Maximum-Likelihood Augmented Discrete Generative Adversarial Networks (MaliGAN)

Tong Che, et al.

Presenting: Yengeny Tkach

2019 Spring @ https://qdata.github.io/deep2Read/

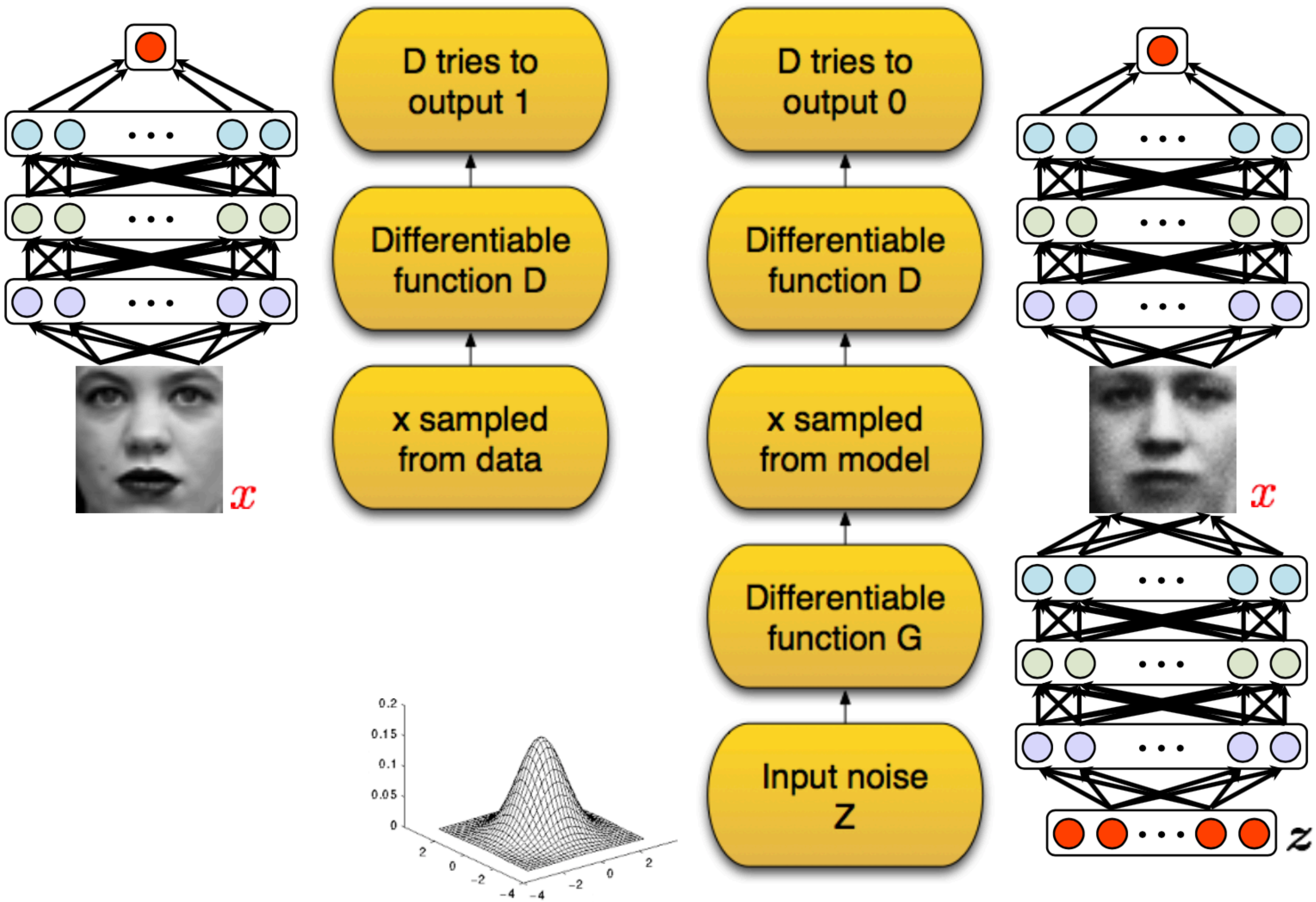
Executive Summary

- MaliGAN is a GAN based generative model for discrete sequences, trained using RL methods for variance reduction.
- The optimization objective of the generative function is replaced in this work with $KL(Q||P\downarrow G)$ where $P\downarrow G$ is the distribution of the generated data and Q is a self-normalized importance sampling (SIS) estimation of the data distribution.
- To reduce the variance of the gradient signal the authors mix sampling from the true data and the generated data distributions.

Outline

- GAN – Basic Idea
- Discrete data challenges
- Importance Sampling
- MaliGAN – Basic
- policy gradient
- Sequential MaliGAN with Mixed MLE Training
- seqGAN
- Experiments

Basic Idea of GAN



GAN Formally

- Value Function:

$$V(\mathbb{P}, G_\theta, D_\phi) = E_{x\sim P}[\log D(x)] + E_{x\sim Q}[\log(1-D(x))]$$
$$= E_{x\sim P}[\log D(x)] + E_{z\sim h(z)}[\log(1-D(G(z)))]$$

- Monte-Carlo Approximation:

$$V(\mathbb{P}, G_\theta, D_\phi) = 1/m \sum_{i=1}^{m} \log D(x^{i}) + 1/m \sum_{i=1}^{m} \log(1-D(G(z^{i})))$$

- Discriminator target:

$$\max_{\phi} V(\mathbb{P}, G_\theta, D_\phi)$$

- Generator target:

$$\min_{\theta} \max_{\phi} V(\mathbb{P}, G_\theta, D_\phi)$$

Algorithm

Initialize ϕ_d for D and θ_g for G

- In each training iteration:

Learning D

Repeat k times

  - Sample m examples $\{x^1, x^2, \ldots, x^m\}$ from data distribution $P(x)$
  - Sample m noise samples $\{z^1, z^2, \ldots, z^m\}$ from the prior $h(z)$
  - Obtaining generated data $\{x^1, x^2, \ldots, x^m\}$, $x^i = G(z^i)$
  - Update discriminator parameters θ_d to maximize
    - $V = \frac{1}{m}\sum_{i=1}^{m} \log D(x^i) + \frac{1}{m}\sum_{i=1}^{m} \log(1 - D(x^i))$
    - $\phi_d \leftarrow \phi_d + \eta \nabla V(\phi_d)$

Learning G

Only Once

  - Sample another m noise samples $\{z^1, z^2, \ldots, z^m\}$ from the prior $P_{prior}(z)$
  - Update generator parameters θ_g to minimize
    - $V = \frac{1}{m}\sum_{i=1}^{m} \log D(x^i) + \frac{1}{m}\sum_{i=1}^{m} \log(1 - D(G(z^i)))$
    - $\theta_g \leftarrow \theta_g - \eta \nabla V(\theta_g)$

GAN for Discrete sequences

Adapting GAN to generating discrete data is challenging:

- How do we calculate $\nabla V(\theta_g)$? G(z) is discontinuous.
- How can we reduce the variance of $\nabla V(\theta_g)$ for long sequence generation

Importance Sampling

$$E_{x\sim P}[f(x)] = \int f(x)p(x)dx$$
$$= \int f(x)\frac{p(x)}{q(x)}q(x)dx$$
$$= \int f(x)w(x)q(x)dx$$
$$= E_{x\sim Q}[f(x)w(x)]$$
$$= E_{x\sim Q}[f(x)w(x)]/E_{x\sim Q}[w(x)]$$
$$w(x) = p(x)/q(x)$$

In case p or q are scaled density functions

$w(x)$ - Importance Weights

Importance sampling in MaliGAN

Basic idea: optimal discriminator $D^*(x)$ holds:

$$D^*(x)=\frac{p_d(x)}{p_\theta(x)+p_d(x)} \Leftrightarrow p_d(x)=\frac{D(x)}{1-D(x)}p_\theta(x)$$

Where $p_d(x)$ in true data distribution and $p_\theta(x)$ is generated. We can estimate $p_d(x)$ by $q(x)$:

$$q(x)=\frac{r_D(x)}{\mathbb{E}[r_D(x)]}p_\theta(x) \;, r_D(x)=\frac{D(x)}{1-D(x)}$$

Generator loss:

$$L_G(\theta)=KL(q(x)||p_\theta(x))$$

$$\nabla L_G(\theta)=-\mathbb{E}_{p_d}[\nabla_\theta \log p_\theta(x)]=-\mathbb{E}_{p_\theta}\left[\frac{r_D(x)}{\mathbb{E}[r_D(x)]}\nabla_\theta \log p_\theta(x)\right]$$

Why self normalization?

If we would use $r_D(x)$:

- In the beginning of the training $D(x)$ close to 0 and $r_D(x)$ will offer a very poor gradient direction with very little change.
- For some instances during the training $D(x)$ will be close to 1 and $r_D(x)$ will explode.
- This ensures that the model can always learn something as long as there exist some generations better than others and controls the decreases the gradient variance.

MaliGAN Algorithm

Algorithm 1 MaliGAN

Require: A generator p with parameters θ.
A discriminator $D(x)$ with parameters θ_d.
A baseline b.

for number of training iterations **do**
    for k steps **do**
        Sample a minibatch of samples $\{\mathbf{x}_i\}_{i=1}^m$ from p_θ.
        Sample a minibatch of samples $\{\mathbf{y}_i\}_{i=1}^m$ from p_d.
        Update the parameter of discriminator by taking gradient ascend of discriminator loss

$$\sum_i [\nabla_{\theta_d} \log D(\mathbf{y}_i)] + \sum_i [\nabla_{\theta_d} \log(1 - D(\mathbf{x}_i))]$$

    end for
    Sample a minibatch of samples $\{\mathbf{x}_i\}_{i=1}^m$ from p_θ.
    Update the generator by applying gradient update

$$\sum_{i=1}^{m} \left(\frac{r_D(\mathbf{x}_i)}{\sum_i r_D(\mathbf{x}_i)} - b\right) \nabla \log p_\theta(\mathbf{x}_i)$$

end for

Policy Gradient

- $J(\theta)$ *the expected reward under a stochastic policy* π_θ
- $r(\tau)$ *is the reward of trajectory* τ

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau$$

- Stochastic policy gradient:

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau$$

$$= E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

- In discrete GANs π_θ is the generator G_θ that produces a distribution over discrete objects (actions)
- $r(\tau)$ in MaliGAN is $r_D(x)/\mathbb{E}[r_D(x)]$

$\pi(\tau)$ is defined as:

$$\underbrace{\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

Take the log:

$$\log \pi_\theta(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

The first and the last term does not depend on θ and can be removed.

$$\nabla_\theta \left[\cancel{\log p(\mathbf{s}_1)} + \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \cancel{\log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)} \right]$$

mixed MLE-MaliGAN

To further decrease the variance that maybe accumulated over long sequences:

- use the training data for N time steps and switch to free running mode for the remaining T-N time steps.
- For the first N tokens, that are from the training data, the generator objective is MLE and for the rest is the MaliGAN

$$
\begin{aligned}
\nabla L_G =& \mathbb{E}_q[\nabla \log p_\theta(\mathbf{x})] \\
=& \mathbb{E}_{p_d}[\nabla \log p_\theta(\mathbf{x}_{\leq N})] + \mathbb{E}_q[\nabla \log p_\theta(\mathbf{x}_{>N} | \mathbf{x}_{<N})] \\
=& \mathbb{E}_{p_d}[\nabla \log p_\theta(x_0, x_1, \cdots x_T)] \\
& + \frac{1}{Z}\mathbb{E}_{p_\theta}[\sum_{t=N+1}^{L} r_D(\mathbf{x}) \nabla \log p_\theta(a_t | \mathbf{s}_t)]
\end{aligned}
$$

mixed MLE-MaliGAN

- for each $0 \leq N \leq T$:

$$\nabla L_G^N \approx \sum_{i=1,j=1}^{m,n} \left(\frac{r_D(\mathbf{x}_{i,j})}{\sum_j r_D(\mathbf{x}_{i,j})} - b\right) \nabla \log p_\theta(\mathbf{x}_{i,j}^{>N} | \mathbf{x}_i^{\leq N}) + \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{N} p_\theta(a_t^i | \mathbf{s}_t^i) = E_N(\mathbf{x}_{i,j}) \tag{4}$$

- During the training procedure N is decreased from T towards 0

Algorithm 2 Sequential MaliGAN with Mixed MLE Training

Require: A generator p with parameters θ.
A discriminator $D(x)$ with parameters θ_d.
Maximum sequence length T, step size K.
A baseline b, sampling multiplicity m.

$N = T$
Optional: Pretrain model using pure MLE with some epochs.
for number of training iterations **do**
  $N = N - K$
  for k steps **do**
    Sample a minibatch of sequences $\{\mathbf{y}_i\}_{i=1}^{m}$ from p_d.
    While keeping the first N steps the same as $\{\mathbf{y}_i\}_{i=1}^{m}$, sample a minibatch of sequences $\{\mathbf{x}_i\}_{i=1}^{m}$ from p_θ from time step N.
    Update the discriminator by taking gradient ascend of discriminator loss.

$$\sum_i [\nabla_{\theta_d} \log D(\mathbf{y}_i)] + \sum_i [\nabla_{\theta_d} \log(1 - D(\mathbf{x_i}))]$$

  end for
  Sample a minibatch of sequences $\{\mathbf{x}_i\}_{i=1}^{m}$ from p_d.
  For each sample $\mathbf{x}_i$ with length larger than N in the mini-batch, clamp the generator to the first N words of s, and freely run the model to generate m samples $\mathbf{x}_{i,j}, j = 1, \cdots m$ till the end of the sequence.
  Update the generator by applying the mixed MLE-Mali gradient update

$$\nabla L_G^N \approx \sum_{i=1,j=1}^{m,n} \left(\frac{r_D(\mathbf{x}_{i,j})}{\sum_j r_D(\mathbf{x}_{i,j})} - b\right) \nabla \log p_\theta(\mathbf{x}_{i,j}^{>N} | \mathbf{x}_i^{\leq N}) + \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{N} p_\theta(a_t^i | \mathbf{s}_t^i)$$

end for

Comparison to seqGAN

- The general formulation of the algorithm is similar to seqGAN by Yu et al.
- seqGAN is also a policy gradient based approach with slightly different formulation, one of the many popular forms in the RL literature.
- MaliGAN also mention a MCTS approximation that becomes slightly clearer in notation after reviewing the seqGAN algorithm that performs classic MCS.

Policy Gradient

- *Alternative forms*:

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right], \qquad (1)$$

where Ψ_t may be one of the following:

1. $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
2. $\sum_{t'=t}^{\infty} r_{t'}$: reward following action a_t.
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: baselined version of previous formula.
4. $Q^\pi(s_t, a_t)$: state-action value function.
5. $A^\pi(s_t, a_t)$: advantage function.
6. $r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$: TD residual.

The latter formulas use the definitions

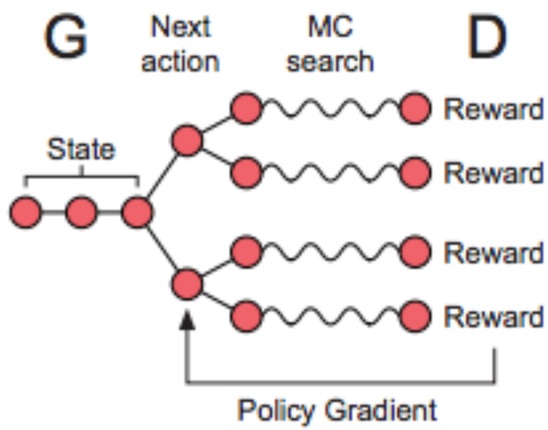
$$V^\pi(s_t) := \mathbb{E}_{\substack{s_{t+1:\infty},\\ a_{t:\infty}}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \qquad Q^\pi(s_t, a_t) := \mathbb{E}_{\substack{s_{t+1:\infty},\\ a_{t+1:\infty}}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \qquad (2)$$

$$A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t), \quad \text{(Advantage function)}. \qquad (3)$$

Fig. 1. A general form of policy gradient methods. (Image source: Schulman et al., 2016)

SeqGAN Algorithm

$$\nabla J(\theta)=E\sum_{t=1}^{T} Q(y_t, Y_{1:t-1})\nabla \log p_\theta(y_t | Y_{1:t-1})$$



$$Q_{D_\phi}^{G_\theta}(s=Y_{1:t-1}, a=y_t) = \tag{4}$$

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

SeqGAN 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

$$\theta \leftarrow \theta + \alpha_h \nabla_\theta J(\theta), \quad (8)$$

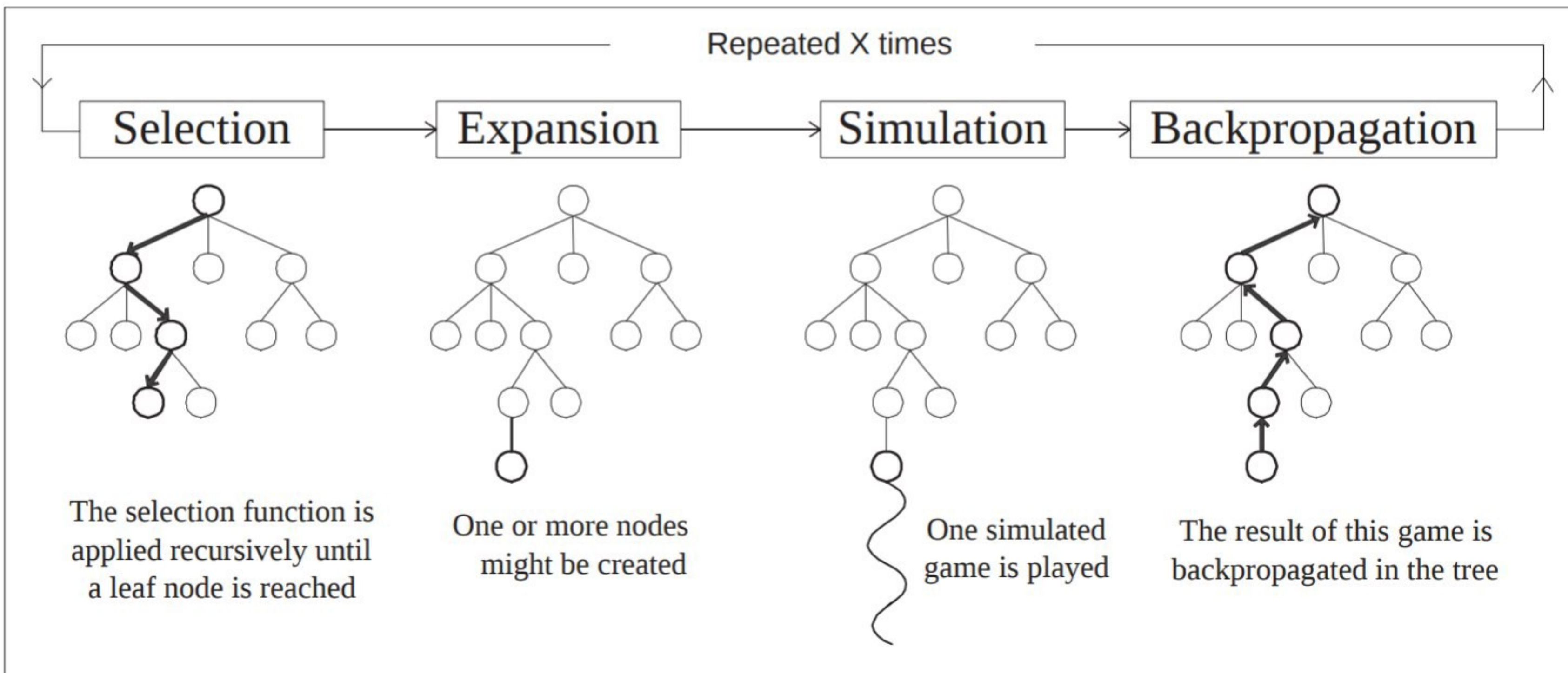
$$\min_\phi -\mathbb{E}_{Y \sim p_{\text{data}}}[\log D_\phi(Y)] - \mathbb{E}_{Y \sim G_\theta}[\log(1 - D_\phi(Y))]. \quad (5)$$

MaliGAN with MCTS

- Alternative loss function where $r(\tau)$ is replaced by $Q(a,s)$

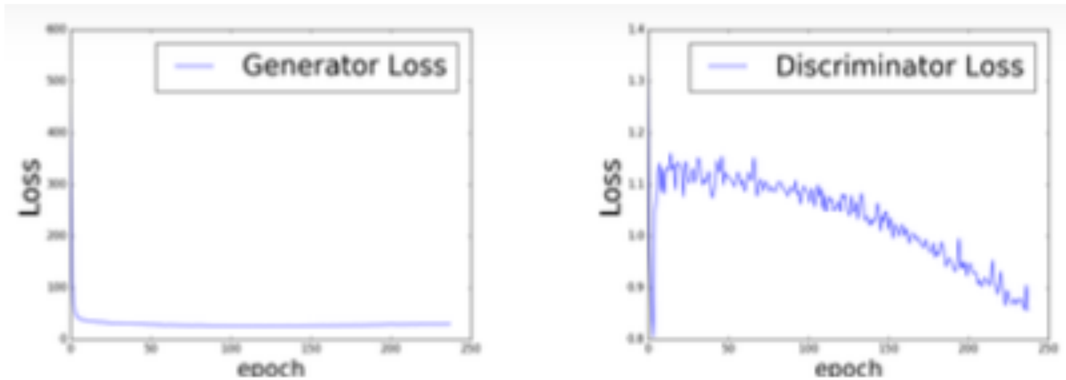
$$\nabla L_G(\theta) \approx \frac{\sum_i L_i}{m \sum Q(a_t^i, \mathbf{s}_t^i)} \sum_{i,t}^{m,L_i} Q(a_t^i, \mathbf{s}_t^i) \nabla \log p_\theta(a_t^i | \mathbf{s}_t^i)$$

- Che et al. use MCTS



Experiments

- Discrete MNIST



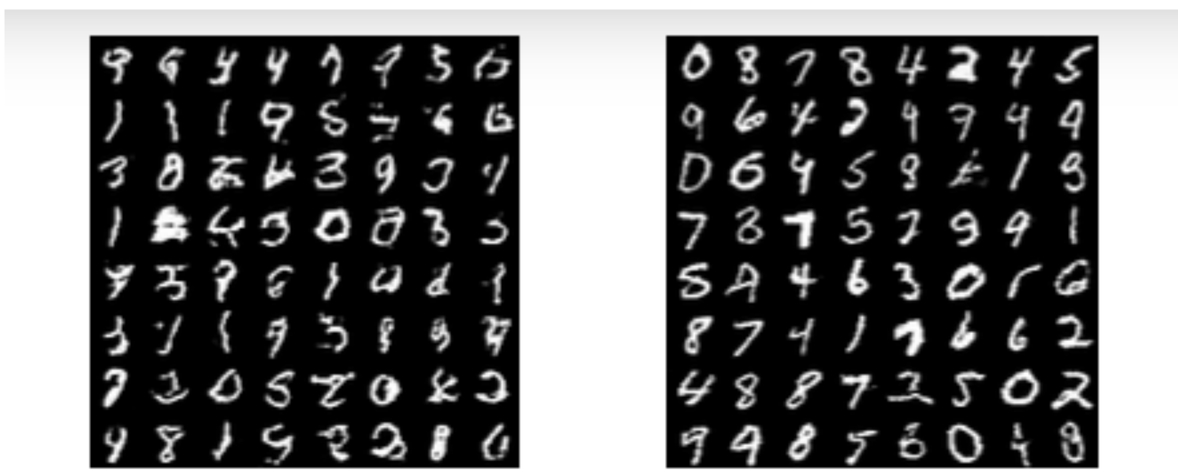
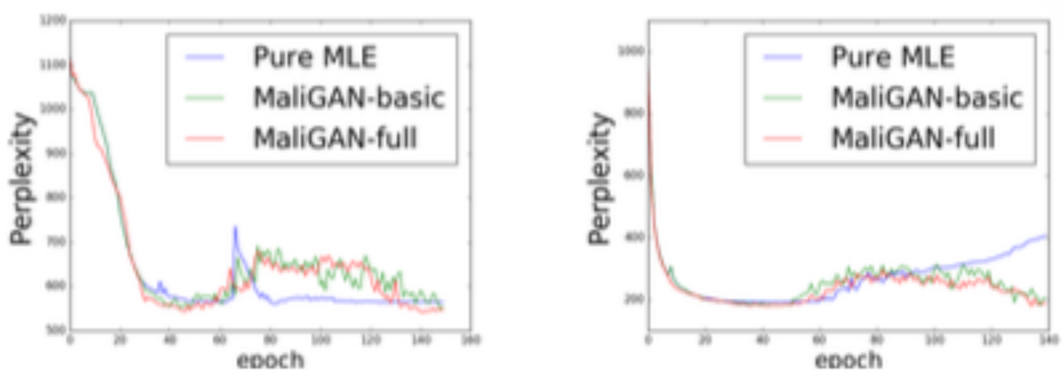


Figure 2. Samples generated by REINFORCE-like model (left) and by MaliGAN (right).

Experiments

- Chinese poem generation

Model	Poem-5		Poem-7	
	BLEU-2	PPL	BLEU-2	PPL
MLE	0.6934	564.1	0.3186	192.7
SeqGAN	0.7389	-	-	-
MaliGAN-basic	0.7406	548.6	0.4892	182.2
MaliGAN-full	**0.7628**	**542.7**	**0.5526**	**180.2**



- Sentence-Level Language Modeling

	MLE	MaliGAN-basic	**MaliGAN-full**
Valid-Perplexity	141.9	131.6	**128.0**
Test-Perplexity	138.2	125.3	**123.8**

Discussion

Main takeaways:

- Try to reduce the variance and keep the bias unchanged to stabilize learning.
- Off-policy gives us better exploration and helps us use data samples more efficiently.
- Experience replay (training data sampled from a replay memory buffer);
- Batch normalization;

References

T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio. Maximum-likelihood augmented discrete generative adversarial networks. arXiv preprint arXiv:1702.07983, 2017.

Yu, Lantao, Zhang, Weinan, Wang, Jun, and Yu, Yong. Seqgan: sequence generative adversarial nets with policy gradient. In Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), 2017.

https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146

https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html