial \theta} r_{sa}$$

Likelihood Ratio

- Likelihood ratios exploit the following identity

$$\begin{aligned}\frac{\partial \pi_{\theta}(a|s)}{\partial \theta} &= \pi_{\theta}(a|s) \frac{1}{\pi_{\theta}(a|s)} \frac{\partial \pi_{\theta}(a|s)}{\partial \theta} \\ &= \pi_{\theta}(a|s) \frac{\partial \log \pi_{\theta}(a|s)}{\partial \theta}\end{aligned}$$

- Thus the policy's expected value

$$\begin{aligned}J(\theta) &= \mathbb{E}_{\pi_{\theta}}[r] = \sum_{s \in S} d(s) \sum_{a \in A} \pi_{\theta}(a|s) r_{sa} \\ \frac{\partial J(\theta)}{\partial \theta} &= \sum_{s \in S} d(s) \sum_{a \in A} \frac{\partial \pi_{\theta}(a|s)}{\partial \theta} r_{sa} \\ &= \sum_{s \in S} d(s) \sum_{a \in A} \pi_{\theta}(a|s) \frac{\partial \log \pi_{\theta}(a|s)}{\partial \theta} r_{sa} \\ &= \mathbb{E}_{\pi_{\theta}} \left[\frac{\partial \log \pi_{\theta}(a|s)}{\partial \theta} r_{sa} \right] \quad \text{This can be approximated by sampling} \\ &\quad \text{state } s \text{ from } d(s) \text{ and action } a \text{ from } \pi_{\theta}\end{aligned}$$

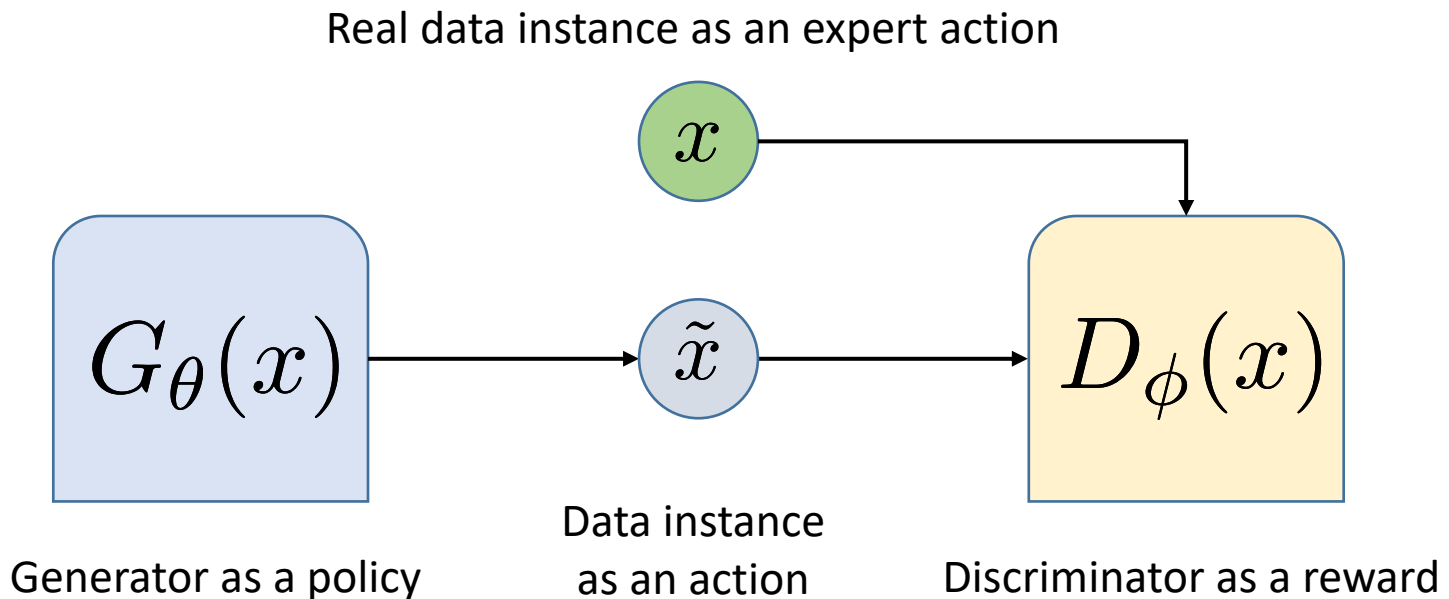
Policy Gradient Theorem

- The policy gradient theorem generalizes the likelihood ratio approach to multi-step MDPs
 - Replaces instantaneous reward r_{sa} with long-term value $Q^{\pi_\theta}(s, a)$
- Policy gradient theorem applies to
 - start state objective J_1 , average reward objective J_{avR} , and average value objective J_{avV}
- Theorem
 - For any differentiable policy $\pi_\theta(a|s)$, for any of policy objective function $J = J_1, J_{avR}, J_{avV}$, the policy gradient is

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta} \left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s, a) \right]$$

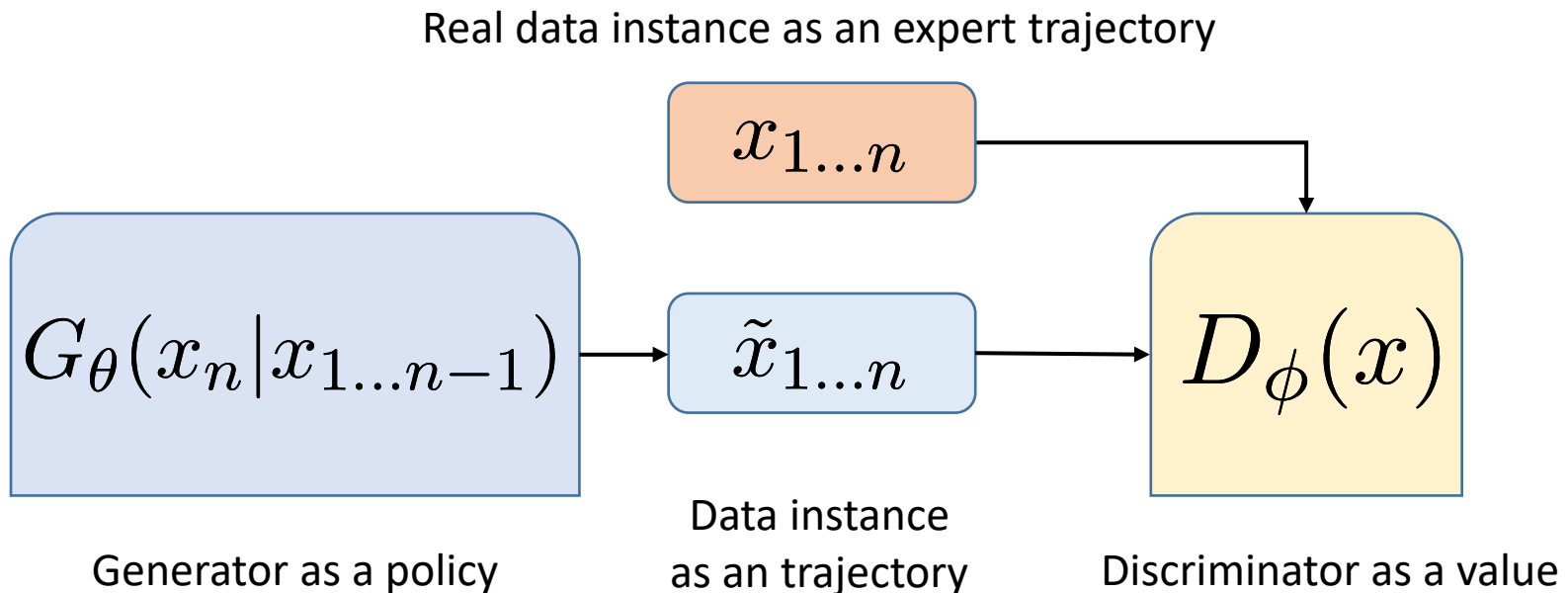
Connection between GAN and IL

- Analogy to **Imitation learning**
 - In imitation learning, a value function is learned from expert data to guide the policy optimization
 - In GAN, a discriminator is trained with real (positive) data and generated (negative) data to guide the generator optimization
- One step generation: stateless or one-step MDP



Connection between GAN and IL

- Analogy to **Imitation learning**
 - In imitation learning, a value function is learned from expert data to guide the policy optimization
 - In GAN, a discriminator is trained with real (positive) data and generated (negative) data to guide the generator optimization
- Multi-step generation: MDP



GAN and RL on Learning Rules

For continuous data/action

- Deterministic policy gradient (DPG)
$$\frac{\partial J(\pi_\theta)}{\partial \theta} = \mathbb{E}_{s \sim \rho^\pi} \left[\frac{\partial Q^\pi(s, a)}{\partial a} \frac{\partial \pi_\theta(s)}{\partial \theta} \Big|_{a=\pi_\theta(s)} \right]$$
- GAN for continuous data
$$\frac{\partial J(G_\theta, D)}{\partial \theta} = \mathbb{E}_{z \sim p(z)} \left[\frac{\partial J(G_\theta, D(x))}{\partial x} \frac{\partial G_\theta(z)}{\partial \theta} \Big|_{x=G_\theta(z)} \right]$$

For discrete data/action

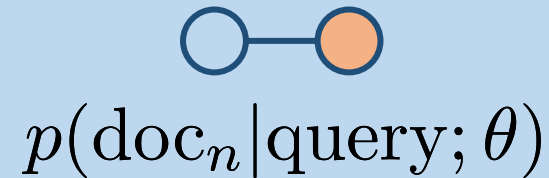
- Stochastic policy gradient (PG)
$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta} \left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s, a) \right]$$
- GAN for discrete data
$$\frac{\partial J(G_\theta, D)}{\partial \theta} = \mathbb{E}_{x \sim G_\theta} \left[\frac{\partial \log G_\theta(x)}{\partial \theta} D(x) \right]$$

Content

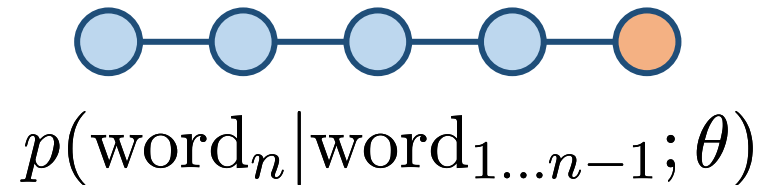
1. Introduction to Generative Adversarial Nets
2. GANs on Continuous Data
3. GAN and RL
4. GANs on Discrete Data
5. Summary

GANs on Discrete Data

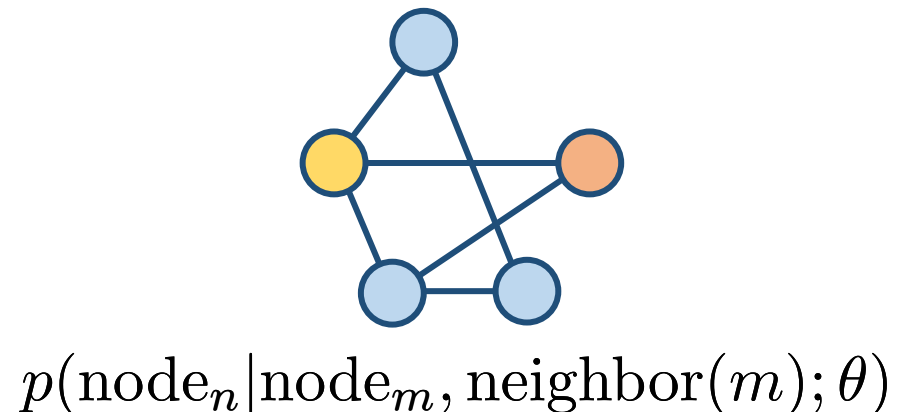
- Single token
 - Information retrieval



- Sequence
 - Text
 - Music score
 - DNA/RNA pieces
 - ...



- Graph
 - Social network
 - User-item shopping behavior
 - Paper citations
 - ...

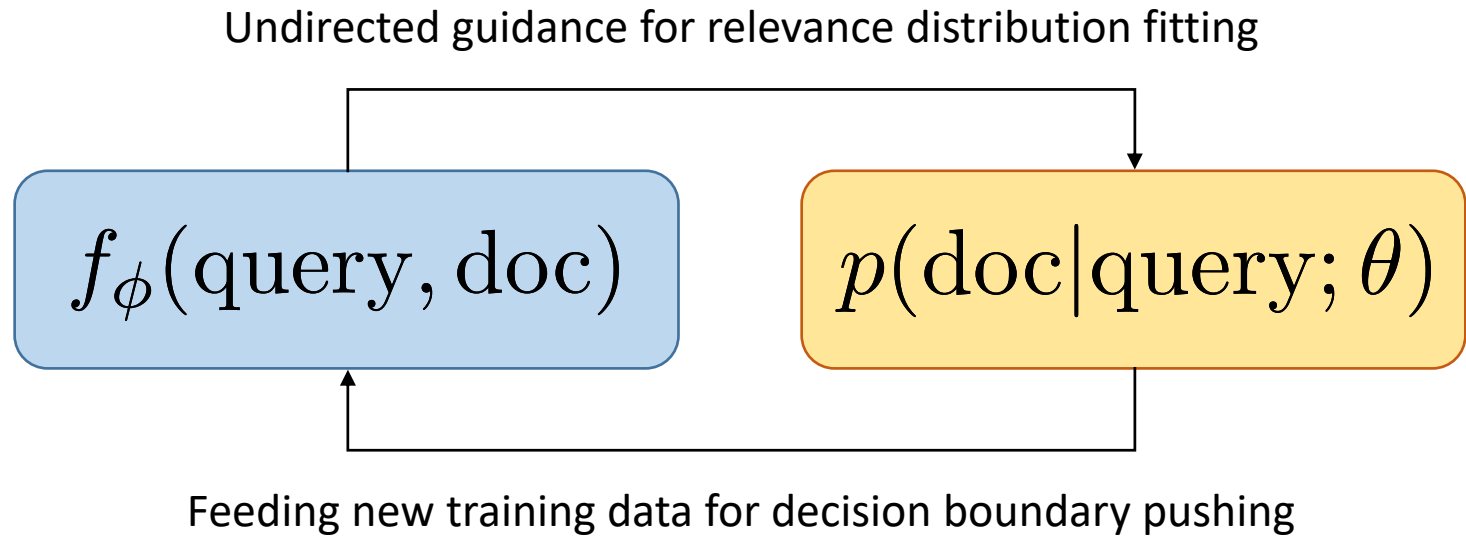


IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu,
Benyou Wang, Peng Zhang and Dell Zhang. SIGIR 2017.

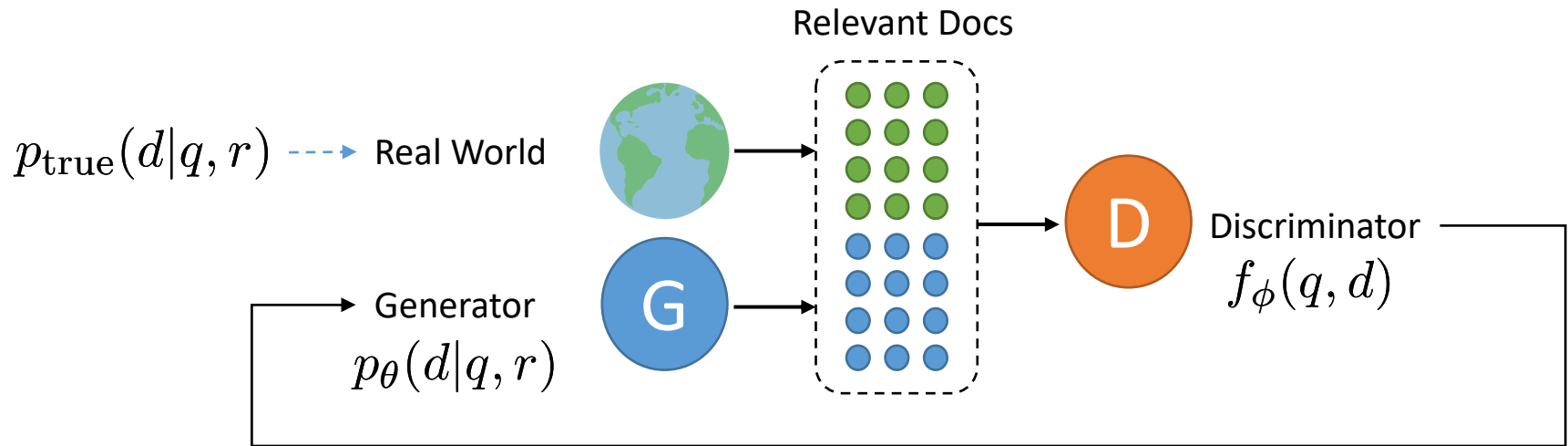
<https://arxiv.org/abs/1705.10513>

A Two Agent Framework for IR



- Deep discriminative models
 - Flexible to fit complex relevance ranking & scoring
 - Obtaining training data (negative cases) from the generative model
- Deep generative models
 - Flexible to fit complex relevance distribution
 - Trainable
 - Guided from the discriminative model

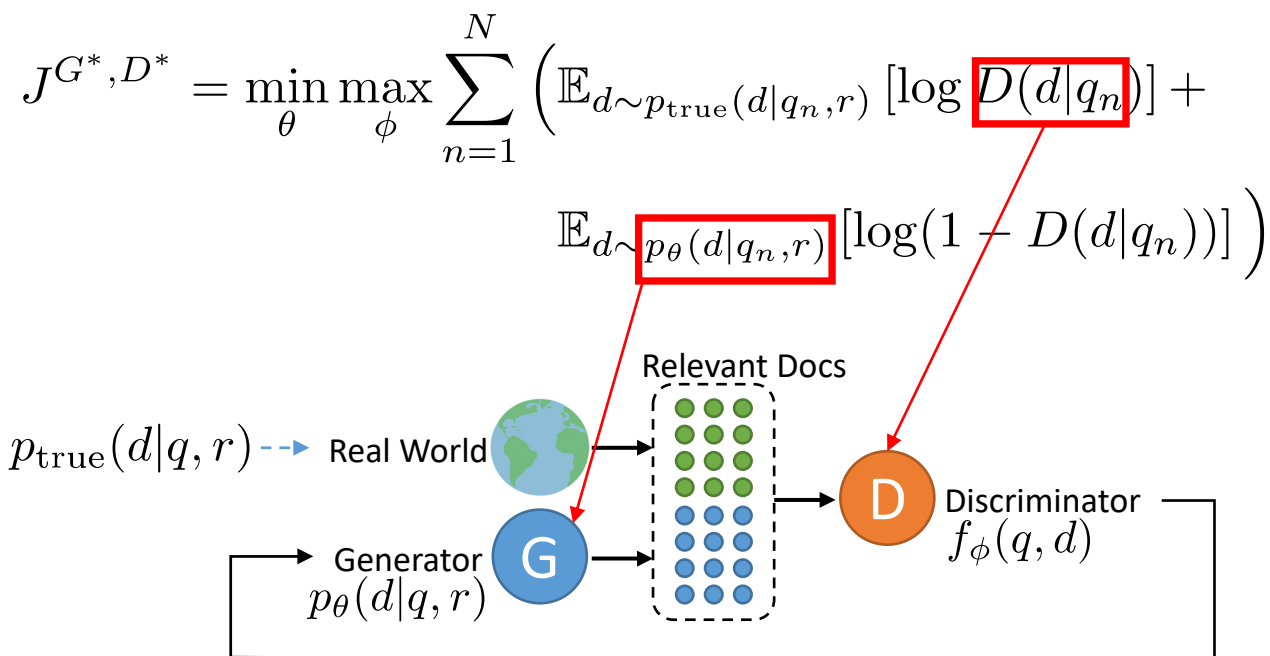
IRGAN Formulation



- **Underlying real relevance distribution** $p_{\text{true}}(d|q, r)$ depicts the user's relevance preference distribution over the candidate documents with respect to his submitted query
 - **Training set:** A set of samples from $p_{\text{true}}(d|q, r)$
- **Generative retrieval model** $p_{\theta}(d|q, r)$
 - Goal: approximate the real relevance distribution
- **Discriminative retrieval model** $f_{\phi}(q, d)$
 - Goal: distinguish between relevant documents and non-relevant documents

A Minimax Game Unifying Both Models

- Objective



where

$$p_{\theta}(d|q, r) = \frac{\exp(g_{\theta}(q, d))}{\sum_{d'} \exp(g_{\theta}(q, d'))}$$

$$D(d|q) = \sigma(f_{\phi}(d, q)) = \frac{\exp(f_{\phi}(d, q))}{1 + \exp(f_{\phi}(d, q))}$$

Optimizing Generative Retrieval via Policy Gradient

- Optimizing Generative Retrieval
 - Samples documents from the whole document set to fool its opponent

$$\begin{aligned}\theta^* &= \arg \min_{\theta} \sum_{n=1}^N \left(\mathbb{E}_{d \sim p_{\text{true}}(d|q_n, r)} [\log \sigma(f_{\phi}(d, q_n))] + \right. \\ &\quad \left. \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\log(1 - \sigma(f_{\phi}(d, q_n)))] \right) \\ &= \arg \max_{\theta} \sum_{n=1}^N \underbrace{\mathbb{E}_{d \sim p_{\theta}(d|q_n, r)}}_{\text{Generator as Policy}} \underbrace{[\log(1 + \exp(f_{\phi}(d, q_n)))]}_{\text{denoted as } J^G(q_n) \text{ Reward Term}}\end{aligned}$$

- REINFORCE (with advantage function)

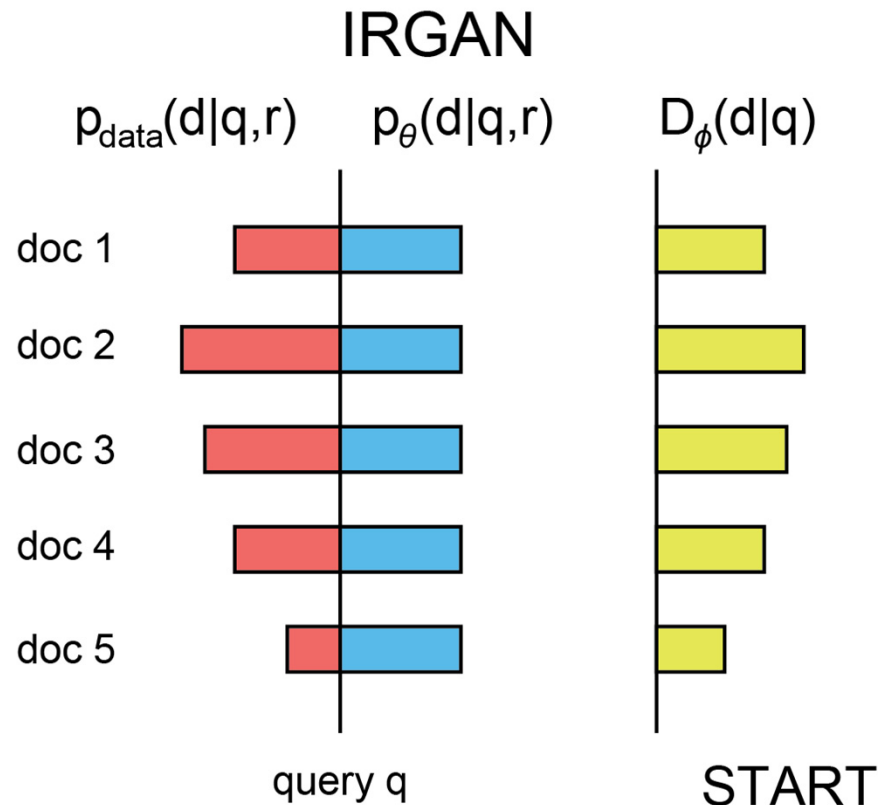
$$\log(1 + \exp(f_{\phi}(d, q_n))) - \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\log(1 + \exp(f_{\phi}(d, q_n)))]$$

IRGAN REINFORCE

- Likelihood ratio

$$\begin{aligned} & \nabla_{\theta} J^G(q_n) \\ &= \nabla_{\theta} \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\log(1 + \exp(f_{\phi}(d, q_n)))] \\ &= \sum_{i=1}^M \nabla_{\theta} p_{\theta}(d_i|q_n, r) \log(1 + \exp(f_{\phi}(d_i, q_n))) \\ &= \sum_{i=1}^M p_{\theta}(d_i|q_n, r) \nabla_{\theta} \log p_{\theta}(d_i|q_n, r) \log(1 + \exp(f_{\phi}(d_i, q_n))) \\ &= \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\nabla_{\theta} \log p_{\theta}(d|q_n, r) \log(1 + \exp(f_{\phi}(d, q_n)))] \\ &\simeq \frac{1}{K} \sum_{k=1}^K \nabla_{\theta} \log p_{\theta}(d_k|q_n, r) \log(1 + \exp(f_{\phi}(d_k, q_n))) \end{aligned}$$

The Interplay between Generative and Discriminative Retrieval



$$D_{\phi}(d|q) = \frac{p_{\text{data}}(d|q, r)}{p_{\text{data}}(d|q, r) + p_{\theta}(d|q, r)}$$

Extension to Pairwise Case

- It is common that the dataset is a set of ordered document pairs for each query rather than a set of relevant documents.

- Capture relative preference judgements

$$R_n = \{ \langle d_i, d_j \rangle \mid d_i \succ d_j \}$$

rather than absolute relevance judgements

- Generator would try to generate document pairs that are similar to those in R_n , i.e., with the correct ranking.

Experiments: Web Search

- Dataset
 - MQ-2008 (Million-query Track in LETOR 4.0)
 - Semi-supervised learning: a large amount of unlabeled query-document pairs
- Task
 - Rank the candidate documents for each query

Table 1: Webpage ranking performance comparison on MQ2008-semi dataset, where * means significant improvement in a Wilcoxon signed-rank test.

	P@3	P@5	P@10	MAP
MLE	0.1556	0.1295	0.1029	0.1604
RankNet [3]	0.1619	0.1219	0.1010	0.1517
LambdaRank [5]	0.1651	0.1352	0.1076	0.1658
LambdaMART [4]	0.1368	0.1026	0.0846	0.1288
IRGAN-pointwise	0.1714	0.1657	0.1257	0.1915
IRGAN-pairwise	0.2000	0.1676	0.1248	0.1816
Impv-pointwise	3.82%	22.56%*	16.82%*	15.50%*
Impv-pairwise	21.14%*	23.96%*	15.98%	9.53%
	NDCG@3	NDCG@5	NDCG@10	MRR
MLE	0.1893	0.1854	0.2054	0.3194
RankNet [3]	0.1801	0.1709	0.1943	0.3062
LambdaRank [5]	0.1926	0.1920	0.2093	0.3242
LambdaMART [4]	0.1573	0.1456	0.1627	0.2696
IRGAN-pointwise	0.2065	0.2225	0.2483	0.3508
IRGAN-pairwise	0.2148	0.2154	0.2380	0.3322
Impv-pointwise	7.22%	15.89%	18.63%	8.20%
Impv-pairwise	11.53%	12.19%	13.71%	2.47%

Experiments: Item Recommendation

IRGAN-pointwise Generator Performance on Movielens

	P@3	P@5	P@10	MAP
MLE	0.3369	0.3013	0.2559	0.2005
BPR [35]	0.3289	0.3044	0.2656	0.2009
LambdaFM [45]	0.3845	0.3474	0.2967	0.2222
IRGAN-pointwise	0.4072	0.3750	0.3140	0.2418
Impv-pointwise	5.90%*	7.94%*	5.83%*	8.82%*
	NDCG@3	NDCG@5	NDCG@10	MRR
MLE	0.3461	0.3236	0.3017	0.5264
BPR [35]	0.3410	0.3245	0.3076	0.5290
LambdaFM [45]	0.3986	0.3749	0.3518	0.5797
IRGAN-pointwise	0.4222	0.4009	0.3723	0.6082
Impv-pointwise	5.92%*	6.94%*	5.83%*	4.92%*

IRGAN-pointwise Generator Performance on Netflix

	P@3	P@5	P@10	MAP
MLE	0.2941	0.2945	0.2777	0.0957
BPR [35]	0.3040	0.2933	0.2774	0.0935
LambdaFM [45]	0.3901	0.3790	0.3489	0.1672
IRGAN-pointwise	0.4456	0.4335	0.3923	0.1720
Impv-pointwise	14.23%*	14.38%*	12.44%*	2.87%*
	NDCG@3	NDCG@5	NDCG@10	MRR
MLE	0.3032	0.3011	0.2878	0.5085
BPR [35]	0.3077	0.2993	0.2866	0.5040
LambdaFM [45]	0.3942	0.3854	0.3624	0.5857
IRGAN-pointwise	0.4498	0.4404	0.4097	0.6371
Impv-pointwise	14.10%*	14.27%*	13.05%*	8.78%*

- Datasets

- Movielens: 943 users, 1.7k items, 100k ratings
- Netflix: 480k users, 17k items, 100M ratings

- Task: Top-N item recommendation with implicit feedback data

- Key observations

- Although generative retrieval model in IRGAN does not explicitly learn to optimize the final ranking measures like what LambdaFM does, it still performs consistently better than LambdaFM.

GANs on Discrete Data

- Single token
 - Information retrieval



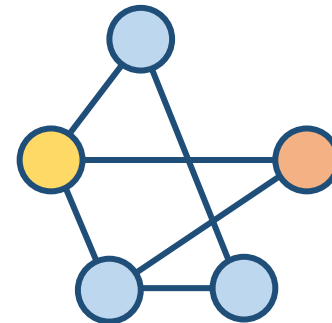
$$p(\text{doc}_n | \text{query}; \theta)$$

- Sequence
 - Text
 - Music score
 - DNA/RNA pieces
 - ...



$$p(\text{word}_n | \text{word}_{1\dots n-1}; \theta)$$

- Graph
 - Social network
 - User-item shopping behavior
 - Paper citations
 - ...

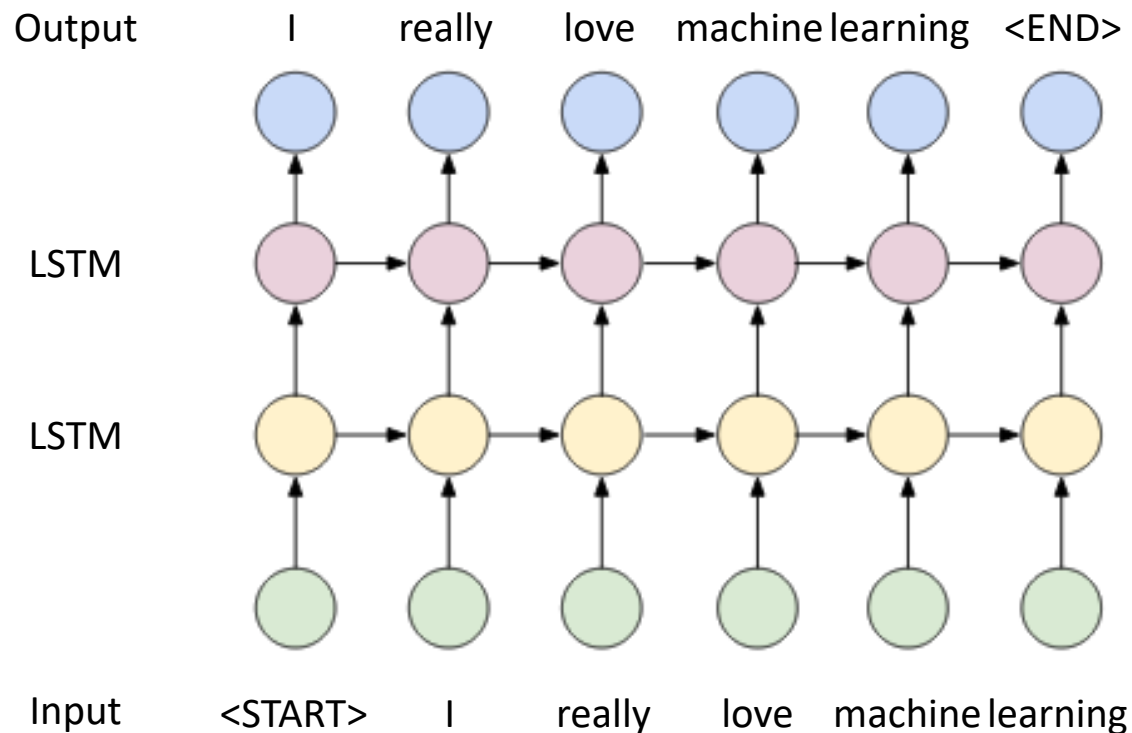


$$p(\text{node}_n | \text{node}_m, \text{neighbor}(m); \theta)$$

RNN based Language Model

- Trained via maximum likelihood estimation (MLE)

$$\max_{\theta} \mathbb{E}_{x \sim p(x)} [\log q_{\theta}(x)]$$



Exposure Bias

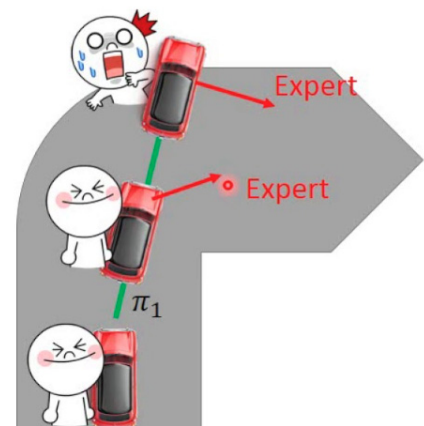
- Exposure bias
 - In MLE, the prefix is always from the real data

$$p(\text{learning} | \text{I really love machine})$$

- But during generation, the prefix is the output of the model, which could never occur in real data

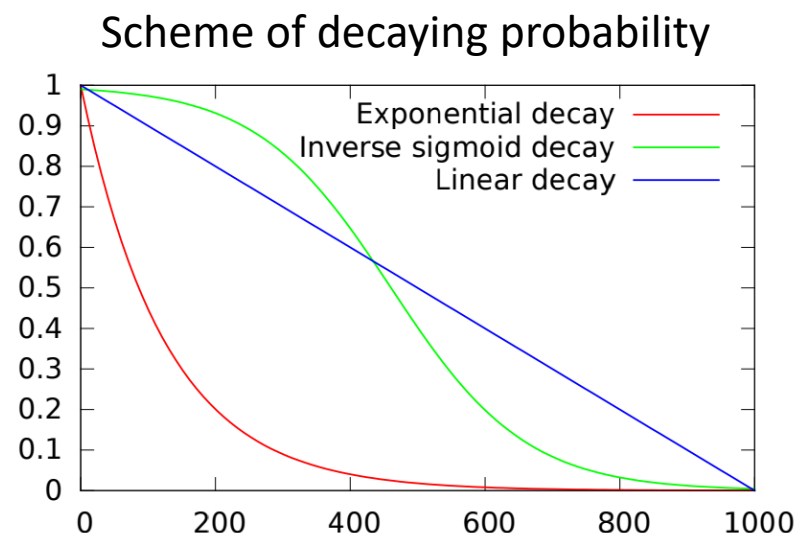
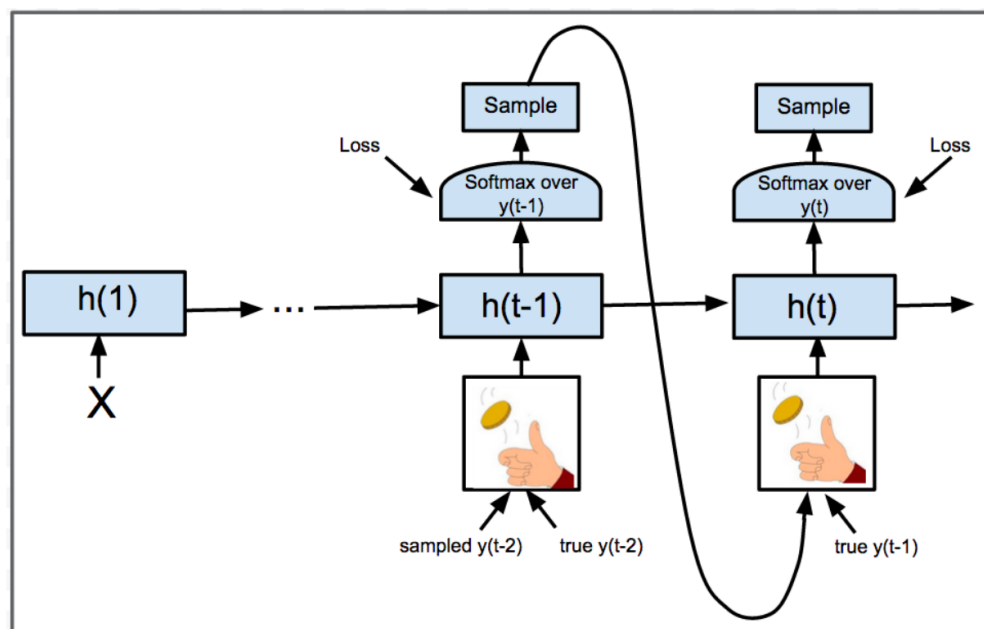
$$p(? | \text{I machine love really})$$

- Similar in self-driving car training
 - Problem of behavior cloning



Exposure Bias & Scheduled Sampling

- Scheduled sampling
 - With a decaying probability, use the prefix from real data, otherwise use the generated prefix to train



Inconsistency of Evaluation and Use

- Given a generator q with a certain generalization ability

$$\max_{\theta} \mathbb{E}_{x \sim p(x)} [\log q_{\theta}(x)]$$

Training/evaluation

- Check whether a real data is with a high mass density of the learned model
- Approximated by

$$\max_{\theta} \frac{1}{|D|} \sum_{x \in D} [\log q_{\theta}(x)]$$

$$\max_{\theta} \mathbb{E}_{x \sim q_{\theta}(x)} [\log p(x)]$$

Use

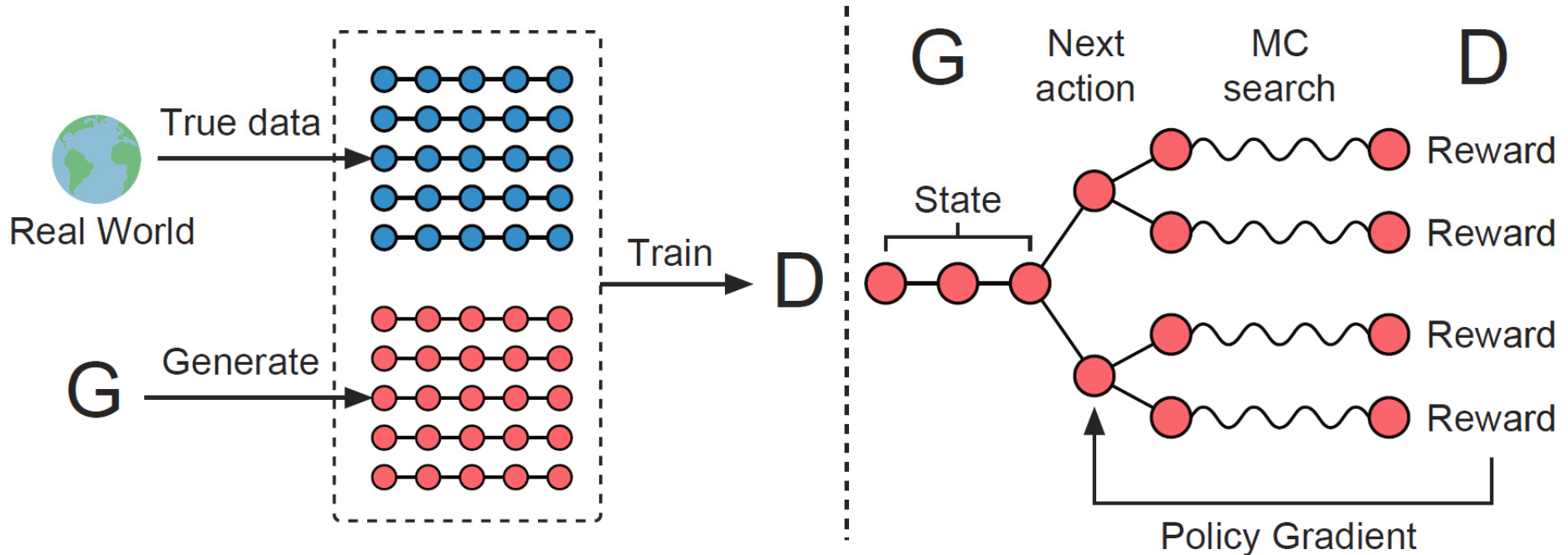
- Check whether a model-generated data is considered as real as possible
- More straightforward but it is hard or impossible to directly calculate $p(x)$

SeqGAN: Sequence Generation via GANs with Policy Gradient

Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. AAAI 2017.

<https://arxiv.org/abs/1609.05473>

SeqGAN



- Generator is a reinforcement learning policy $G_{\theta}(y_t|Y_{1:t-1})$ of generating a sequence
 - decide the next word to generate given the previous ones
- Discriminator provides the reward (i.e. the probability of being real data) $D_{\phi}(Y_{1:T}^n)$ for the whole sequence

Sequence Generator

- Objective: to maximize the expected reward

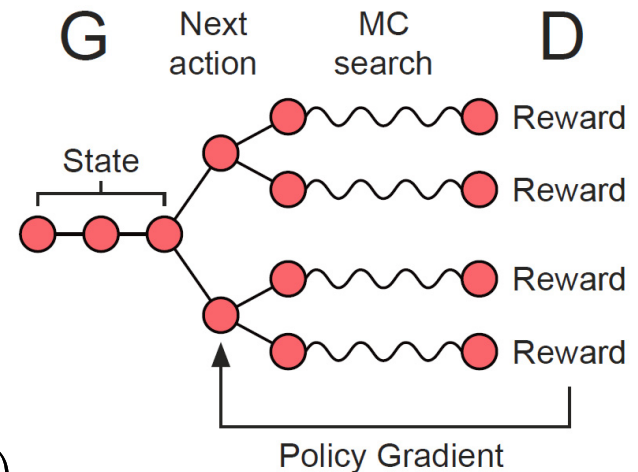
$$J(\theta) = \mathbb{E}_{Y_{1:t-1} \sim G_\theta} \left[\sum_{y_t \in \mathcal{Y}} G_\theta(y_t | Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t) \right]$$

- State-action value function $Q_{D_\phi}^{G_\theta}(s, a)$ is the expected accumulative reward that

- Start from state s
- Take action a
- And follow policy G until the end

- Reward is only on completed sequence (no immediate reward)

$$Q_{D_\phi}^{G_\theta}(a = y_T, s = Y_{1:T-1}) = D_\phi(Y_{1:T})$$



State-Action Value Setting

- Reward is only on completed sequence
 - No immediate reward
 - Then the last-step state-action value

$$Q_{D_\phi}^{G_\theta}(a = y_T, s = Y_{1:T-1}) = D_\phi(Y_{1:T})$$

- For intermediate state-action value

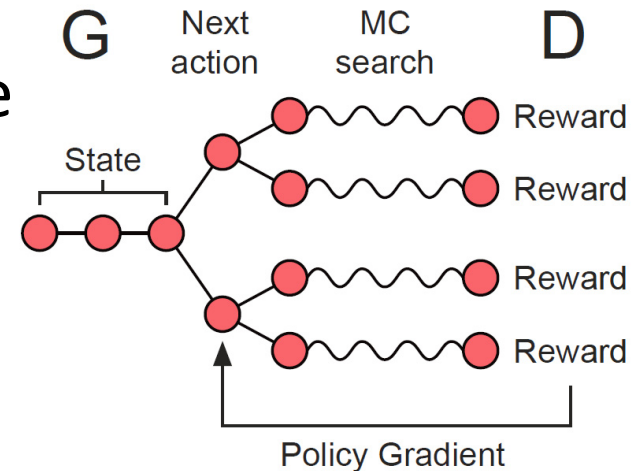
- Use Monte Carlo search to estimate

$$\{Y_{1:T}^1, \dots, Y_{1:T}^N\} = \text{MC}^{G_\theta}(Y_{1:t}; N)$$

- Following a roll-out policy G_θ

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) =$$

$$\begin{cases} \frac{1}{N} \sum_{n=1}^N D_\phi(Y_{1:T}^n), & Y_{1:T}^n \in \text{MC}^{G_\theta}(Y_{1:t}; N) & \text{for } t < T \\ D_\phi(Y_{1:t}) & & \text{for } t = T \end{cases}$$



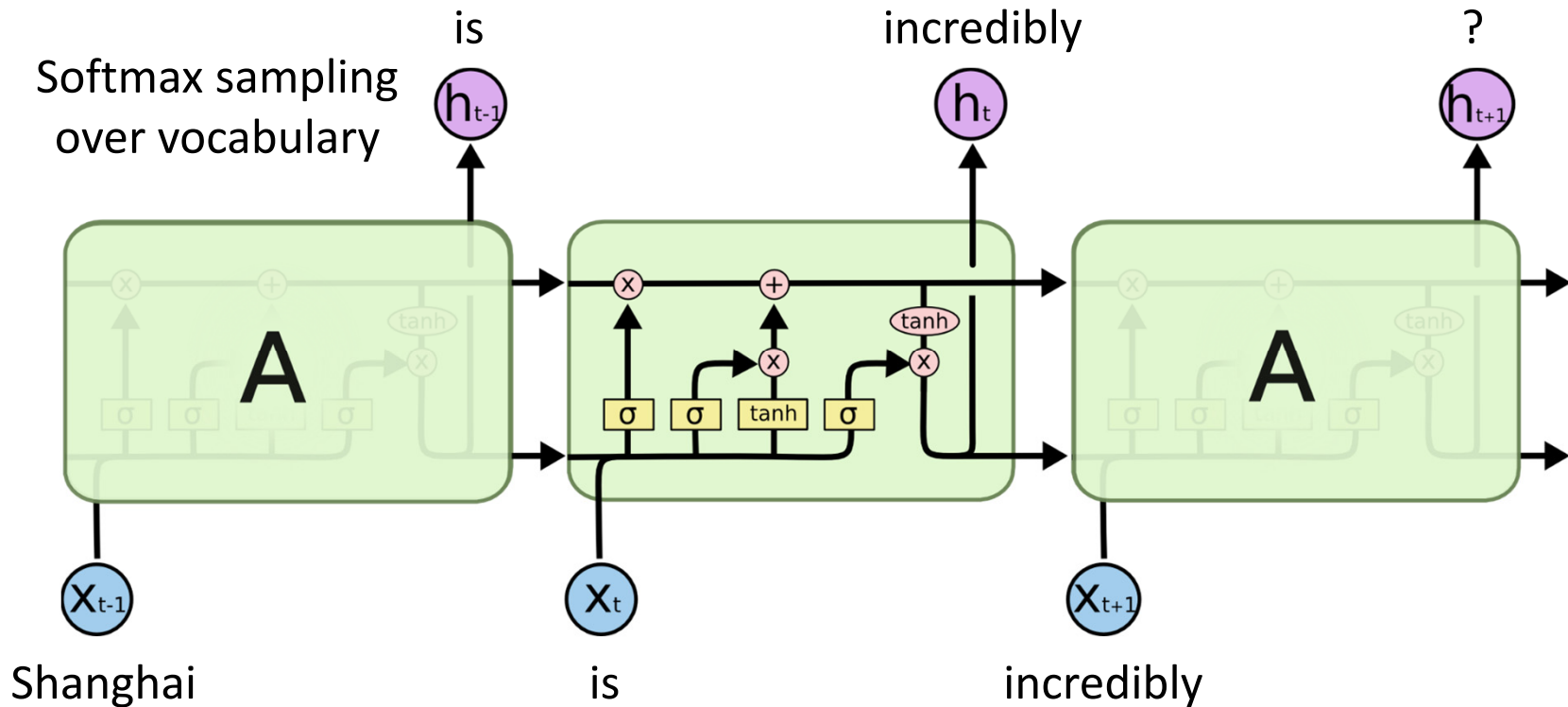
Training Sequence Generator

- Policy gradient (REINFORCE)

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{Y_{1:t-1} \sim G_{\theta}} \left[\sum_{y_t \in \mathcal{Y}} \nabla_{\theta} G_{\theta}(y_t | Y_{1:t-1}) \cdot Q_{D_{\phi}}^{G_{\theta}}(Y_{1:t-1}, y_t) \right] \\ &\simeq \frac{1}{T} \sum_{t=1}^T \sum_{y_t \in \mathcal{Y}} \nabla_{\theta} G_{\theta}(y_t | Y_{1:t-1}) \cdot Q_{D_{\phi}}^{G_{\theta}}(Y_{1:t-1}, y_t) \\ &= \frac{1}{T} \sum_{t=1}^T \sum_{y_t \in \mathcal{Y}} G_{\theta}(y_t | Y_{1:t-1}) \nabla_{\theta} \log G_{\theta}(y_t | Y_{1:t-1}) \cdot Q_{D_{\phi}}^{G_{\theta}}(Y_{1:t-1}, y_t) \\ &= \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{y_t \sim G_{\theta}(y_t | Y_{1:t-1})} [\nabla_{\theta} \log G_{\theta}(y_t | Y_{1:t-1}) \cdot Q_{D_{\phi}}^{G_{\theta}}(Y_{1:t-1}, y_t)]\end{aligned}$$

$$\theta \leftarrow \theta + \alpha_h \nabla_{\theta} J(\theta)$$

Sequence Generator Model



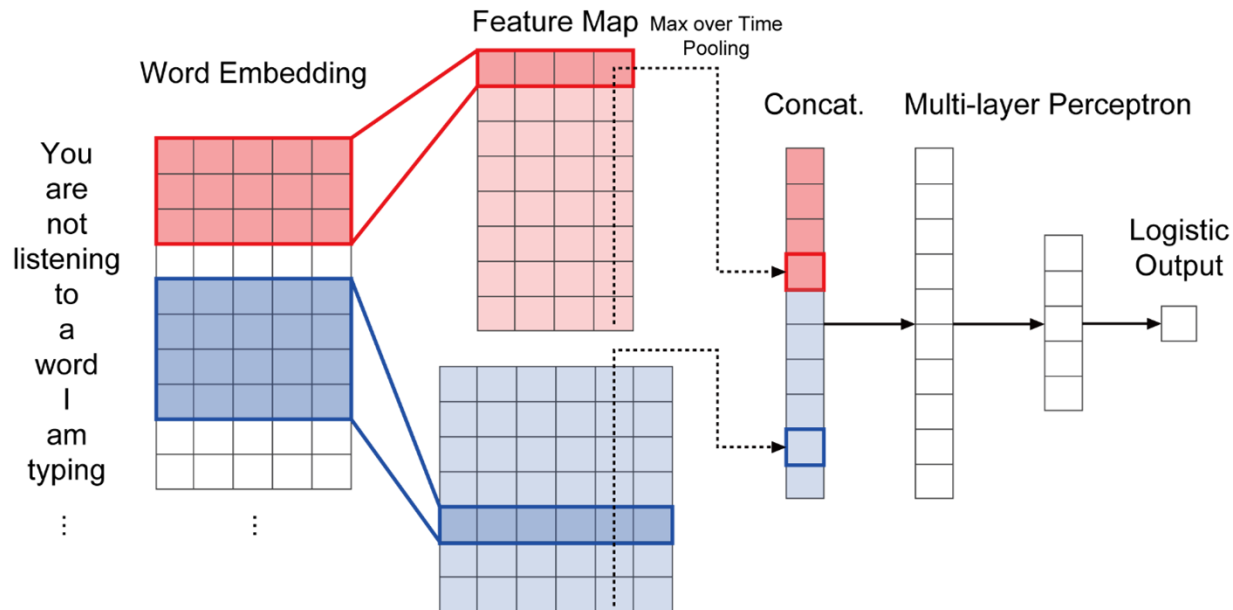
- RNN with LSTM cells for $G_{\theta}(y_t|Y_{1:t-1})$

Training Sequence Discriminator

- Objective: standard binary classification

$$\min_{\phi} -\mathbb{E}_{Y \sim p_{\text{data}}} [\log D_{\phi}(Y)] - \mathbb{E}_{Y \sim G_{\theta}} [\log(1 - D_{\phi}(Y))]$$

- A CNN implementation

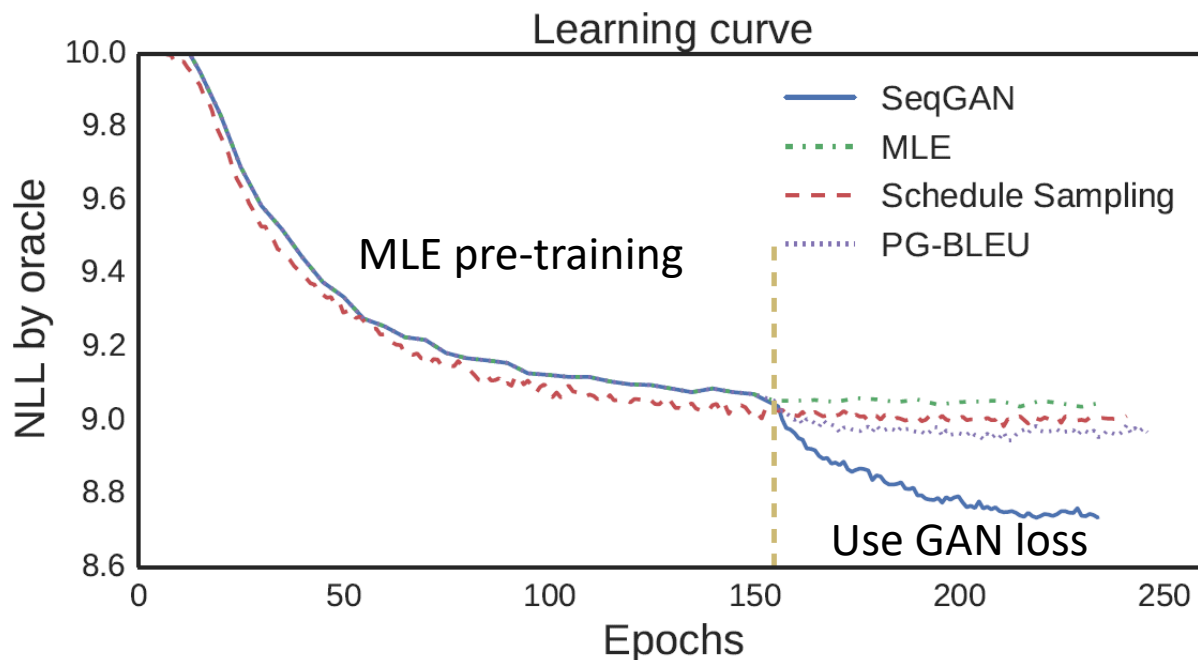


Experiments on Synthetic Data

- Evaluation measure with Oracle $\max_{\theta} \mathbb{E}_{x \sim q_{\theta}(x)} [\log p(x)]$

$$\text{NLL}_{\text{oracle}} = -\mathbb{E}_{Y_{1:T} \sim G_{\theta}} \left[\sum_{t=1}^T \log G_{\text{oracle}}(y_t | Y_{1:t-1}) \right]$$

Algorithm	Random	MLE	SS	PG-BLEU	SeqGAN
NLL	10.310	9.038	8.985	8.946	8.736
p -value	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	



Experiments on Real-World Data

- Chinese poem generation

Algorithm	Human score	p -value	BLEU-2	p -value
MLE	0.4165	0.0034	0.6670	$< 10^{-6}$
SeqGAN	0.5356		0.7389	
Real data	0.6011		0.746	

- Obama political speech text generation

Algorithm	BLEU-3	p -value	BLEU-4	p -value
MLE	0.519	$< 10^{-6}$	0.416	0.00014
SeqGAN	0.556		0.427	

- Midi music generation

Algorithm	BLEU-4	p -value	MSE	p -value
MLE	0.9210	$< 10^{-6}$	22.38	0.00034
SeqGAN	0.9406		20.62	

Experiments on Real-World Data

- Chinese poem generation

南陌春风早，东邻去日斜。

紫陌追随日，青门相见时。

胡风不开花，四气多作雪。

Human

山夜有雪寒，桂里逢客时。

此时人且饮，酒愁一节梦。

四面客归路，桂花开青竹。

Machine

Obama Speech Text Generation

- When he was told of this extraordinary honor that he was the most trusted man in America
- But we also remember and celebrate the journalism that Walter practiced -- a standard of honesty and integrity and responsibility to which so many of you have committed your careers. It's a standard that's a little bit harder to find today
- I am honored to be here to pay tribute to the life and times of the man who chronicled our time.

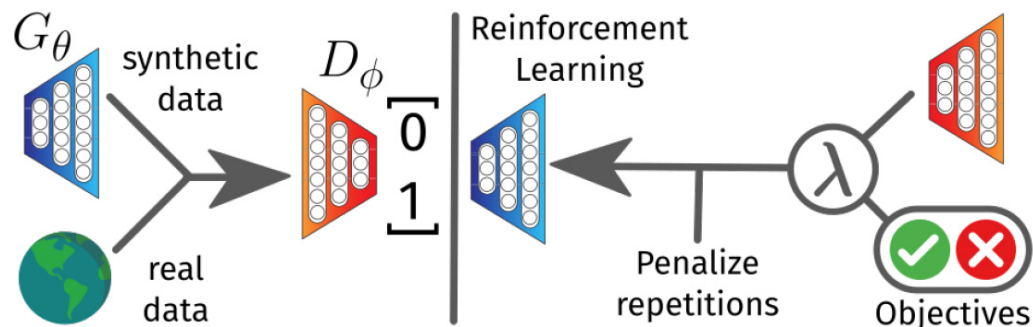
Human

- i stood here today i have one and most important thing that not on violence throughout the horizon is OTHERS american fire and OTHERS but we need you are a strong source
- for this business leadership will remember now i cant afford to start with just the way our european support for the right thing to protect those american story from the world and
- i want to acknowledge you were going to be an outstanding job times for student medical education and warm the republicans who like my times if he said is that brought the

Machine

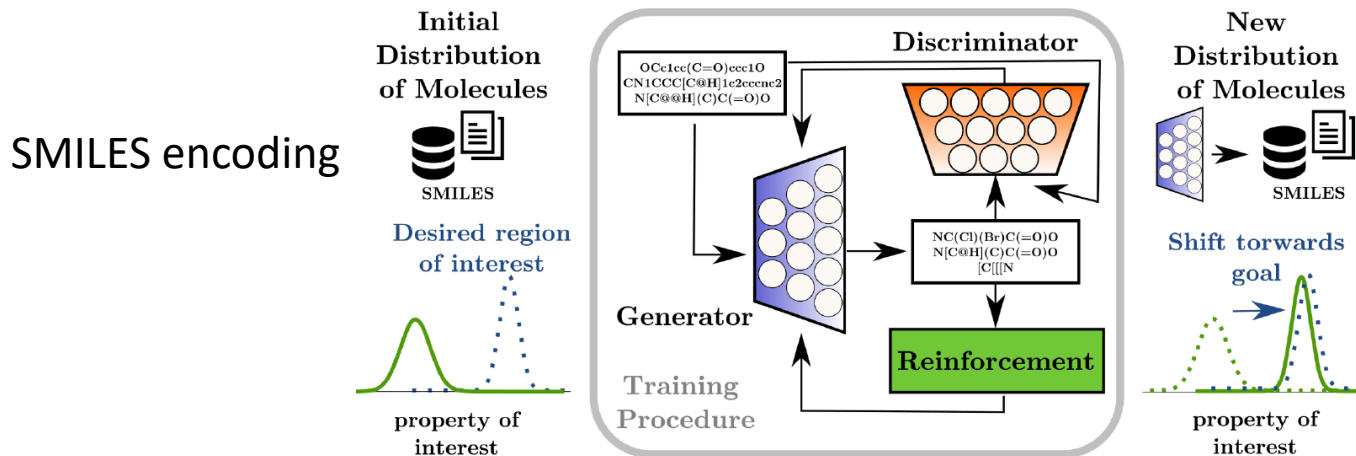
SeqGAN for Inverse-Design Chemistry

- Objective-Reinforced Generative Adversarial Networks (ORGAN)



$$R(Y_{1:T}) = \lambda \cdot D_\phi(Y_{1:T}) + (1 - \lambda) \cdot O_i(Y_{1:T})$$

- ORGANIC: ORGAN for inverse-design Chemistry



GANs on Discrete Data

- Single token
 - Information retrieval



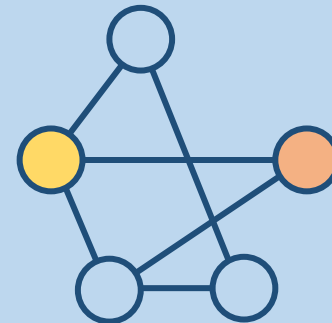
$$p(\text{doc}_n | \text{query}; \theta)$$

- Sequence
 - Text
 - Music score
 - DNA/RNA pieces
 - ...



$$p(\text{word}_n | \text{word}_{1...n-1}; \theta)$$

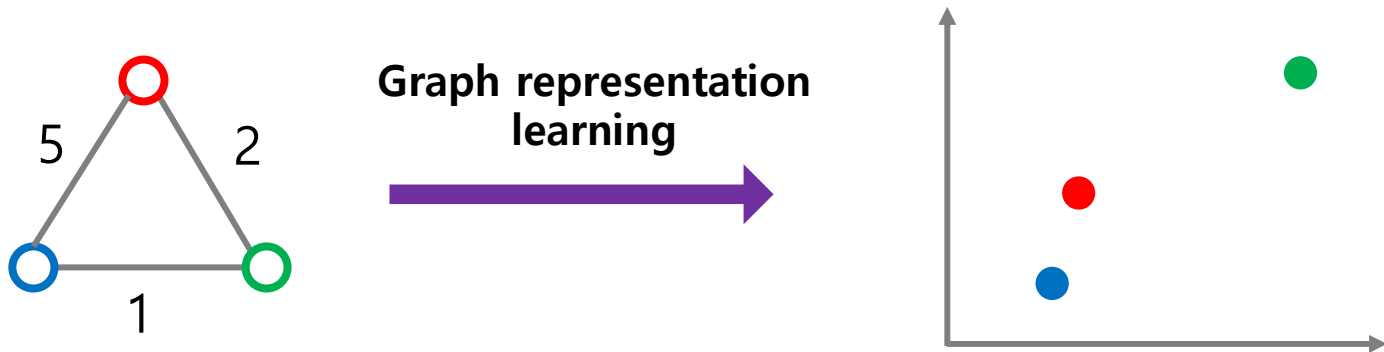
- Graph
 - Social network
 - User-item shopping behavior
 - Paper citations
 - ...



$$p(\text{node}_n | \text{node}_m, \text{neighbor}(m); \theta)$$

Background of GRL

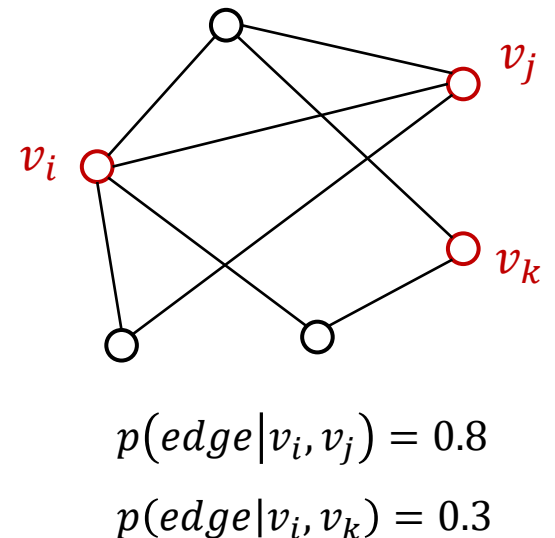
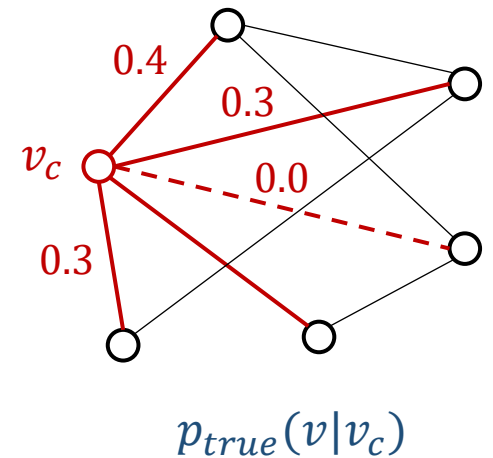
- Graph representation learning (GRL) learns a vector for each node in a graph
 - a.k.a. graph embedding / network embedding / network representation learning



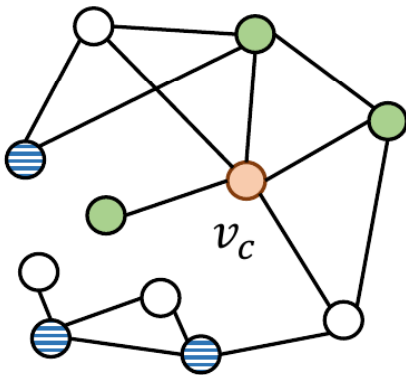
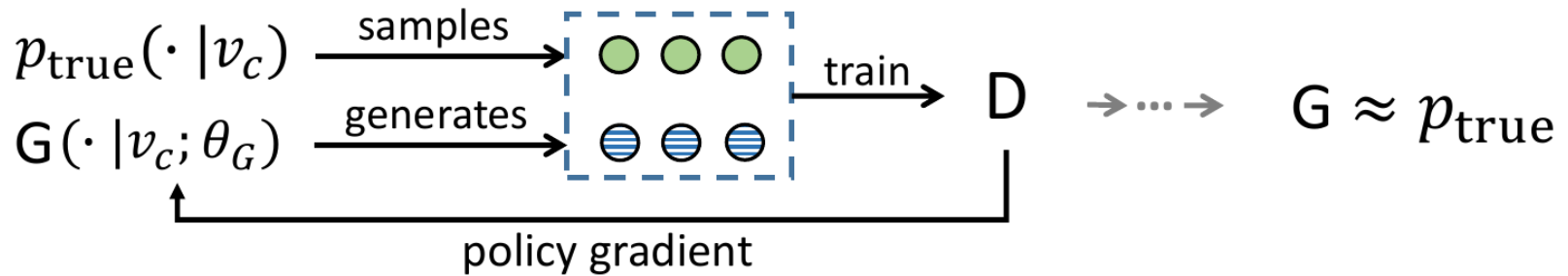
- Graph representation learning applications
 - Link prediction
 - Node classification
 - Recommendation
 - Visualization
 - Clustering
 - Social network analysis

Motivation of GraphGAN

- **Generative** graph representation learning model assumes an underlying real connectivity distribution $p_{true}(v|v_c)$ for each vertex v_c
 - E.g., DeepWalk (KDD 2014) and node2vec (KDD 2016)
- **Discriminative** graph representation learning model aims to learn a classifier for predicting the existence of edges directly
 - E.g., SDNE (KDD 2016) and PPNE (DASFAA 2017)

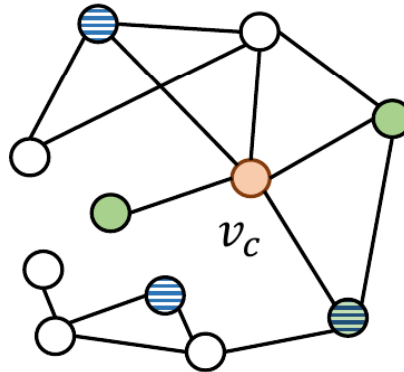


GraphGAN: the Minimax Game



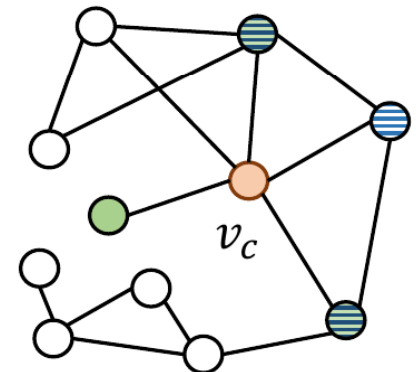
G underperforms
in initial stage

→ ... →



G is approaching p_{true}
during adversarial training

→ ... →



G is hardly distinguishable
from p_{true}

$$\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^V \left(\mathbb{E}_{v \sim p_{\text{true}}(\cdot | v_c)} [\log D(v, v_c; \theta_D)] + \mathbb{E}_{v \sim G(\cdot | v_c; \theta_G)} [\log (1 - D(v, v_c; \theta_D))] \right).$$

Implementation & Optimization of D

$$\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^V \left(\mathbb{E}_{v \sim p_{\text{true}}(\cdot | v_c)} [\log D(v, v_c; \theta_D)] + \mathbb{E}_{v \sim G(\cdot | v_c; \theta_G)} [\log (1 - D(v, v_c; \theta_D))] \right).$$

- A simple implementation of D

$$D(v, v_c) = \sigma(\mathbf{d}_v^\top \mathbf{d}_{v_c}) = \frac{1}{1 + \exp(-\mathbf{d}_v^\top \mathbf{d}_{v_c})}$$

- Note that any other discriminative model of link prediction can be implemented here, e.g., SDNE
- Gradient of $V(G, D)$ w.r.t. the parameters of D

$$\nabla_{\theta_D} V(G, D) = \begin{cases} \nabla_{\theta_D} \log D(v, v_c), & \text{if } v \sim p_{\text{true}} \\ \nabla_{\theta_D} (1 - \log D(v, v_c)), & \text{if } v \sim G \end{cases}$$

(a normal replacement of loss in GAN)

Graph Softmax in GraphGAN

- Breadth First Search (BFS) on \mathcal{G} from every vertex v_c
 - BFS-tree T_c rooted at v_c
- For a given vertex v and one of its neighbors $v_i \in \mathcal{N}_c(v)$, the relevance probability of v_i given v as

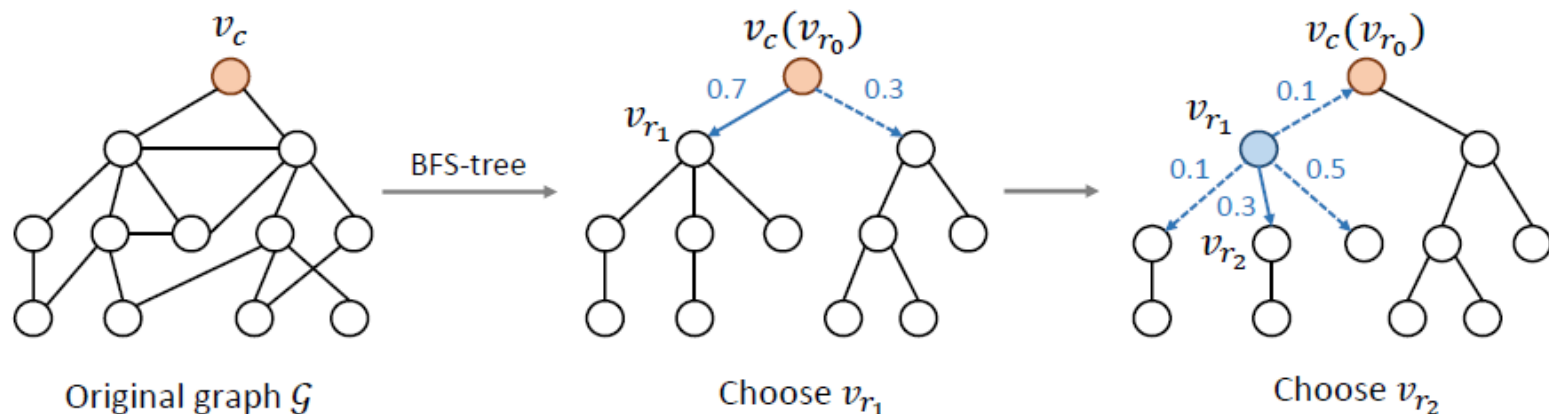
$$p_c(v_i|v) = \frac{\exp(\mathbf{g}_{v_i}^\top \mathbf{g}_v)}{\sum_{v_j \in \mathcal{N}_c(v)} \exp(\mathbf{g}_{v_j}^\top \mathbf{g}_v)}$$

- Graph softmax Go to an unvisited neighbor Get back to the parent

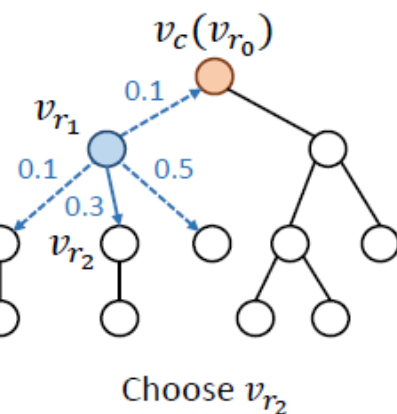
$$G(v|v_c) \triangleq \left(\prod_{j=1}^m p_c(v_{r_j}|v_{r_{j-1}}) \right) \cdot p_c(v_{r_{m-1}}|v_{r_m})$$

given the unique path from v_c to v in tree T_c : $P_{v_c \rightarrow v} = (v_{r_0}, v_{r_1}, \dots, v_{r_m})$, where $v_{r_0} = v_c$ and $v_{r_m} = v$

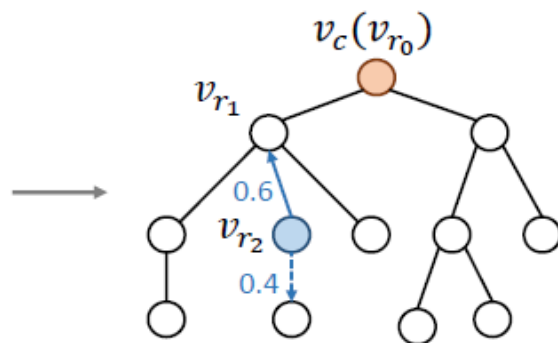
Graph Softmax in GraphGAN



$$p_c(v_{r_1}|v_c) = 0.7$$

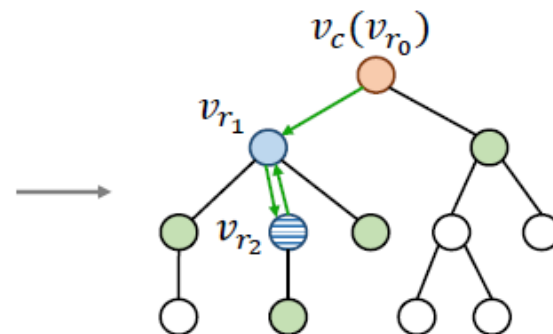


$$p_c(v_{r_2}|v_{r_1}) = 0.3$$



Choose v_{r_1} , sampling completed
 v_{r_2} is the sampled vertex

$$p_c(v_{r_1}|v_{r_2}) = 0.6$$



Update all vertexes along the green
 path and all vertexes in green

$$G(v_{r_2}|v_c; \theta_G) = 0.7 \times 0.3 \times 0.6 = 0.126$$

Link Prediction Experiments

- Overall link prediction performance

Model	arXiv-AstroPh		arXiv-GrQc	
	Accuracy	Macro-F1	Accuracy	Macro-F1
DeepWalk	0.841	0.839	0.803	0.812
LINE	0.820	0.814	0.764	0.761
Node2vec	0.845	0.854	0.844	0.842
Struc2vec	0.821	0.810	0.780	0.776
GraphGAN	0.855	0.859	0.849	0.853

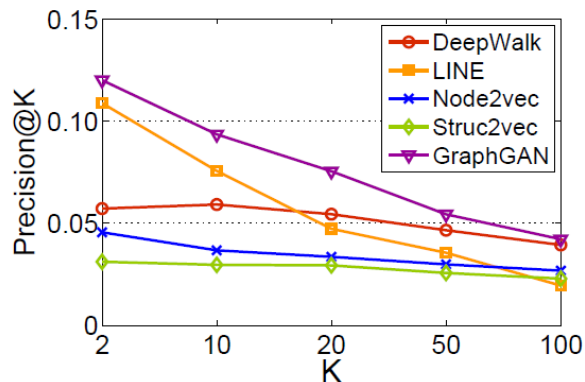
- LINE and struc2vec is relatively poor in link prediction, as they cannot quite capture the pattern of edge existence in graphs.
- DeepWalk and node2vec perform better than LINE and struc2vec probably because of the random-walk-based Skip-Gram model, which is graph-structure-aware and better at extracting proximity information among vertices.
- GraphGAN performs the best

Experiments on Other Tasks

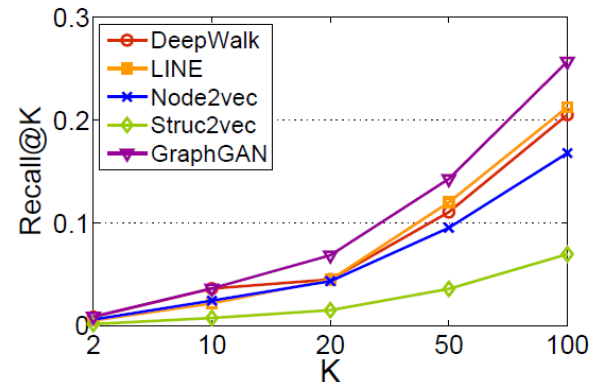
- Node Classification

Model	BlogCatalog		Wikipedia	
	Accuracy	Macro-F1	Accuracy	Macro-F1
DeepWalk	0.225	0.214	0.194	0.183
LINE	0.205	0.192	0.175	0.164
Node2vec	0.215	0.206	0.191	0.179
Struc2vec	0.228	0.216	0.211	0.190
GraphGAN	0.232	0.221	0.213	0.194

- Recommendation (Movielens-1M)



(a) Precision@K



(b) Recall@K

Content

1. Introduction to Generative Adversarial Nets
2. GANs on Continuous Data
3. GAN and RL
4. GANs on Discrete Data
5. Summary

SUMMARY

GANs on Continuous and Discrete Data

- GANs on continuous data $\frac{\partial L(D(x))}{\partial x} \cdot \frac{\partial x}{\partial \theta}$
 - Taking gradient on the generated data, then back-prop to generator parameters
 - Gradient penalty problems, with Wasserstein distance solution and Lipschitz constraints
- GANs on discrete data $\mathbb{E}_{x \sim G_\theta} \left[\frac{\partial \log G_\theta(x)}{\partial \theta} L(D(x)) \right]$
 - No gradient on the generated data
 - Leverage stochastic policy gradient method from RL
 - Data efficiency problems

References of GANs on Continuous Data

- Lipschitz Continuity
 - Arjovsky et al. Wasserstein GANs. ICML 2017.
 - Gulrajani et al. Improved Training of Wasserstein GANs. NIPS 2017.
 - Petzka et al. On the regularization of Wasserstein GANs. ICLR 2018.
 - Miyato et al. Spectral normalization for generative adversarial networks. ICLR 2018.
 - Zhou et al. Lipschitz Generative Adversarial Nets. ICML 2019.
- GAN Applications
 - Tero Karras et al. Progressive Growing of GANs for Improved Quality, Stability, and Variation. ICLR 2018.
 - Andrew Brock et al. Large Scale GAN Training for High Fidelity Natural Image Synthesis. ICLR 2019.
 - Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." CVPR 2017.
 - Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." CVPR 2017.
 - Yun Cao, Weinan Zhang etc. Unsupervised Diverse Colorization via Generative Adversarial Networks. ECML-PKDD 2017.

References of GANs on Discrete Data

- Single token



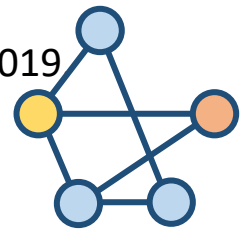
- Wang, Jun, et al. "IRGAN: A minimax game for unifying generative and discriminative information retrieval models." *SIGIR 2017*.

- Sequence



- Yu, Lantao, et al. "Seqgan: Sequence generative adversarial nets with policy gradient." *AAAI 2017*.
 - Guo, Jiaxian, et al. "Long text generation via adversarial training with leaked information." *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
 - Zhu, Yaoming, et al. "Texygen: a benchmarking platform for text generation models." *SIGIR 2018*.
 - Lu, Sidi, et al. "Improved Training for Sequence Generative Models." *ICML 2019*

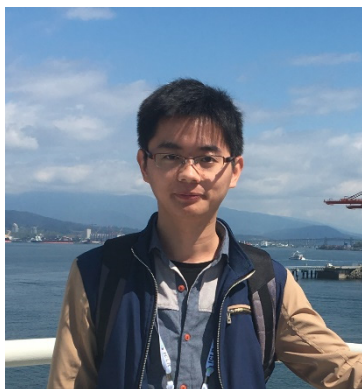
- Graph



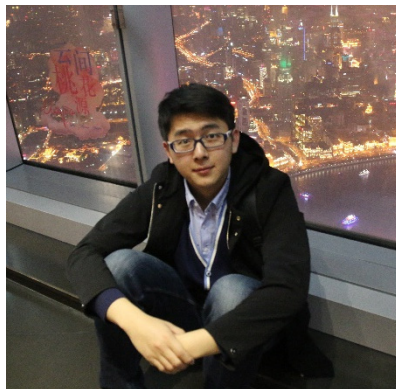
- Wang, Hongwei, et al. "GraphGAN: graph representation learning with generative adversarial nets." *AAAI 2018*.
 - Jia, Yuting, et al. "CommunityGAN: Community Detection with Generative Adversarial Nets." *WWW 2019*.

Acknowledgements

Thanks to My Students & Collaborators



Zhiming Zhou
Lipschitz GAN



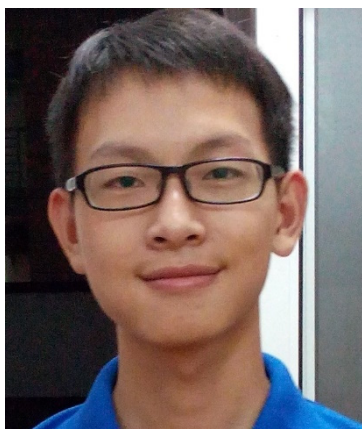
Lantao Yu
SeqGAN/IRGAN



Yaoming Zhu
Texygen



Yuting Jia
CommunityGAN



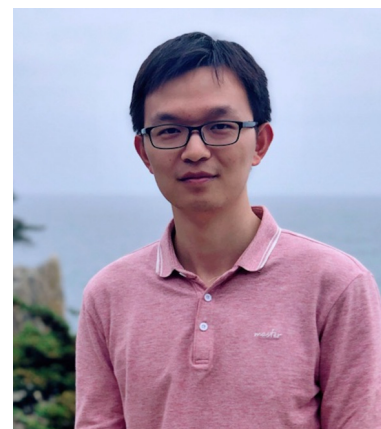
Jiaxian Guo
LeakGAN



Sidi Lu
CoT



Yun Cao
ColorGAN



Hongwei Wang
GraphGAN

Thank You! Questions?



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

Weinan Zhang

Associate Professor

APEX Data & Knowledge Management Lab

John Hopcroft Center for Computer Science

Shanghai Jiao Tong University

<http://wnzhang.net>