Generative Adversarial Networks (GANs) for Discrete Data

Lantao Yu
Shanghai Jiao Tong University
http://lantaoyu.com
July 26, 2017

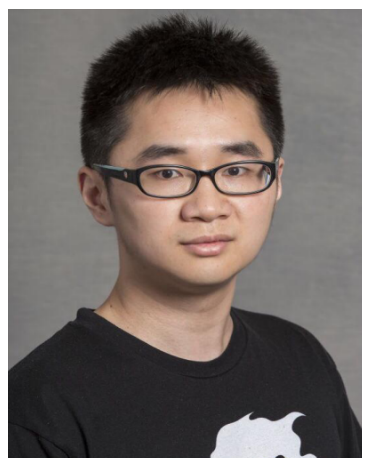
Self Introduction – Lantao Yu

- Position
  - Research Assistant at CS Dept. of SJTU 2015-now
  - Apex Data and Knowledge Management Lab
  - Research on deep learning, reinforcement learning and multi-agent systems
- Education
  - Third-year Undergraduate Student, Computer Science Dept., Shanghai Jiao Tong University, China, 2014-now
- Contact
  - lantaoyu@hotmail.com
  - yulantao@apex.sjtu.edu.cn

Apex Data & Knowledge Management Lab

- Machine learning and data science
  - with applications of recommender systems, computational ads, social networks, crowdsourcing, urban computing etc.

Professor Yong Yu



A.P. Weinan Zhang

- Students
  - 8 PhDs
  - 8 Masters
  - 24 Undergrads
- www.apexlab.org

Content

- Fundamentals - Generative Adversarial Networks
  - Connection and difference between generating discrete data and continuous data with GANs
- Advances - GANs for Discrete Data
  - SeqGAN: Sequence Generation via GANs with Policy Gradient
  - IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models

Generative Adversarial Networks (GANs)

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

Problem Definition

- Given a dataset $D = \{x\}$, build a model $q(x)$ of the data distribution that fits the true one $p(x)$

- Traditional objective: maximum likelihood estimation (MLE)

$$\max_q \frac{1}{|D|} \sum_{x \in D} \log q(x) \approx \max_q \mathbb{E}_{x \sim p(x)} [\log q(x)]$$

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

Problems of MLE

- Inconsistency of Evaluation and Use

$$\max_q \mathbb{E}_{x\sim p(x)}[\log q(x)]$$

Training/evaluation

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

$$\max_q \frac{1}{|D|}\sum_{x\in D}\log q(x)$$

$$\max_q \mathbb{E}_{x\sim q(x)}[\log p(x)]$$

Use (Turing Test)

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

Problems of MLE

- Equivalent to minimizing asymmetric $KL(p(x)||q(x))$:

$$KL(p(x)||q(x)) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x$$

  - When $p(x) > 0$ but $q(x) \to 0$, the integrand inside the KL grows quickly to infinity, which means MLE assigns an extremely high cost to such situation
  - When $p(x) \to 0$ but $q(x) > 0$, the value inside the KL divergence goes to 0, which means MLE pays extremely low cost for generating fake looking samples

[Arjovsky M, Bottou L. Towards principled methods for training generative adversarial networks. arXiv, 2017.]

Problems of MLE

- *Exposure bias* in sequential data generation:

Training

Update the model as follows:

$$\max_{\theta} \mathbb{E}_{Y \sim p_{\text{true}}} \sum_{t} \log G_{\theta}(y_t | \boxed{Y_{1:t-1}})$$

The **real** prefix

Inference

When generating the next token y_t , sample from:

$$G_{\theta}(\hat{y}_t | \boxed{\hat{Y}_{1:t-1}})$$

The **guessed** prefix

Generative Adversarial Nets (GANs)

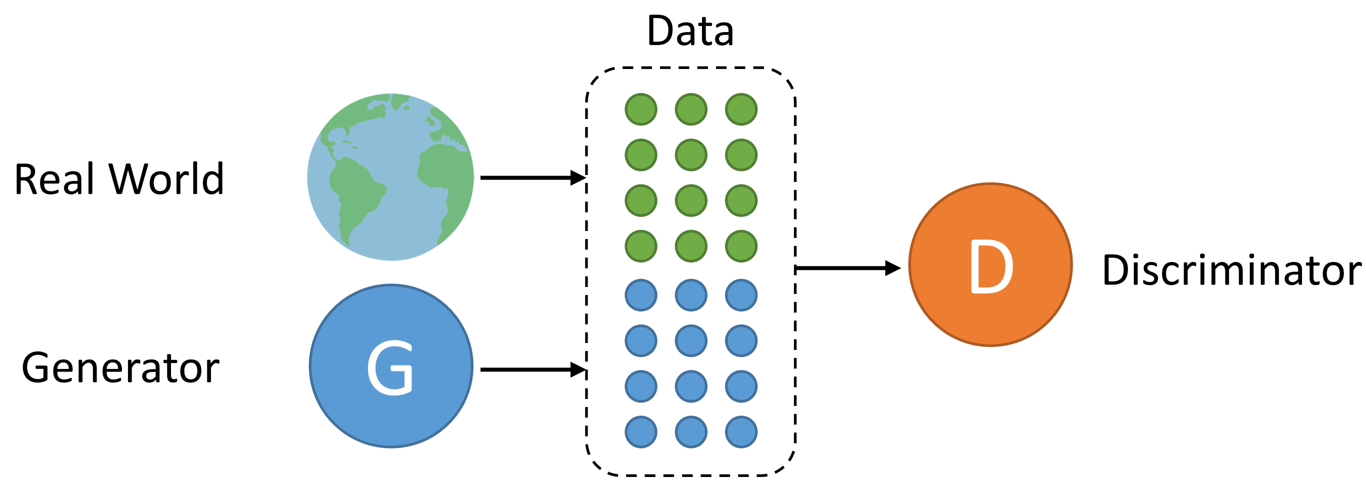
- What we really want

$$\max_q \mathbb{E}_{x \sim q(x)}[\log p(x)]$$

- But we cannot directly calculate $\mathrm{p(x)}$

- 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

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

GANs: A Minimax Game

- The most general form:

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

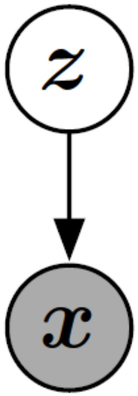
$$= \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{x\sim G(x)}[\log(1 - D(x))]$$

$$= \int_x p_{\text{data}}(x) \cdot \log(D(x)) + \boxed{G(x)} \cdot \log(1 - D(x))dx$$

Probability Density Function

Generating continuous data

- The generative model is a differentiable mapping from the prior noise space to data space
- First sample from a simple prior $z \sim p(z)$, then apply a deterministic function $G : \mathcal{Z} \rightarrow \mathcal{X}$
- No explicit Probability Density Function for data x



GANs for continuous data

- The most general form:

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

$$= \int_x p_{\text{data}}(x) \cdot \log(D(x)) + \boxed{G(x)} \cdot \log(1 - D(x)) dx$$

Probability Density Function

- Without explicit P.D.F., we can rewrite the minimax game as:

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

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

$$= \int_x p_{\text{data}}(x) \cdot \log(D(x)) dx + \int_z p_z(z) \cdot \log(1 - D(\boxed{G(z)})) dx$$

Directly optimize the differentiable mapping!

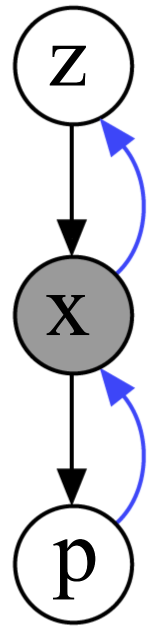
GANs for continuous data

1. Generation

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

2. Discrimination

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



4. Further gradient on generator

$$\frac{\nabla L(p)}{\nabla x} \frac{\nabla x}{\nabla \theta^{(G)}}$$

3. Gradient on generated data

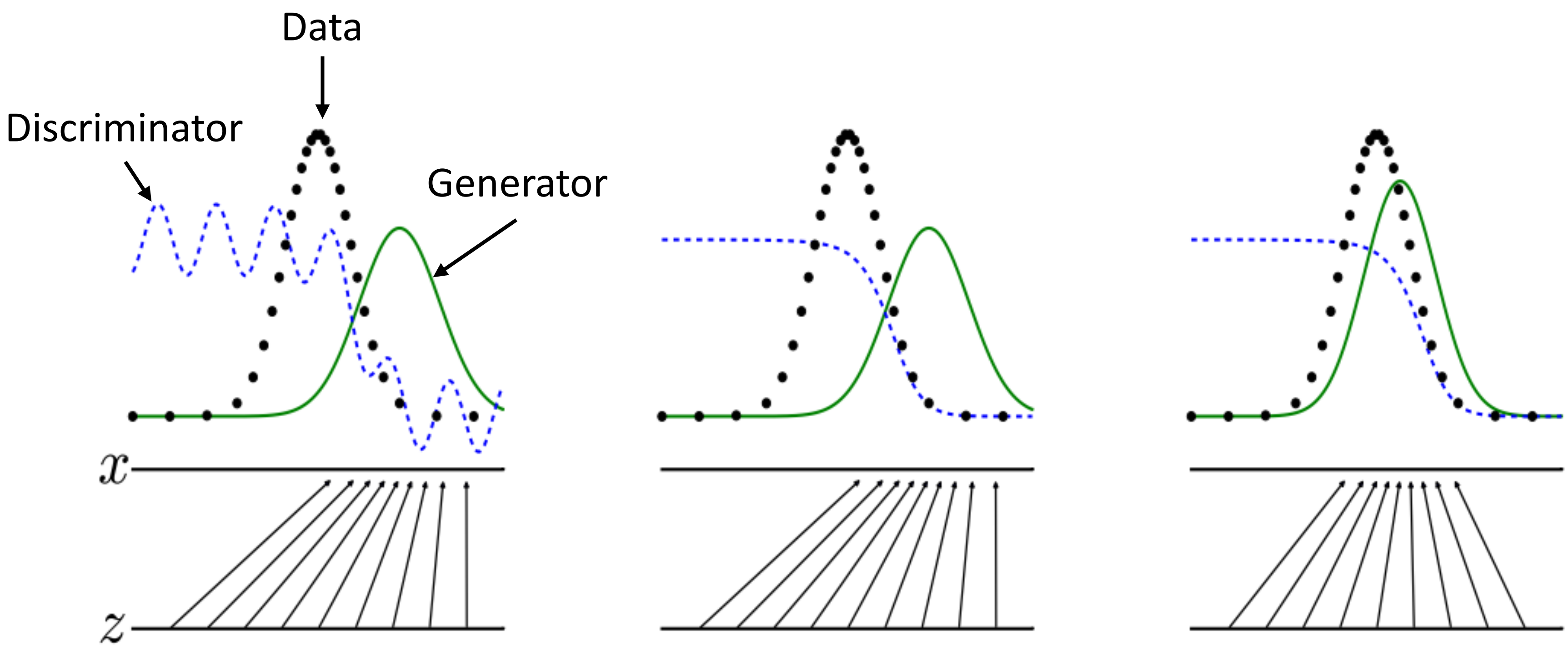
$$\frac{\nabla L(p)}{\nabla x}$$

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

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

Generator $\min_{G} \max_{D} J^{(D)}$ Discriminator $\max_{D} J^{(D)}$

GANs for continuous data



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

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

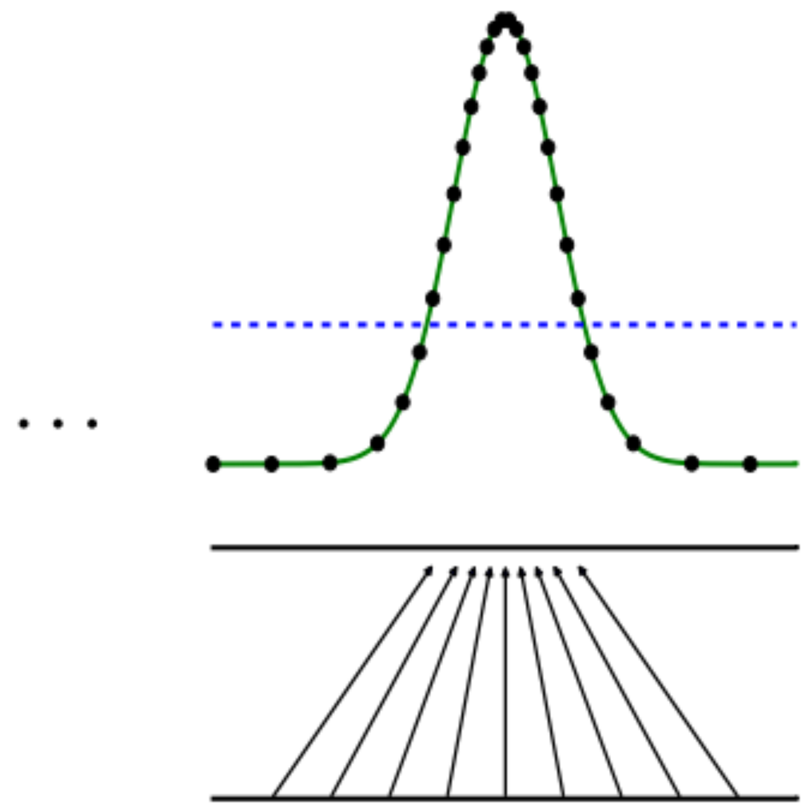
$$= \int_x p_{\text{data}}(x) \cdot \log(D(x))dx + \int_z p_z(z) \cdot \log(1 - D(\boxed{G(z)}))dx$$

Directly optimize the differentiable mapping!

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:

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

end for

Training generator

- 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:

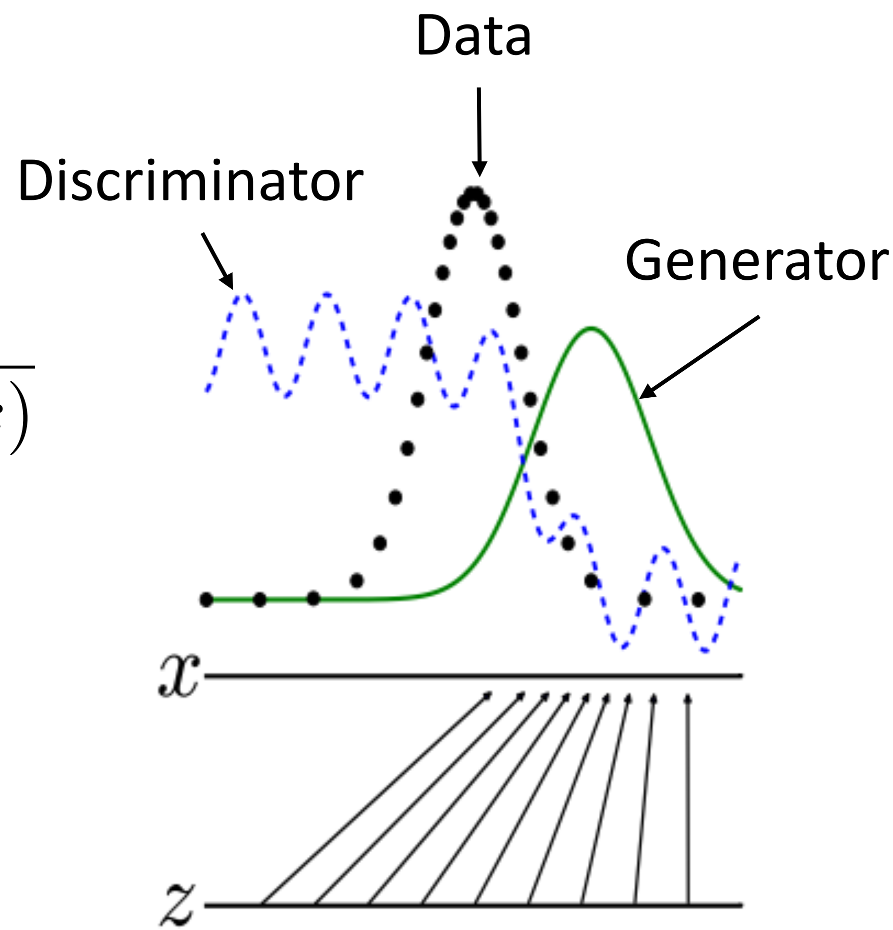
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Optimal Strategy for Discriminator

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

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



Reformulate the Minimax Game

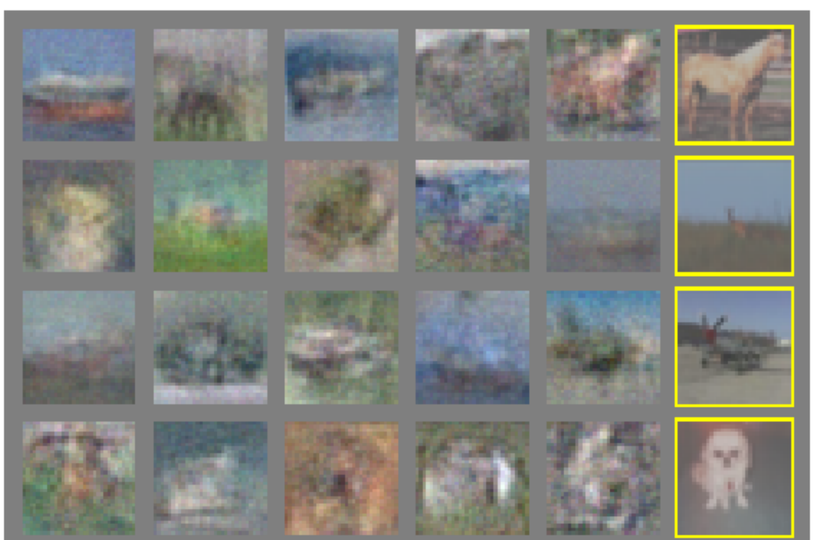
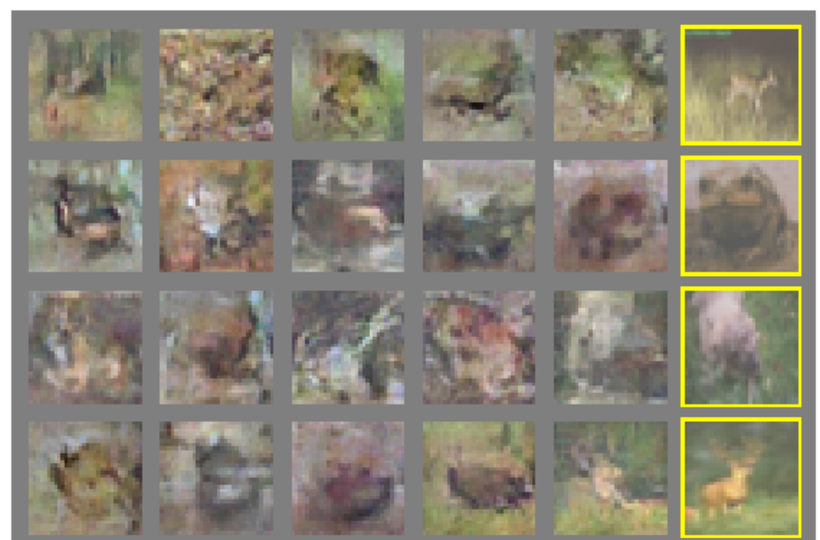
$$
\begin{aligned}
J^{(D)} &= \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \\
&= \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{x\sim p_G(x)}[\log(1 - D(x))] \\
&= \mathbb{E}_{x\sim p_{\text{data}}(x)} \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)} \\
&+ \mathbb{E}_{x\sim p_G(x)} \log \frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)} \\
&= \log(4) + KL(p_{\text{data}} || \frac{p_{\text{data}} + p_G}{2}) + KL(p_G || \frac{p_{\text{data}} + p_G}{2})
\end{aligned}
$$

$\min_G J^{(D)}$ 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).]

Case Study



Generating discrete data

- The generative model computes a probability distribution over the candidate choices
- First computes the Probability Density Function of data x (e.g. softmax over the vocabulary), then sample from it.

GANs for discrete data

- With explicit P.D.F. we can simply start with the most general form of the minimax game:

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

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

$$= \int_x p_{\text{data}}(x)\cdot \log(D(x)) + \boxed{G(x)} \cdot \log(1-D(x))dx$$

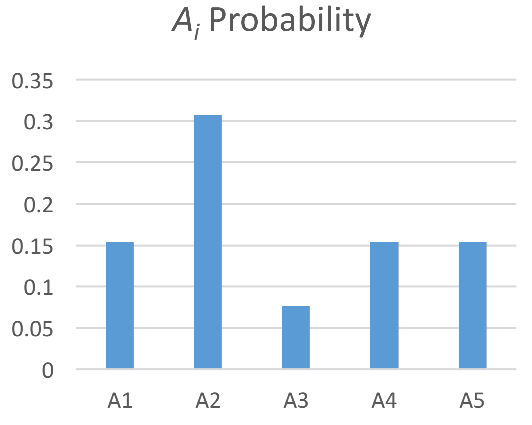
Probability Density Function

- Now, instead of optimizing a transformation, we can directly optimize the P.D.F. with the guidance of the discriminator.
- Note that even for discrete data, $G(x)$ is differentiable!

GANs for Discrete Data

- We could direct build a parametric distribution for discrete data
  - For example of the discrete data

$$\{A_1; A_2; A_3; A_4; A_5\}$$



  - The data P.D.F. could be defined as

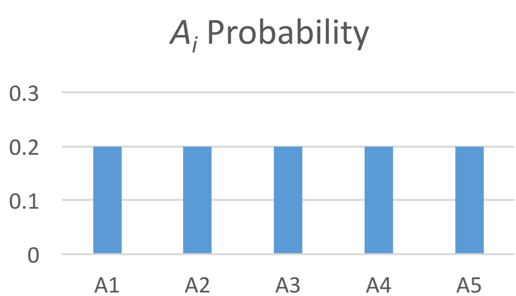
$$p(A_i) = \frac{e^{f(A_i)}}{\sum_j e^{f(A_j)}}$$

where the scoring function could be defined based on domain knowledge, e.g., a neural network with A_i embedding as the input

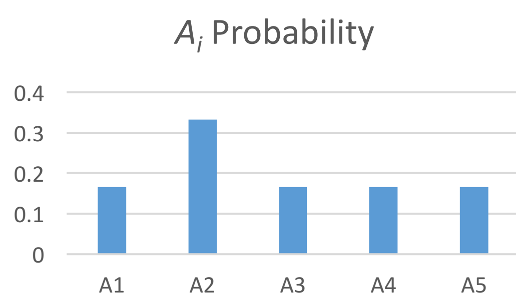
Borrow the Idea from RL

- For a generator $G(A_i) = P(A_i; \theta^{(G)})$
- Intuition
  - lower the probability of the choice that leads to low value/reward
  - higher the probability of the choice that leads to high value/reward
- The one-field 5-category example

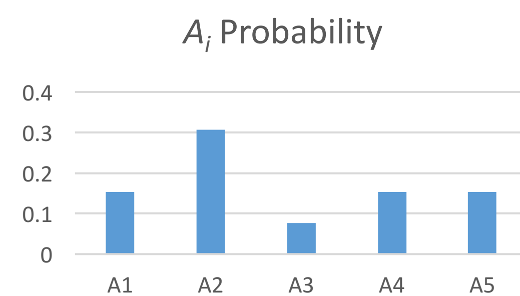
1. Initialize ϑ



3. Update ϑ by policy gradient



5. Update ϑ by policy gradient



2. Generate A_2
Observe positive reward (from D)

4. Generate A_3
Observe negative reward (from D)

Advances: GANs for Discrete Data

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

SeqGAN: Sequence Generation via GANs with Policy Gradient

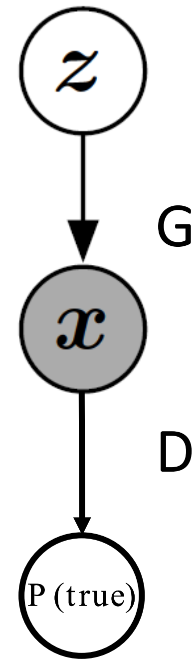
[Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. AAAI 2017.]

Problem for Discrete Data

- On continuous data, there is direct gradient

$$\nabla_{\theta^{(G)}} \frac{1}{|m|} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)})))$$

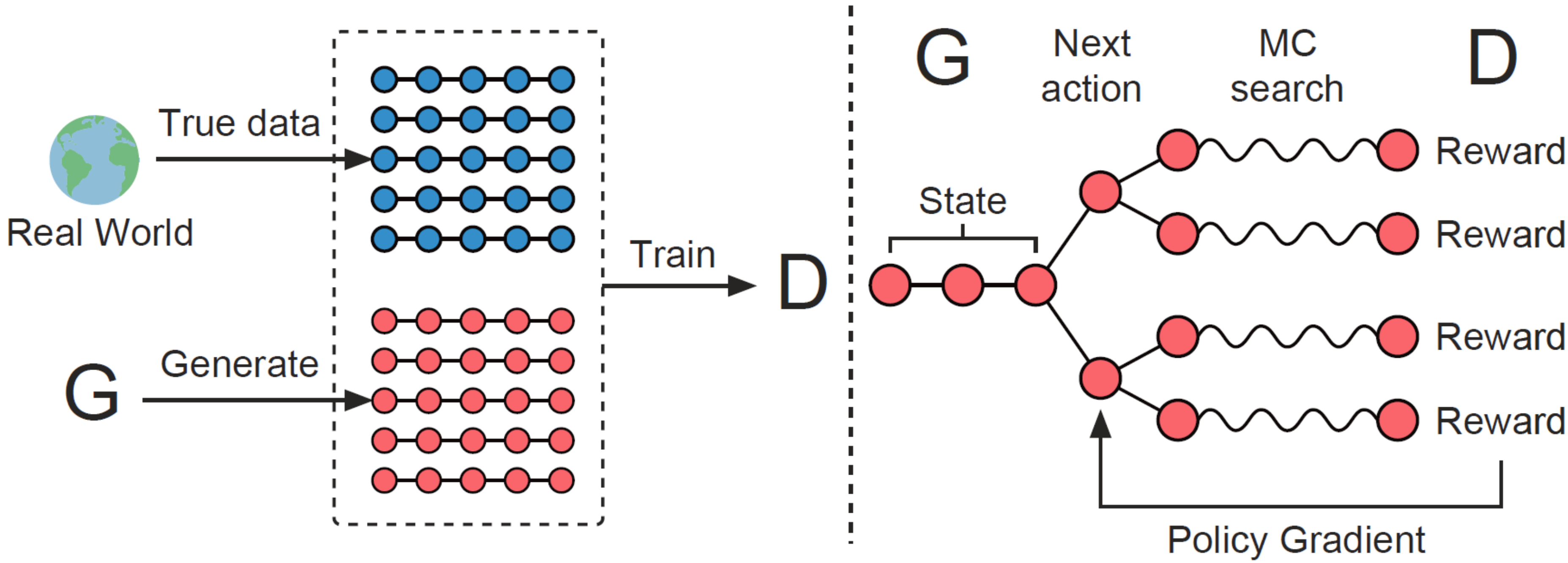
  - Guide the generator to (slightly) modify the output



- No direct gradient on discrete data
  - Text generation example
    - “床前明月光”
    - “I caught a penguin in the park”
    - From Ian Goodfellow: “If you output the word ‘penguin’, you can't change that to "penguin + .001" on the next step, because there is no such word as "penguin + .001". You have to go all the way from "penguin" to "ostrich".”

[https://www.reddit.com/r/MachineLearning/comments/40ldq6/generative_adversarial_networks_for_text/]

SeqGAN



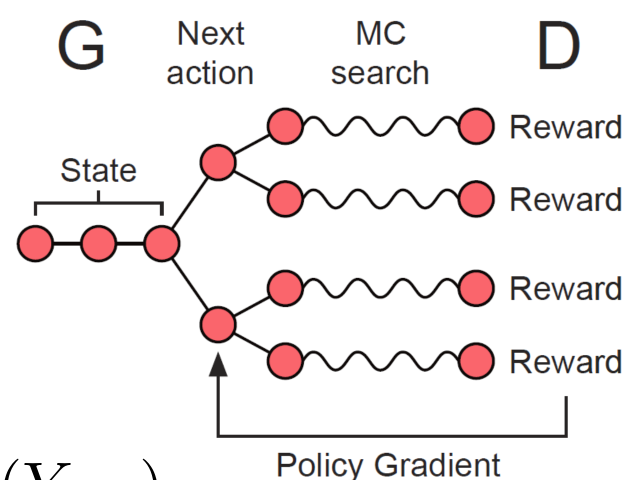
- Generator is a reinforcement learning policy $G(y_t|Y_{1:t-1})$ of generating a sequence
  - decide the next word to generate given the previous ones
- Discriminator provides the reward (i.e. the probability of being true data) $D(Y_{1:T}^n)$ for the whole sequence

Sequence Generator

- Objective: to maximize the expected reward

$$J(\theta) = \mathbb{E}[R_T | s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1 | s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

  - 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)



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

State-Action Value Setting

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

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

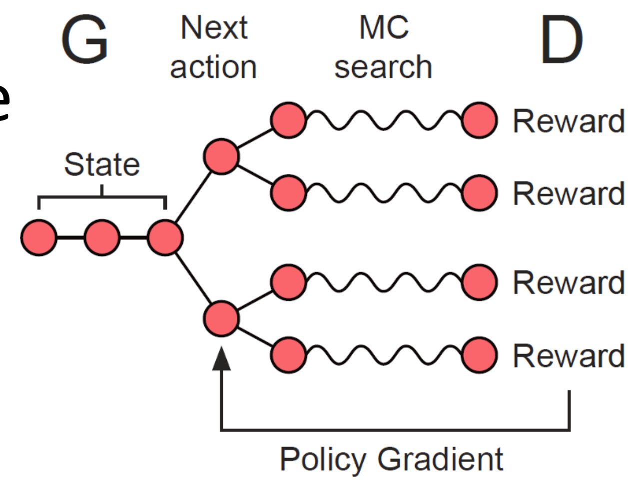
- For intermediate state-action value
  - Use Monte Carlo search to estimate

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

  - Following a roll-out policy G

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

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



Training Sequence Generator

- Policy gradient (REINFORCE)

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

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

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

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

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

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

Training Sequence Discriminator

- Objective: standard bi-classification

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

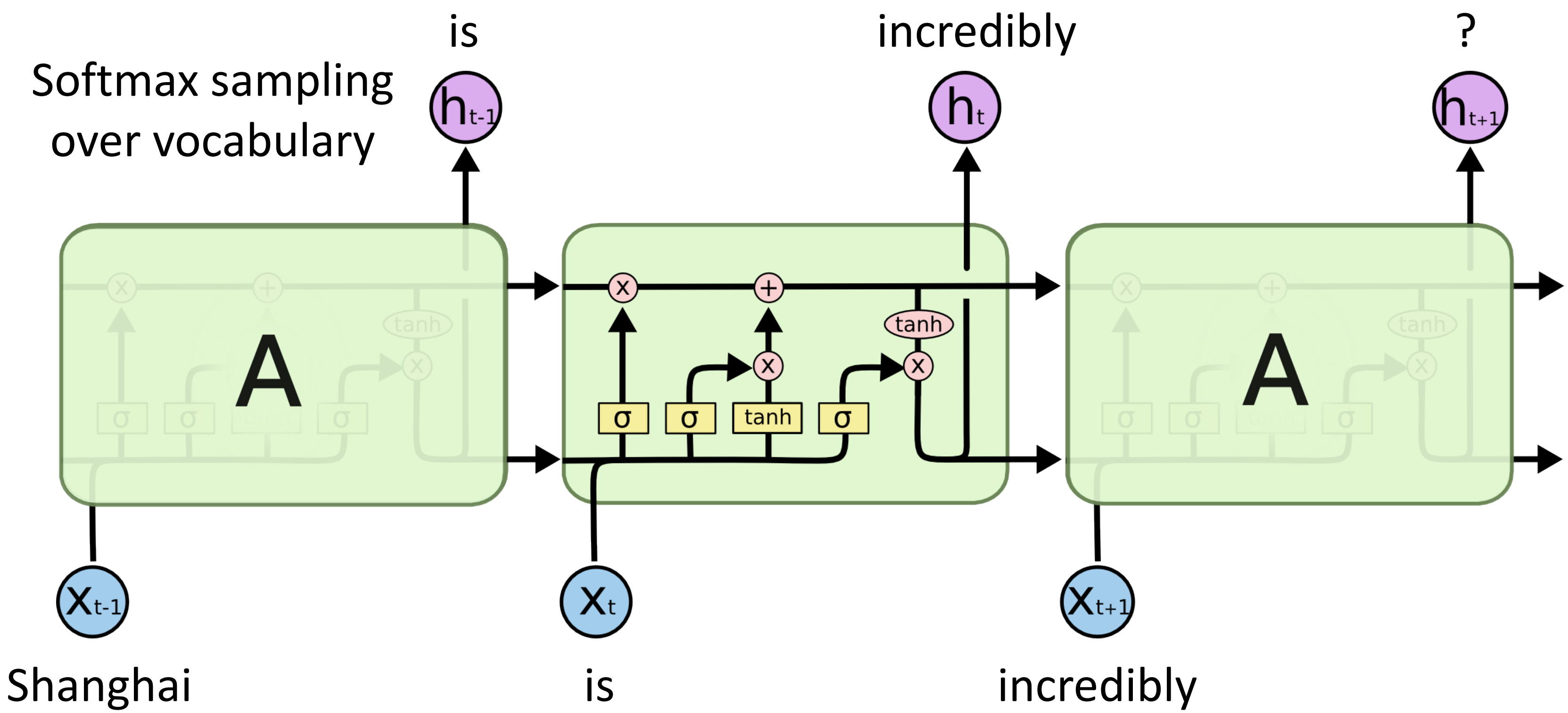
Overall Algorithm

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

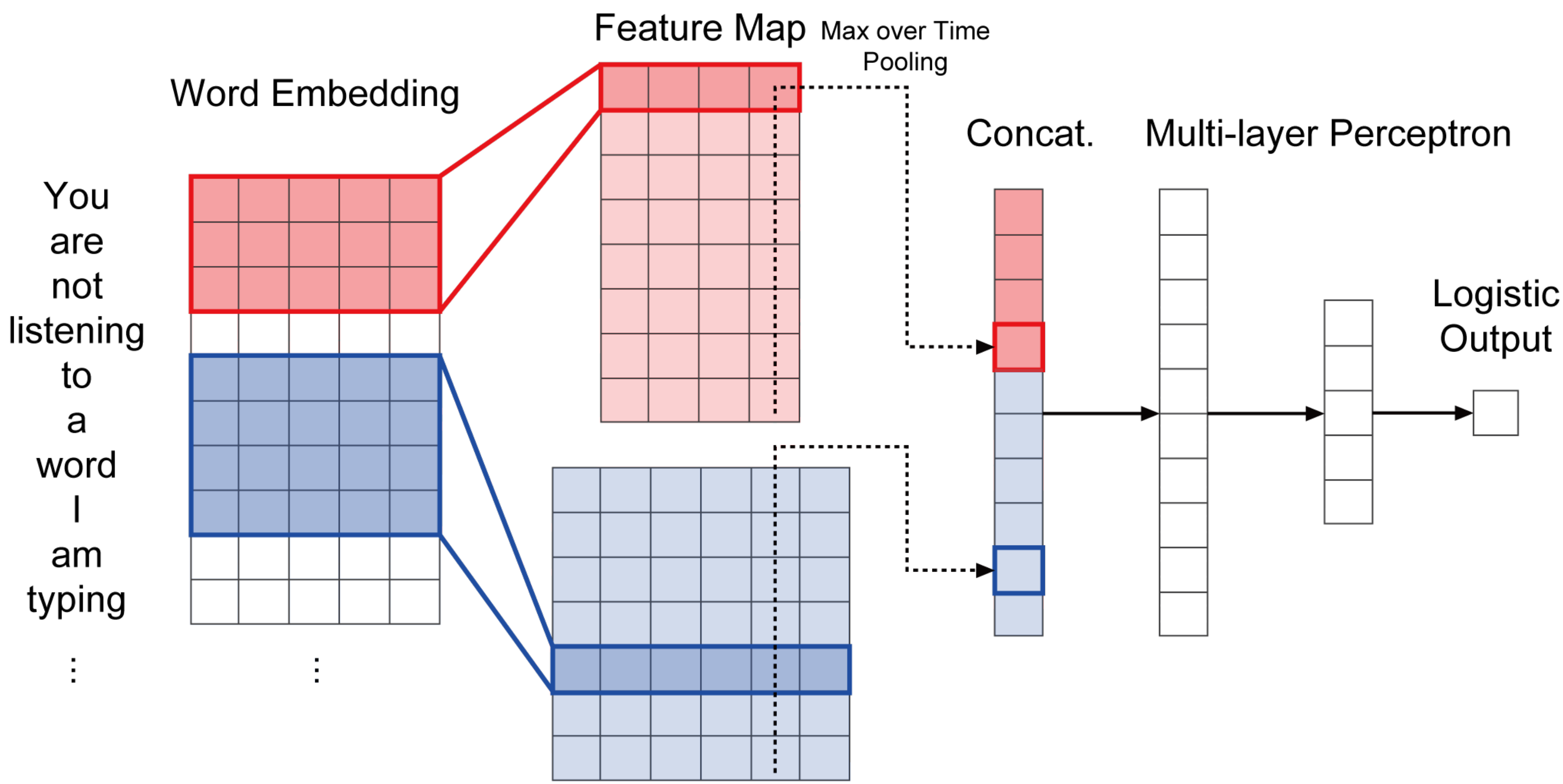
Sequence Generator Model



- RNN with LSTM cells

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

Sequence Discriminator Model



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

Experiments on Synthetic Data

- Evaluation measure with Oracle

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

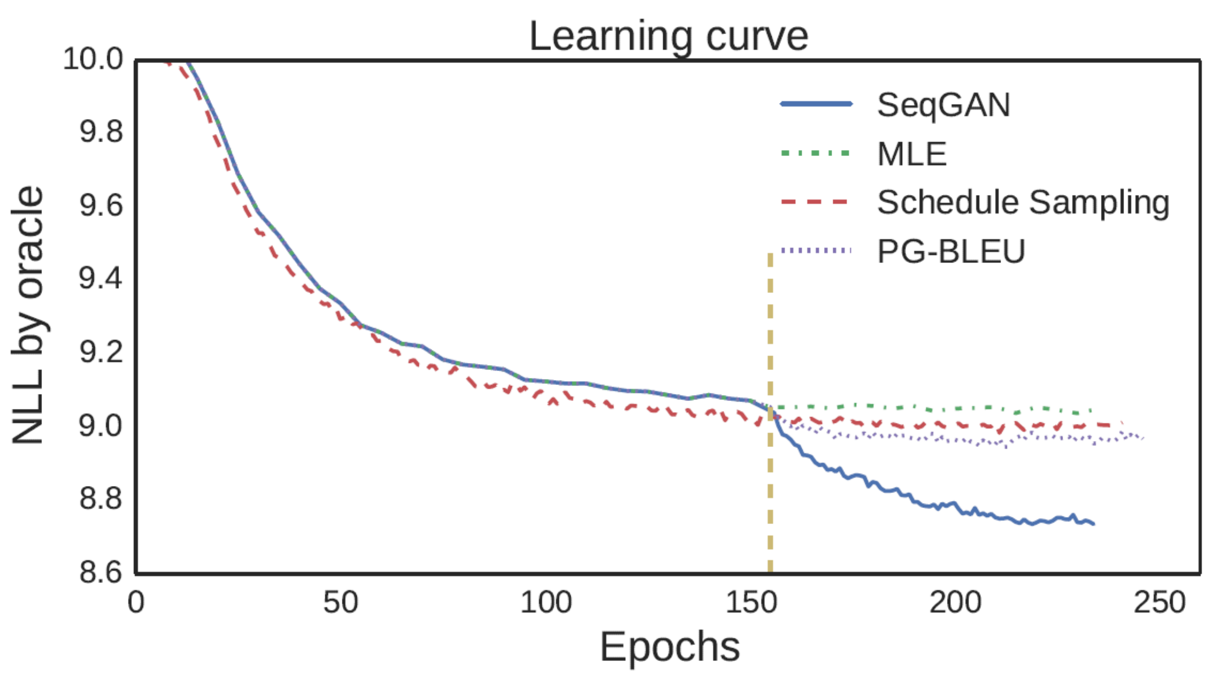
- An oracle model (e.g. the randomly initialized LSTM)
  - Firstly, the oracle model produces some sequences as training data for the generative model
  - Secondly the oracle model can be considered as the human observer to accurately evaluate the perceptual quality of the generative model

Experiments on Synthetic Data

- Evaluation measure with Oracle

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

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

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。

Can you distinguish which part is from human or machine?

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

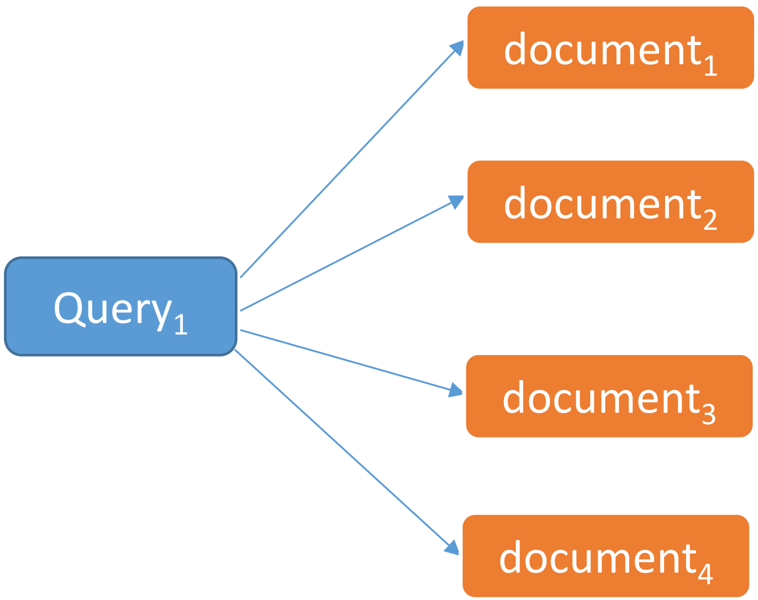
Machine

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.

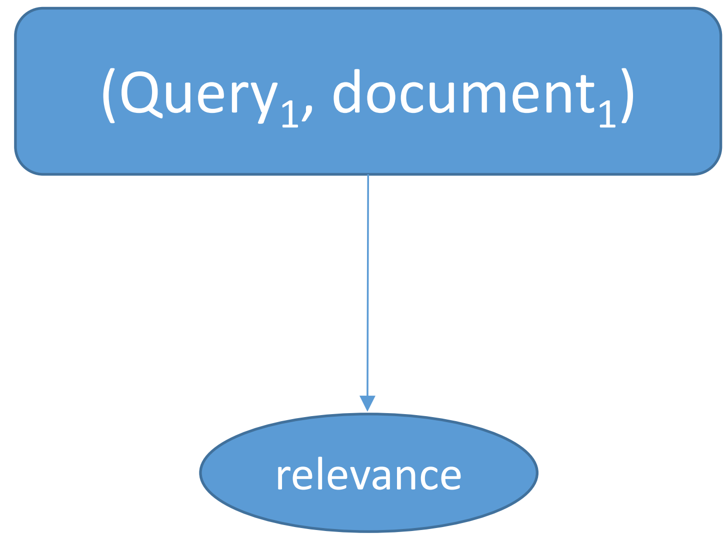
Two schools of thinking in IR modeling

Generative Retrieval



- Assume there is an underlying stochastic generative process between documents and queries
- Generate/Select relevant documents given a query

Discriminative Retrieval



- Learns from labeled relevant judgments
- Predict the relevance given a query-document pair

Three paradigms in Learning to Rank (LTR)

- **Pointwise:** learn to approximate the relevance estimation of each document to the human rating
- **Pairwise:** distinguish the more-relevant document from a document pair
- **Listwise:** learn to optimise the (smoothed) loss function defined over the whole ranking list for each query

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 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 for document ranking.

IRGAN Formulation

- The 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

IRGAN Formulation

- Overall Objective

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

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

IRGAN Formulation

- Optimizing Discriminative Retrieval

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

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

IRGAN Formulation

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

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

Reward Term

- REINFORCE (Advantage Function)

IRGAN Formulation

- Algorithm

Algorithm 1 Minimax Game for IR (a.k.a IRGAN)

Input: generator $p_\theta(d|q,r)$; discriminator $f_\phi(\boldsymbol{x}_i^q)$;
training dataset $\mathcal{S} = \{\boldsymbol{x}\}$

Initialise $p_\theta(d|q,r)$, $f_\phi(q,d)$ with random weights θ, ϕ.
Pre-train $p_\theta(d|q,r)$, $f_\phi(q,d)$ using $\mathcal{S}$
repeat
  for g-steps **do**
    $p_\theta(d|q,r)$ generates K documents for each query q
    Update generator parameters via policy gradient Eq. (5)
  end for
  for d-steps **do**
    Use current $p_\theta(d|q,r)$ to generate negative examples and combine with given positive examples $\mathcal{S}$
    Train discriminator $f_\phi(q,d)$ by Eq. (3)
  end for
until IRGAN converges

IRGAN Formulation

- 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 rather than absolute relevance judgements

- Now, for each query q_n ,we have a set of labelled document pairs $R_n = \{\langle d_i, d_j \rangle | d_i \succ d_j\}$

IRGAN Formulation

- Extension to Pairwise Case
  - Discriminator would try to predict if a document pair is correctly ranked, which can be implemented as many pairwise ranking loss function:
    - RankNet: $\log(1 + \exp(-z))$
    - Ranking SVM (Hinge Loss): $(1 - z)_+$
    - RankBoost: $\exp(-z)$

  where $z = f_\phi(d_u, q) - f_\phi(d_v, q)$

IRGAN Formulation

- Extension to Pairwise Case
  - Generator would try to generate document pairs that are similar to those in R_n , i.e., with the correct ranking.
  - A softmax function over the Cartesian Product of the document sets, where the logits is the advantage of d_i over d_j in a document pair (d_i, d_j)

An Intuitive Explanation of IRGAN

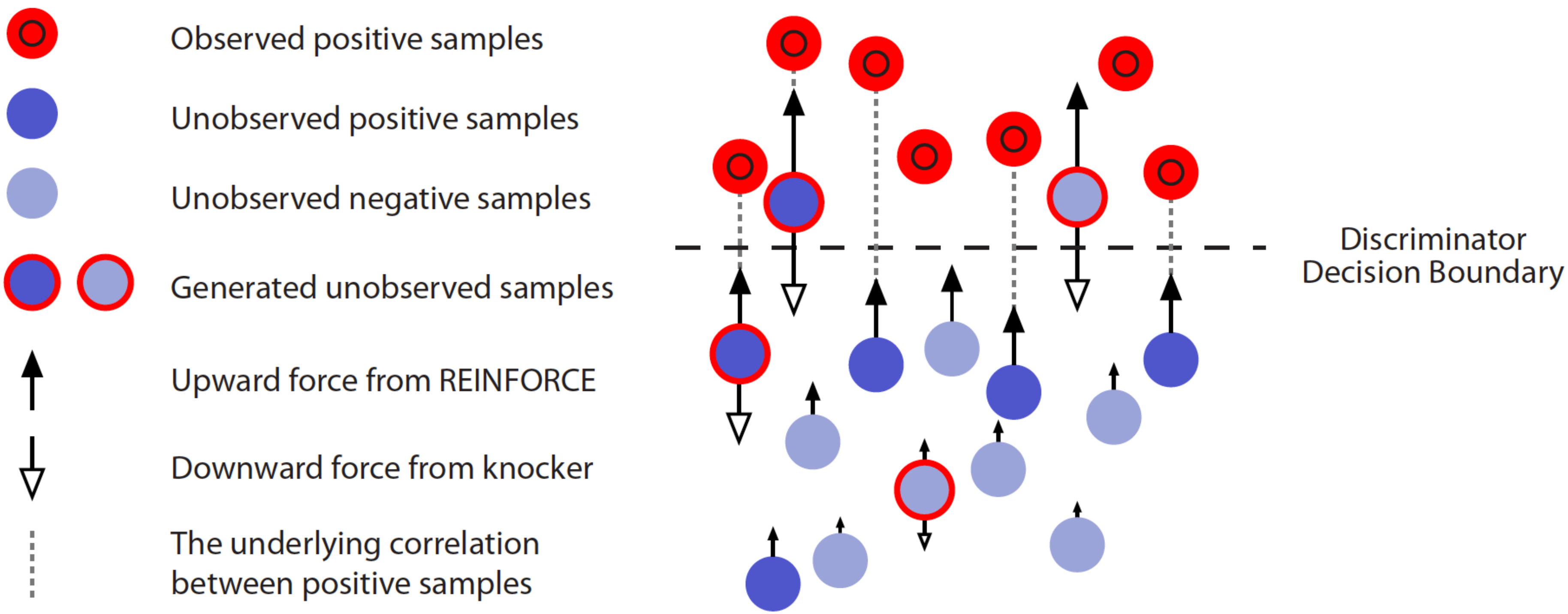


Figure 1: An illustration of IRGAN training.

An Intuitive Explanation of IRGAN

- The generative retrieval model is guided by the signal provided from the discriminative retrieval model, which makes it more favorable than the non-learning methods or the Maximum Likelihood Estimation (MLE) scheme.

- The discriminative retrieval model could be enhanced to better rank top documents via a strategic negative sampling from the generator.

Experiments: Web Search

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

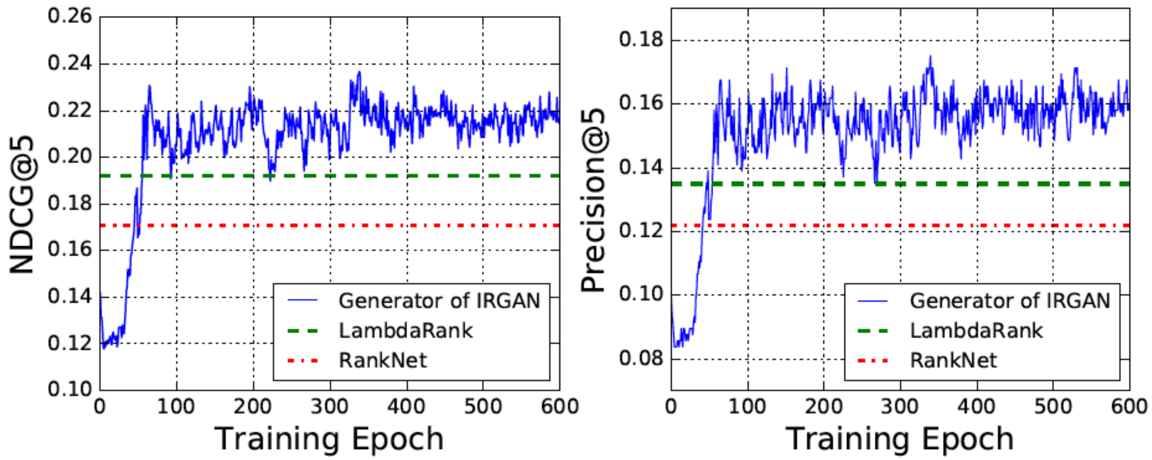


Figure 2: Learning curves of the pointwise IRGAN on web search task.

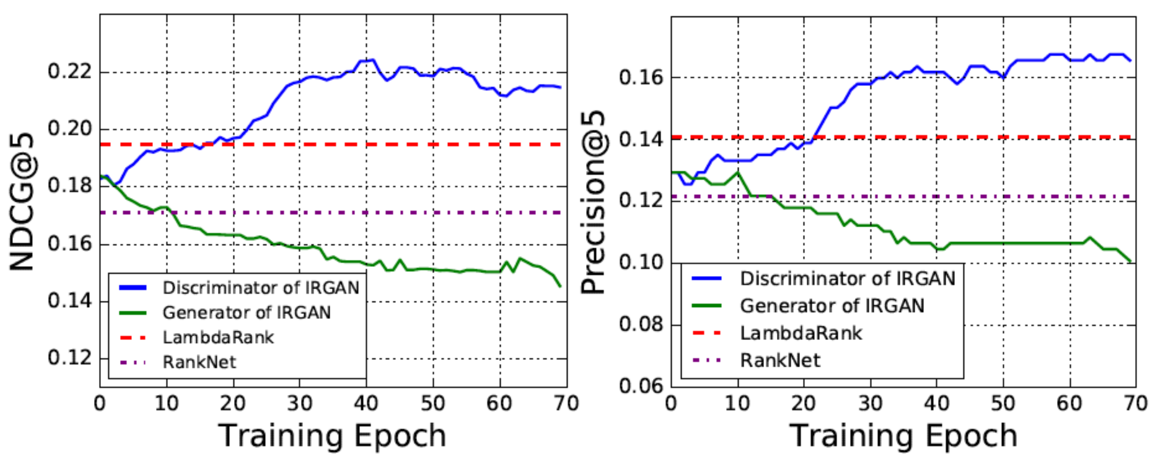


Figure 3: Learning curves of the pairwise IRGAN on web search task.

Experiments: Item Recommendation

Table 3: Item recommendation results (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%*

Table 4: Item recommendation results (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%*

Experiments: Item Recommendation

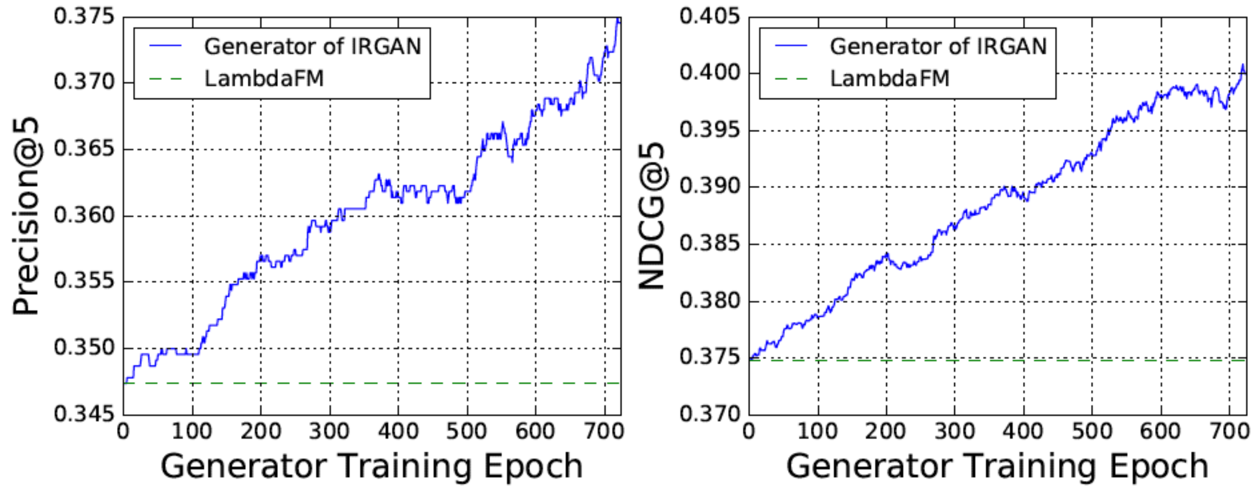


Figure 6: Learning curve of precision and NDCG of the generative retrieval model for top-5 item recommendation task on Movielens dataset.

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%

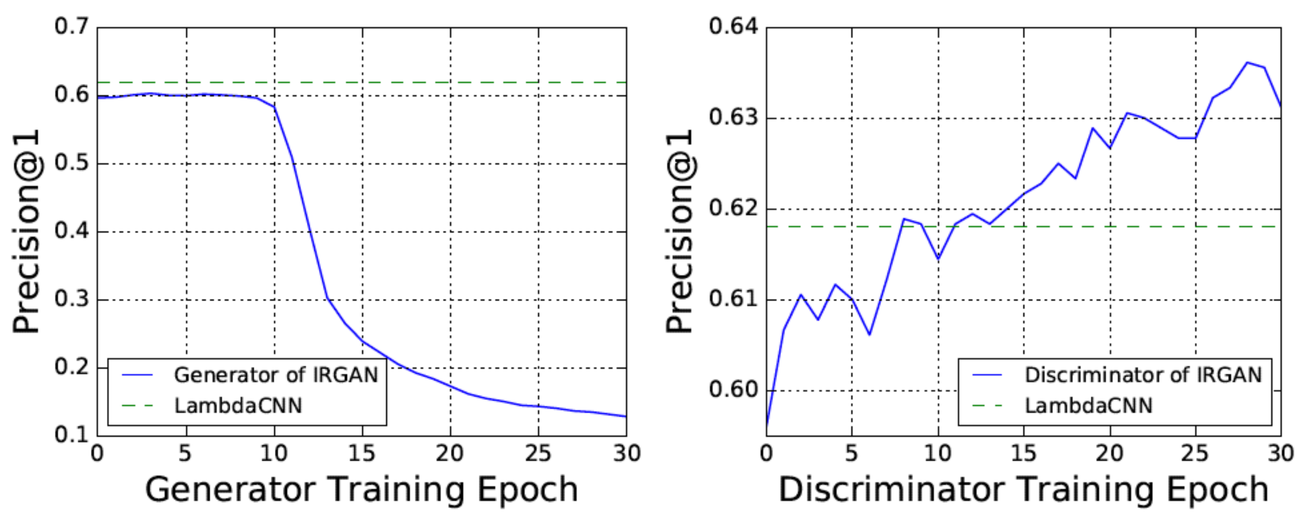


Figure 8: The experimental results in QA task.

Summary of IRGAN

- We proposed IRGAN framework that unifies two schools of information retrieval methodologies, via adversarial training in a minimax game, which takes advantage of both schools of thinking.
- Significant performance gains were observed in three typical information retrieval tasks.
- Experiments suggest that different equilibria could be reached in the end depending on the tasks and settings.

Summary of This Talk

- Generative Adversarial Networks (GANs) are so popular in deep unsupervised learning research

- GANs provide a new training framework closer to the target use of the generative model

$$\max \mathbb{E}_{x \sim p(x)}[\log q(x)]$$

Training/evaluation

$$\max \mathbb{E}_{x \sim q(x)}[\log p(x)]$$

Use (Turing Test)

- For discrete data generation, one could directly define the parametric distribution and optimize the P.D.F. by policy gradient methods, e.g. REINFORCE

Thank You

Lantao Yu

Shanghai Jiao Tong University

http://lantaoyu.com

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