Generative Adversarial Nets for Information Retrieval

Fundamentals and Advances

Weinan Zhang
Shanghai Jiao Tong University
http://wnzhang.net

SIGIR 2018 Tutorial, Ann Arbor, USA

http://wnzhang.net/tutorials/sigir2018/

Self Introduction – Weinan Zhang

- Position
  - Assistant Professor at APEX Data & Knowledge Management Lab, John Hopcroft Center, Dept. of Computer Science, Shanghai Jiao Tong University, 2016-now
  - Research on machine learning and data mining topics

- Education
  - Ph.D. on Computer Science from University College London (UCL), United Kingdom, 2012-2016
  - B.Eng. on Computer Science from ACM Class 07 of Shanghai Jiao Tong University, China, 2007-2011

Motivations of this Tutorial

- Deep learning methods get explosive growth in IR
  - Lots of new works are implemented with deep neural networks
  - NeuIR workshop in SIGIR, deep learning for recommender system workshop in RecSys etc.
- But almost all attentions are put on discriminant models, i.e., how to use deep networks to implement a scoring function

$$f_\phi(\text{query}, \text{doc})$$

- We can definitely consider more on the generative modeling side of IR

Motivations of this Tutorial

- Many classic generative models in IR

From document to query

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

From query to document

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

- Compared with the scoring for a particular query-doc pair, generative models provide relevance distribution over documents
- We can definitely consider more on the generative modeling side of IR

Content of this Tutorial

1. Introduction to Generative Adversarial Nets
2. Reinforcement Learning
3. GANs for Information Retrieval
4. GANs for Text Generation
5. GANs for Graph/Network Learning
6. Beyond GANs, Cooperative Training
7. Future Perspective and Summarization

Content of this Tutorial

1. Introduction to Generative Adversarial Nets
2. Reinforcement Learning
3. GANs for Information Retrieval
4. GANs for Text Generation
5. GANs for Graph/Network Learning
6. Beyond GANs, Cooperative Training
7. Future Perspective and Summarization

Problem Definition of Data Generation

- Given a dataset $D = \{x\}$, build a model $q_\theta(x)$ of the data distribution that fits the true 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 true 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 true 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 true as possible
- More straightforward but it is hard or impossible to directly calculate $p(x)$

Generative Adversarial Nets (GANs)

- What we really want

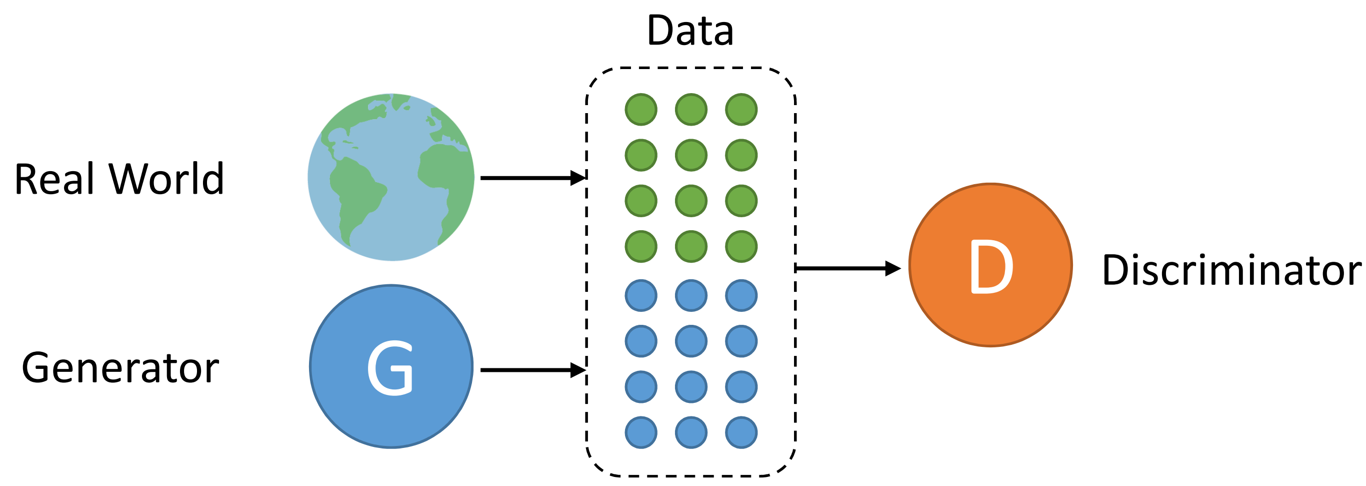
$$\max_{\theta} \mathbb{E}_{x \sim q_\theta(x)}[\log p(x)]$$

- But we cannot directly calculate $p(x)$

- Idea: what if we build a discriminator to judge whether a data instance is true or fake (artificially generated)?
  - Leverage the strong power of deep learning based discriminative models

[Goodfellow, I., et al. 2014. Generative adversarial nets. In NIPS 2014.]

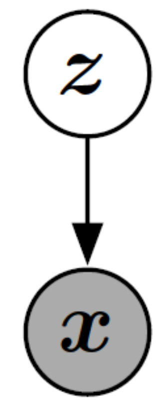
Generative Adversarial Nets (GANs)



- Discriminator tries to correctly distinguish the true 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 true and generated data, G nicely fits the true underlying data distribution

Generator Network

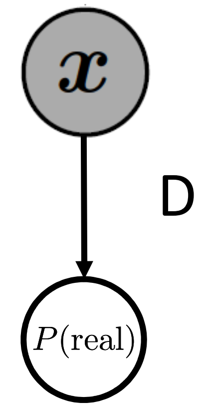
$$\boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{\theta})$$



- Must be differentiable
- No invertibility requirement
- Trainable for any size of *z*
- Can make *x* conditionally Gaussian given *z* but need not do so
  - e.g. Variational Auto-Encoder
- Popular implementation: multi-layer perceptron

Discriminator Network

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



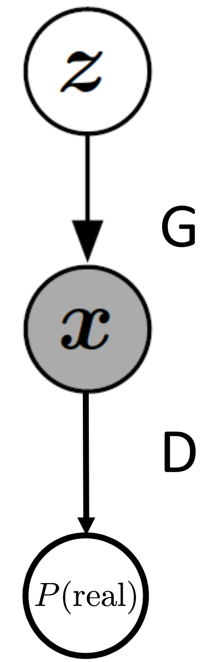
- Can be implemented by any neural networks with a probabilistic prediction
- For example
  - Multi-layer perceptron with logistic output
  - AlexNet etc.

Generator and Discriminator Nets

- Generator network

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

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

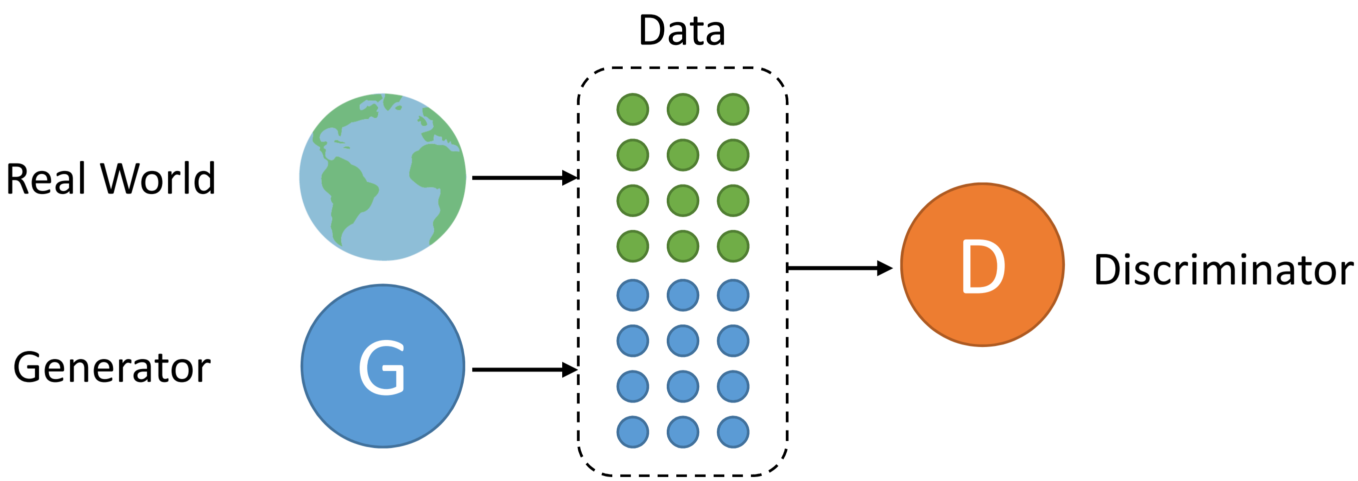


- Discriminator network

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

  - 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

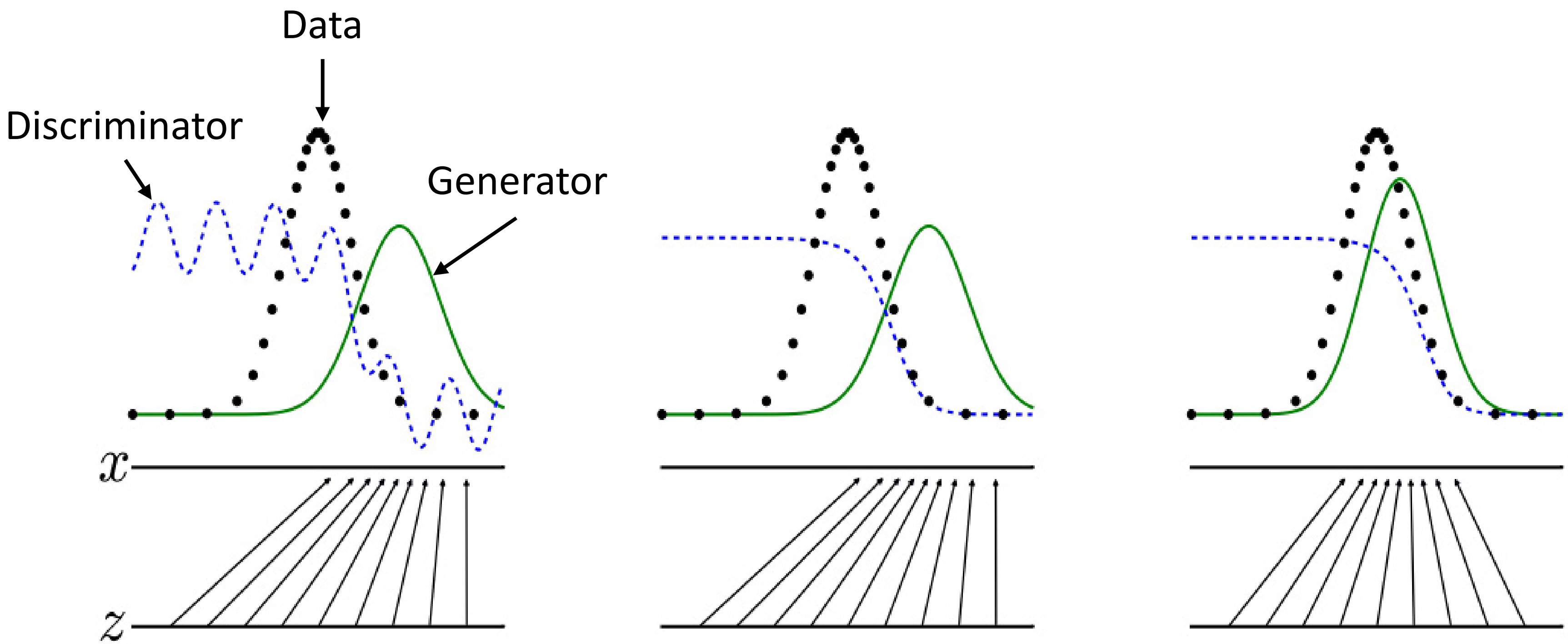


$$\min_G \max_D J(G,D) \qquad \max_D J(G,D)$$

The joint objective function

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

Illustration of GANs

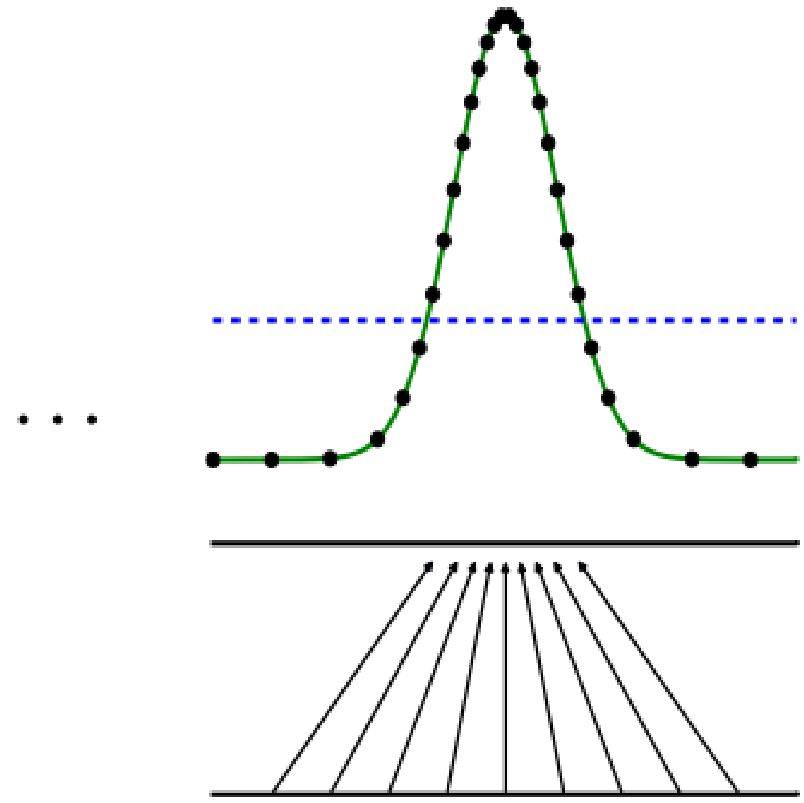


Generator $\min_G \max_D J^{(D)}$ Discriminator $\max_D J^{(D)}$

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

Ideal Final Equilibrium

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



Training GANs

Training discriminator

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Training GANs

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

Training generator

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

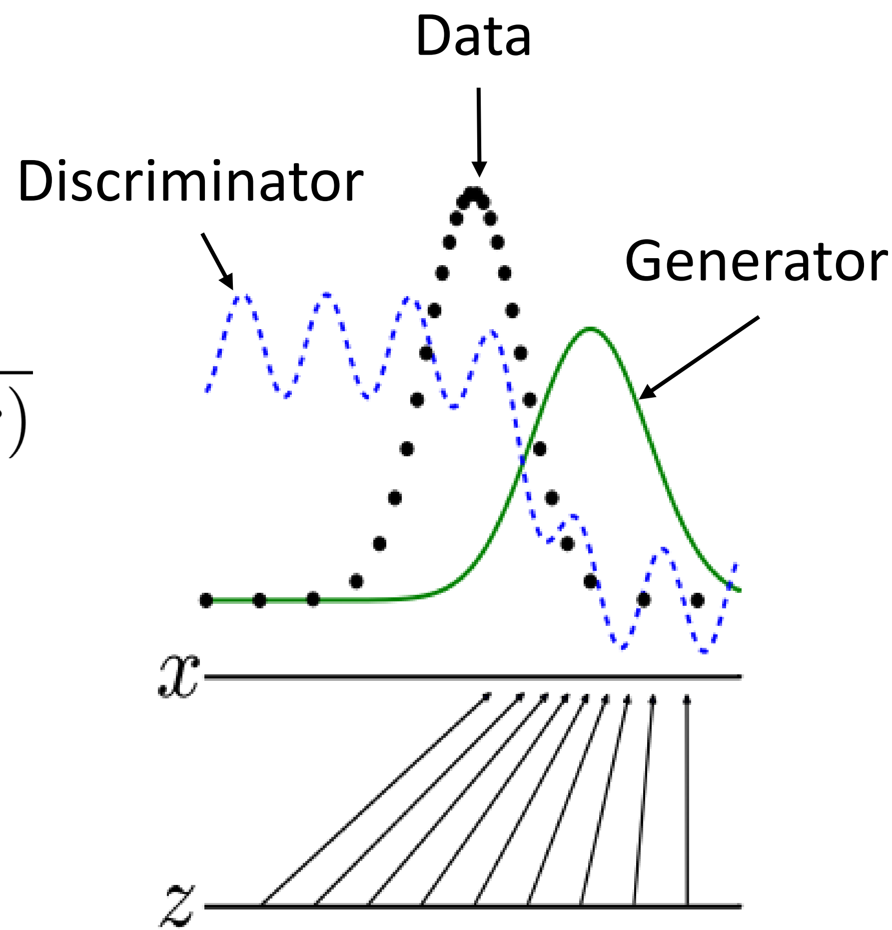
end for

Optimal Strategy for Discriminator

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

G: $\min_G \max_D J(G, D)$ D: $\max_D J(G, D)$

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

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

Equilibrium for the Minimax Game

G: $\min_G \max_D J(G, D)$ D: $\max_D J(G, D)$

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

$\min_G J^{(D)}$ is something between $\min_G \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[p_G(\boldsymbol{x})]$ and $\min_G \mathbb{E}_{\boldsymbol{x} \sim p_G}[p_{\text{data}}(\boldsymbol{x})]$

[Huszár, Ferenc. "How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?." *arXiv* (2015).]

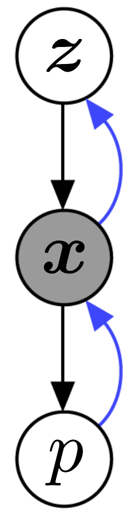
GANs for Continuous Data

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

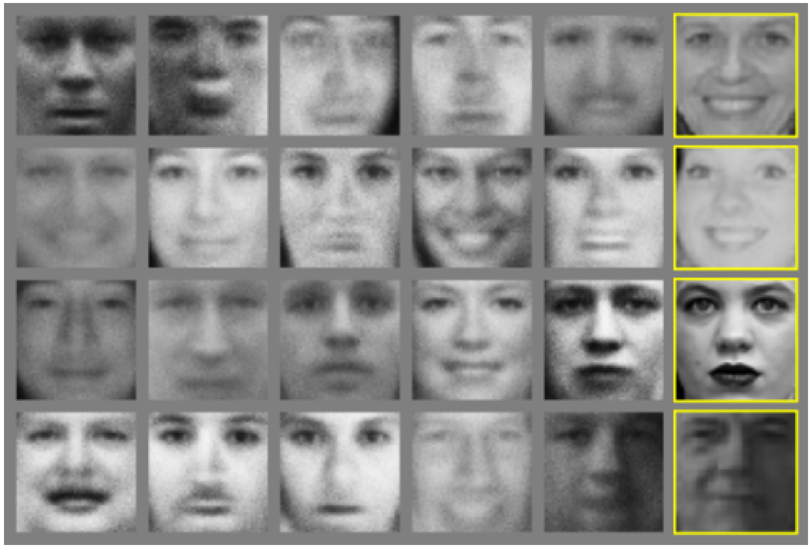
$$\frac{\partial J(G, D)}{\partial x}$$

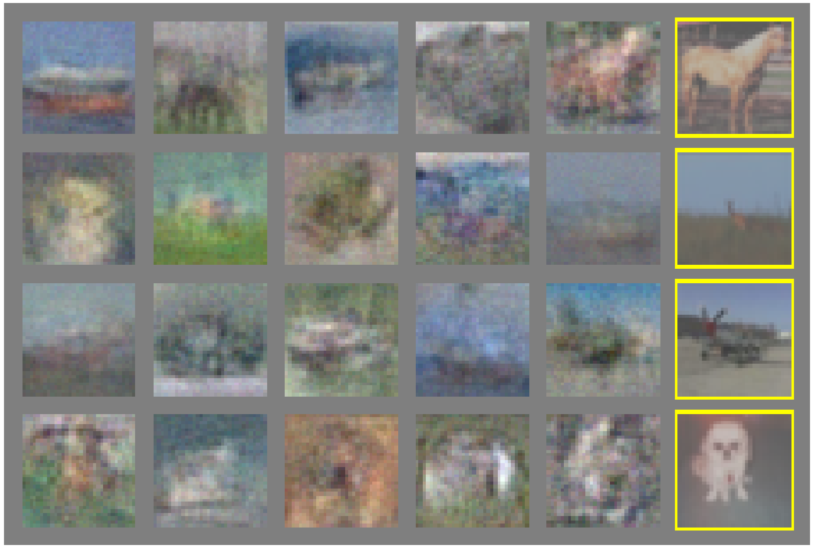
- In order to take gradient on the generator parameter, *x* has to be continuous

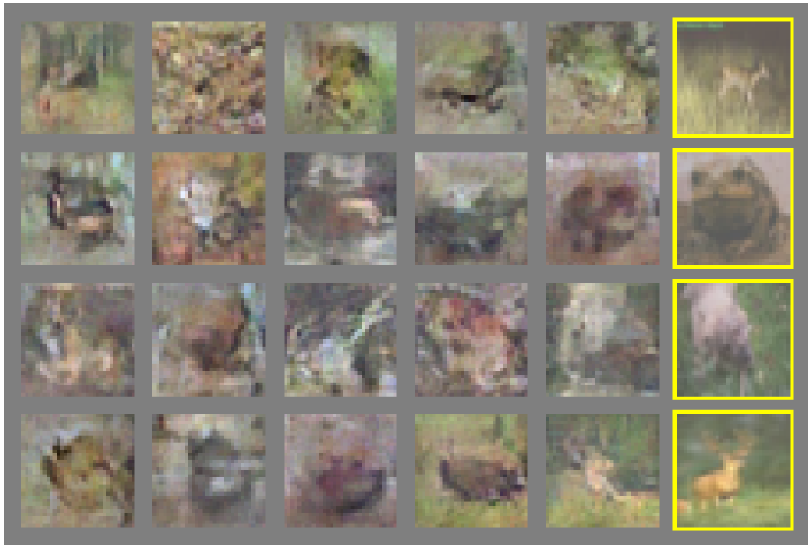
$$J(G, D) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Generator $\min_G \max_D J(G, D)$ Discriminator $\max_D J(G, D)$

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

(Goodfellow NIPS 2016 Tutorial: Generative Adversarial Networks)

High Resolution and Quality Images

- Progressive Growing of GANs

Two imaginary celebrities that were dreamed up by a random number generator.

Tero Karras et al. Progressive Growing of GANs for Improved Quality, Stability, and Variation. ICLR 2018.

Single Image Super-Resolution

bicubic (21.59dB/0.6423) | SRResNet (23.53dB/0.7832) | SRGAN (21.15dB/0.6868) | original

[4× upscaling]

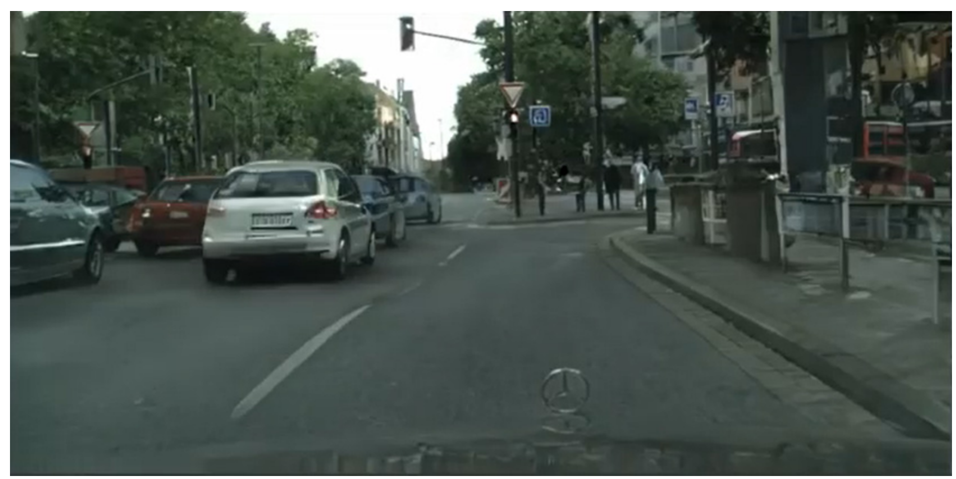
deep residual generative adversarial network optimized for a loss more sensitive to human perception

Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." CVPR 2017.

Image to Image Translation

Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." *CVPR 2017*.

High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs

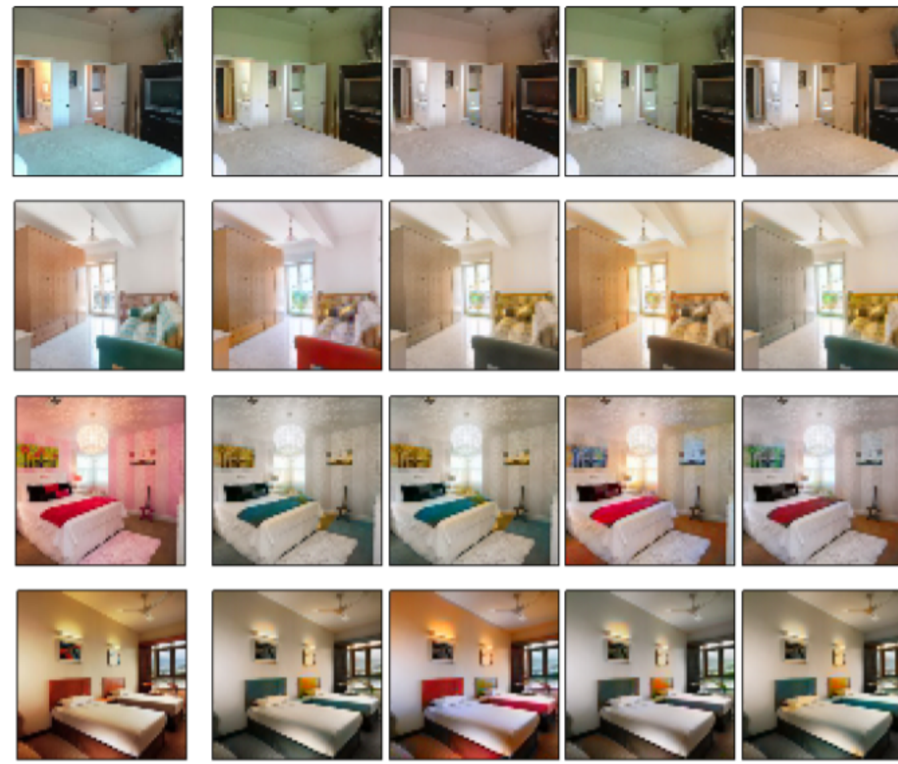


Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs", arXiv preprint arXiv:1711.11585.

Grayscale Image Colorization

Ground Truth

Generated Colorization after Performing Grayscale



Ground Truth

Generated Colorization after Performing Grayscale

Yun Cao, Weinan Zhang etc. Unsupervised Diverse Colorization via Generative Adversarial Networks. ECML-PKDD 2017.

GANs for Continuous Data

- All above applications are based on (conditional) GANs oriented to continuous data
- In information retrieval tasks, the data are mostly discrete
  - IDs in collaborative filtering
  - Text in web search
  - Graph nodes and edges in social networks
- The original GANs framework cannot handle such discrete data generation tasks

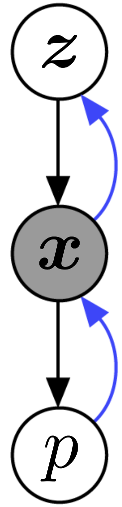
GANs for Continuous Data

1. Generation

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

2. Discrimination

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



4. Further gradient on generator

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

3. Gradient on generated data

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

- The chain rule in step 4 enables the generative to
  - Tune the parameter to slightly change the output *x* on the direction of $\partial J(G,D)/\partial x$ from the discriminator

$$x \leftarrow x - \eta \cdot \boxed{\frac{\partial J(G,D)}{\partial x}}$$

The loss function should be differentiable w.r.t. the instance *x*, which requires the data space is continuous

Discrete Data Generation

- How to generate discrete data?

- Sample the discrete token from a parametric distribution

$$x \sim P(x; \theta)$$

  and optimize the distribution w.r.t. its parameter

- Compare to the original GAN for continuous data
  - Sample a noise vector from a known distribution
  - Map the noise vector to a data instance

Content of this Tutorial

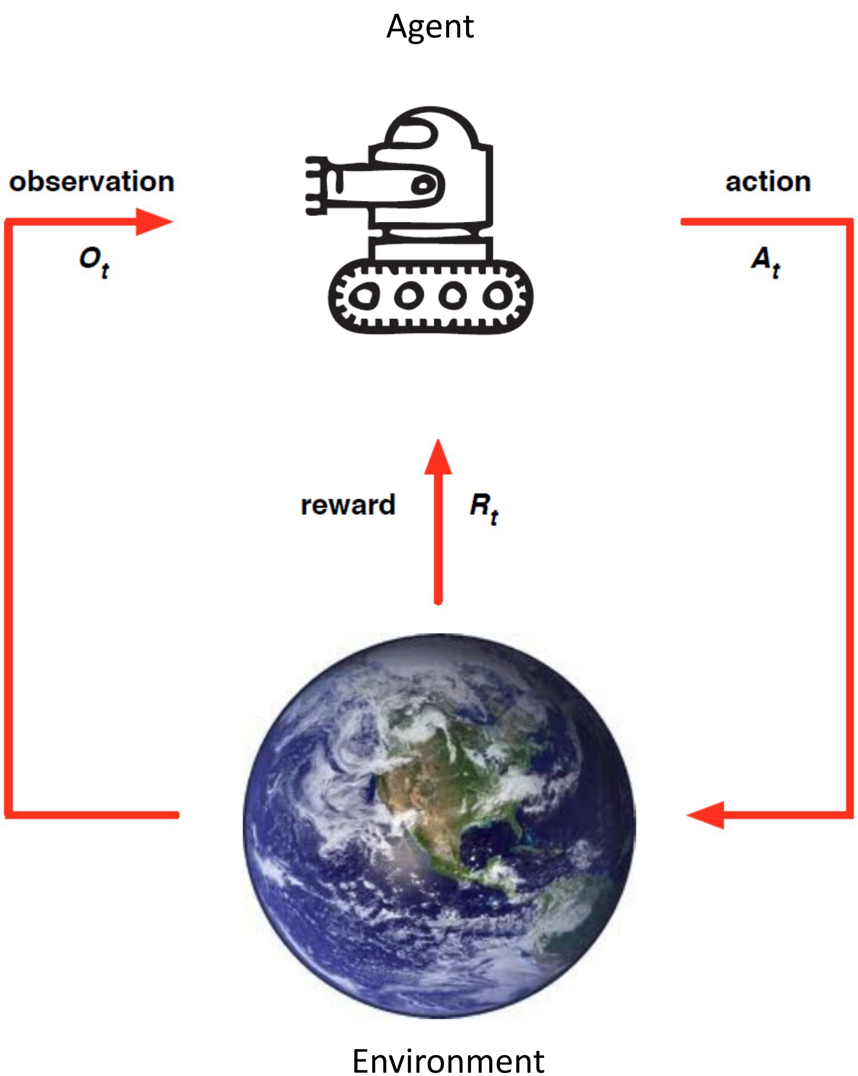
1. Introduction to Generative Adversarial Nets
2. Reinforcement Learning
3. GANs for Information Retrieval
4. GANs for Text Generation
5. GANs for Graph/Network Learning
6. Beyond GANs, Cooperative Training
7. Future Perspective and Summarization

Two Kinds of Machine Learning

- Prediction
  - Predict the desired output given the data (supervised learning)
  - Generate data instances (unsupervised learning)

- Decision Making
  - Take actions based on a particular state in a dynamic environment (reinforcement learning)
    - to transit to new states
    - to receive immediate reward
    - to maximize the accumulative reward over time
  - Learning from interaction

Reinforcement Learning



- At each step t, the agent
  - Receives observation O_t
  - Receives scalar reward R_t
  - Executes action A_t
- The environment
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at environment step
- Goal of RL: take actions to maximize cumulative rewards over time

Model-free Reinforcement Learning

- In realistic problems, we have no access to the environment (model) and only observed some episodes

Episode 1: $$s_0^{(1)} \xrightarrow[R(s_0)^{(1)}]{a_0^{(1)}} s_1^{(1)} \xrightarrow[R(s_1)^{(1)}]{a_1^{(1)}} s_2^{(1)} \xrightarrow[R(s_2)^{(1)}]{a_2^{(1)}} s_3^{(1)} \cdots s_T^{(1)}$$

Episode 2: $$s_0^{(2)} \xrightarrow[R(s_0)^{(2)}]{a_0^{(2)}} s_1^{(2)} \xrightarrow[R(s_1)^{(2)}]{a_1^{(2)}} s_2^{(2)} \xrightarrow[R(s_2)^{(2)}]{a_2^{(2)}} s_3^{(2)} \cdots s_T^{(2)}$$

- Model-free RL is to directly learn value & policy from experience without building an MDP
- Key steps: (1) estimate value function; (2) optimize policy

Value Function Estimation

- In RL, the value function is calculated by dynamic programming

$$V^{\pi}(s) = \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi]$$

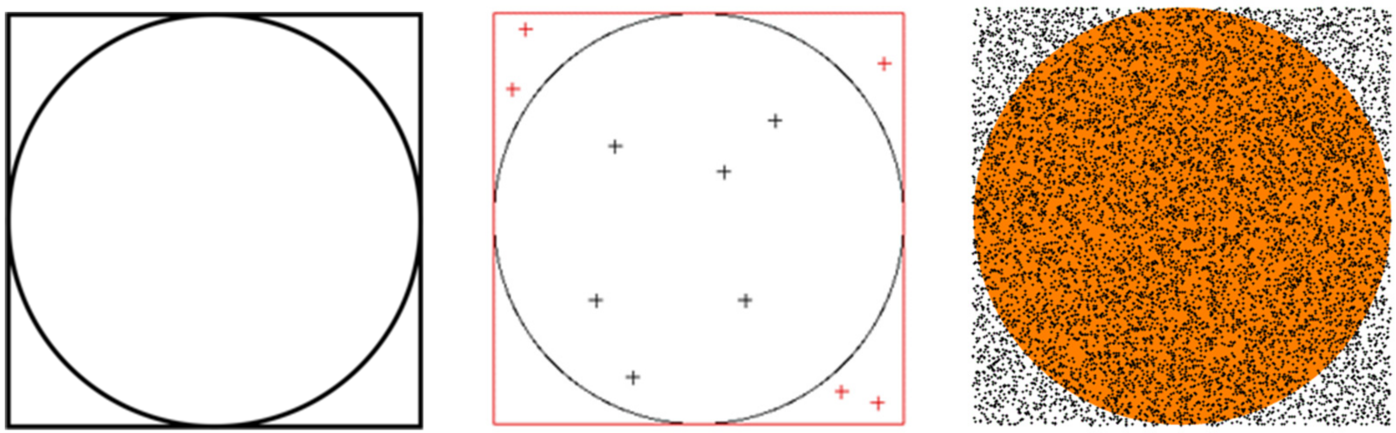
- Now in model-free RL
  - We cannot directly know the environment
  - But we have a list of experiences to estimate the values

Episode 1: $s_0^{(1)} \xrightarrow[R(s_0)^{(1)}]{a_0^{(1)}} s_1^{(1)} \xrightarrow[R(s_1)^{(1)}]{a_1^{(1)}} s_2^{(1)} \xrightarrow[R(s_2)^{(1)}]{a_2^{(1)}} s_3^{(1)} \cdots s_T^{(1)}$

Episode 2: $s_0^{(2)} \xrightarrow[R(s_0)^{(2)}]{a_0^{(2)}} s_1^{(2)} \xrightarrow[R(s_1)^{(2)}]{a_1^{(2)}} s_2^{(2)} \xrightarrow[R(s_2)^{(2)}]{a_2^{(2)}} s_3^{(2)} \cdots s_T^{(2)}$

Monte-Carlo Methods

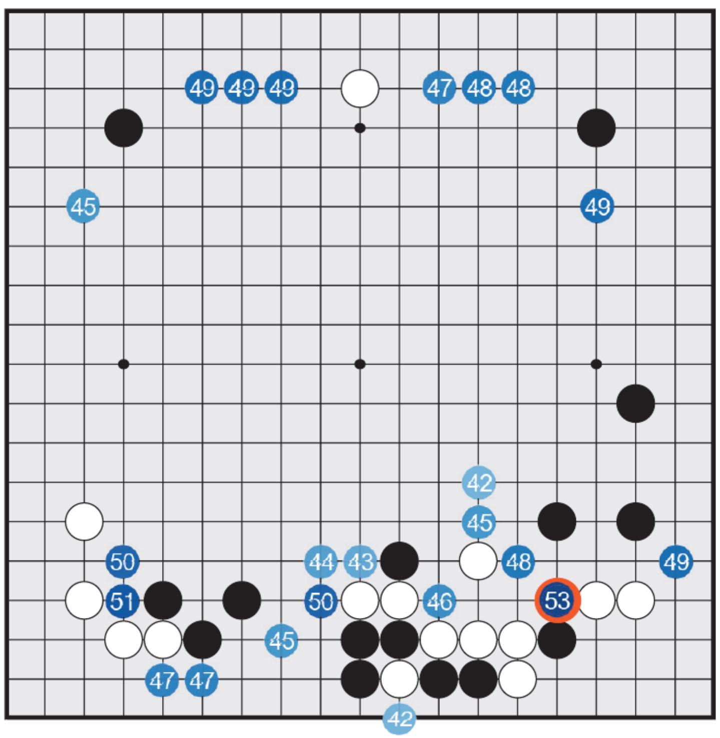
- Monte-Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.
- Example, to calculate the circle's surface

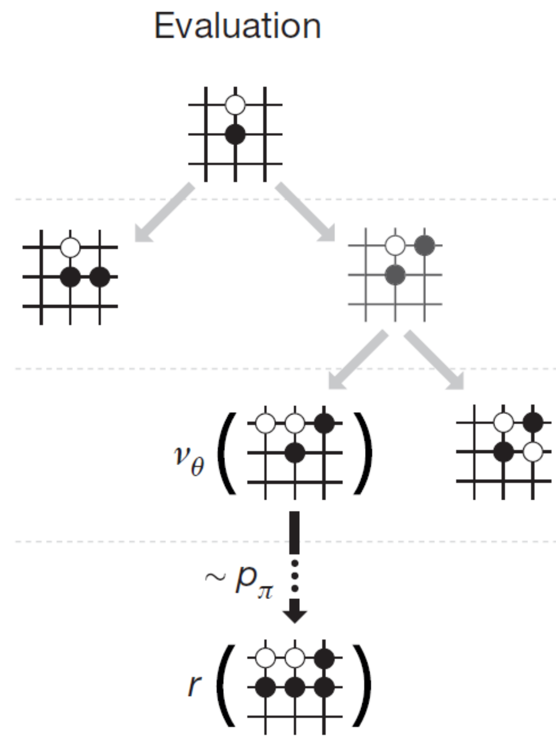


$$\text{Circle Surface} = \text{Square Surface} \times \frac{\#\text{points in circle}}{\#\text{points in total}}$$

Monte-Carlo Methods

- Go: to estimate the winning rate given the current state





$$\text{Win Rate}(s) = \frac{\#\text{win simulation cases started from } s}{\#\text{simulation cases started from } s \text{ in total}}$$

Monte-Carlo Value Estimation

- Goal: learn V^π from episodes of experience under policy π

$$s_0^{(i)} \xrightarrow[R_1^{(i)}]{a_0^{(i)}} s_1^{(i)} \xrightarrow[R_2^{(i)}]{a_1^{(i)}} s_2^{(i)} \xrightarrow[R_3^{(i)}]{a_2^{(i)}} s_3^{(i)} \cdots s_T^{(i)} \sim \pi$$

- Recall that the return is the total discounted reward

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots \gamma^{T-1} R_T$$

- Recall that the value function is the expected return

$$\begin{aligned} V^\pi(s) &= \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi] \\ &= \mathbb{E}[G_t | s_t = s, \pi] \\ &\simeq \frac{1}{N} \sum_{i=1}^{N} G_t^{(i)} \end{aligned}$$

- Sample N episodes from state s using policy π
- Calculate the average of cumulative reward

- Monte-Carlo policy evaluation uses empirical mean return instead of expected return

Parametric Policy

- We can parametrize the stochastic policy

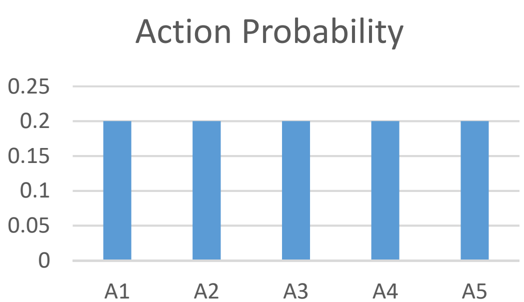
$$\pi_\theta(a|s) = P(a|s;\theta)$$

  - ϑ is the parameters of the policy
  - Generalize from seen states to unseen states
  - We focus on model-free reinforcement learning

Policy Gradient

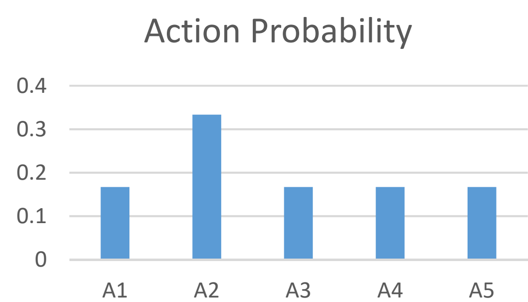
- For stochastic policy $\pi_\theta(a|s) = P(a|s;\theta)$
- Intuition
  - lower the probability of the action that leads to low value/reward
  - higher the probability of the action that leads to high value/reward
- A 5-action example

1. Initialize ϑ



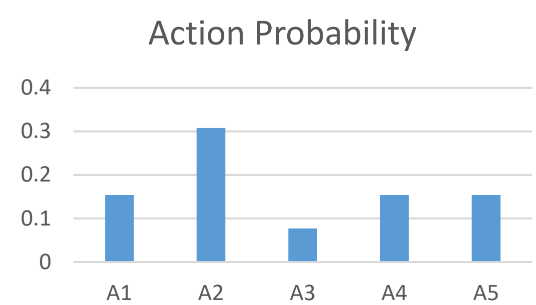
2. Take action A2
Observe positive reward

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

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

- Thus the policy's expected value

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

This can be approximated by sampling state *s* from *d*(*s*) and action *a* from π_ϑ

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

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

Please refer to appendix of the slides for detailed proofs

Monte-Carlo Policy Gradient (REINFORCE)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return G_t as an unbiased sample of $Q^{\pi_\theta}(s,a)$

$$\Delta\theta_t = \alpha \frac{\partial \log \pi_\theta(a_t|s_t)}{\partial\theta} G_t$$

- REINFORCE Algorithm

Initialize ϑ arbitrarily

for each episode $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ do

for t=1 to T-1 do

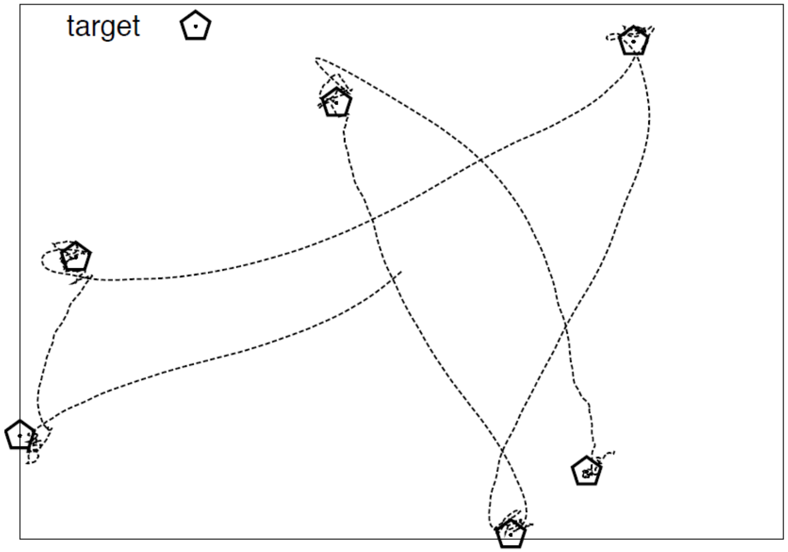
$$\theta \leftarrow \theta + \alpha \frac{\partial}{\partial\theta} \log \pi_\theta(a_t|s_t) G_t$$

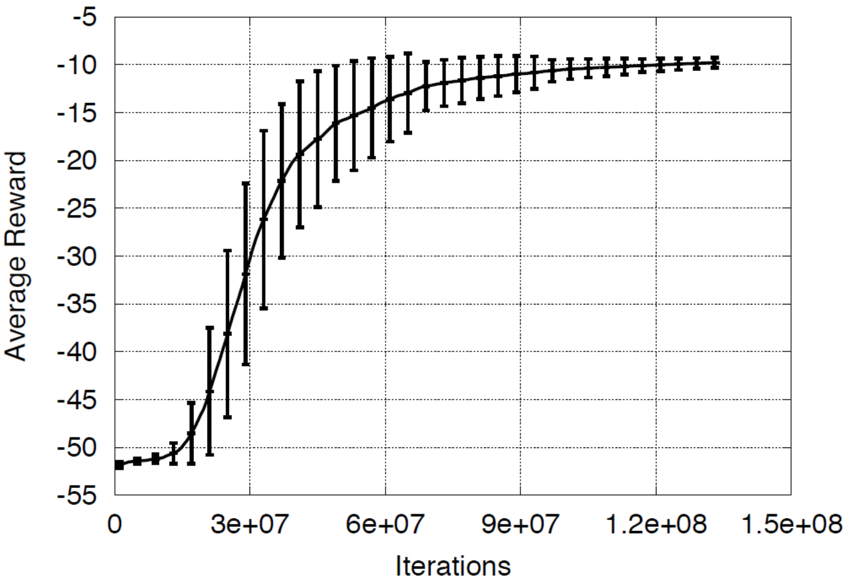
end for

end for

return ϑ

Puck World Example





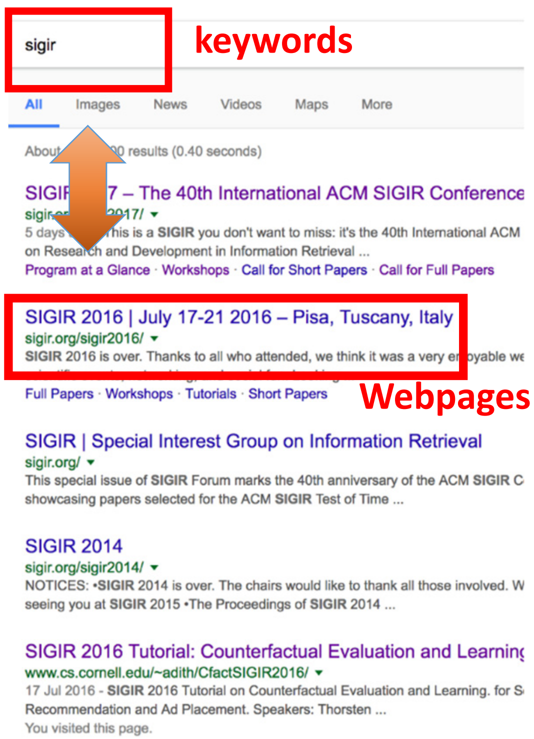
- Continuous actions exert small force on puck
- Puck is rewarded for getting close to target
- Target location is reset every 30 seconds
- Policy is trained using variant of MC policy gradient

Content of this Tutorial

1. Introduction to Generative Adversarial Nets
2. Reinforcement Learning
3. GANs for Information Retrieval
4. GANs for Text Generation
5. GANs for Graph/Network Learning
6. Beyond GANs, Cooperative Training
7. Future Perspective and Summarization

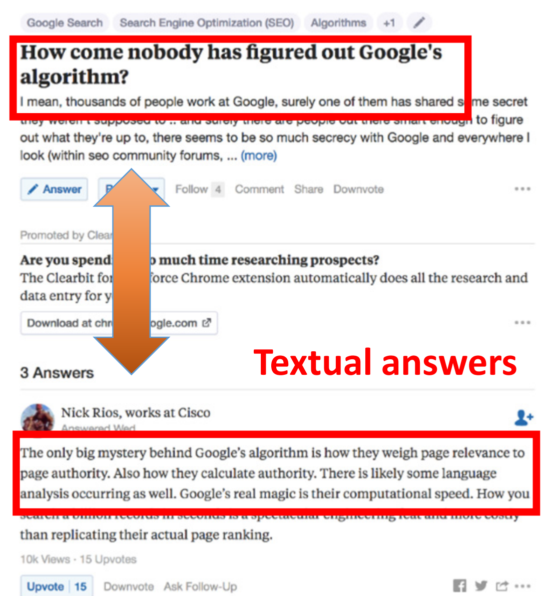
IR Theory: Relevancy is the Key

Web Search



Question Answers

Textual questions

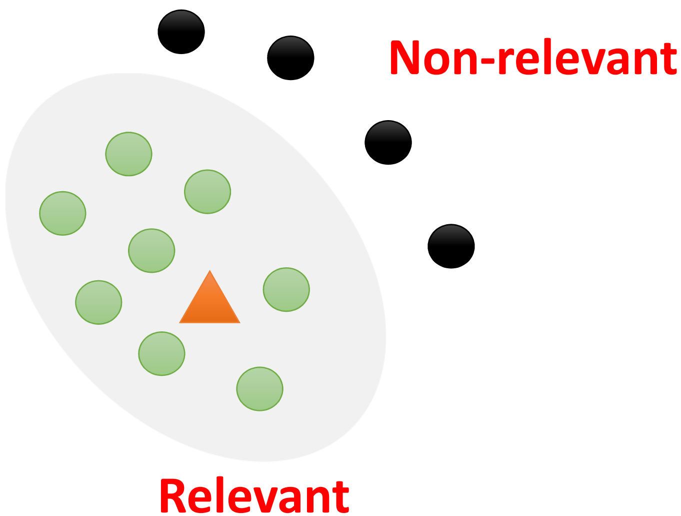


Recommender Systems

Recommended item

The classic school:
Generative Retrieval

D -> Q, Q -> D



Relevant document or query distribution

- Assume there is an underlying stochastic generative process between documents and information needs
  - D -> Q

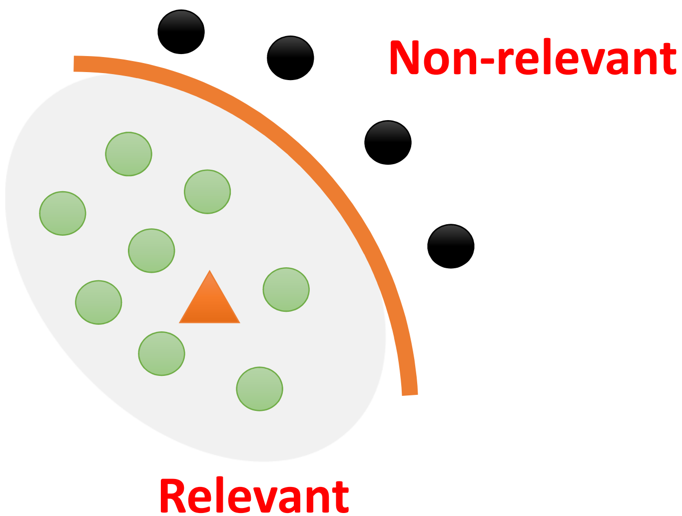
  e.g., From [Maron and Kuhns' Probabilistic Indexing, 60s] to [Statistical language models of text retrieval, 90s]

  - Q -> D

  e.g., [Robertson and Sparck Jones's Binary Independence Model, 70s]

The modern school:
Discriminative models

Q + D -> R



Decision boundary between relevance and non-relevance

- Discriminative models learned from labeled relevant judgements or their proxies such as clicks or ratings
- Consider documents and queries jointly as features and predicts their relevancy or rank order labels
  - Q + D -> R

  e.g., [Learning to rank, 2000s]

  [Neural information retrieval, 2010s]

Two Schools IR Thinkings: Pro/Con

Generative models of IR	Discriminative models of IR

• **Pros:** theoretically sound and very successful in modelling features
• **Cons:**
  • Difficult in leveraging relevancy signals from largely observable data, e.g., links, clicks
  • **Typically not trainable**
	• **Pros:** learn a retrieval ranking function implicitly from labeled data
	• **Cons:** lack a principled way of
	  • Obtaining useful features,
	  • Gathering helpful signals from the massive unlabeled data available, e.g., text statistics, the collection distribution

How to take advantage of both schools of thinking?

Generative models of IR	Discriminative models of IR
• Learns to fit the relevance distribution over documents via the signal from the discriminative model • **-> Trainable!!**	• Able to exploit the unlabeled data selected by the generative model to achieve a better estimation for document ranking • **-> automatically obtain needed training data!!**

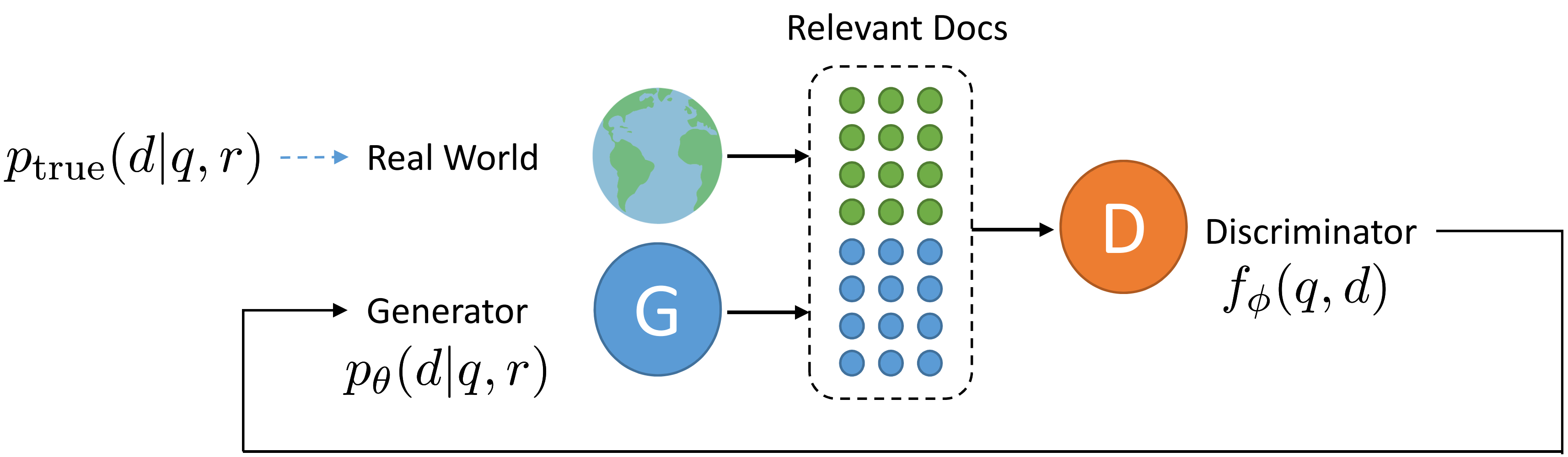
IRGAN: A Minimax Game for Information Retrieval

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang and Dell Zhang. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. SIGIR 2017.

IRGAN: A Minimax Game Unifying both Models

- Take advantage of both schools of thinking:
  - The generative model learns to fit the relevance distribution over documents $p_{\mathrm{true}}(d|q, r)$ via the signal from the discriminative model.
  - The discriminative model is able to exploit the unlabeled data selected by the generative model to achieve a better estimation $f_\phi(q, d)$ for document ranking.

IRGAN Formulation

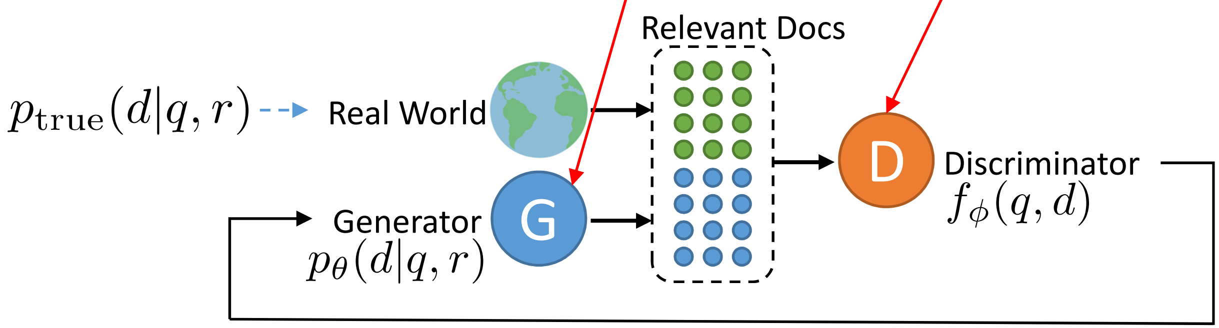


- **Underlying true relevance distribution** $p_{\text{true}}(d|q,r)$ depicts the user's relevance preference distribution over the candidate documents with respect to his submitted query
  - **Training set:** A set of samples from $p_{\text{true}}(d|q,r)$
- **Generative retrieval model** $p_\theta(d|q,r)$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

$$J^{G^*,D^*} = \min_{\theta} \max_{\phi} \sum_{n=1}^{N} \Big(\mathbb{E}_{d \sim p_{\text{true}}(d|q_n,r)} \left[\log \boxed{D(d|q_n)}\right] + \mathbb{E}_{d \sim \boxed{p_\theta(d|q_n,r)}} \left[\log(1 - D(d|q_n))\right] \Big)$$



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

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

Optimizing Generative Retrieval via Policy Gradient

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

$$\theta^* = \arg\min_{\theta} \sum_{n=1}^{N} \Big(\mathbb{E}_{d\sim p_{\text{true}}(d|q_n,r)} \left[\log \sigma(f_\phi(d,q_n))\right] + \mathbb{E}_{d\sim p_\theta(d|q_n,r)} \left[\log(1-\sigma(f_\phi(d,q_n)))\right] \Big)$$

$$= \arg\max_{\theta} \sum_{n=1}^{N} \underbrace{\boxed{\mathbb{E}_{d\sim p_\theta(d|q_n,r)}} \boxed{\left[\log(1+\exp(f_\phi(d,q_n)))\right]}}_{\text{denoted as } J^G(q_n)}$$

Generator as Policy denoted as $J^G(q_n)$ **Reward Term**

- REINFORCE (with Advantage Function)

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

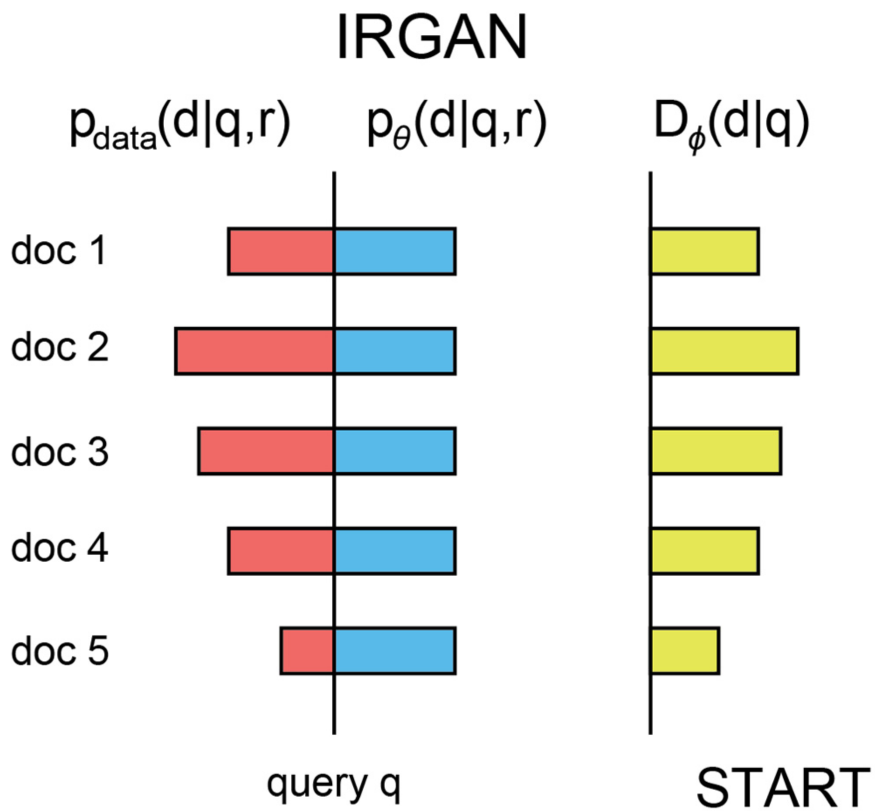
Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *NIPS 2000.*

IRGAN REINFORCE

- Likelihood ratio

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

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

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

  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.

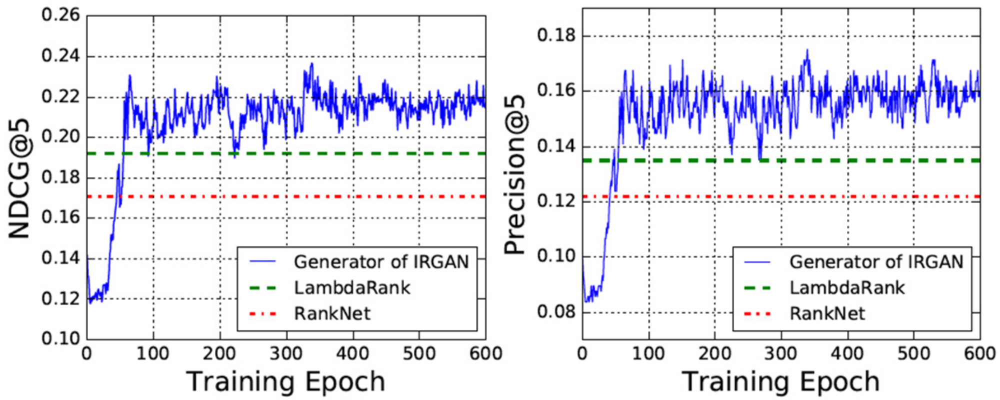
Experiments: Web Search

- 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

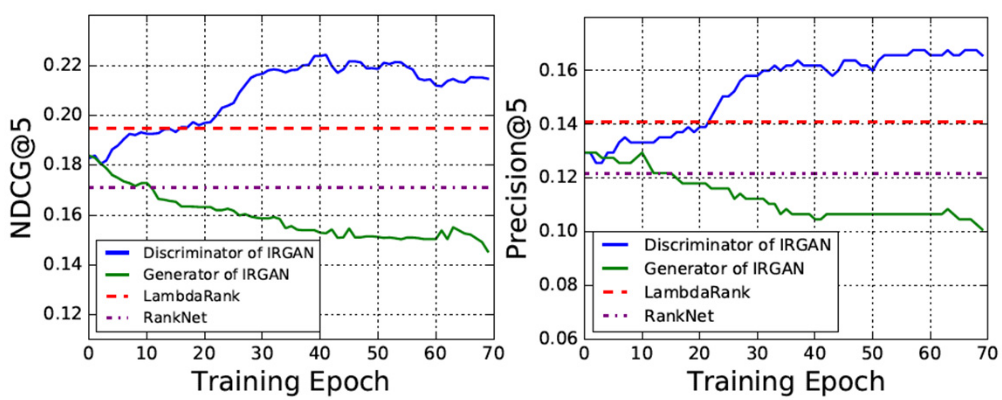
Table 1: Webpage ranking performance comparison on MQ2008-semi dataset, where ∗ means significant improvement in a Wilcoxon signed-rank test.

	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: Web Search



IRGAN-Pointwise Generator Performance



IRGAN-Pairwise Discriminator Performance

Key Observations

- In both setting, IRGAN consistently and significantly (see the table) outperforms other algorithms
- Typically, when one player (G or D) starts to outperforms the baseline discriminative model, the other player (D or G) would get worse than the baseline

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%*

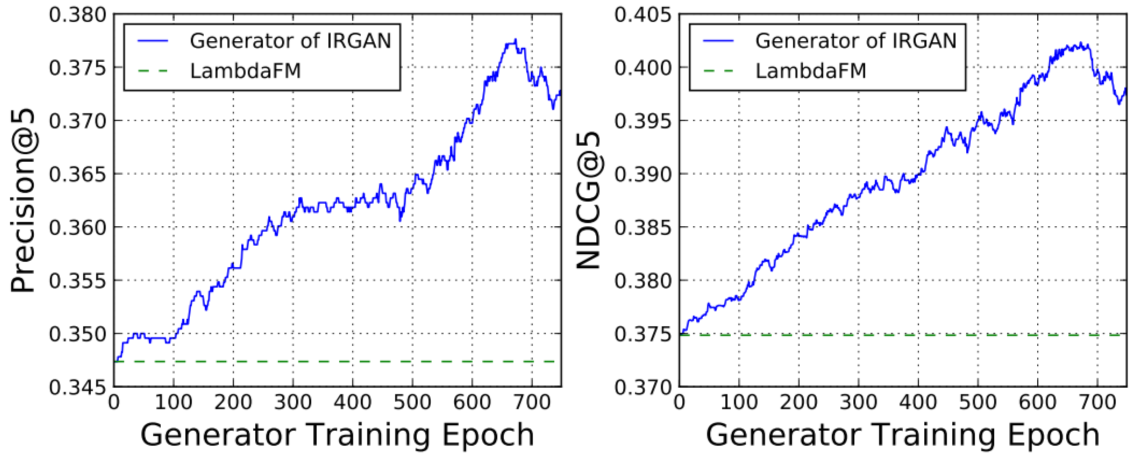
IRGAN-pointwise Generator Performance on Netflix

	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.

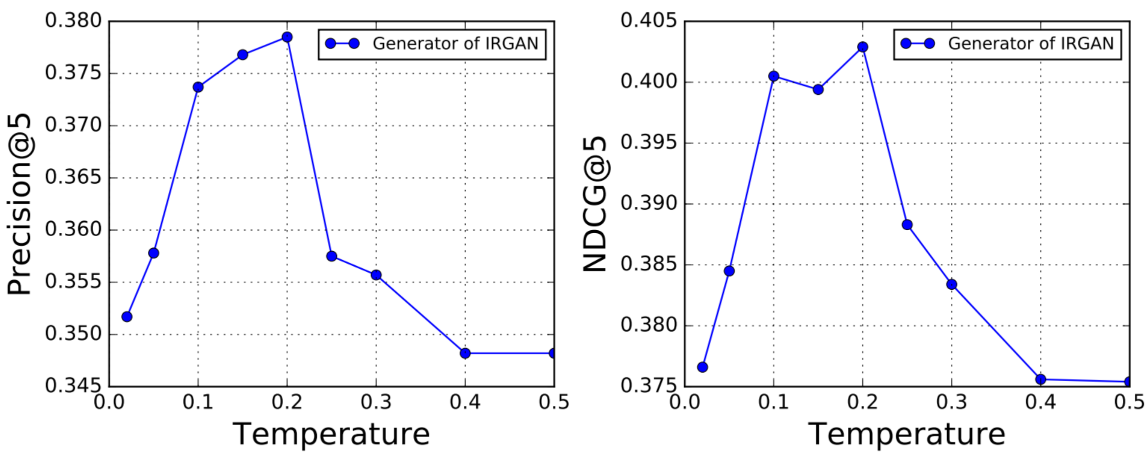
Experiments: Item Recommendation



Top-5 item recommendation task on Movielens

Key observations

- A reliable training process where IRGAN owns a consistent superiority over LambdaFM from the beginning of adversarial training



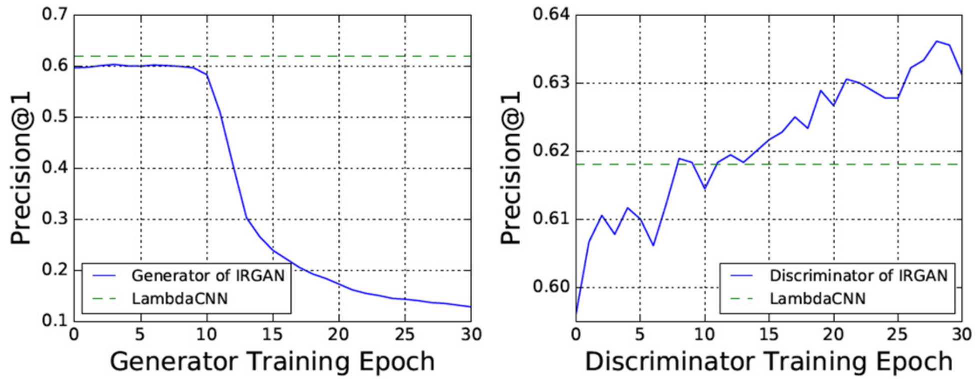
Temperature hyperparameter tuning

- The empirically optimal sampling temperature is 0.2 but not 0 or 1
- Such a low temperature means optimal ranking is achieved by setting a low (but not none) randomness

Experiments: Question Answering

Table 5: The Precision@1 of InsuranceQA.

	test-1	test-2
QA-CNN [9]	0.6133	0.5689
LambdaCNN [9, 49]	0.6183	0.5838
IRGAN-pairwise	0.6383	0.5978
Impv-pairwise	3.23%*	2.74%

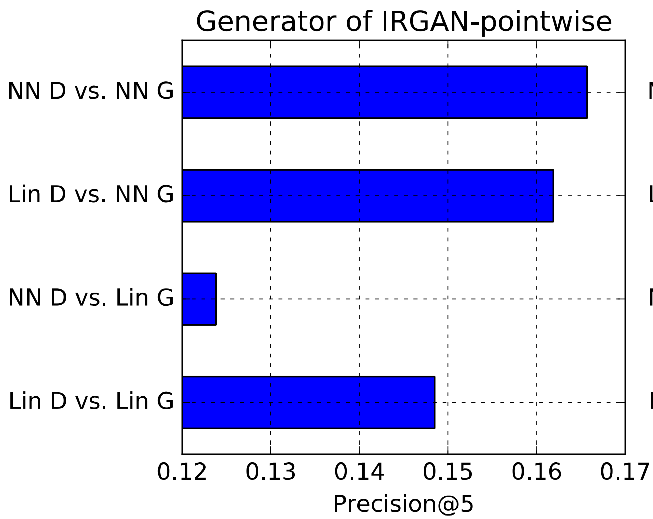


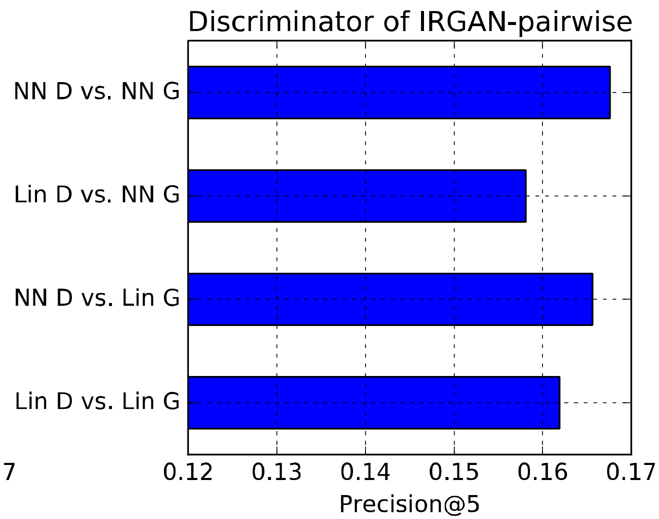
G and D performance on InsuranceQA

- InsuranceQA Dataset
  - 12k question answer pairs
  - Two test sets with 1.8k pairs
- Task
  - rank top-1 answer for each question

- Key observations
  - Discriminator performs better than LambdaCNN while the generator tends to perform less effectively
  - The reason could be the high sparsity of the answer distribution

Different generator and discriminator scoring functions





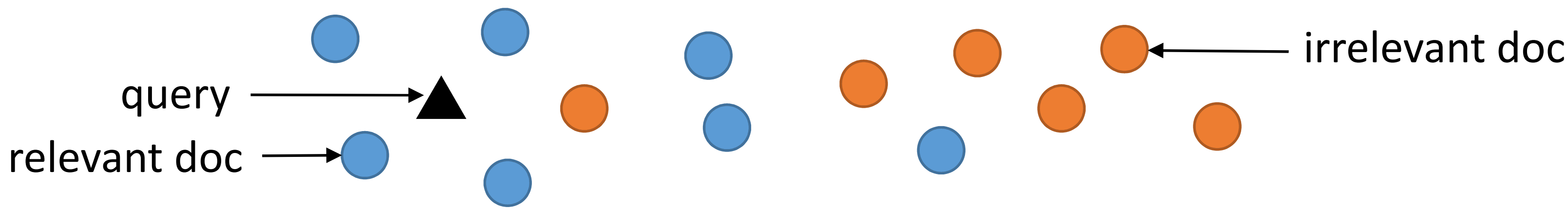
a) For IRGAN-pointwise, the NN implemented generator works be better than its linear version, while the NN implemented discriminator may not offer a good guidance if the generator has lower model complexity (i.e. linear).

b) For IRGAN-pairwise, the NN implemented discriminator outperforms its linear version. The one used for performing the prediction should be implemented with a capacity at least as high as its opponent.

On the Equilibrium of Query Reformulation and Document Retrieval

Shihao Zou, Guanyu Tao, Jun Wang, Weinan Zhang, Dell Zhang. On the Equilibrium of Query Reformulation and Document Retrieval. ICTIR 2018.

Two Challenges in Information Retrieval



- How to formulate optimal queries to best represent the user's information needs → Query reformulation (relevance feedback)

- Relevance estimation for the document given the information need representation → Retrieval model

Equilibrium theory of information retrieval

- a strategic game, simultaneously played between the query reformulation and the retrieval model

Intuition

- The query reformulation would refine the query that is the best response to the actions from the given retrieval model player
- The retrieval model would also need to produce the document relevant estimation that is the best response toward the formulated query
- Two components cooperate to achieve the best response to each other. (an equilibrium state)

Definition: IR Strategic Game

- $P = \{Q, M\}$ is the set of two players: query formulator Q and retrieval model M.
- $S = S_Q \times S_M$ are finite sets of strategies available to player Q and M.
  - $s_q \in S_Q$ denotes whether the term is included in the query or not.
  - $s_m \in S_M$ denotes relevance estimation by retrieval model.
- An equilibrium state: both players have no incentive to change their strategies s_m^* and s_q^*, so that

$$u_Q\left(s_q^*, s_m^*\right) \geq u_Q\left(s_q, s_m^*\right),\ u_M\left(s_q^*, s_m^*\right) \geq u_M\left(s_q^*, s_m\right)$$

IR Game with Relevance Feedback

- Common utility

$$u(\mathbf{s}_q, \mathbf{s}_m) = \frac{1}{|D_r|} \sum_{\mathbf{d}_i \in D_r} \log p(r = 1|\mathbf{d}_i, \mathbf{q}; \phi) - \frac{1}{|D_n|} \sum_{\mathbf{d}_i \in D_n} \log p(r = 0|\mathbf{d}_i, \mathbf{q}; \phi),$$

- A toy example

Table 1: An IR game example (relevance feedback).

	$\mathbf{d}_1$	$\mathbf{d}_2$
t_1	1	0
t_2	0	1
r	1	0

(a) Corpus

	$\mathbf{s}_{m1} = \{1, 0.2\}$	$\mathbf{s}_{m2} = \{0.2, 1\}$
$\mathbf{s}_{q1} = \{1, 0\}$	-1.0064	-1.2913
$\mathbf{s}_{q2} = \{0, 1\}$	-1.4913	-2.0064

(b) Utilities of Strategies

$$p(r = 1|\mathbf{d}_i, \mathbf{q}; \phi) = \text{sigmoid}\Big(\phi_1 \mathbf{q}_1 \mathbf{d}_{i1} + \phi_2 \mathbf{q}_2 \mathbf{d}_{i2}\Big)$$

$$u(\mathbf{s}_q, \mathbf{s}_m) = \log p(r = 1|\mathbf{d}_1, \mathbf{q}; \phi) + \log p(r = 0|\mathbf{d}_2, \mathbf{q}; \phi) = -1.0064$$

IR Game with Pseudo Relevance Feedback

- Utility for retrieval model

$$u_M(\mathbf{s}_q, \mathbf{s}_m) = \frac{1}{|D_r|} \sum_{\mathrm{d}_i \in D_r} \log p(r = 1|\mathbf{d}_i, \mathbf{q}; \phi) - \frac{1}{|D_n|} \sum_{\mathrm{d}_i \in D_n} \log p(r = 0|\mathbf{d}_i, \mathbf{q}; \phi).$$

- Utility for query reformulation

$$u_Q(\mathbf{s}_q, \mathbf{s}_m) = \frac{1}{|D_k|} \sum_{\mathrm{d}_i \in D_k} \log p(r = 1|\mathbf{d}_i, \mathbf{q}; \phi) - \frac{1}{N - |D_k|} \sum_{\mathrm{d}_i \notin D_k} \log p(r = 0|\mathbf{d}_i, \mathbf{q}; \phi),$$

It makes the two utilities different

IR Game with Pseudo Relevance Feedback

- A toy example

Table 2: An IR game example (pseudo relevance feedback).

	$\mathbf{d}_1$	$\mathbf{d}_2$
t_1	1	0
t_2	0	1
r	1	0

(a) Corpus

	$\mathbf{s}_{m1} = \{1, 0.2\}$	$\mathbf{s}_{m2} = \{0.2, 1\}$
$\mathbf{s}_{q1} = \{1, 0\}$	(-1.0064, -1.0064)	(-1.2913, -1.2913)
$\mathbf{s}_{q2} = \{0, 1\}$	(-1.2913, -1.4913)	(-1.0064, -2.0064)

(b) Utilities of Strategies (u_Q, u_M)

$$u_Q(\mathbf{s}_{q2}, \mathbf{s}_{m1}) = \log p(r = 1|\mathbf{d}_2, \mathbf{q}; \phi) + \log p(r = 0|\mathbf{d}_1, \mathbf{q}; \phi) = -1.2913$$

$$u_M(\mathbf{s}_{q2}, \mathbf{s}_{m1}) = \log p(r = 1|\mathbf{d}_1, \mathbf{q}; \phi) + \log p(r = 0|\mathbf{d}_2, \mathbf{q}; \phi) = -1.4913$$

Three Training Schemes

- Case 1: Query Iteration (Conv-Q)

$$\theta_i = \text{sigmoid}(\mathbf{q}^\top \mathbf{d}_i) = \frac{1}{1 + e^{-\mathbf{q}^\top \mathbf{d}_i}}$$

$$\frac{\partial u_Q(\mathbf{s}_q, \mathbf{s}_m)}{\partial \mathbf{q}} = \frac{1}{|D_r|} \sum_{d_i \in D_r} (1 - \theta_i)\mathbf{d}_i - \frac{1}{|D_n|} \sum_{d_i \in D_n} \theta_i \mathbf{d}_i$$

- Case 2: Retrieval Model Iteration (Conv-M)

$$\theta_i = \text{sigmoid}\left(\sum_{k=1}^{K} w_k \cdot (\mathbf{d}_i^k)^\top \mathbf{q}^k\right)$$

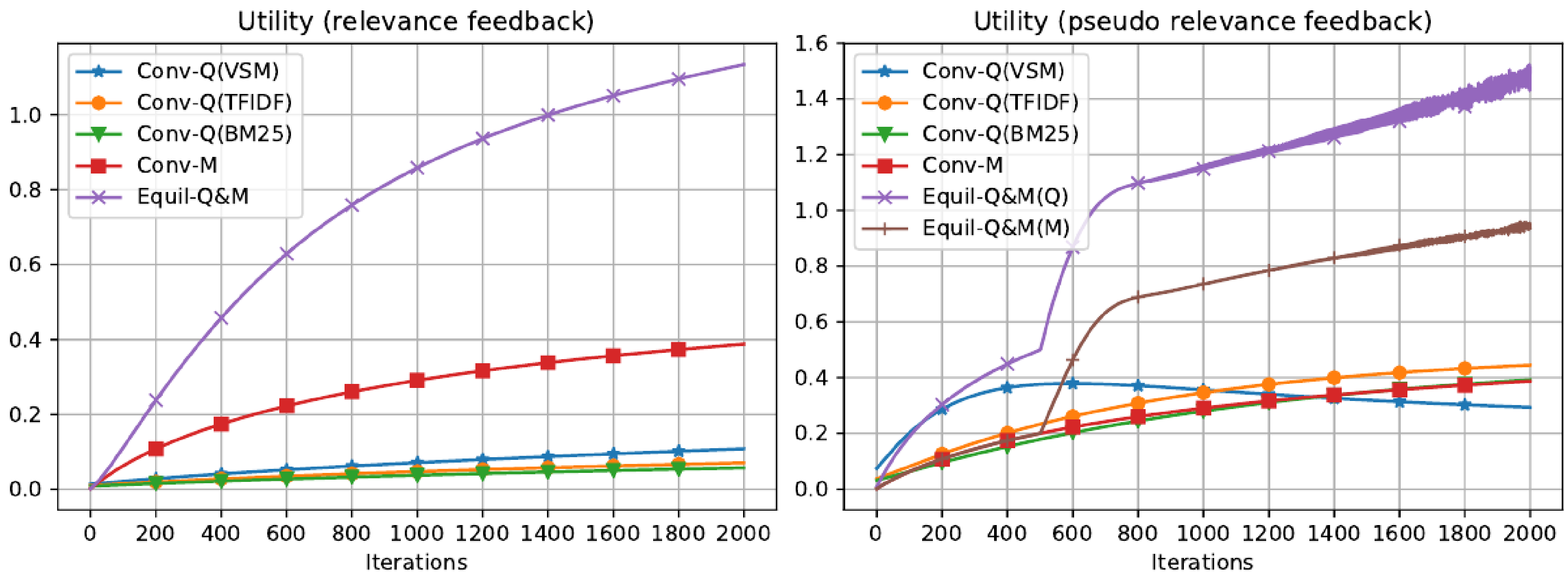
Logistic regression of *K* weight schemes

$$\frac{\partial u_M(\mathbf{s}_q, \mathbf{s}_m)}{\partial w_k} = \frac{1}{|D_r|} \sum_{\mathrm{d}_i \in D_r} (1 - \theta_i) \cdot (\mathbf{d}_i^k)^\top \mathbf{q}^k - \frac{1}{|D_n|} \sum_{\mathrm{d}_i \in D_n} \theta_i \cdot (\mathbf{d}_i^k)^\top \mathbf{q}^k$$

- Case 3: Equilibrium of the Query and Retrieval Model.

Experiment: text retrieval

- Utility in each iteration of training stage



Observations

- Iterations on Q, M and both parts help improve the ranking performance (utility) in training stage

Text retrieval results on RF

Table 3: Text retrieval results (relevance feedback).

Algorithm	NDCG@10 mean±std	NDCG@30 mean±std	MRR mean±std
Naive (VSM)	0.395±0.37	0.412±0.32	0.352±0.38
Naive (TFIDF)	0.511±0.37	0.528±0.33	0.478±0.41
Naive (BM25)	0.504±0.37	0.517±0.32	0.459±0.40
Rocchio (VSM)	0.407±0.37	0.422±0.32	0.367±0.39
Rocchio (TFIDF)	0.519±0.38	0.536±0.33	0.487±0.41
Rocchio (BM25)	0.518±0.37	0.531±0.32	0.474±0.40
Conv-Q (VSM)	0.527±0.34	0.554±0.29	0.475±0.39
Conv-Q (TFIDF)	0.568±0.35	0.571±0.30	0.530±0.40
Conv-Q (BM25)	0.563±0.35	0.573±0.30	0.522±0.40
Conv-M	0.463±0.38	0.482±0.34	0.431±0.41
Equil-Q&M	**0.583±0.34**	**0.601*±0.29**	**0.537±0.39**
	P@10	P@30	MAP
Naive (VSM)	0.152±0.18	0.134±0.15	0.184±0.16
Naive (TFIDF)	0.221±0.22	0.179±0.18	0.263±0.23
Naive (BM25)	0.217±0.22	0.178±0.17	0.262±0.23
Rocchio (VSM)	0.162±0.18	0.139±0.15	0.193±0.17
Rocchio (TFIDF)	0.225±0.22	0.186±0.18	0.276±0.24
Rocchio (BM25)	0.221±0.21	0.183±0.17	0.272±0.24
Conv-Q (VSM)	0.245±0.23	0.212±0.18	0.288±0.22
Conv-Q (TFIDF)	0.264±0.24	0.220±0.20	0.317±0.25
Conv-Q (BM25)	0.265±0.24	0.214±0.20	0.319±0.25
Conv-M	0.190±0.20	0.160±0.16	0.238±0.21
Equil-Q&M	**0.278*±0.24**	**0.233±0.19**	**0.331*±0.25**

- Datasets
  - TREC disks 4 & 5
- Task
  - Text retrieval ranking
- Key observations
  - Conv-Q shows better performance than Naive and Rocchio
  - Conv-M fails to perform well on test set although well on training set
  - The best Equil-Q&M indicates the effectiveness of coordinating Q and M

Text retrieval results on PRF

Table 4: Text retrieval results (pseudo relevance feedback)

Algorithm	NDCG@10 mean±std	NDCG@30 mean±std	MRR mean±std
Naive (VSM)	0.323±0.38	0.378±0.29	0.287±0.36
Naive (TFIDF)	0.463±0.36	0.493±0.30	**0.413±0.38**
Naive (BM25)	0.439±0.35	0.474±0.28	0.375±0.36
Rocchio (VSM)	0.323±0.36	0.378±0.30	0.285±0.36
Rocchio (TFIDF)	0.460±0.36	0.493±0.30	0.410±0.38
Rocchio (BM25)	0.444±0.35	0.477±0.29	0.386±0.37
Conv-Q (VSM)	0.245±0.34	0.308±0.29	0.228±0.33
Conv-Q (TFIDF)	0.428±0.37	0.465±0.32	0.370±0.38
Conv-Q (BM25)	0.400±0.36	0.456±0.30	0.349±0.36
Conv-M	0.415±0.37	0.447±0.31	0.367±0.385
Equil-Q&M	**0.469±0.37**	**0.499±0.31**	0.397±0.38
	P@10	P@30	MAP
Naive (VSM)	0.112±0.14	0.100±0.12	0.158±0.15
Naive (TFIDF)	0.200±0.21	0.142±0.14	0.239±0.22
Naive (BM25)	0.187±0.20	0.137±0.13	0.226±0.21
Rocchio (VSM)	0.108±0.14	0.100±0.12	0.157±0.16
Rocchio (TFIDF)	0.207±0.22	0.145±0.14	0.244±0.23
Rocchio (BM25)	0.193±0.20	0.141±0.14	0.233±0.22
Conv-Q (VSM)	0.095±0.15	0.090±0.12	0.138±0.15
Conv-Q (TFIDF)	0.211±0.23	0.150±0.16	0.253±0.24
Conv-Q (BM25)	0.180±0.21	0.143±0.15	0.234±0.23
Conv-M	0.154±0.17	0.122±0.13	0.205±0.19
Equil-Q&M	**0.223±0.16**	**0.162*±0.16**	**0.257±0.23**

- Datasets
  - TREC disks 4 & 5
- Task
  - Text retrieval ranking
- Key observations
  - Conv-Q shows worse performance than Naive and Rocchio
    - One cannot fully rely on the top-k retrieved docs from the model to update the query
  - The best Equil-Q&M indicates the coordination of Q and M help overcome the issue of bad query representation

Summary of this Part

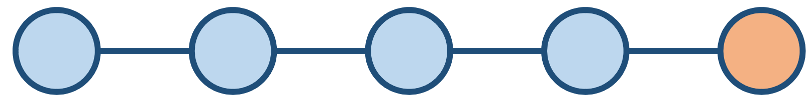
- Study the interactions between query reformulation and retrieval model relevance estimation in a game theoretical framework

- The performance of an equilibrium solution from relevance feedback consistently outperforms others separate cases.

- We shall perform a deeper investigation of the utility design in the proposed normal-form IR

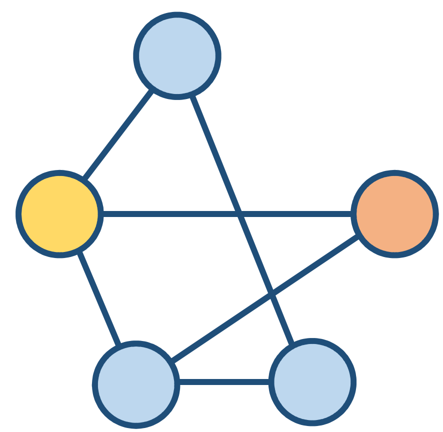
Beyond Single Discrete Token

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



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

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



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

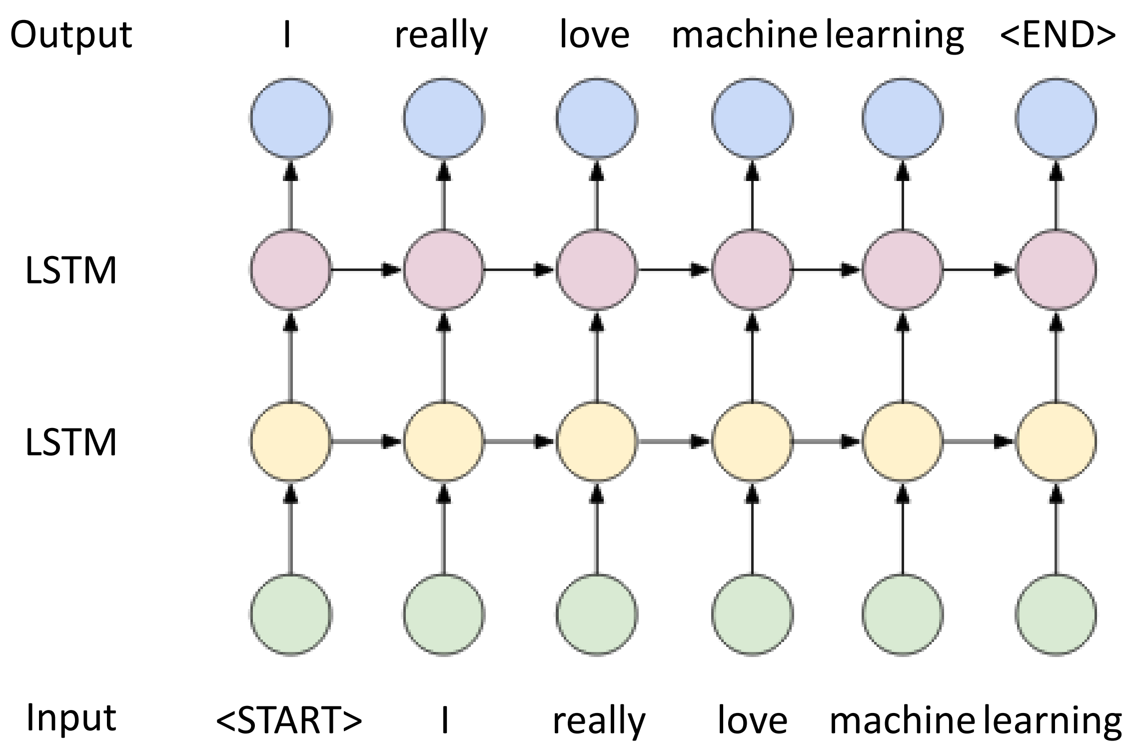
Content of this Tutorial

1. Introduction to Generative Adversarial Nets
2. Reinforcement Learning
3. GANs for Information Retrieval
4. GANs for Text Generation
5. GANs for Graph/Network Learning
6. Beyond GANs, Cooperative Training
7. Future Perspective and Summarization

RNN based Language Model

- Trained via maximum likelihood estimation (MLE)

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



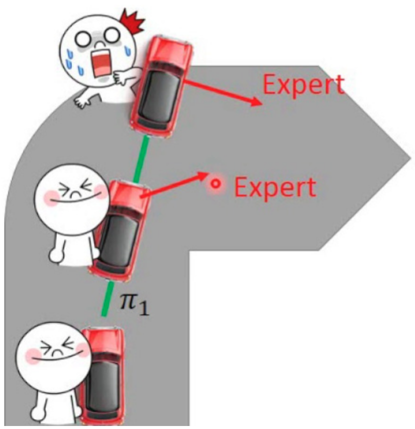
Exposure Bias

- 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

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

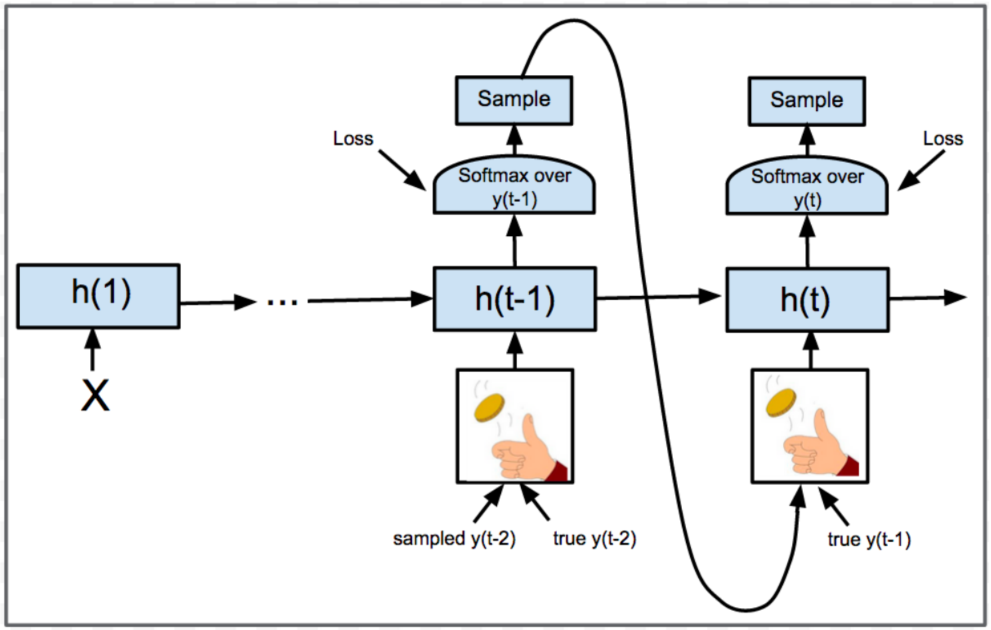


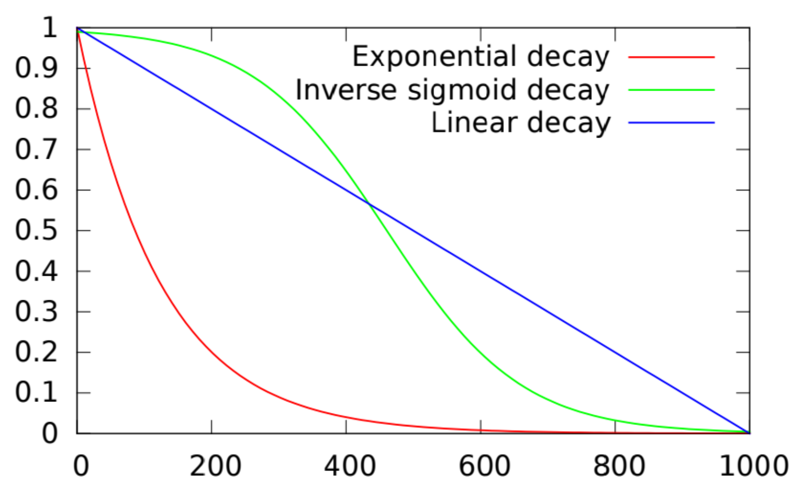
- Similar in self-driving car training
  - Problem of behavior cloning

S Bengio et al. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. NIPS 2015.

Exposure Bias & Schedule Sampling

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

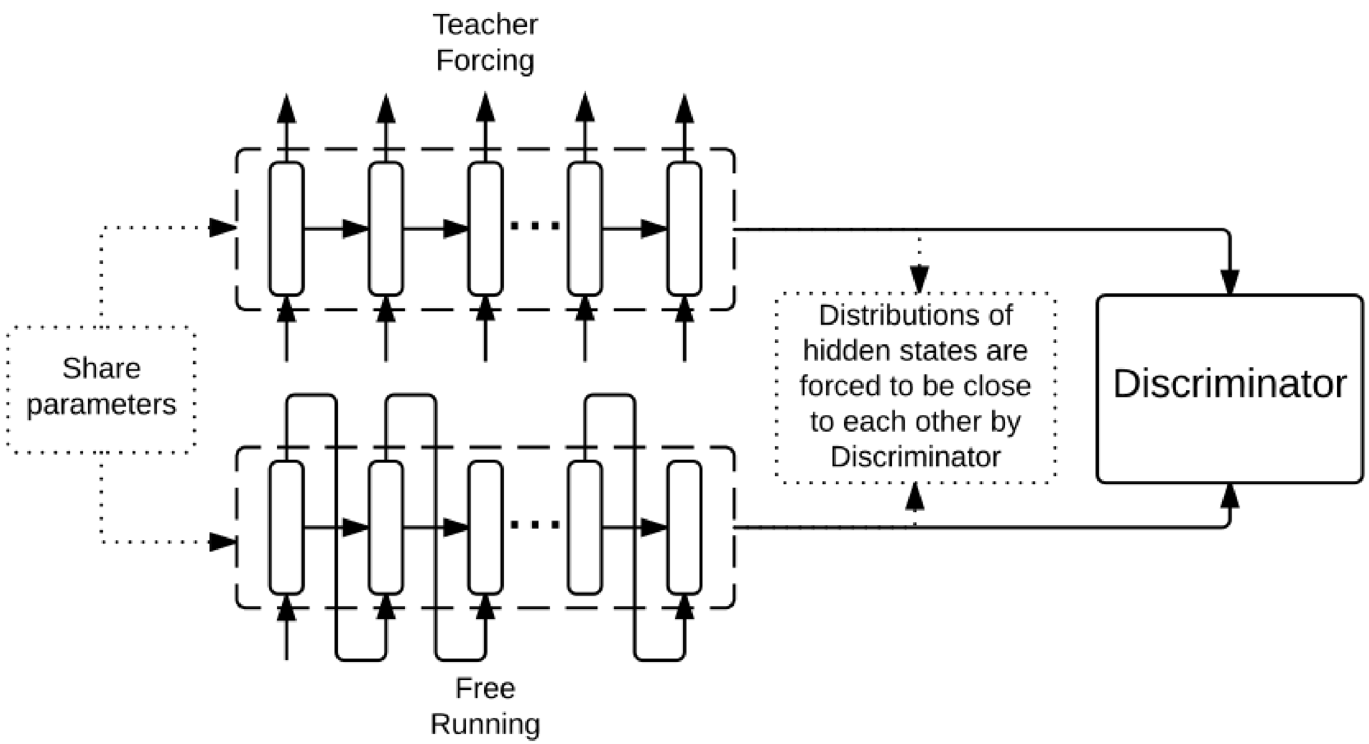




S Bengio et al. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. NIPS 2015.

Professor Forcing

- Professor Forcing reduces the gap between teacher forcing and free generation procedures.



Lamb, Alex M., et al. "Professor forcing: A new algorithm for training recurrent networks." NIPS 2016.

REVIEW

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 true 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 true as possible
- More straightforward but it is hard or impossible to directly calculate $p(x)$

Generative Adversarial Nets (GANs)

- What we really want

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

- But we cannot directly calculate $p(x)$

- Idea: what if we build a discriminator to judge whether a data instance is true or fake (artificially generated)?
  - Leverage the strong power of deep learning based discriminative models

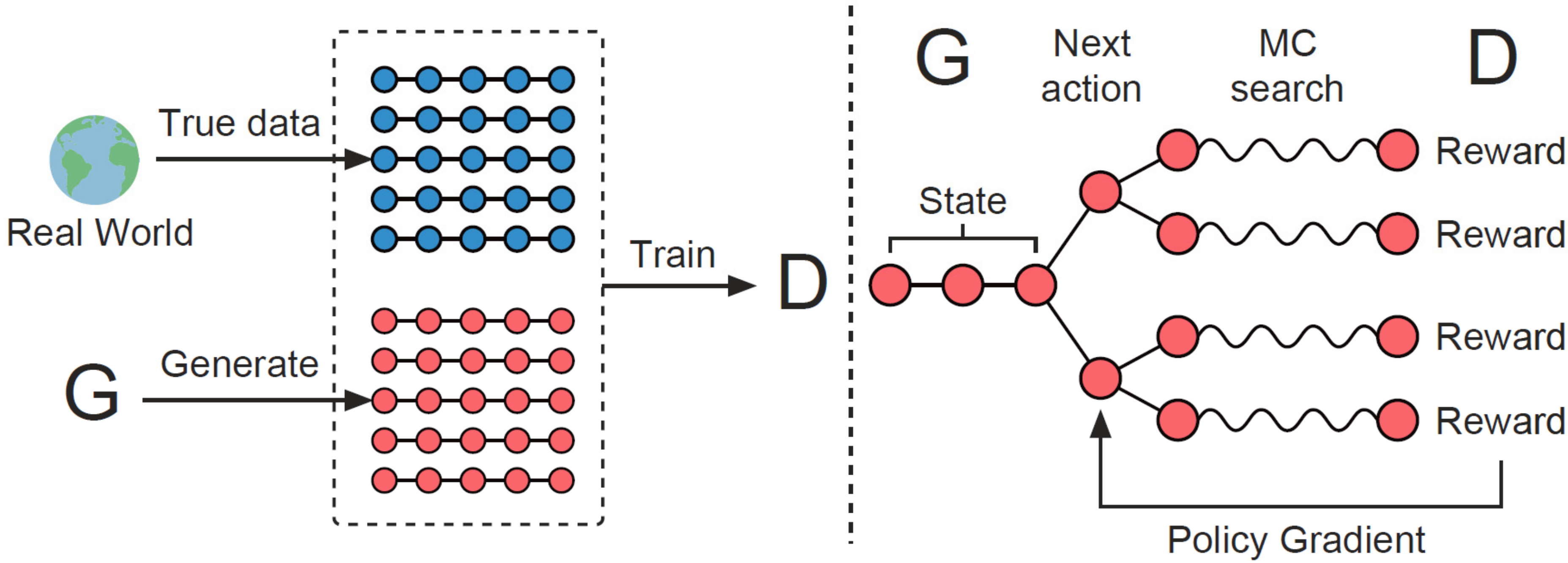
[Goodfellow, I., et al. 2014. Generative adversarial nets. In NIPS 2014.]

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 true data) $D_\phi(Y^n_{1:T})$ for the whole sequence

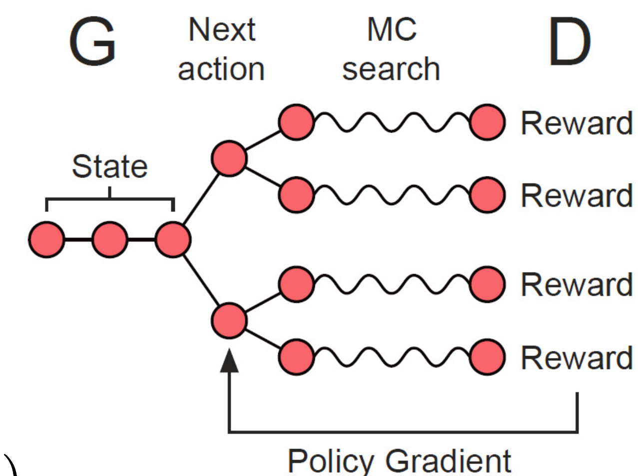
Sequence Generator

- Objective: to maximize the expected reward

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

  - State-action value function $Q_{D_\phi}^{G_\theta}(s, a)$ is the expected accumulative reward that
    - Start from state *s*
    - Taking action *a*
    - And following 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

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

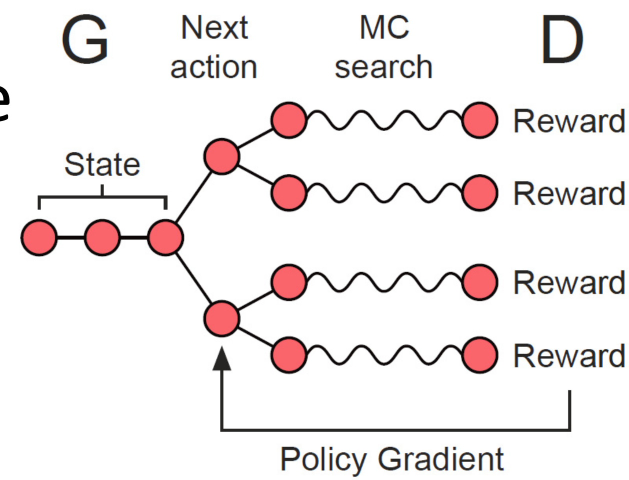
- For intermediate state-action value
  - Use Monte Carlo search to estimate

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

  - Following a roll-out policy G_θ

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

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



Training Sequence Generator

- Policy gradient (REINFORCE)

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

$$\simeq \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t\in\mathcal{Y}} \nabla_\theta G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)$$

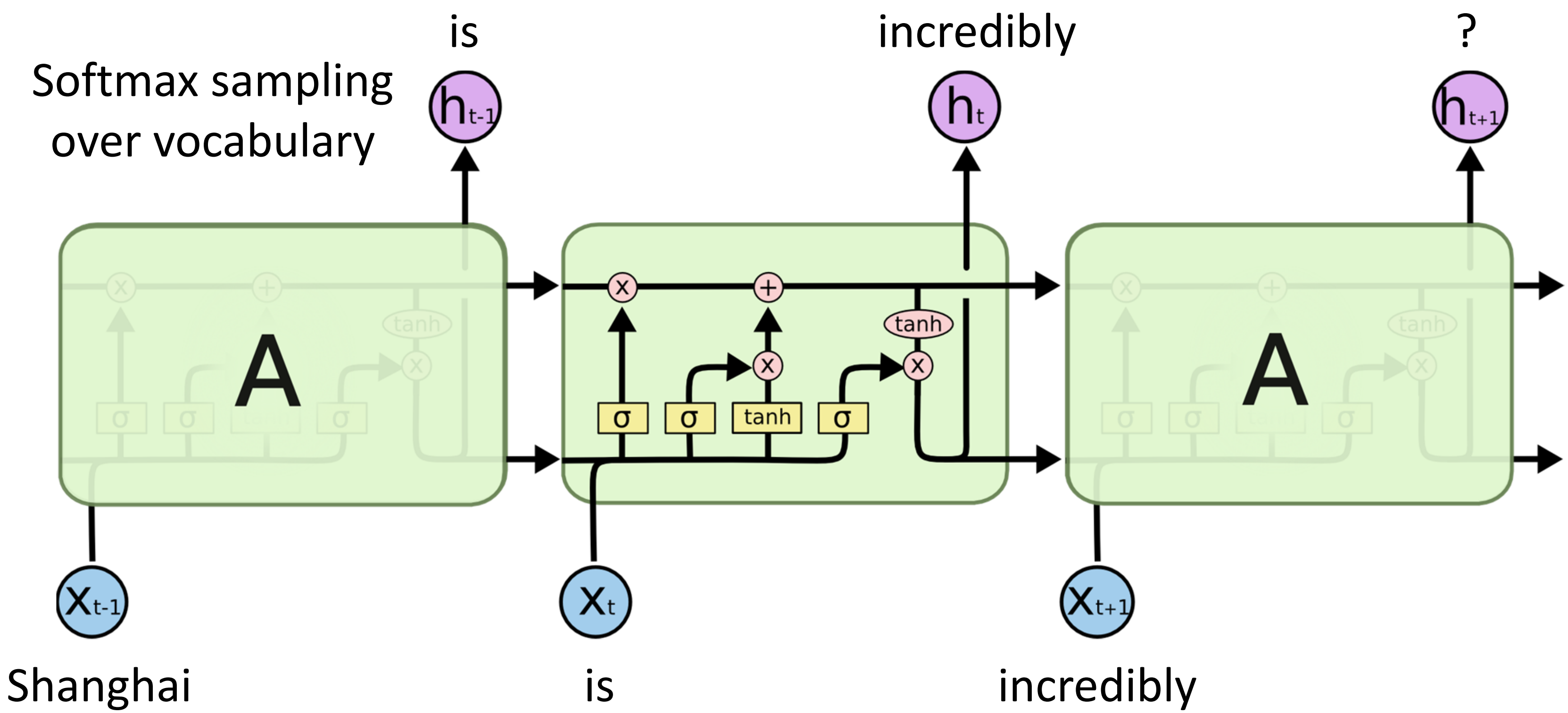
$$= \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t\in\mathcal{Y}} G_\theta(y_t|Y_{1:t-1})\nabla_\theta \log G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)$$

$$= \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{y_t\sim G_\theta(y_t|Y_{1:t-1})}[\nabla_\theta \log G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)]$$

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

Richard Sutton et al. Policy Gradient Methods for Reinforcement Learning with Function Approximation. NIPS 1999.

Sequence Generator Model



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

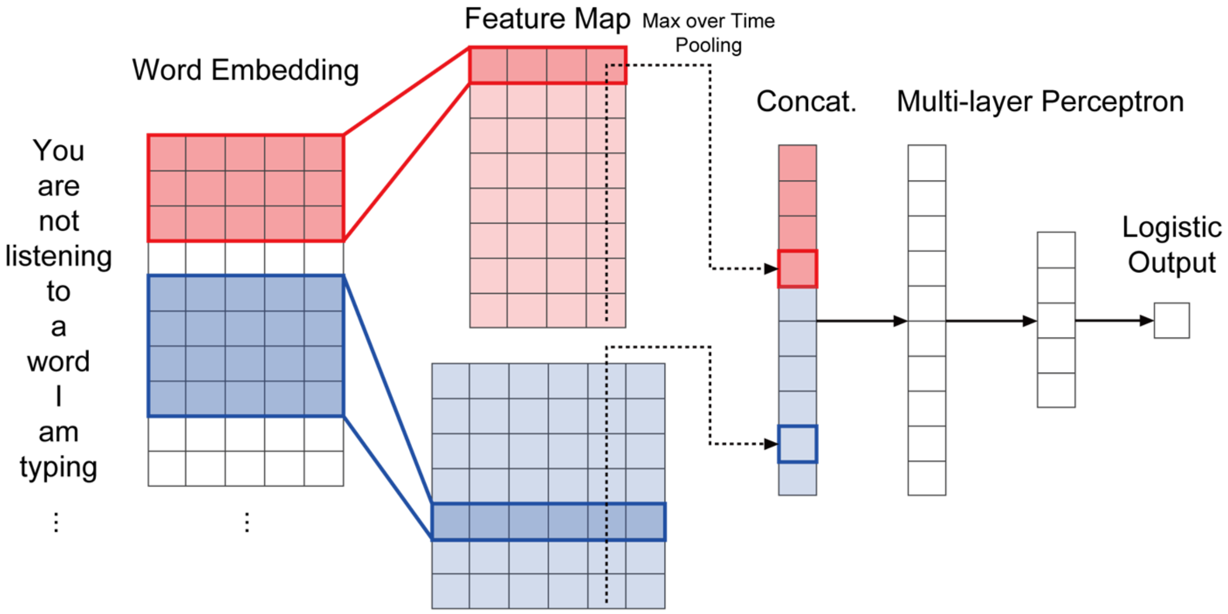
[Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. Neural computation 9(8):1735–1780.]

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



[Kim, Y. 2014. Convolutional neural networks for sentence classification. EMNLP 2014.]

Overall Algorithm

Algorithm 1 Sequence Generative Adversarial Nets

Require: generator policy G_θ; roll-out policy G_β; discriminator D_ϕ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

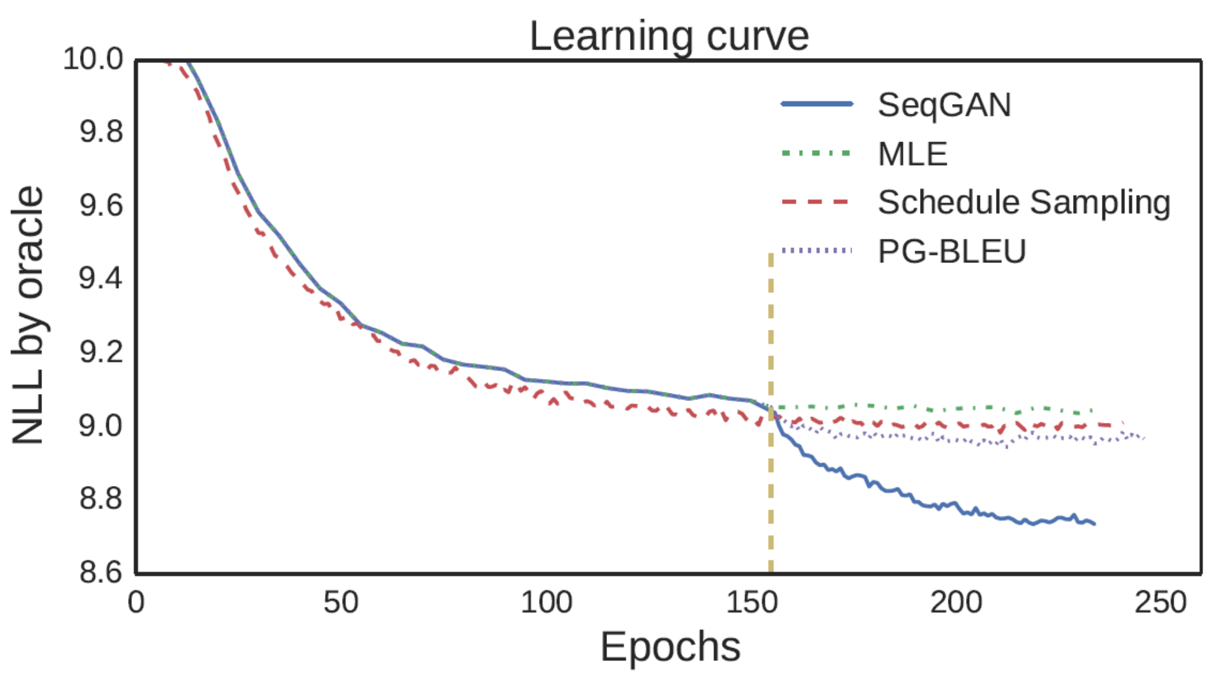
Initialize G_θ, D_ϕ with random weights θ, ϕ.
Pre-train G_θ using MLE on $\mathcal{S}$
$\beta \leftarrow \theta$
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ via minimizing the cross entropy
repeat
    for g-steps **do**
        Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_\theta$
        for t in $1 : T$ **do**
            Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
        end for
        Update generator parameters via policy gradient Eq. (8)
    end for
    for d-steps **do**
        Use current G_θ to generate negative examples and combine with given positive examples $\mathcal{S}$
        Train discriminator D_ϕ for k epochs by Eq. (5)
    end for
    $\beta \leftarrow \theta$
until SeqGAN converges

Experiments on Synthetic Data

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

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

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

Human	Machine
南陌春风早，东邻去日斜。	山夜有雪寒，桂里逢客时。
紫陌追随日，青门相见时。	此时人且饮，酒愁一节梦。
胡风不开花，四气多作雪。	四面客归路，桂花开青竹。

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

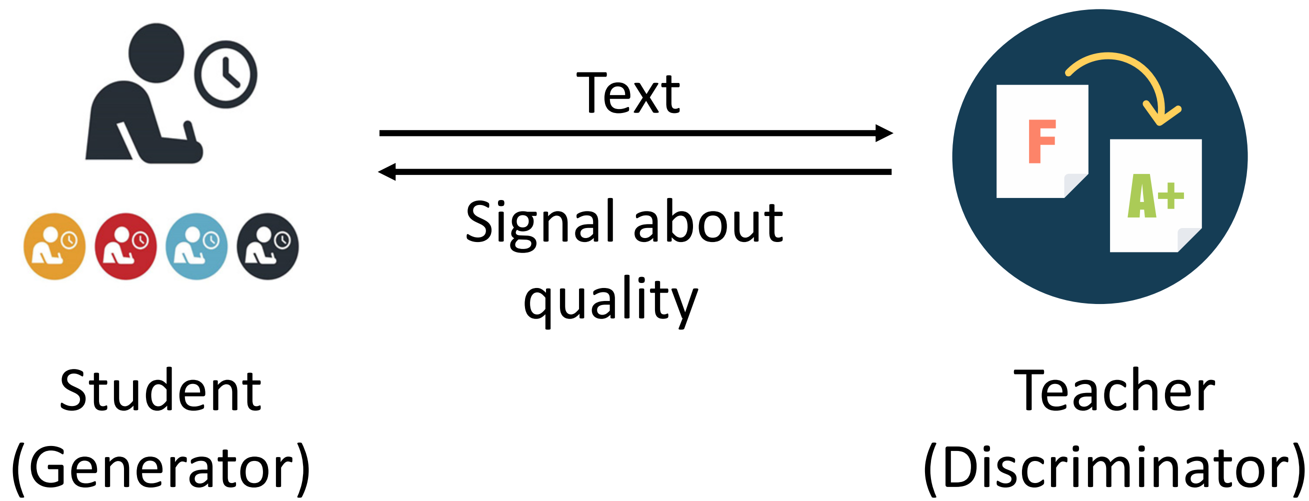
LeakGAN: Long Text Generation via Adversarial Training with Leaked Information

- [Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, Jun Wang. Long Text Generation via Adversarial Training with Leaked Information. AAAI 2018.]

  https://arxiv.org/abs/1709.08624

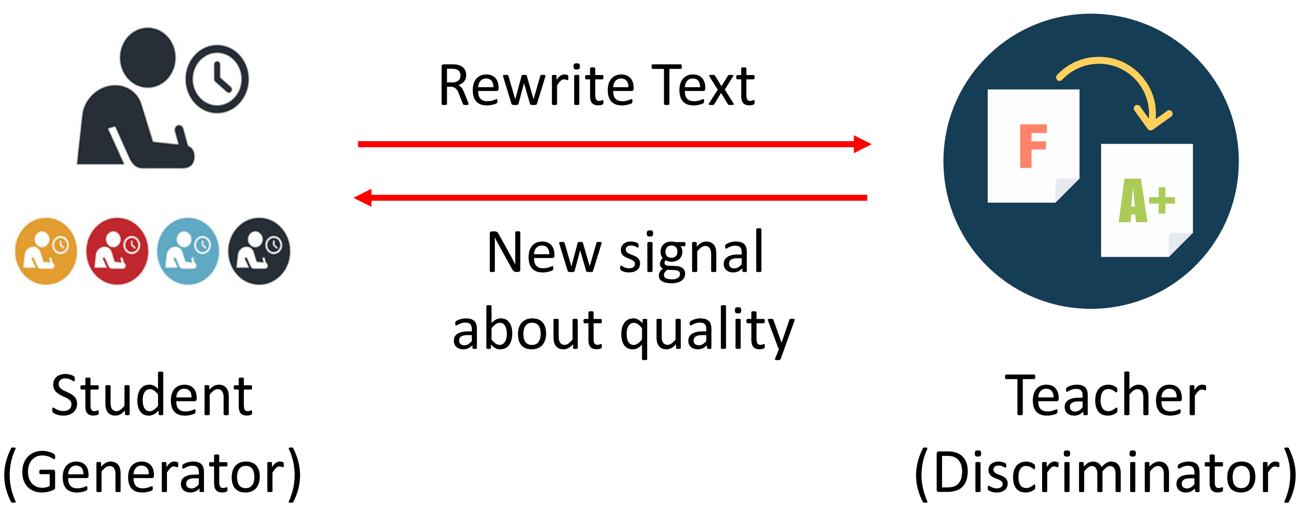
Main Weakness in SeqGAN

- Huge search space: If the vocabulary is 5000, and the text length is 10, the text's number will be up to 5000^{10}, it needs many attempts to find a good policy.



Main Weakness in SeqGAN

- Huge search space: If the vocabulary is 5000, and the text length is 10, the text's number will be up to 5000^{10}, it needs many attempts to find a good policy.

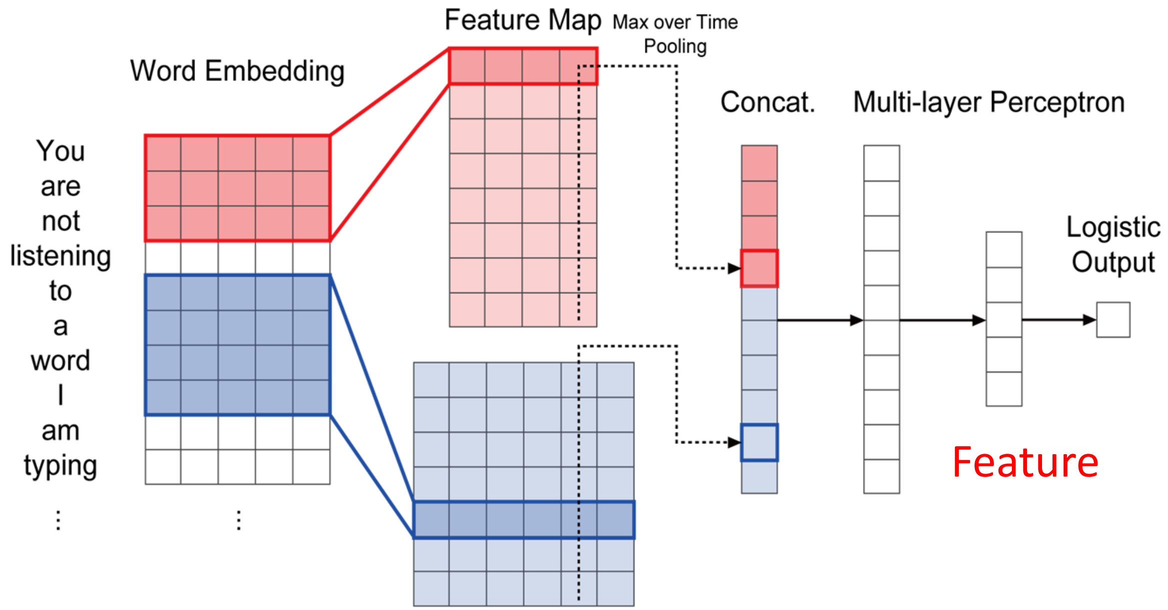


Generator only knows if the text is good, but don't know why!
Huge search cost!
We need more guide information.

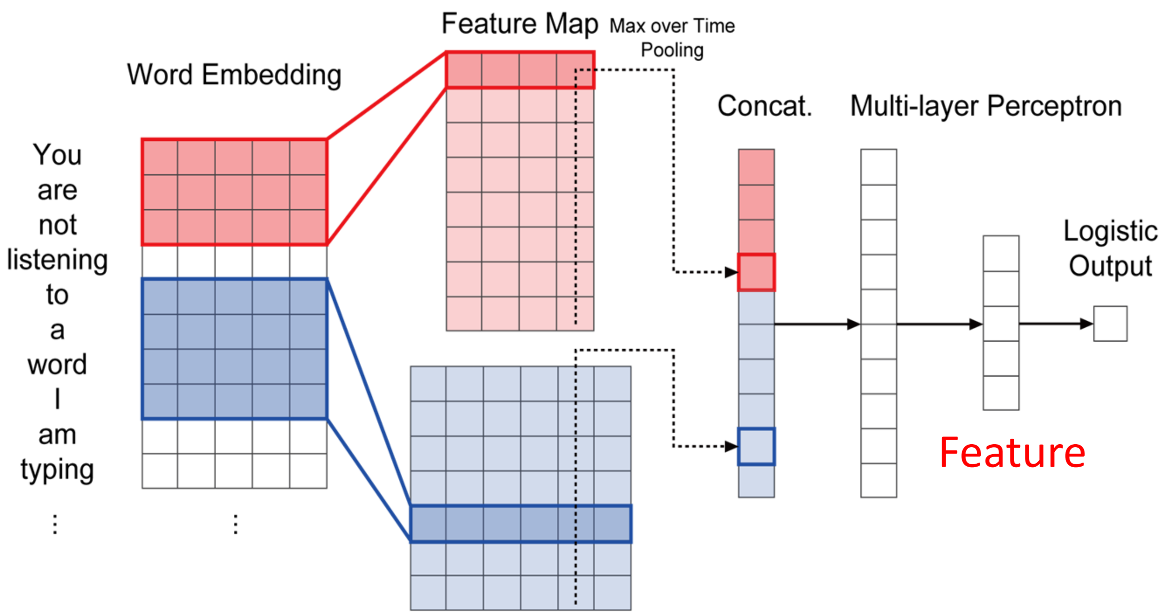
Recall the 2 Players' Tasks

- Generator task: generate high quality's text to fool discriminator.

- Discriminator task: identify the text according to the feature extracted by itself

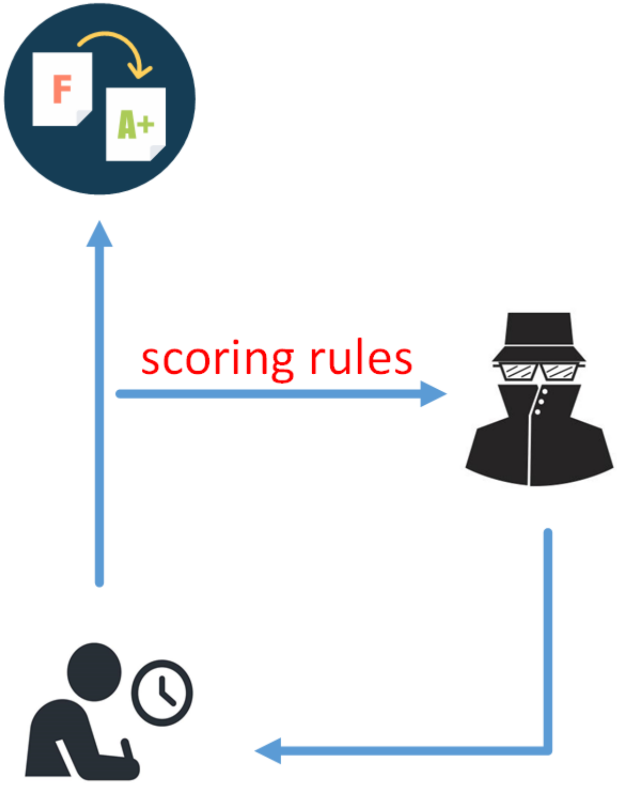


The criterion is the feature in the discriminator



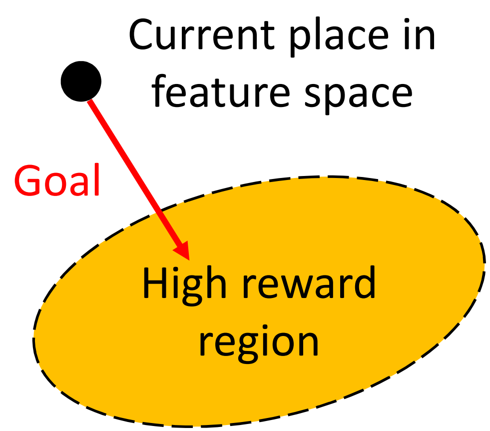
Information Leaking

- If this criterion is leaked to Generator just like student knows the scoring rules. It may reduce the search space.



How to utilize the feature from D?

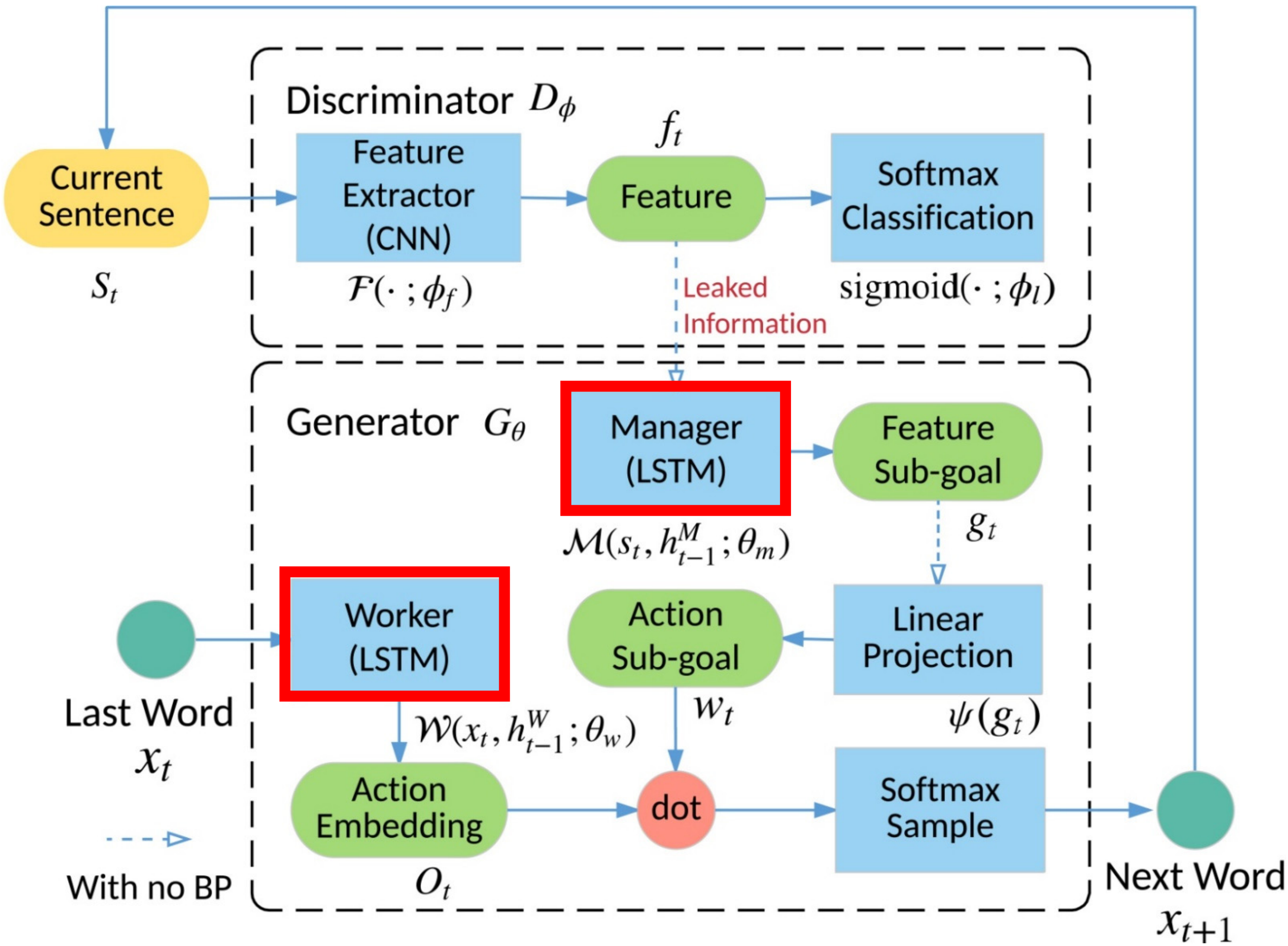
- If G wants to fool D, G needs to generate more realistic text to obtain more realistic feature.
- During the generation, we know current feature, we need to choose next action to obtain more realistic feature, i.e., we need to find a higher reward region in feature space.



- We refer the hierarchical reinforcement learning like FeUdal Networks [Vezhnevets 2017]

Vezhnevets, Alexander Sasha, et al. "Feudal networks for hierarchical reinforcement learning." ICML 2017.

LeakGAN Framework



Manager: receives current text's feature from D, outputs a direction of higher reward feature as the sub-goal vector for Worker.

Worker: produces primitive actions to follow the sub-goals from Manager.

Training of Hierarchical RL

- Manager: a high-level transition policy gradient

$$\pi^{TP}(f_{t+c}|f_t) = p(f_{t+c}|f_t, g_t(\theta_m))$$

what new feature will
be after taking action
and transit c steps

  - Use Mises-Fisher distribution to implement

$$p(f_{t+c}|f_c, g_t(\theta_m)) \propto e^{d_{\cos}(f_{t+c}-f_t,\ g_t(\theta_m))}$$

  - Policy gradient (with likelihood ratio)

$$\nabla_{\theta_m}^{\text{adv}} g_t = -Q_{\mathcal{F}}(s_t, g_t)\nabla_{\theta_m} d_{\cos}\Big(f_{t+c} - f_t, g_t(\theta_m)\Big)$$

  - Value function by Monte Carlo search

$$Q_{\mathcal{F}}(s_t, g_t) = Q(\mathcal{F}(s_t), g_t) = Q(f_t, g_t) = \mathbb{E}[r_t]$$

Vezhnevets, Alexander Sasha, et al. "Feudal networks for hierarchical reinforcement learning." ICML 2017.

Training of Hierarchical RL

- For Generator, we train the Manager and Worker independently to let them focus on their own task

- Worker is trained like SeqGAN

$$\nabla_{\theta_w} \mathbb{E}_{s_{t-1} \sim G} \Big[\sum_{x_t} r_t^I \mathcal{W}(x_t | s_{t-1}; \theta_w) \Big]$$
$$= \mathbb{E}_{s_{t-1} \sim G, x_t \sim \mathcal{W}(x_t | s_{t-1})} [r_t^I \nabla_{\theta_w} \log \mathcal{W}(x_t | s_{t-1}; \theta_w)]$$

where the intrinsic reward is

$$r_t^I = \frac{1}{c} \sum_{i=1}^{c} d_{\cos}(f_t - f_{t-i}, g_{t-i})$$

to follow the Manager's sub-goal.

  - The training method is REINFORCE.

Experiment

- Synthetic data experiments

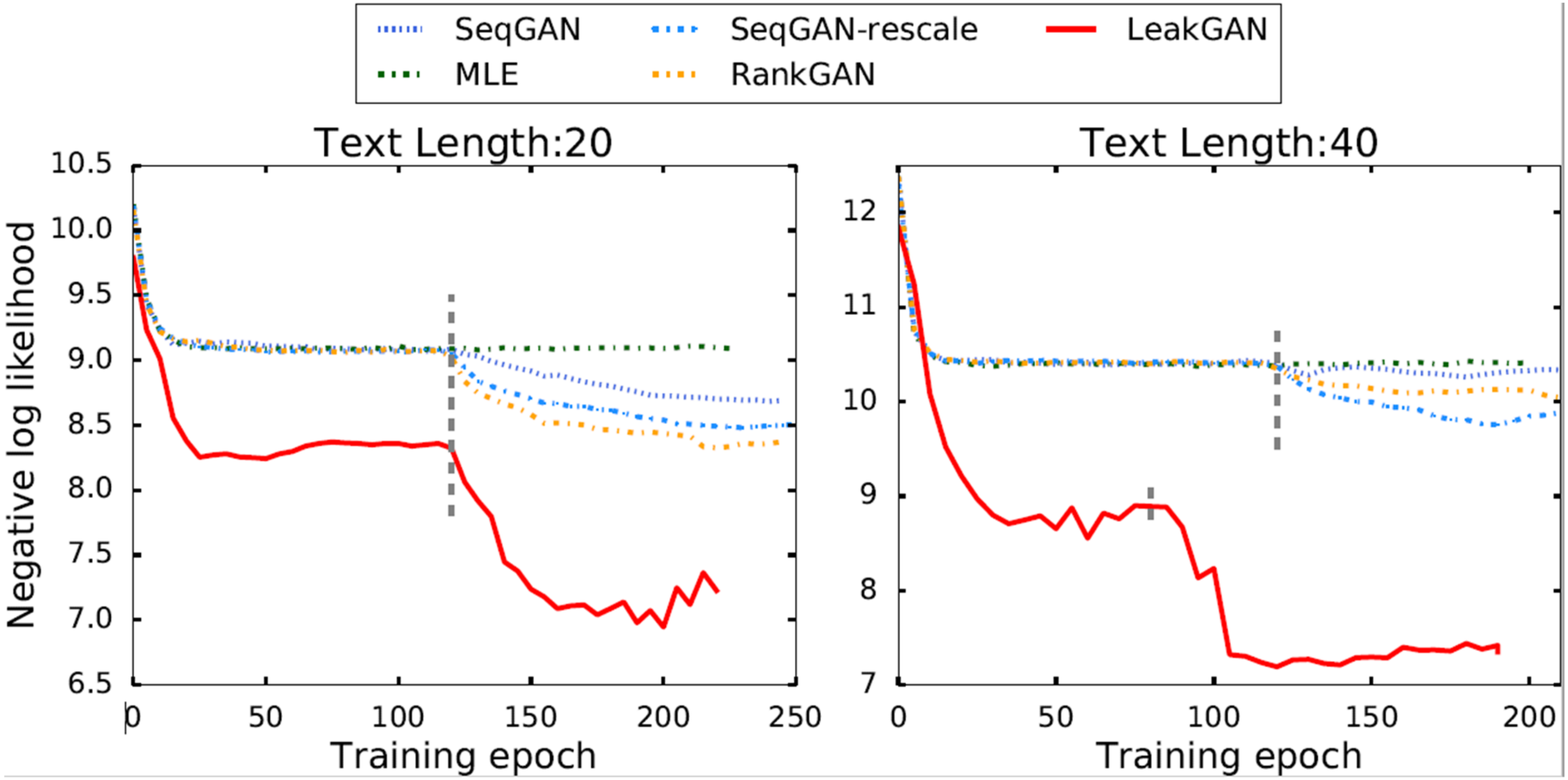


Table 1: The over NLL performance on synthetic data.

Length	MLE	SeqGAN	RankGAN	LeakGAN	Real	p-value
20	9.038	8.736	8.247	**7.038**	5.750	$< 10^{-6}$
40	10.411	10.310	9.958	**7.191**	4.071	$< 10^{-6}$

Experiment

- BLEU Scores in real data
  - Short Text (Chinese Poem):

Table 4: The BLEU performance on Chinese Poems.

Method	SeqGAN	RankGAN	LeakGAN
BLEU-2	0.738	0.812	**0.881**
p-value	$< 10^{-6}$	$< 10^{-6}$	-

  - Middle Text (Image COCO):

Table 3: BLEU scores on COCO Image Captions.

Method	SeqGAN	RankGAN	LeakGAN	p-value
BLEU-2	0.831	0.850	**0.950**	$< 10^{-6}$
BLEU-3	0.642	0.672	**0.880**	$< 10^{-6}$
BLEU-4	0.521	0.557	**0.778**	$< 10^{-6}$
BLEU-5	0.427	0.544	**0.686**	$< 10^{-6}$

Experiment

- BLEU Scores
  - Long Text (EMNLP WMT 2017):

Table 2: BLEU scores performance on EMNLP2017 WMT.

Method	SeqGAN	RankGAN	LeakGAN	p-value
BLEU-2	0.8590	0.778	**0.956**	$< 10^{-6}$
BLEU-3	0.6015	0.478	**0.819**	$< 10^{-6}$
BLEU-4	0.4541	0.411	**0.627**	$< 10^{-6}$
BLEU-5	0.4498	0.463	**0.498**	$< 10^{-6}$

- Human study

Table 5: Turing test results for in real-world experiments.

Dataset	SeqGAN	LeakGAN	Ground Truth	p-value
WMT News	0.236	**0.554**	0.651	$< 10^{-6}$
COCO	0.405	**0.574**	0.675	$< 10^{-6}$

Experiment

- BLEU improvements over baseline models

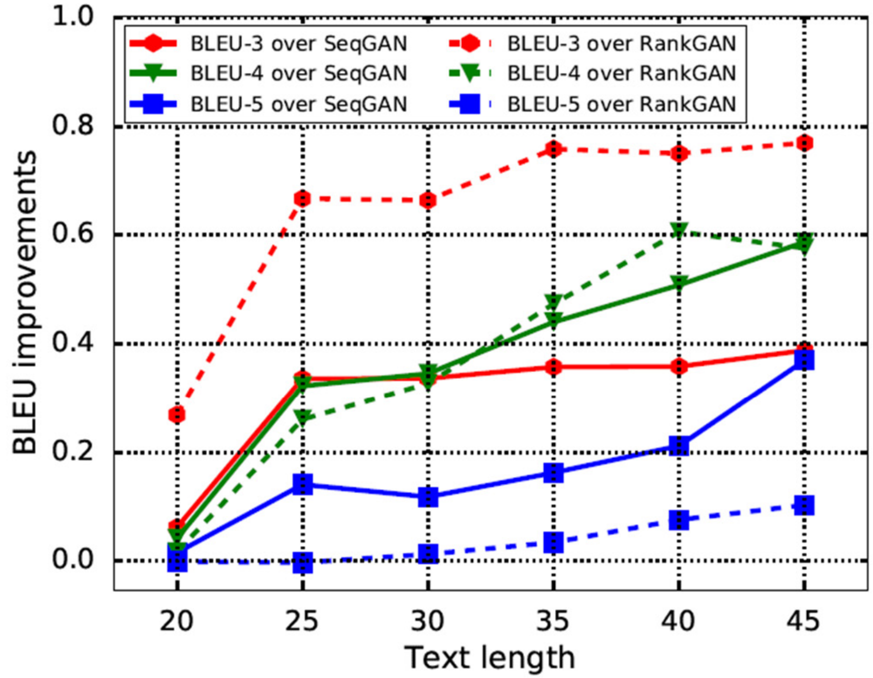


Figure 3: The illustration of BLEU improvement change along with the generated text length on WMT News.

The curves clearly show that LeakGAN yields larger performance gain over the baselines when the generated sentences are longer.

Image COCO Caption Text Generation

- A woman holding an umbrella while standing against a sidewalk.
- The bathroom is clean and ready for us to use .

LeakGAN

- Several metal balls sit in the sand near a group of people.
- A phone lies on the counter in a modern kitchen.

Human

- A silver stove, the refrigerator, sitting in a kitchen.
- A cat and a woman standing by two computer preparing food.

SeqGAN

WMT 2017 News Text Generation

- I also think that’s a good place for us, I’m sure that this would be a good opportunity for me to get in touch.
- That’ s why we’re the most important people for the African American community and we’ve made a good response.

LeakGAN

- This is a part of the population that is notorious for its lack of interest in actually showing up when the political process takes place.
- I was paid far too little to pick up a dead off of the ground and put it back in the box.

Human

- “I think you should really really leave for because we hadn’t been busy, where it goes to one,” he wrote.
- We are thinking about 40, 000 and jobs in what is wrong in the coming and you know.

SeqGAN

Explanation

- Feature Traces

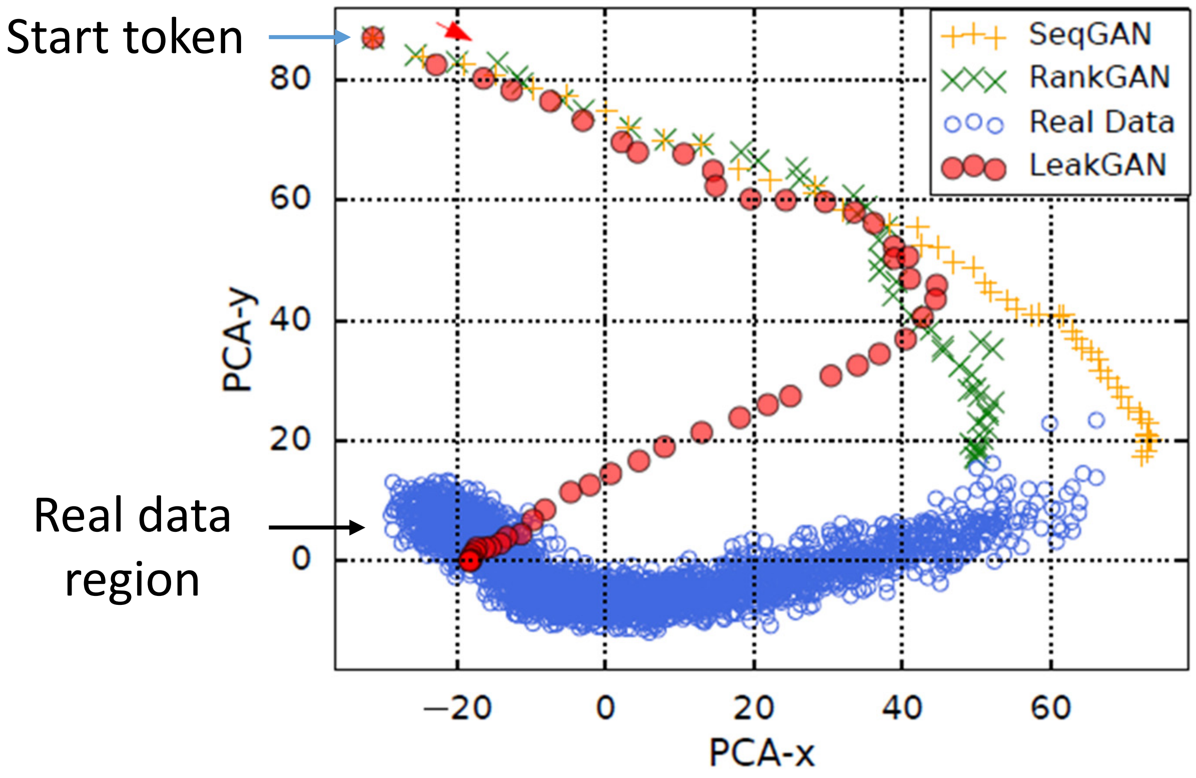


Figure 4: Feature traces during the generation (SeqGAN, RankGAN and LeakGAN) and features of completed real data (all compressed to 2-dim by PCA) on WMT News.

Explanation

- Behaviors of Worker and Manager

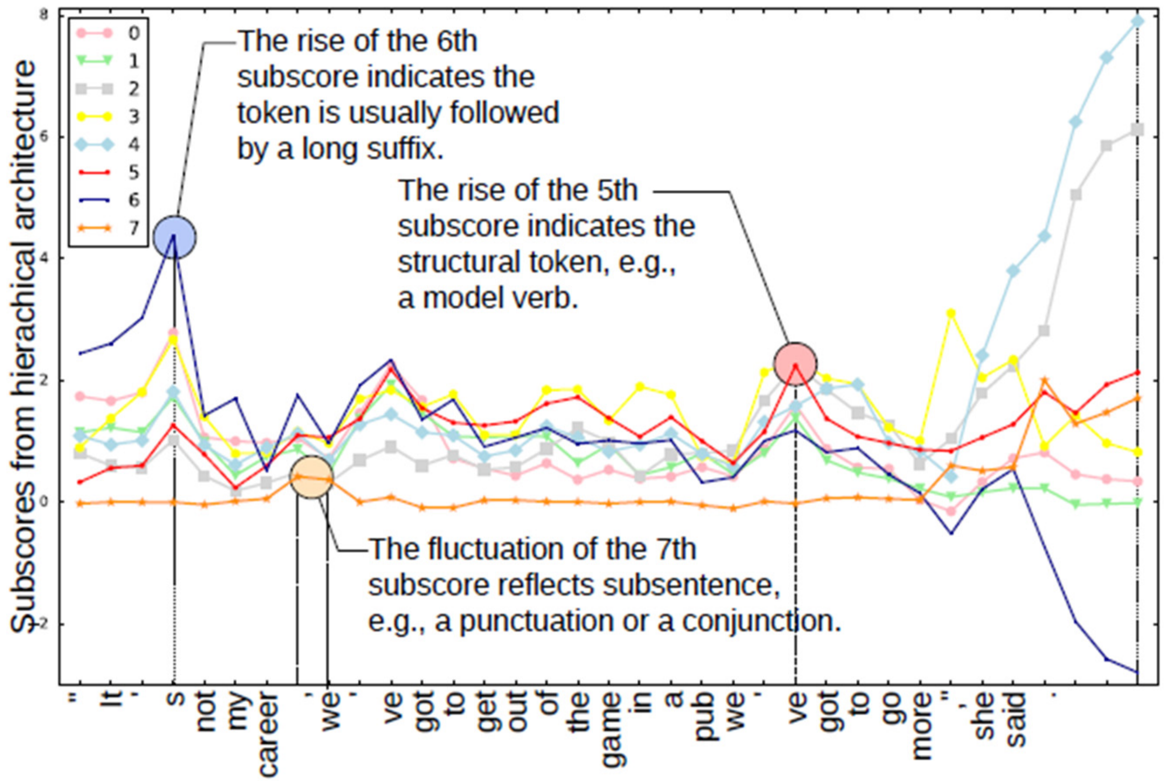
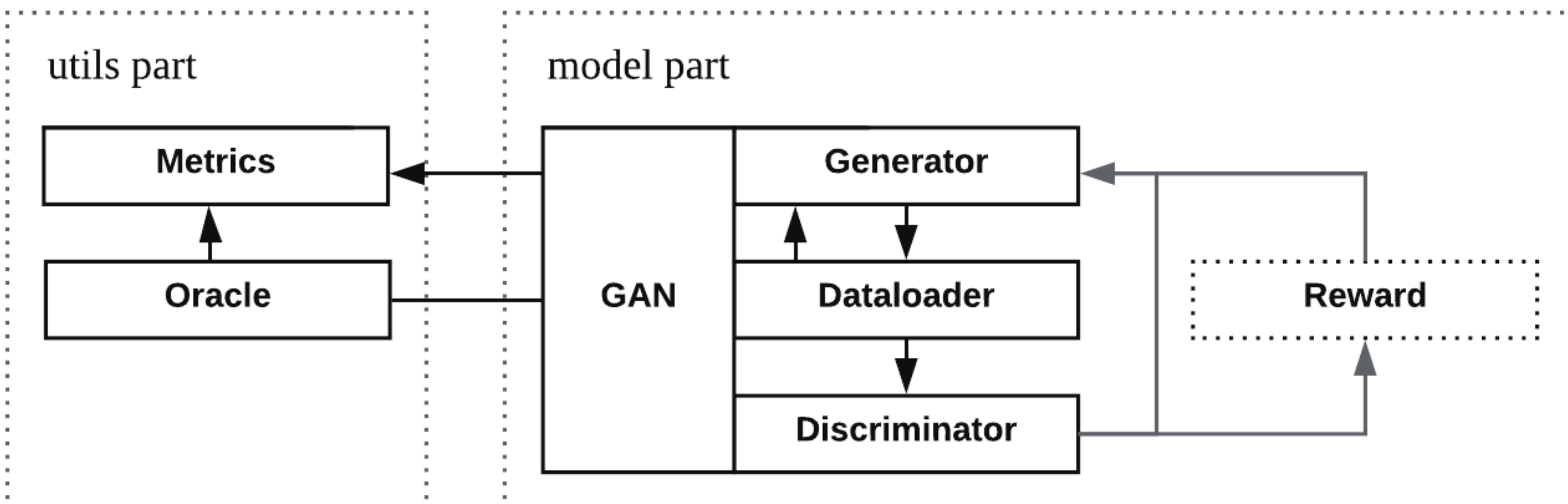


Figure 5: Illustration of WORKER and MANAGER's behaviors during a generation. (Dimension-wise Product of Worker and Manager)

Texygen by Geek.AI

A Text Generation Model Benchmarking Platform

- Motivations
  - Make comparison transparent and comprehensive
  - Open source all models
- Platform advantages
  - Highly decoupled
  - Easy to Run
  - Customization



Yaoming Zhu, Weinan Zhang et al. Texygen: A Benchmarking Platform for Text Generation Models. SIGIR 2018.
https://github.com/geek-ai/Texygen

Evaluation Metrics

- Document Similarity
  - BLEU score [Papineni *et al.*, 2002]
    - Counting of matching n-grams between two sentences
    - Penalty for shorter sentence
    - Clip the duplicated n-grams
    - Average the BLEU score over all true sentences and the generated one
  - Self-BLEU
    - Measure the BLEU score among the generated sentences
    - The lower Self-BLEU means the higher diversity

Evaluation Metrics

- Likelihood-based measurement
  - NLL-oracle [Yu et al., 2017]
    - Use a randomly initialized LSTM as the true model, aka, oracle
    - Minimize the exact opposite average negative log-likelihood

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

  - NLL-test
    - Dual to NLL-oracle
    - Evaluating the model's capacity to fit real test data

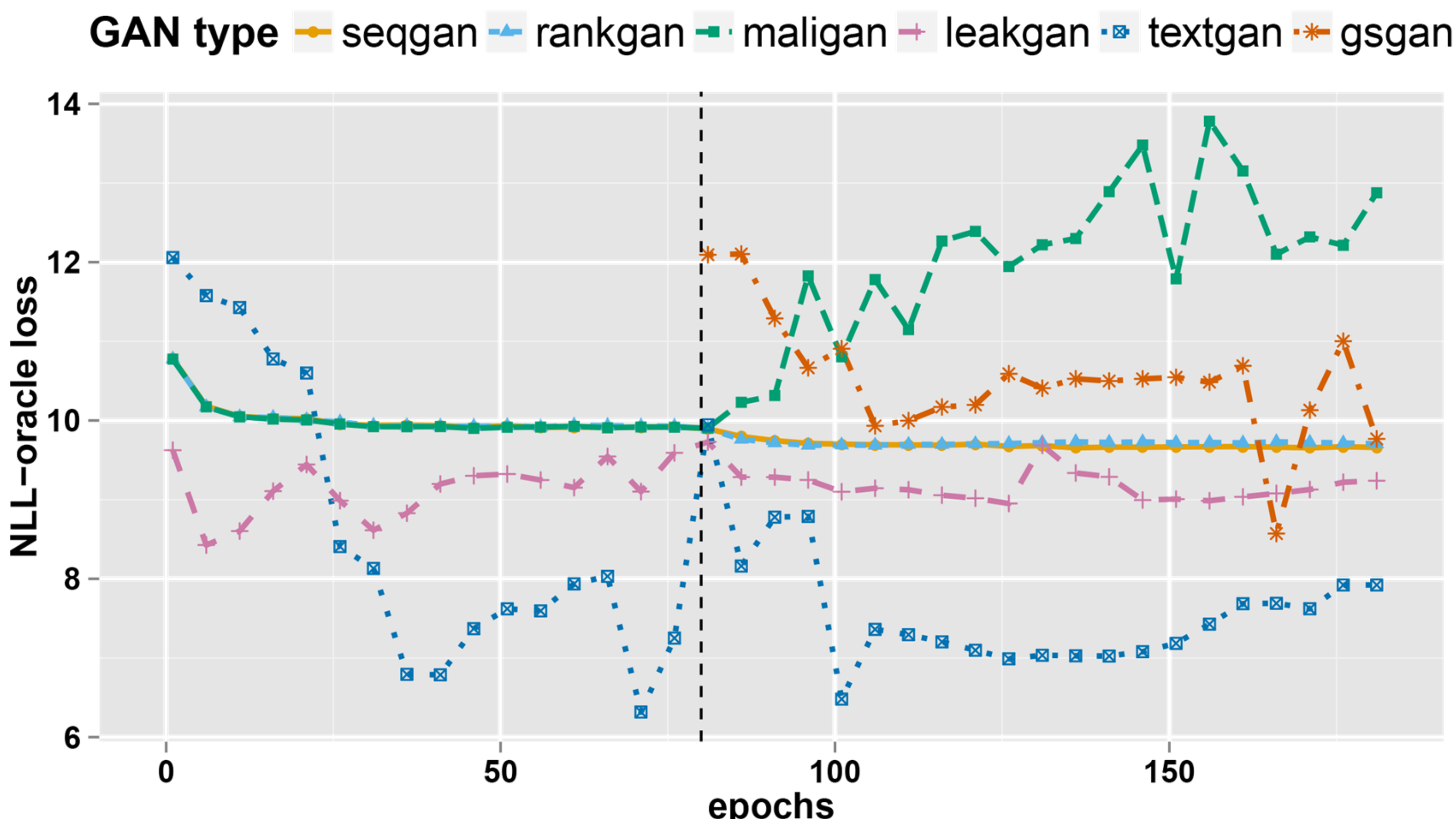
$$\mathrm{NLL}_{\mathrm{test}} = -\mathbb{E}_{Y_{1:T} \sim G_{\mathrm{real}}}\Big[\sum_{t=1}^{T} \log(G_\theta(y_t | Y_{1:t-1}))\Big]$$

Texygen Code

- User friendly APIs for
  - 7 models: SeqGAN, MaliGAN, RankGAN, LeakGAN, GSGAN, TextGAN, MLE
  - 6 Metrics: BLEU, EmbSim, NLL-oracle, NLL-test, Self-BLEU, CFG
  - 3 Training methods: Oracle, CFG, Real data

```
#usage:
python main.py -g <GAN type> -t <training method> -d <data location>
  -g <GAN type>
    specify the GAN type in the experiment
    <GAN type> = seqgan | maligan | rankgan | leakgan | gsgan | textgan | mle
  -t <training method>
    specify the traning method in the experiment
    <training method> = oracle | cfg | real
    default is oracle
  -d <data location>
    use user's own dataset
    only avaiable with real data training
    default is 'data/image_coco.txt'
```

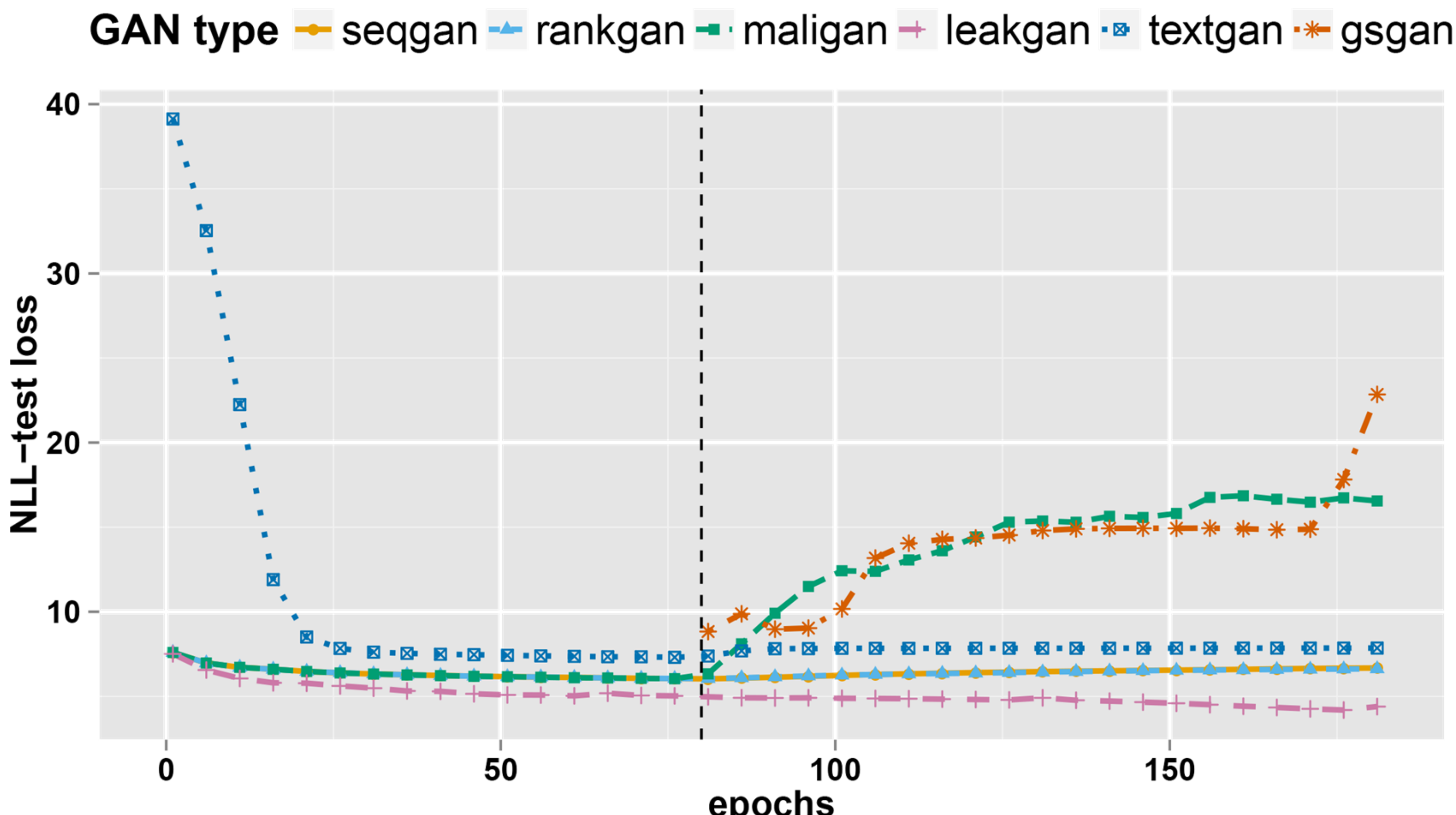
Texygen Experiments



Key observations

- MaliGAN and GSGAN diverges
- LeakGAN and TextGAN performs better than SeqGAN/RankGAN, could be led by mode collapse, i.e. low diversity

Texygen Experiments



Key observations

- LeakGAN performs better than any other compared models on both NLL-test and NLL-oracle metrics

Texygen Experiments

Table 2: BLEU score on test data

	BLEU-2	BLEU-3	BLEU-4	BLEU-5
SeqGAN	0.917	0.747	0.530	0.348
MaliGAN	0.887	0.697	0.482	0.312
RankGAN	**0.937**	0.799	0.601	0.414
LeakGAN	0.926	**0.816**	**0.660**	0.470
TextGAN	0.650	0.645	0.569	**0.523**
MLE	0.921	0.768	0.570	0.392

Table 3: Self-BLEU score

	BLEU-2	BLEU-3	BLEU-4	BLEU-5
SeqGAN	0.950	0.840	0.670	0.489
MaliGAN	0.918	0.781	0.606	0.437
RankGAN	0.959	0.882	0.762	0.618
LeakGAN	0.966	0.913	0.848	0.780
TextGAN	0.942	0.931	0.804	0.746
MLE	0.916	0.769	0.583	0.408

Dataset: Image COCO

Key observations: LeakGAN provides the best BLEU performance but sacrifice on diversity

Summary of this Part

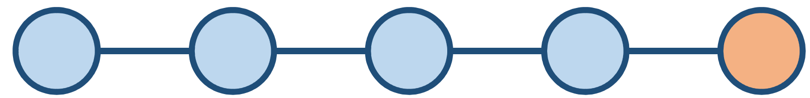
- It looks promising to leverage RL to train GANs for discrete data
- SeqGAN models the sequence generation as a sequential decision making process
  - Next token generation as an RL policy
  - Discriminator provides final reward signals
- LeakGAN addresses two problems of SeqGAN
  - Scalar reward is non-informative
  - Final reward is sparse
  - By leaking information from D to G with HRL
- More models are developed, which need fair comparison

Content of this Tutorial

1. Introduction to Generative Adversarial Nets
2. Reinforcement Learning
3. GANs for Information Retrieval
4. GANs for Text Generation
5. GANs for Graph/Network Learning
6. Beyond GANs, Cooperative Training
7. Future Perspective and Summarization

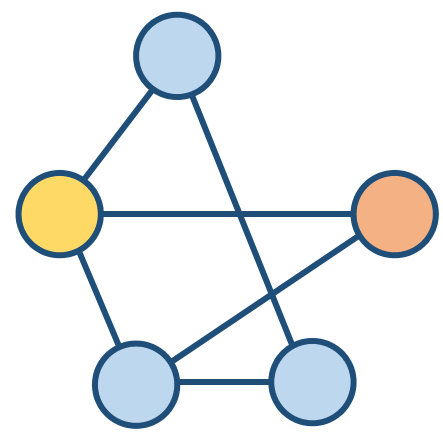
Beyond Single Discrete Token

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



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

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



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

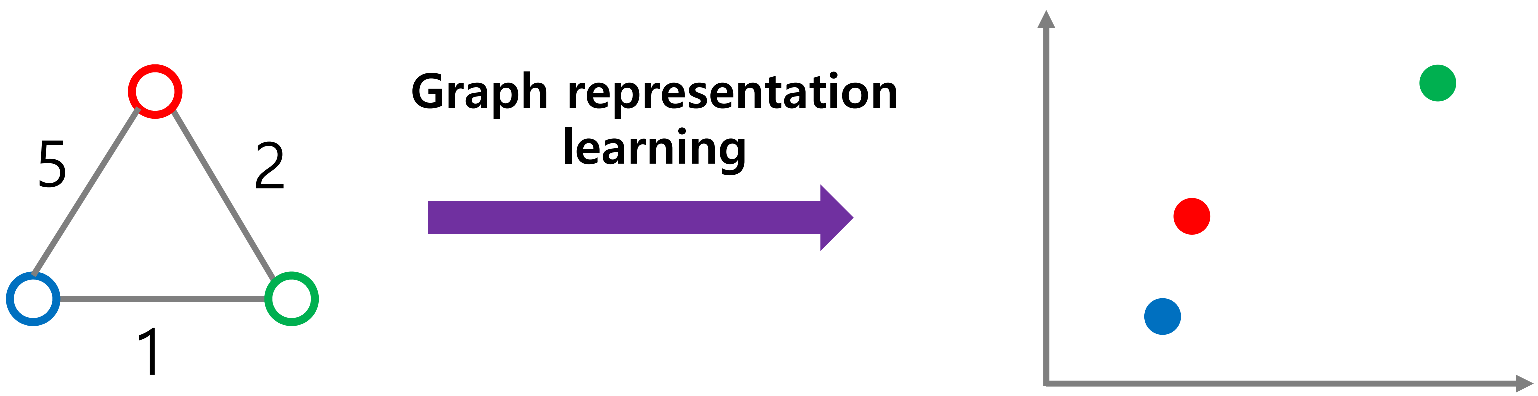
GraphGAN: Graph Representation Learning with Generative Adversarial Nets

[Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, Minyi Guo. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. AAAI 2018.]

https://arxiv.org/abs/1711.08267

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



Background of GRL

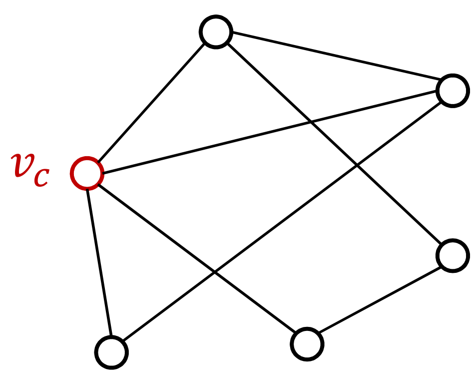
- Graph representation learning applications
  - Link prediction
  - Node classification
  - Recommendation
  - Visualization
  - Knowledge graph representation
  - Clustering
  - Text embedding
  - Social network analysis
  - …

Background of GRL

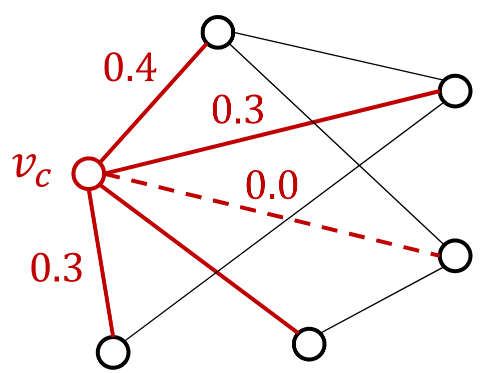
- Researchers have examined applying representation learning methods to various types of graphs:
  - Weighted graphs (Grover and Leskovec, KDD 2016)
  - Directed graphs (Zhou et al., AAAI 2017)
  - Signed graphs (Wang et al., SDM 2017)
  - Heterogeneous graphs (Wang et al., WSDM 2018)
  - Attributed graphs (Huang, Li, and Hu, WSDM 2017)
- Several prior works also try to preserve specific properties during the learning process:
  - Global structures (Wang, Cui, and Zhu, KDD 2017)
  - Community structures (Wang et al., AAAI 2017)
  - Group information (Chen, Zhang, and Huang, CIKM 2016)
  - Asymmetric transitivity (Ou et al., KDD 2016)

Motivation of GraphGAN

- Generative graph representation learning model assumes an underlying true connectivity distribution $p_{true}(v|v_c)$ for each vertex v_c
  - Similar to GMM and LDA
  - The edges can be viewed as observed samples generated by the true distribution $p_{true}(v|v_c)$
  - Vertex embeddings are learned by maximizing the likelihood of edges
  - E.g., DeepWalk (KDD 2014) and node2vec (KDD 2016)



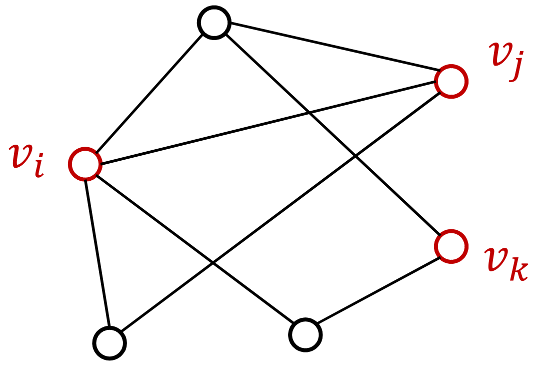
Original graph



$p_{true}(v|v_c)$

Motivation of GraphGAN

- Discriminative graph representation learning model aims to learn a classifier for predicting the existence of edges directly
  - Consider two vertices v_i and v_j jointly as features
  - Predict the probability of an edge existing between them, i.e., $p(edge|v_i,v_j)$
  - E.g., SDNE (KDD 2016) and PPNE (DASFAA, 2017)



$$p(edge|v_i, v_j) = 0.8$$

$$p(edge|v_i, v_k) = 0.3$$

……

Motivation of GraphGAN

- Generative and discriminative models are two sides of the same coin
  - LINE (WWW 2015) has tried to combine these two objectives via edge sampling

- GraphGAN, a framework that unifies generative and discriminative thinking for graph representation learning

Jian Tang et al. LINE: Large-scale Information Network Embedding. WWW 2015.

GraphGAN: the Minimax Game

- Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
  - Set of vertices: $\mathcal{V} = \{v_1, ..., v_V\}$
  - Set of edges: $\mathcal{E} = \{e_{ij}\}_{i,j=1}^{V}$
  - Underlying true connectivity distribution for v_c: $p_{\text{true}}(v|v_c)$

- The objective of GraphGAN is to learn the following two models
  - Generator $G(v|v_c; \theta_G)$ to approximate $p_{\text{true}}(v|v_c)$
  - Discriminator $D(v, v_c; \theta_D)$ to estimate the connectivity for the vertex pair (v, v_c)

GraphGAN: the Minimax Game

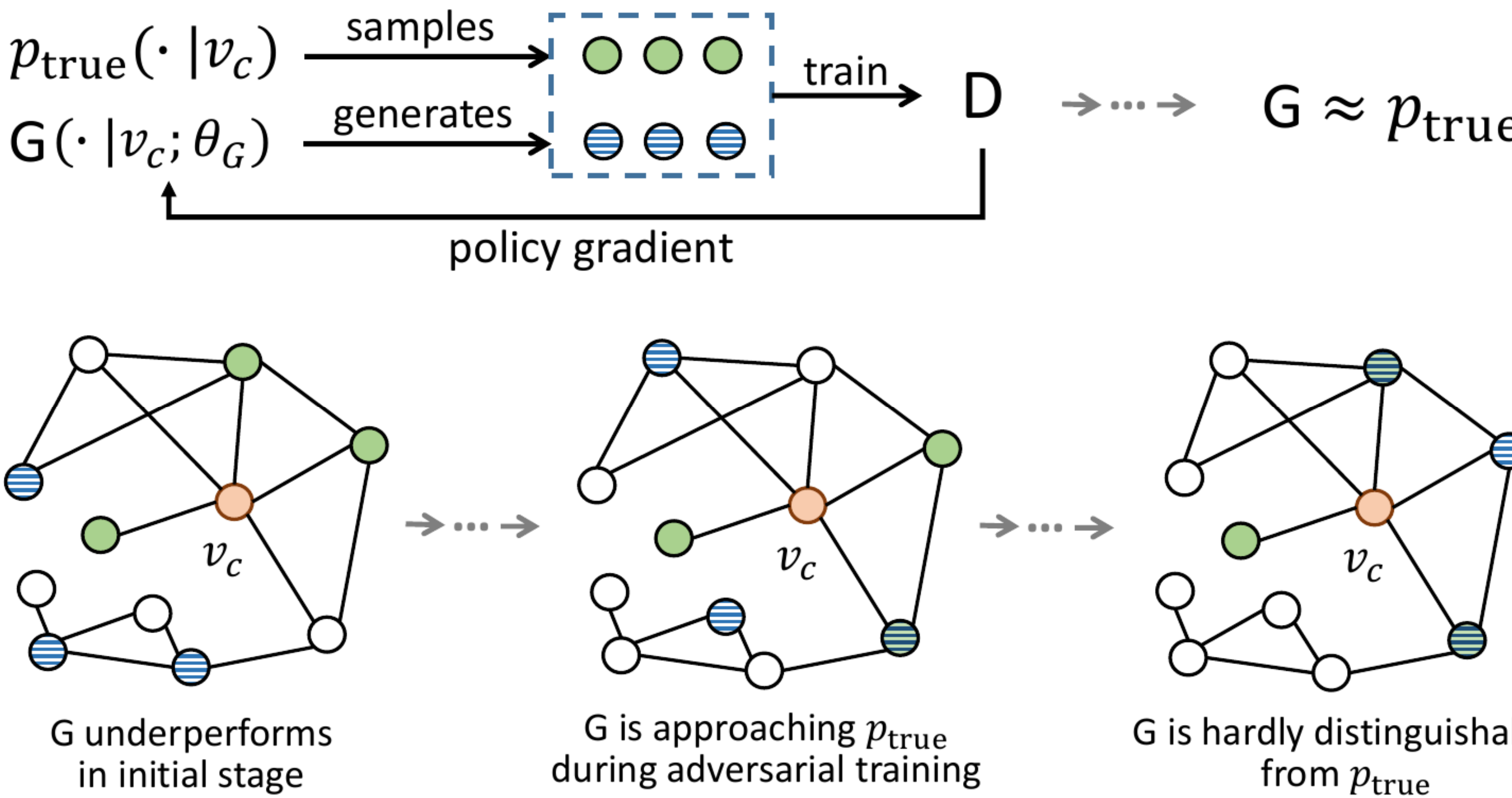
- The objective of GraphGAN is to learn the following two models
  - Generator $G(v|v_c;\theta_G)$ to approximate $p_{\text{true}}(v|v_c)$
  - Discriminator $D(v,v_c;\theta_D)$ to estimate the connectivity for the vertex pair (v,v_c)

- The two-player minimax game:

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

$$+\,\mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\log\big(1-D(v,v_c;\theta_D)\big)\big]\Big).$$

GraphGAN: the Minimax Game



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

Implementation & Optimization of D

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

- 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*

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

(a normal replacement of loss in GAN)

Optimization of G

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

- Gradient of *V*(*G*, *D*) w.r.t. the parameters of *G*

$$\begin{aligned}
\nabla_{\theta_G} V(G,D) &= \nabla_{\theta_G}\sum_{c=1}^{V}\mathbb{E}_{v\sim G(\cdot|v_c)}\big[\log\big(1-D(v,v_c)\big)\big] \\
&= \sum_{c=1}^{V}\sum_{i=1}^{N} \nabla_{\theta_G} G(v_i|v_c)\log\big(1-D(v_i,v_c)\big) \\
&= \sum_{c=1}^{V}\sum_{i=1}^{N} G(v_i|v_c)\nabla_{\theta_G}\log G(v_i|v_c)\log\big(1-D(v_i,v_c)\big) \\
&= \sum_{c=1}^{V}\mathbb{E}_{v\sim G(\cdot|v_c)}\big[\nabla_{\theta_G}\log G(v|v_c)\log\big(1-D(v,v_c)\big)\big]
\end{aligned}$$

Implementation of G

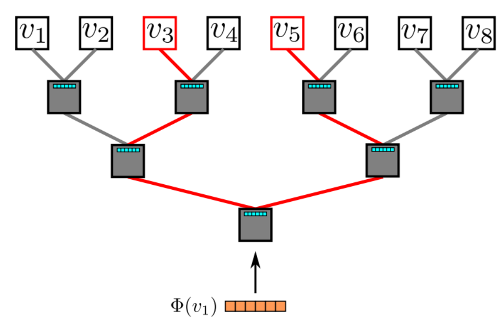
- Softmax?
  - Computationally inefficient
  - Graph-structure-unaware

$$G(v|v_c) = \frac{\exp(\mathbf{g}_v^\top \mathbf{g}_{v_c})}{\sum_{v \neq v_c} \exp(\mathbf{g}_v^\top \mathbf{g}_{v_c})}$$

where $\mathbf{g}_v, \mathbf{g}_{v_c} \in \mathbb{R}^k$ are the k-dimensional vectors of v and v_c for G

- Hierarchical softmax?
  - Graph-structure-unaware

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^\top v_{w_I} \right)$$



- Negative sampling (NCE)?
  - Not a valid probability distribution
  - Graph-structure-unaware

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

Graph Softmax in GraphGAN

- Objectives: The design of graph softmax should satisfy the following three properties
  - Normalized: The generator should produce a valid probability distribution

$$\sum_{v \neq v_c} G(v|v_c;\theta_G) = 1$$

  - Graph-structure-aware: The generator should take advantage of the structural information of a graph
  - Computationally efficient: The computation of $G(v|v_c;\theta_G)$ should only involve a small number of vertices in the graph

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

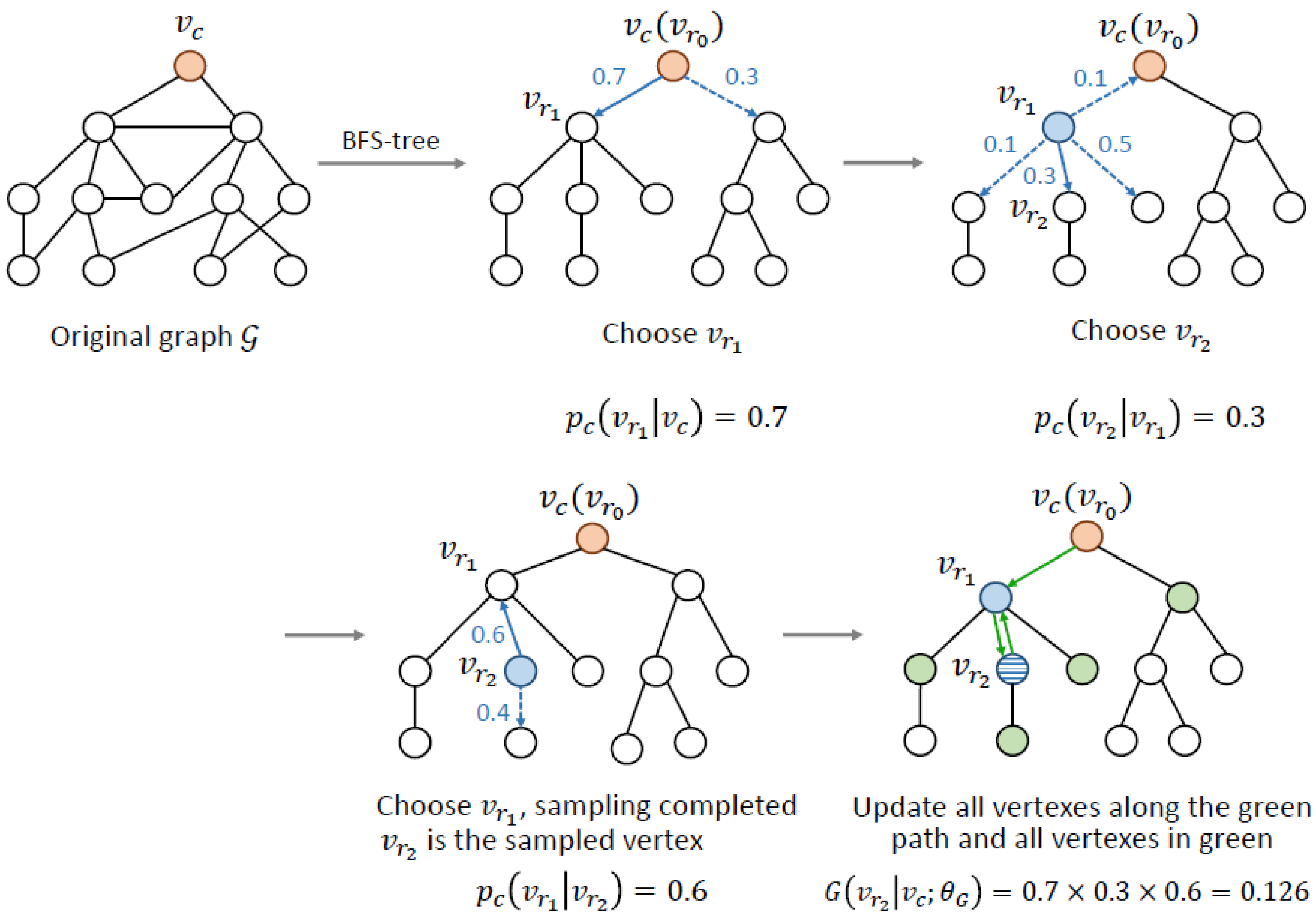
Go to an unvisited neighbor   Get back to the parent

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

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

Graph Softmax in GraphGAN

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



Graph Softmax in GraphGAN

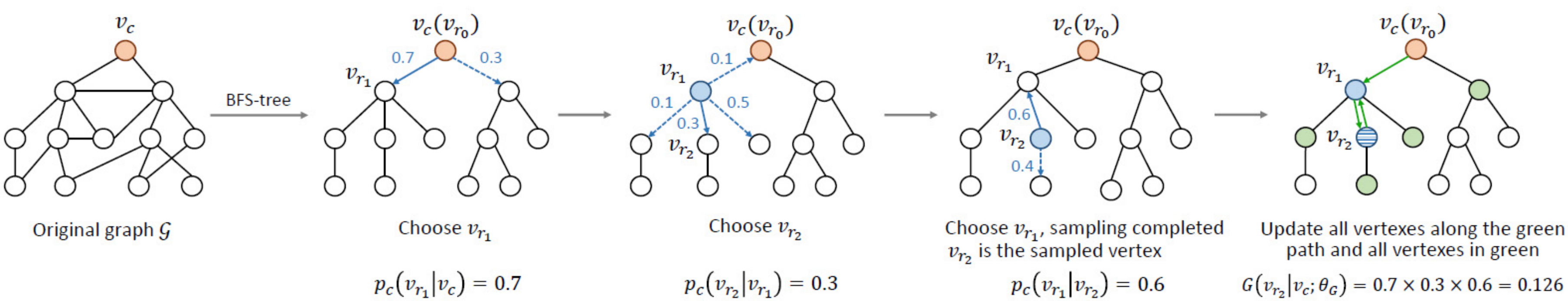
- Some properties for graph softmax in GraphGAN
  - Normalized $\sum_{v \neq v_c} G(v|v_c;\theta_G) = 1$
  - $G(v|v_c;\theta_G)$ decreases exponentially with the increase of the shortest distance between v and v_c in original graph $\mathcal{G}$
  - The calculation of $G(v|v_c;\theta_G)$ depends on $O(d \log V)$ vertices, where d is average degree of vertices and V is the number of vertices in graph $\mathcal{G}$

Graph Softmax Algorithm

Algorithm 1 Online generating strategy for the generator

Input: BFS-tree T_c, representation vectors $\{\mathbf{g}_i\}_{i \in \mathcal{V}}$
Output: generated sample v_{gen}

```
v_pre ← v_c, v_cur ← v_c;
while true do
    Randomly select v_i proportionally to p_c(v_i | v_cur) in Eq. (6);
    if v_i = v_pre then
        v_gen ← v_cur;
        return v_gen
    else
        v_pre ← v_cur, v_cur ← v_i;
    end if
end while
```



GraphGAN Algorithm

Algorithm 2 GraphGAN framework

Input: dimension of embedding k, size of generating samples s, size of discriminating samples t

Output: generator $G(v|v_c;\theta_G)$, discriminator $D(v, v_c;\theta_D)$

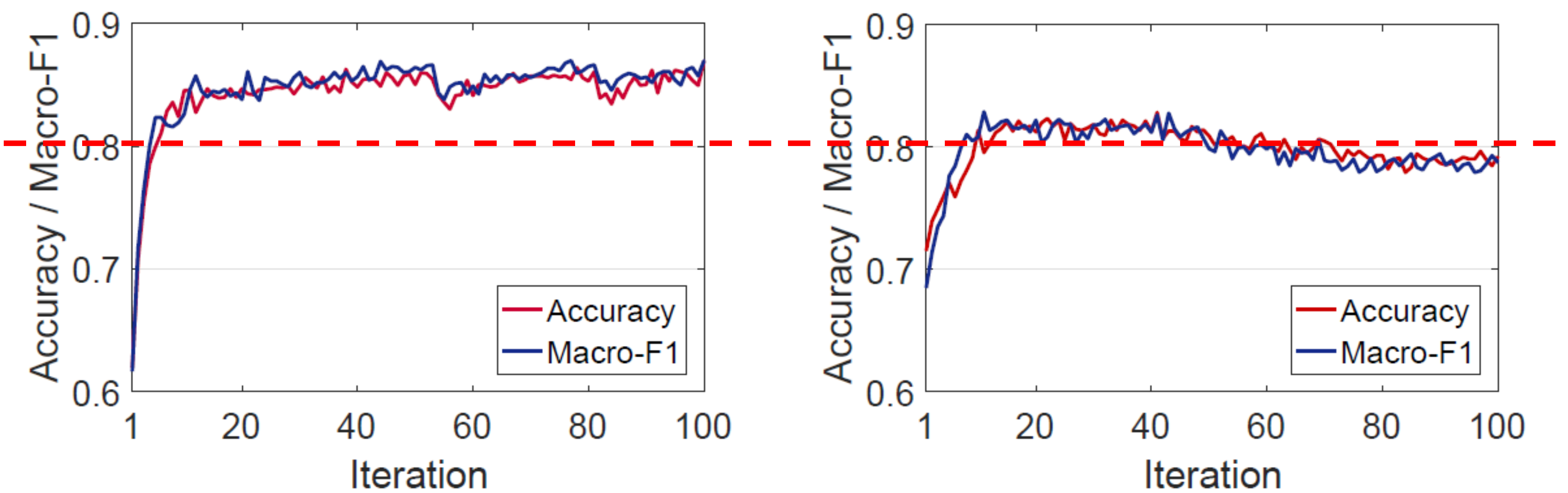
Initialize and pre-train $G(v|v_c;\theta_G)$ and $D(v, v_c;\theta_D)$;
Construct BFS-tree T_c for all $v_c \in \mathcal{V}$;
while GraphGAN not converge **do**
    for G-steps **do**
        $G(v|v_c;\theta_G)$ generates s vertices for each vertex v_c according to Algorithm 1;
        Update θ_G according to Eq. (4), (6) and (7);
    end for
    for D-steps **do**
        Sample t positive vertices from ground truth and t negative vertices from $G(v|v_c;\theta_G)$ for each vertex v_c;
        Update θ_D according to Eq. (2) and (3);
    end for
end while
return $G(v|v_c;\theta_G)$ and $D(v, v_c;\theta_D)$

Experiments of GraphGAN

- Datasets
  - arXiv-AstroPh: 18,772 vertices and 198,110 edges
  - arXiv-GrQc: 5,242 vertices and 14,496 edges
  - BlogCatalog: 10,312 vertices, 333,982 edges and 39 labels
  - Wikipedia: 4,777 vertices, 184,812 edges and 40 labels
  - MovieLens-1M: 6,040 users and 3,706 movies
- Baselines
  - DeepWalk (KDD 2014)
  - LINE (WWW 2015)
  - Node2vec (KDD 2016)
  - Struc2vec (KDD 2017)

Link Prediction Experiments

- Learning curves
  - Generator outperforms discriminator



(a) Generator (b) Discriminator

Dataset: arXiv-GrQc: 5,242 vertices and 14,496 edges

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**

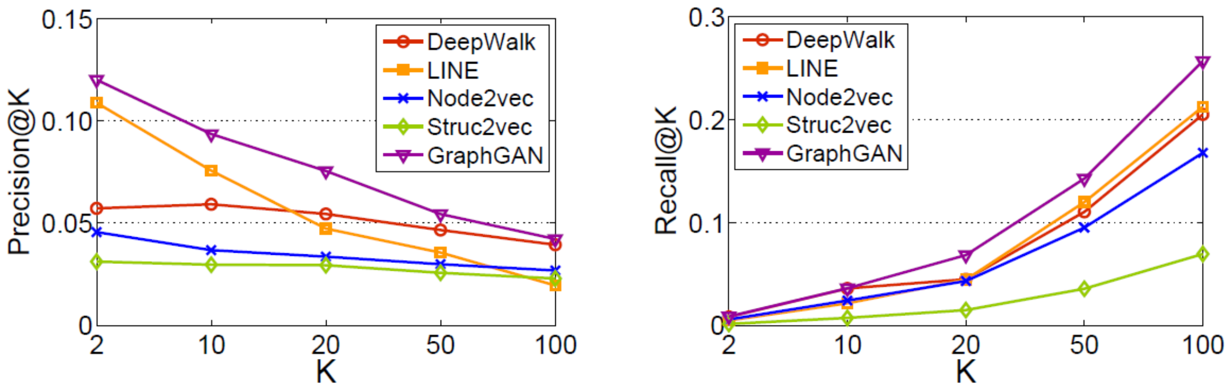
- LINE and struc2vec is relatively poor in link prediction, as they cannot quite capture the pattern of edge existence in graphs.
- DeepWalk and node2vec perform better than LINE and struc2vec probably because of the random-walk-based Skip-Gram model, which is graph-structure-aware and better at extracting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

Summary of This Part

- GraphGAN is a novel framework that unifies generative and discriminative thinking for graph representation learning
  - Generator $G(v|v_c)$ tries to fit $p_{true}\,(v|v_c)$ as much as possible
  - Discriminator $D(v,\, v_c)$ tries to tell whether an edge exists between v and v_c

- G and D act as two players in a minimax game:
  - G tries to produce the most indistinguishable “fake” vertices under guidance provided by D
  - D tries to draw a clear line between the ground truth and “counterfeits” to avoid being fooled by G

- Graph softmax is leveraged as the implementation of G
  - Graph softmax overcomes the limitations of softmax and hierarchical softmax
  - Graph softmax satisfies the properties of normalization, graph structure awareness and computational efficiency

Content of this Tutorial

1. Introduction to Generative Adversarial Nets
2. Reinforcement Learning
3. GANs for Information Retrieval
4. GANs for Text Generation
5. GANs for Graph/Network Learning
6. Beyond GANs, Cooperative Training
7. Future Perspective and Summarization

Rethink about GAN

- Why is GAN significantly better than many supervised approaches?
  - This is because of the nice properties of GAN's objective, i.e., Jensen-Shannon Divergence.

G: $\min_G \max_D J(G, D)$ D: $\max_D J(G, D)$

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

$$= -\log(4) + \underbrace{\mathbf{KL}\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_G}{2}\right) + \mathbf{KL}\left(p_G \middle\| \frac{p_{\text{data}} + p_G}{2}\right)}_{\text{Jensen-Shannon Divergence}}$$

$$JSD(P \| G) = \frac{1}{2}\Big(KL(P \| M) + KL(G \| M)\Big)$$

$$M = \frac{1}{2}(P + G)$$ M: the mediate distribution

Disadvantages of GANs

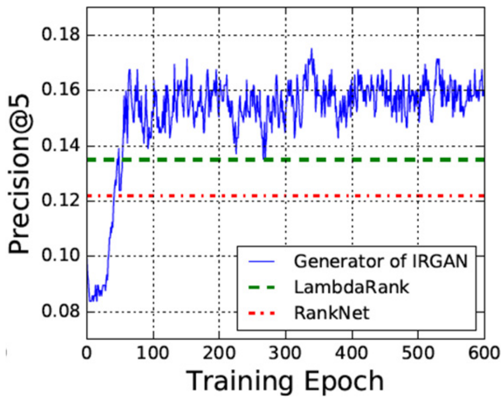
- Model collapse
  - The generator trends to generate some particular samples that fools the current discriminator

$$\min_G \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

  - This problem also occurs for RL based generator like SeqGAN, i.e., the generator policy trends to take the action leading to higher value without considering the diversity

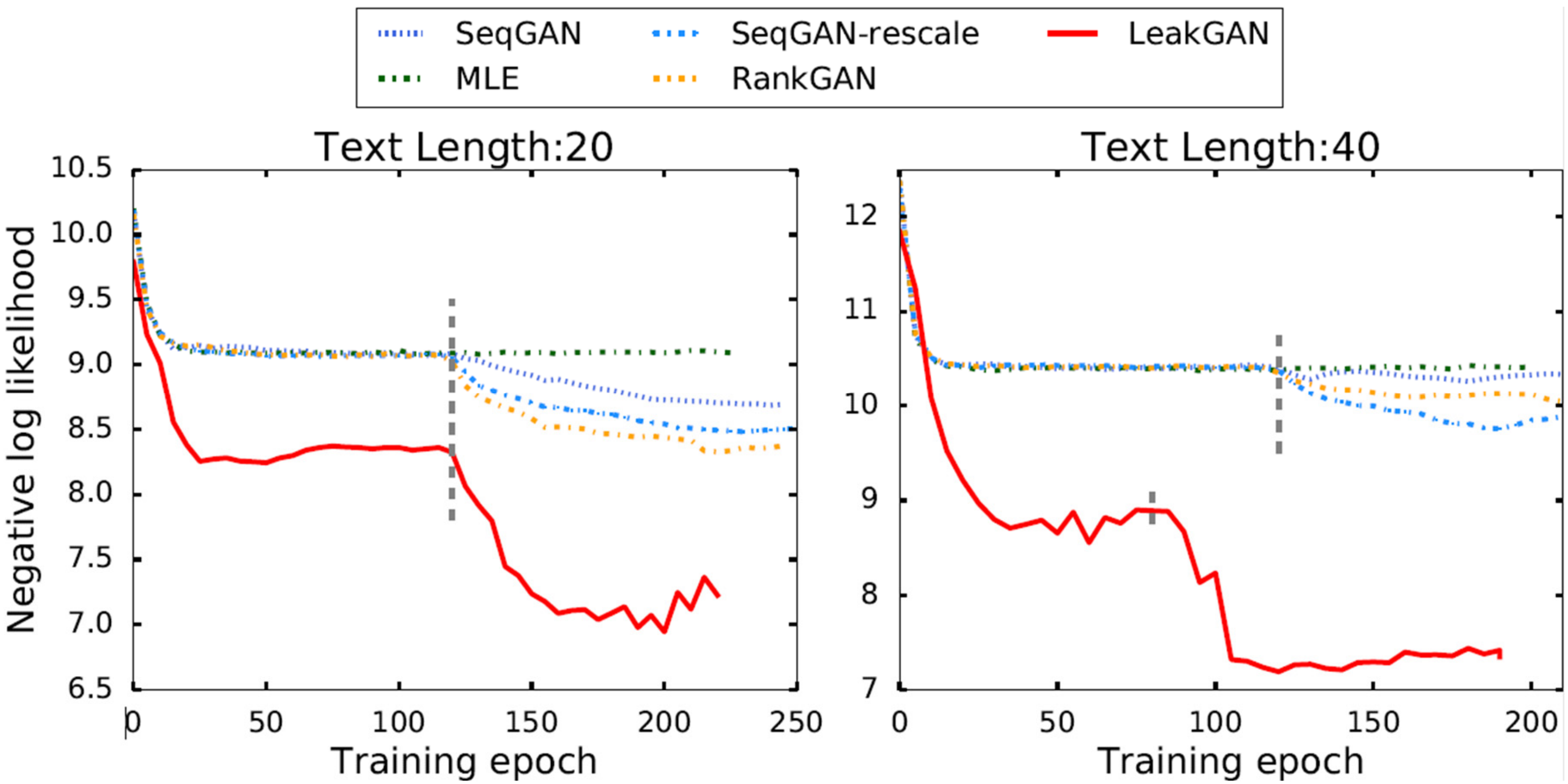
- Unstable training
  - Minimax objective
  - Adversarial training

G: $\min_G \max_D J(G, D)$ D: $\max_D J(G, D)$



Disadvantages of Discrete Data GANs

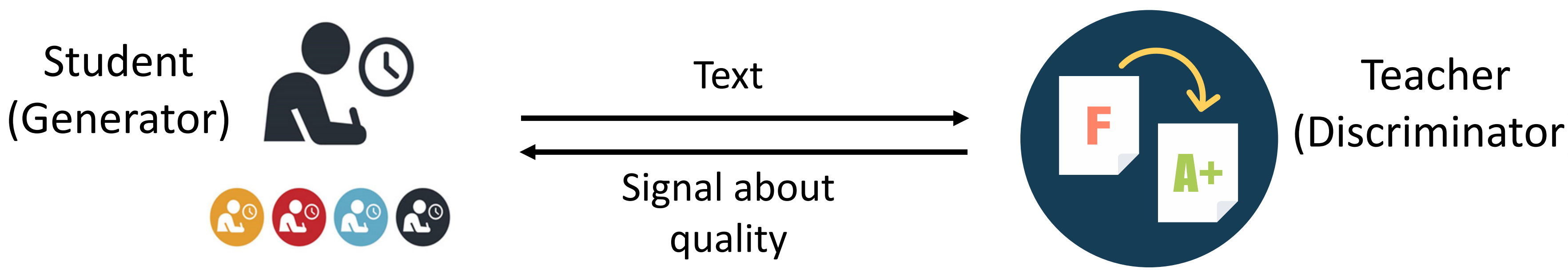
- It is crucial for SeqGAN or LeakGAN to perform model pre-training via MLE



Guo et al. Long Text Generation via Adversarial Training with Leaked Information. AAAI 2018.

Disadvantages of Discrete Data GANs

- It is crucial for SeqGAN or LeakGAN to perform model pre-training via MLE
- The guidance from discriminator is not sufficiently informative and is of high variance
  - Leading to low data efficiency, i.e., one may need a large amount of training data & effort to find a good generator policy



Guo et al. Long Text Generation via Adversarial Training with Leaked Information. AAAI 2018.

Beyond GANs, Cooperative Training

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

$$= -\log(4) + \underbrace{\mathrm{KL}\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_G}{2}\right) + \mathrm{KL}\left(p_G \middle\| \frac{p_{\text{data}} + p_G}{2}\right)}_{\text{Jensen-Shannon Divergence}}$$

$$JSD(P\|G) = \frac{1}{2}\Big(KL(P\|M) + KL(G\|M)\Big)$$

$$M = \frac{1}{2}(P+G)$$ M: the mediate distribution

- To find an algorithm that is at least as good as GANs, a simple solution is to find a way to
  - Optimize an accurate calculation of JSD
  - Or, **find an unbiased estimation of JSD at any time during the training.**

Unbiased Estimation of JSD

$$JSD(P\|G) = \frac{1}{2}\left(KL(P\|M) + KL(G\|M)\right)$$

- where M = 0.5 (P + G)
- If we can find an unbiased estimation for M, the problem will be solved.
- Can we? YES!

Unbiased Estimation of JSD

- Note that for probability prediction, MLE is unbiased.
- We can simply create a balanced mixture dataset ***B*** of samples from both distributions
  - learned model *G* and training data batch *P*
- Then we train a model M_ϕ via MLE using ***B***.

$$\max_{\phi} \mathop{\mathbb{E}}_{s\sim p_{\text{data}}} [\log(M_\phi(s))] + \mathop{\mathbb{E}}_{s\sim G_\theta} [\log(M_\phi(s))] \,.$$

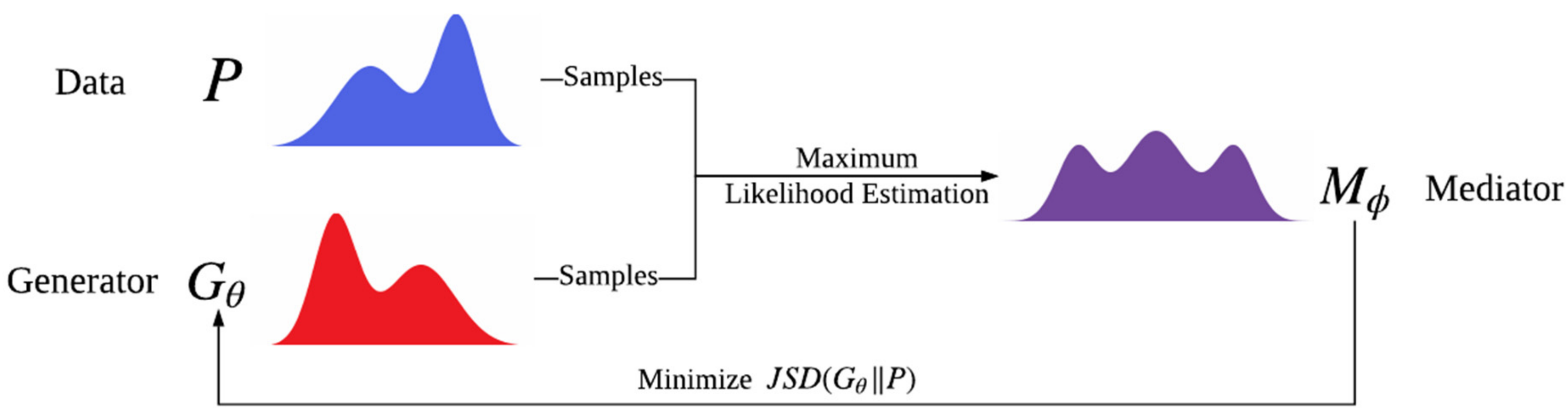
- M_ϕ can be used to provide with an unbiased estimation of JSD.

$$M_\phi \simeq \frac{1}{2}(P + G_\theta)$$

$$J\hat{S}D(G_\theta \| P) = \frac{1}{2}\big[KL(G_\theta \| M_\phi) + KL(P \| M_\phi)\big]$$

Unbiased Estimation of JSD

- Note that M_ϕ is a continuous and white-box distribution.
- We can perform better utilization of it than simply using Policy Gradient.
- We can directly compute the distribution of data (i.e. policy at each state) and perform update on it.



Cooperative Training

CoT: Cooperative Training for Generative Modeling of Discrete Data

Sidi Lu
Shanghai Jiao Tong University
steve_lu@apex.sjtu.edu.cn

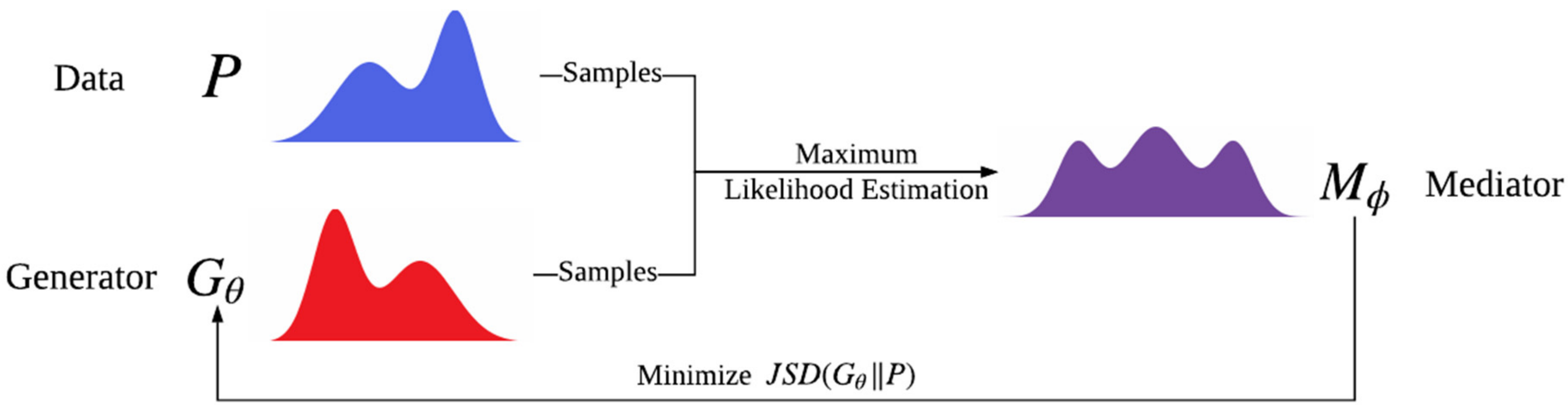
Lantao Yu
Shanghai Jiao Tong University
yulantao@apex.sjtu.edu.cn

Weinan Zhang
Shanghai Jiao Tong University
wnzhang@apex.sjtu.edu.cn

Yong Yu
Shanghai Jiao Tong University
yyu@apex.sjtu.edu.cn

Sidi Lu et al. CoT: Cooperative Training for Generative Modeling. ArXiv:1804.03782 2018.

Cooperative Training



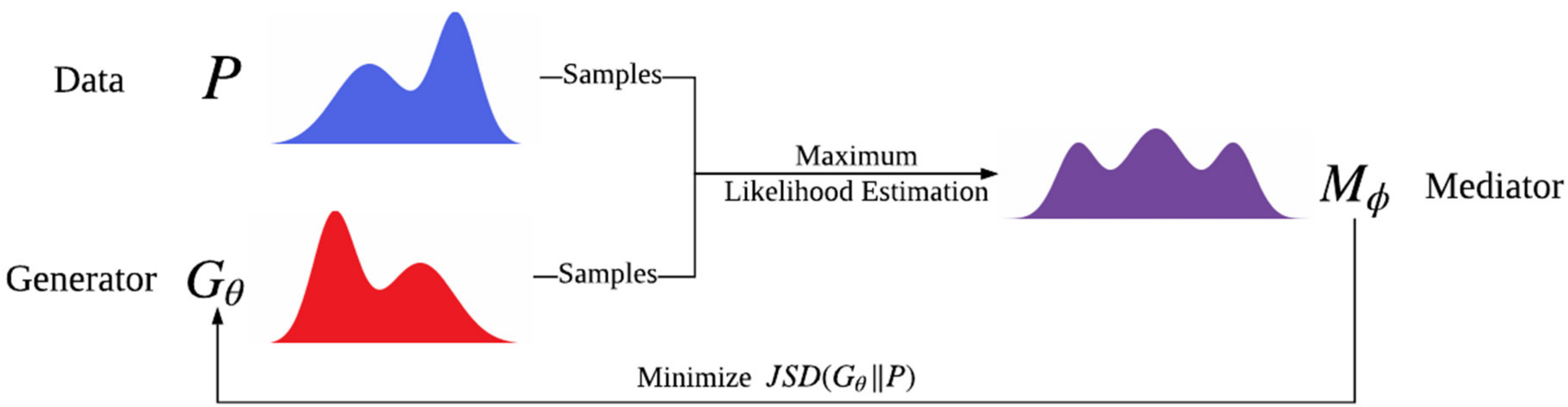
- The overall objective

$$\max_\theta \max_\phi \mathop{\mathbb{E}}_{s\sim p_{\text{data}}} [\log(M_\phi(s))] + \mathop{\mathbb{E}}_{s\sim G_\theta} [\log(M_\phi(s))]$$

Two max operations for cooperative training

Unbiased estimation of Jensen-Shannon Divergence

Cooperative Training



- Mediator: MLE objective

$$J_m(\phi) = \frac{1}{2}\Big(\mathop{\mathbb{E}}_{s\sim G_\theta}[-\log(M_\phi(s))] + \mathop{\mathbb{E}}_{s\sim P}[-\log(M_\phi(s))] \Big)$$

- Generator: maximize estimated JSD

$$J_g(\theta) = J\hat{S}D(G_\theta \| P) = \frac{1}{2}\left[KL(G_\theta \| M_\phi) + KL(P \| M_\phi) \right]$$

$$\nabla_\theta J_g(\theta) = \nabla_\theta \mathop{\mathbb{E}}_{s\sim G_\theta}\left[\sum_{t=0}^{n-1} \pi_g(s_t)^\top (\log \pi_m(s_t) - \log \pi_g(s_t)) \right]$$

Cooperative Training

- The overall objective

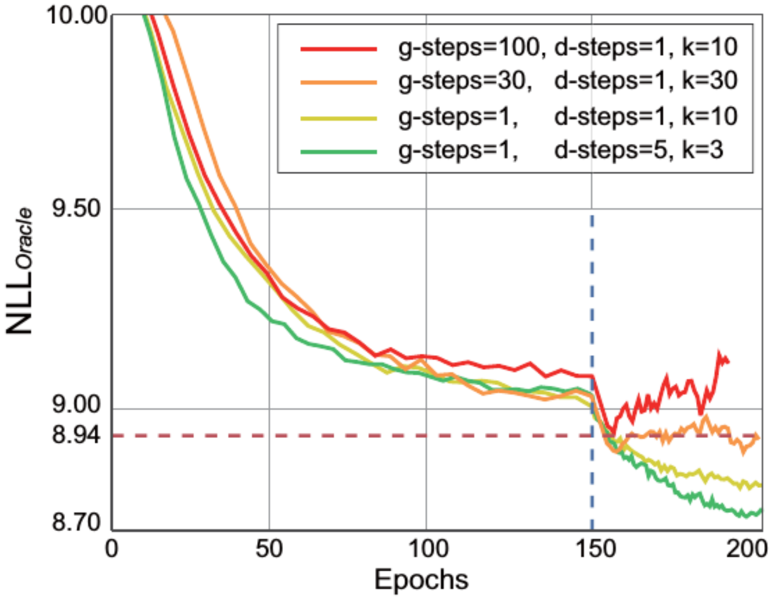
$$\max_{\theta} \max_{\phi} \mathop{\mathbb{E}}_{s \sim p_{\text{data}}} [\log(M_\phi(s))] + \mathop{\mathbb{E}}_{s \sim G_\theta} [\log(M_\phi(s))]$$

Algorithm 1 Cooperative Training

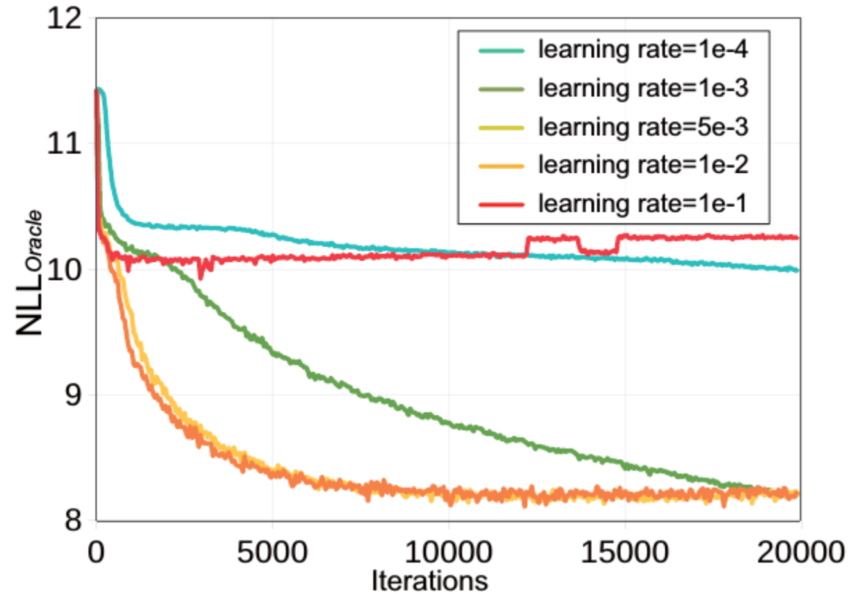
Require: Generator G_θ; mediator M_ϕ; Samples from real data distribution P; Hyper-parameter m.

Initialize G_θ, M_ϕ with random weights θ, ϕ.
repeat
    for m steps **do**
        Collect a mini-batch of mixed balanced samples $\{s\}$ from both G_θ and P
        Update mediator M_ϕ with $\{s\}$ via Eq. (9)
    end for
    Generate a mini-batch of sequences $\{s\} \sim G_\theta$
    Update generator G_θ with $\{s\}$ via Eq. (13)
until CoT converges

Experiment: JSD on Synthetic Data



NLL-oracle of SeqGAN

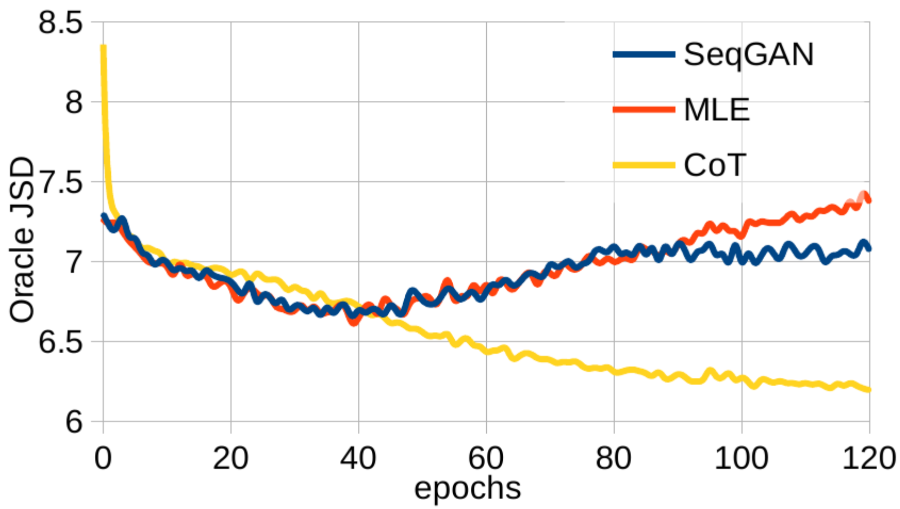


NLL-oracle of CoT

- Compared to SeqGAN, CoT is significantly stable w.r.t. its hyperparameters and requires no pre-training

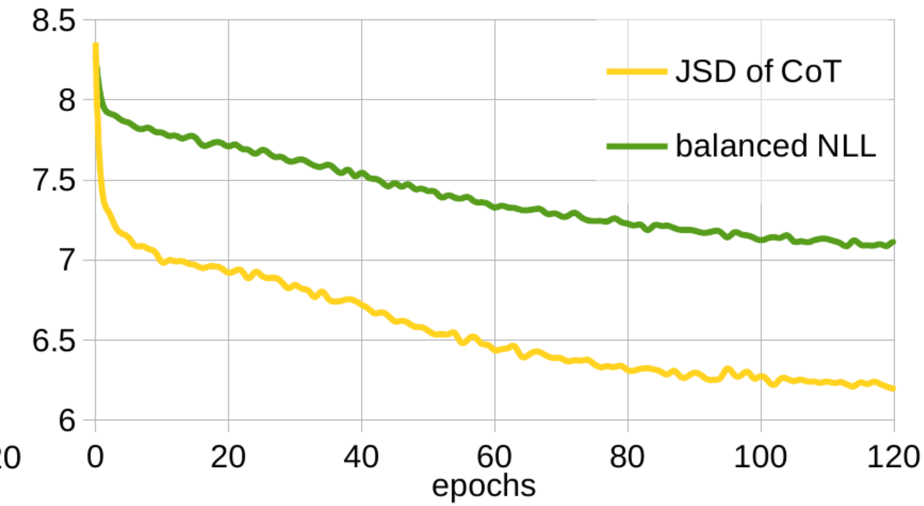
Experiment: JSD on Synthetic Data

Curves of training time JSD on synthetic data



CoT provides a much more stable training curve than SeqGAN

Curves of balanced NLL and real JSD



Balanced NLL is a good estimation of real JSD, i.e.,

$$balanced\ NLL = JSD(G\|P) + H(G) + H(P)$$

In the Sense of Philosophy

- The game thus becomes:

$$\max_{\theta} \max_{\phi} \mathop{\mathbb{E}}_{s \sim p_{\text{data}}} [\log M_\phi(s)] + \mathop{\mathbb{E}}_{s \sim G_\theta} [\log M_\phi(s)]$$

  with maximized entropy of M.

- Compared with the adversarial training of GAN

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

- Cooperative training can also achieve the goal of adversarial training!

Summary of this Part

- Cooperative training (CoT) has potential to be better than GAN

- Along with Equil Q&M (Zou et al. ICTIR 2018), there are multiple ways of formulation for multi-agent IR modeling

- Deeper thinking
  - M is actually a multi-task learning (to model data from both P and G) module. How can we further improve M?
  - GAN is significantly improved as Wasserstein GAN. Does there exist a similar improvement for CoT?

Content of this Tutorial

1. Introduction to Generative Adversarial Nets
2. Reinforcement Learning
3. GANs for Information Retrieval
4. GANs for Text Generation
5. GANs for Graph/Network Learning
6. Beyond GANs, Cooperative Training
7. Future Perspective and Summarization

REVIEW

Motivations of this Tutorial

- Review the two schools of thinking in IR

- Discriminative models estimate the relevance of each query-doc pair

$$f_\phi(\text{query}, \text{doc})$$

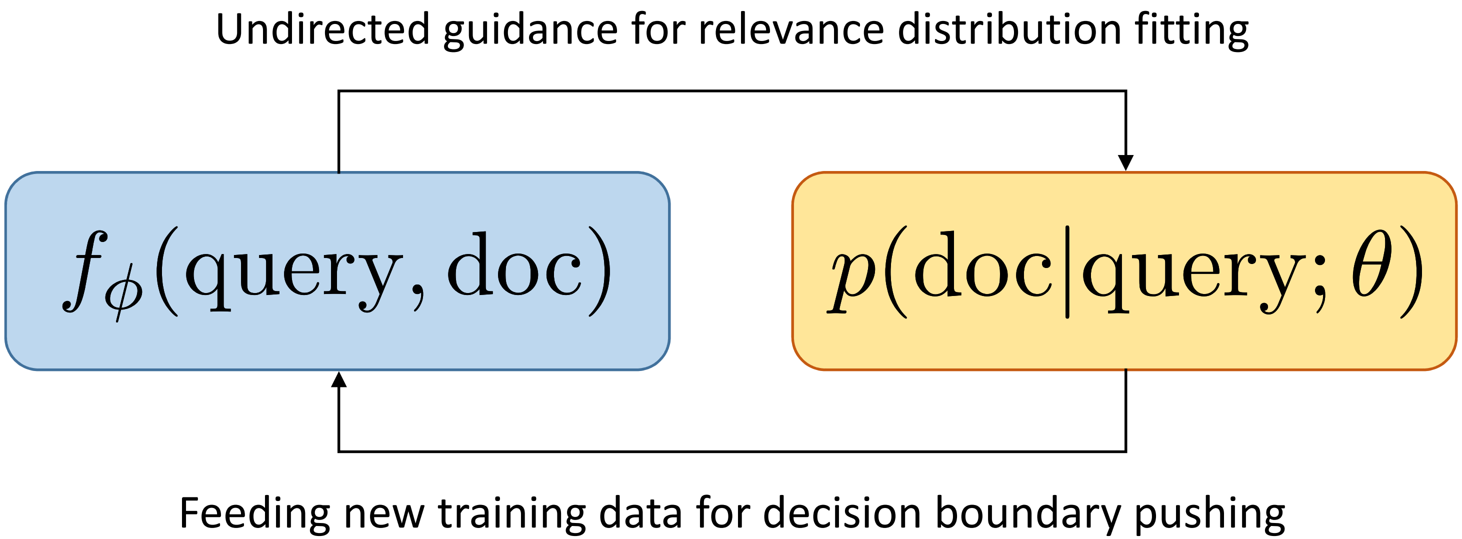
- **Pros:** learn a retrieval ranking function implicitly from labeled data
- **Cons:** lack a principled way of
  - Obtaining useful features,
  - Gathering helpful signals from the massive unlabeled data available, e.g., text statistics, the collection distribution

- Generative models estimate the preference distribution over docs given the query

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

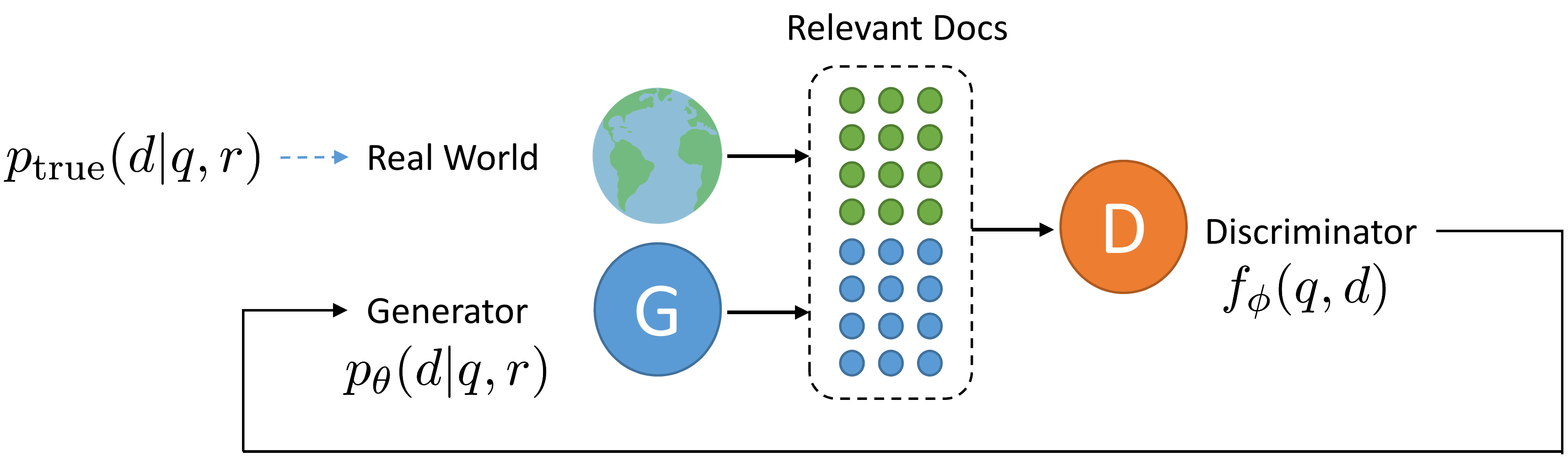
- **Pros:** theoretically sound and very successful in modelling features
- **Cons:** typically difficult in
  - leveraging relevancy signals from largely observable data, e.g., links, clicks
  - Being formulated in a trainable framework

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

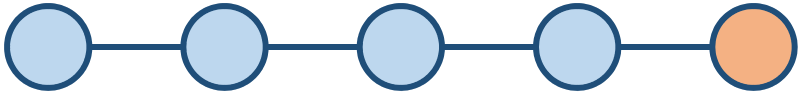
IRGAN Formulation



- **Underlying true 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 true relevance distribution
- **Discriminative retrieval model** $f_\phi(q,d)$
    - Goal: distinguish between relevant documents and non-relevant documents

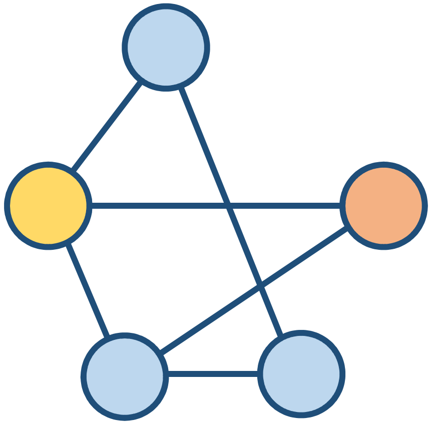
Beyond Single Discrete Token

- 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
  - …

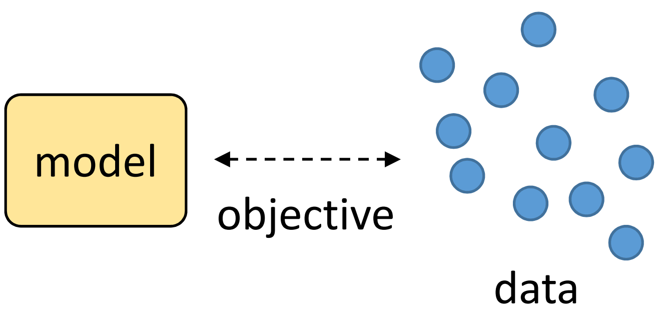


$$p(\text{node}_n|\text{node}_m, \text{neighbor}(m);\theta)$$

From Machine Learning Perspective

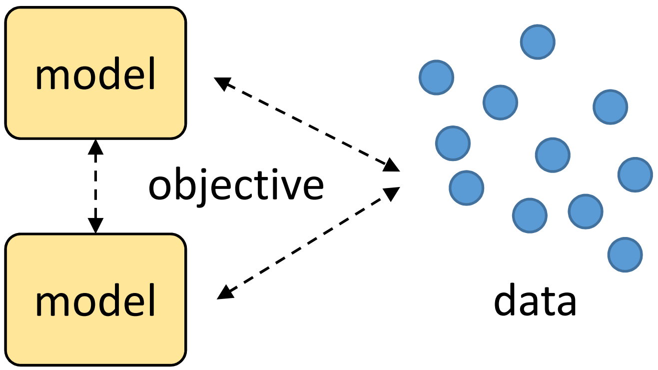
- Traditional machine learning is to build
  - a loss function
  - a likelihood estimation
  - an expectation of value

  from a machine and the training data and to optimize the objective

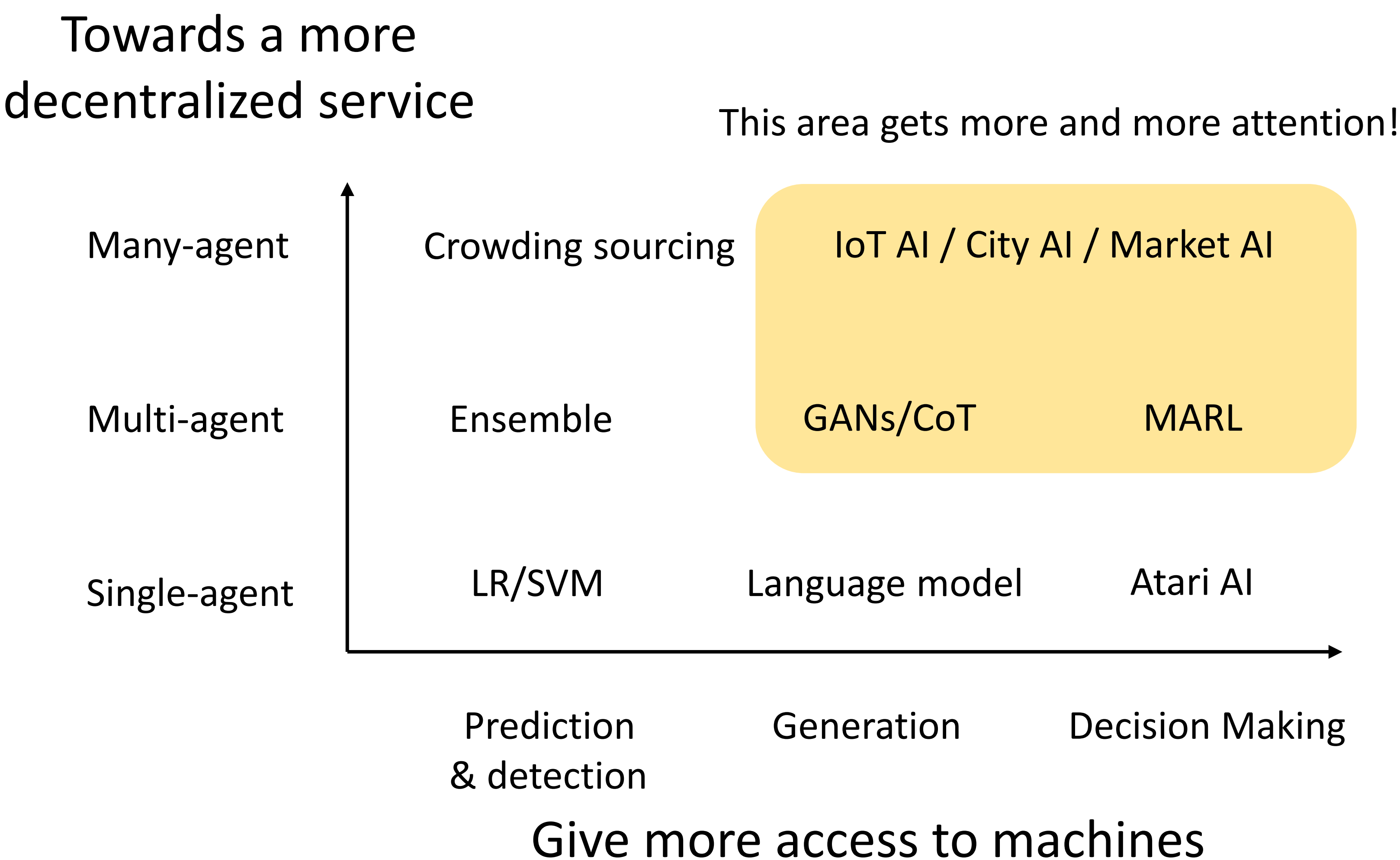


- Two-agent machine learning is to build
  - a loss function
  - a likelihood estimation
  - an expectation of value

  from the two machines and the training data and to optimize the objective



Machine Learning Paradigm Extension



Thank You!
Questions?

Weinan Zhang

Assistant Professor

APEX Data & Knowledge Management Lab

John Hopcroft Center for Computer Science

Shanghai Jiao Tong University

http://wnzhang.net

ACM SIGIR 2018