Lecture 9: GAN1

Ki Pyo	완료
선택	GAN

ReCap

- Model families
  - Autoregressive Models: $p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i|\mathbf{x}_{<i})$
  - Variational Autoencoders: $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})d\mathbf{z}$
  - Normalizing Flow Models: $p_X(\mathbf{x};\theta) = p_Z\left(\mathbf{f}_\theta^{-1}(\mathbf{x})\right)\left|\det\left(\frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right)\right|$
- All the above families are trained by minimizing KL divergence $D_{KL}(p_{\text{data}}||p_\theta)$, or equivalently maximizing likelihoods (or approximations)

$$\hat{\theta} = \underset{\theta}{\mathrm{argmax}} \sum_{i=1}^{M} \log p_\theta(\mathbf{x}_i), \quad \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M \sim p_{\text{data}}(\mathbf{x})$$

- 지금까지 배워왔던 모델들 (VAE, Autoregressive Models, NF etc) real probability distribution과 model에 의해 define된 parameterized distribution사이의 “similarity metric”인 d(pdata, ptheta)을 KL divergence로 두고 접근한 방법들이며 이는 MLE로 이어진다.
- 이러한 학습 방식은 정의된 모델에 대해 각 데이터 포인트의 likelhood를 구할 수 있어야한다.

Limitations of Likelihood based learning

- Case 1: Test 데이터 샘플 X에 대해 실제 모집단에서의 likelihood와 비슷한 값을 가지나(likelihood를 높게 줌), sampling quality(eg. 실제 모집단에서 나올 법한 이미지)가 떨어지는 경우

Case 2: Great test log-likelihoods, poor samples. E.g., For a discrete noise mixture model $p_\theta(\mathbf{x}) = 0.01p_{\text{data}}(\mathbf{x}) + 0.99p_{\text{noise}}(\mathbf{x})$

- Taking logs, we get a lower bound

$$\log p_\theta(\mathbf{x}) = \log[0.01p_{\text{data}}(\mathbf{x}) + 0.99p_{\text{noise}}(\mathbf{x})]$$
$$\geq \log 0.01p_{\text{data}}(\mathbf{x}) = \log p_{\text{data}}(\mathbf{x}) - \log 100$$

- For expected log-likelihoods, we know that
  - Lower bound

$$E_{p_{\text{data}}}[\log p_\theta(\mathbf{x})] \geq E_{p_{\text{data}}}[\log p_{\text{data}}(\mathbf{x})] - \log 100$$

  - Upper bound (via non-negativity of $D_{KL}(p_{\text{data}}||p_\theta) \geq 0$)

$$E_{p_{\text{data}}}[\log p_{\text{data}}(\mathbf{x}))] \geq E_{p_{\text{data}}}[\log p_\theta(\mathbf{x})]$$

- As we increase the dimension n of $\mathbf{x} = (x_1, \cdots, x_n)$, absolute value of $\log p_{\text{data}}(\mathbf{x}) = \sum_{i=1}^{n} \log p_\theta(x_i|\mathbf{x}_{<i})$ increases proportionally to n but $\log 100$ remains constant. Hence, likelihoods are great $E_{p_{\text{data}}}[\log p_\theta(\mathbf{x})] \approx E_{p_{\text{data}}}[\log p_{\text{data}}(\mathbf{x})]$ in very high dimensions

⇒ 원래 모집단 데이터와 완전히 벗어나는 noise를 뽑아낼 확률이 큰 모델 분포 (low sampling quality)

⇒ 차원이 커질 수록 모델의 test data X에 대한 likelihood와 P data의 likelihood가 비슷해진다.

- Case 2: Overfitting (Good Sampling, Low Quality)
    - training data를 기준으로만 likelihood를 최대화하는 방향으로 학습, training data를 그대로 외워서 sampling 시에 뽑아냄
    - 그럴 듯한 Sampling을 하여 Quality가 좋지만, 같은 P data에서 나왔지만 훈련 시에는 보지 못한 test data에 대한 likelihood는 낮음
    - generalization 성능이 떨어짐

Summary - Limitations of Likelihood Based Training

지금까지는 모집단과 모델 분포의 similarity metric을 KL항으로 두었음(likelihood maximize로 이어짐). 이렇게 likelihood를 기반으로 한 훈련방식은

1. 샘플 퀄리티의 저하
2. Overfitting 문제 (샘플 퀄리티는 훈련데이터를 추출하여서 좋으나 일반화 능력이 부족)

와 같은 두가지 한계점이 존재

이제 실제 모집단과 모델 분포의 similarity metric을 바꾸어 likelihood free한 model training 방식을 취할 것임

Likelihood Free Learning : Use different D(p data || p theta)

Given $S_1 = \{\mathbf{x} \sim P\}$ and $S_2 = \{\mathbf{x} \sim Q\}$, a **two-sample test** considers the following hypotheses

- Null hypothesis H_0: $P = Q$
- Alternative hypothesis H_1: $P \neq Q$

- Two-Sample test : 두 종류의 샘플 집합이 같은 distribution으로부터 sampling 되었는지를 판단하는 과정
- 이때의 귀무가설은 P=Q이고 대립가설은 P/=Q

Test statistic T compares S_1 and S_2. For example: difference in means, variances of the two sets of samples

- $T(S_1, S_2) = \left| \frac{1}{|S_1|} \sum_{x \in S_1} x - \frac{1}{|S_2|} \sum_{x \in S_2} x \right|$

If T is larger than a threshold α, then reject H_0 otherwise we say H_0 is consistent with observation.

- 귀무가설을 받아들일지 아닌지는 test statistics(eg. difference in two sample means)의 크기를 통해 결정
- Likelihood-Free : 이러한 방식은 Sample 들로부터 통계량을 뽑아내는 방식이기에 likelihood 계산이 필요 없음

GAN in aspect of Two-sample test?

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

- 각각의 네트워크가 각각의 가설을 지지하며 가설 검정에 이용되는 test statistic값(V(G, D))을 각각 최대/최소화 하려고 경쟁 ⇒ Min/Max Game
- 보통 test statistic값은 샘플 평균의 차이와 같이 지정하는 경우도 있지만 high dimension data에 대해서는 이런 좋은 test statistic 값에 대한 공식?을 명확히 지정하는 것이 어려움
- 그래서 Data로부터 Discriminator에서 나오는 값을 통해 test statistics 값을 구성함. Discriminator는 binary classifier neural network인데 실제 모집단에서 나왔는지 아니면 모델 분포에서 나왔는지를 구별하는 네트워크. 이 Discriminator의 Classification Loss에 음수를 부여한 값이 Test-Statistics

Discriminator

$$V(p_\theta, D_\phi) = \quad E_{\mathbf{x}\sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + E_{\mathbf{x}\sim p_\theta}[\log(1 - D_\phi(\mathbf{x}))]$$

Training objective for discriminator:

$$\max_{D_\phi} V(p_\theta, D_\phi) = \quad E_{\mathbf{x}\sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + E_{\mathbf{x}\sim p_\theta}[\log(1 - D_\phi(\mathbf{x}))]$$

$$\approx \quad \sum_{\mathbf{x}\in S_1} \log D_\phi(\mathbf{x}) + \sum_{\mathbf{x}\in S_2} [\log(1 - D_\phi(\mathbf{x}))]$$

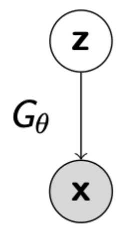
- Discriminator는 실제 모집단의 데이터와 모델 분포의 데이터를 구분하는 역할이기에 Test-Statistics를 최대화 하여 alternative hypothesis를 지지. 즉 binary classification loss를 최소화하도록, 실제 데이터에 대해서는 1의 확률을 부여하고 fake data에 대해서는 0의 확률을 부여하도록 학습을 진행.

  Optimal한 Discriminator의 상태

$$D_\theta^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_\theta(\mathbf{x})}$$

  → 만약 generator의 분포가 실제 모집단의 분포와 같다면 Optimal한 discriminator의 최선의 선택은 1/2의 확률을 부여하는 것

Generator



- **Generator**
  - Directed, latent variable model with a deterministic mapping between **z** and **x** given by G_θ z -> x로 deterministic mapping을 하는 neural network
    - Sample $\mathbf{z} \sim p(\mathbf{z})$, where $p(\mathbf{z})$ is a simple prior, e.g. Gaussian
    - Set $\mathbf{x} = G_\theta(\mathbf{z})$

- prior z에서 sampling한 z를 deterministic하게 x로 mapping하는 network

  Deterministic vs Stochastic

- 반면 Generator의 목표는 실제 모집단의 데이터와 가장 비슷하도록 하는 분포를 만들어내는 것이기에 two-test statistics를 최소화하는 것이 목적, binary classification loss를 최대화 하도록 학습을 진행 ⇒ Null hypothesis 지지

Training Objective of GAN

Training objective for generator:

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

For the optimal discriminator $D_G^*(\cdot)$, we have

$$\begin{aligned} & V(G, D_G^*(\mathbf{x})) \\ &= E_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}\right] + E_{\mathbf{x}\sim p_G}\left[\log \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}\right] \\ &= E_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}{2}}\right] + E_{\mathbf{x}\sim p_G}\left[\log \frac{p_G(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}{2}}\right] - \log 4 \\ &= \underbrace{D_{KL}\left[p_{\text{data}}, \frac{p_{\text{data}}+p_G}{2}\right] + D_{KL}\left[p_G, \frac{p_{\text{data}}+p_G}{2}\right]}_{2\times\text{Jensen-Shannon Divergence (JSD)}} - \log 4 \\ &= 2D_{JSD}[p_{\text{data}}, p_G] - \log 4 \end{aligned}$$

- 앞서 언급한 min-max 게임을 진행
- 먼저 discriminator를 학습 ⇒ maximize test-statistic
- 그후 generator를 학습 ⇒ minimize test-statistic

이론적으로 discriminator를 학습시켜 discriminator가 optimal한 상태에 도달했다면 아래와 같고

$$D_\theta^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_\theta(\mathbf{x})}$$

이때 generator가 최소화해야할 test-statistic은 아래와 같음

For the optimal discriminator $D_G^*(\cdot)$, we have

$$\begin{aligned} & V(G, D_G^*(\mathbf{x})) \\ &= E_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}\right] + E_{\mathbf{x}\sim p_G}\left[\log \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}\right] \\ &= E_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}{2}}\right] + E_{\mathbf{x}\sim p_G}\left[\log \frac{p_G(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}{2}}\right] - \log 4 \\ &= \underbrace{D_{KL}\left[p_{\text{data}}, \frac{p_{\text{data}}+p_G}{2}\right] + D_{KL}\left[p_G, \frac{p_{\text{data}}+p_G}{2}\right]}_{2\times\text{Jensen-Shannon Divergence (JSD)}} - \log 4 \\ &= 2D_{JSD}[p_{\text{data}}, p_G] - \log 4 \end{aligned}$$

이는 실제 분포와 generator 분포 사이의 distribution similarity metric값을 나타냄(Scaled-Shifted JSD). 그리고 generator는 이를 최소화 하도록 업데이트한다면 P_G는 실제 모집단과 비슷해질 것임!

*JSD항의 성질

- Also called as the symmetric KL divergence

$$D_{JSD}[p, q] = \frac{1}{2}\left(D_{KL}\left[p, \frac{p+q}{2}\right] + D_{KL}\left[q, \frac{p+q}{2}\right]\right)$$

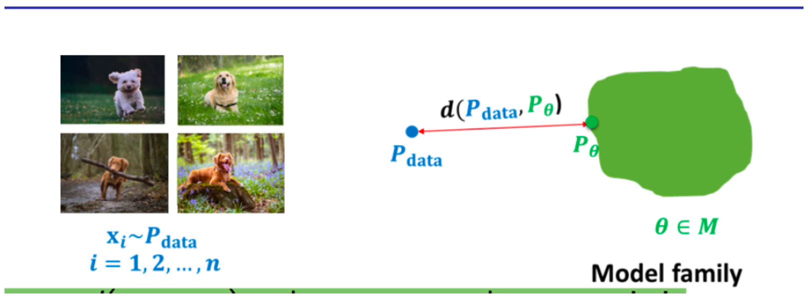
- Properties
  - $D_{JSD}[p, q] \geq 0$
  - $D_{JSD}[p, q] = 0$ iff $p = q$
  - $D_{JSD}[p, q] = D_{JSD}[q, p]$

- KL과 같이 P=Q일 때, 0의 값
- Generator는 이를 최소화 하는 방향으로 학습 될 것이니 P_Data = P_G에 가까워 질 것임
- Optimal한 Discriminator와 Generator가 모두 완성되었다면 test-statistics(**minimax objective function**, Value Function)는 아래와 같음

For the optimal discriminator $D^*_{G^*}(\cdot)$ and generator $G^*(\cdot)$, we have

$$V(G^*, D^*_{G^*}(\mathbf{x})) = -\log 4$$

Summary of GAN - Likelihood-Free Learning



- Distribution Similarity Metric을 Two-Sample test statistic으로 둠 (작을 수록 두 분포가 같다는 가설 입증)
- 이 Two-Sample Test Statistic는 Discriminator에 의한 -(binary classification loss)값
- D와 G는 서로 다른 가설을 지지하여 Two-Sample Test Statistic를 각각 최대, 최소화하려는 방향 순차적으로 학습 ⇒ Min-Max Game
- 서로 목적이 다르기에 적대적 학습이라고도 함.
- 이러한 학습 방식은 관측 변수에 대한 likelihood 계산을 요하지 않음, 샘플 기반 분류 진행 및 분류 손실 계산하면됨 ⇒ Likelihood-Free Training
- 1) Discriminator가 optimal할 때 Generator에게 좋은 학습 metric(두 분포 이만큼 멀어!라는 정확한 척도)을 제공 ⇒ Scaled-Shifted JSD
- 2)이를 최소화하는 방향으로 Generator를 학습시키고
- 1), 2)의 과정을 반복하면 generator는 P_data에 가까운 분포를 가지게됨

⇒ GAN은 두 분포가 같은지 다른지를 검정하는 이표본검정(Two-Sample Test) 문제로 볼 수 있으며, likelihood 계산 없이도 Discriminator의 classification loss를 통해 Generator를 학습시킬 수 있는 구조입니다.

Training Algorithm of GAN

$$\min_\theta \max_\phi V(G_\theta, D_\phi) = E_{\mathbf{x}\sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + E_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

- Sample minibatch of m training points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$ from $\mathcal{D}$
- Sample minibatch of m noise vectors $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(m)}$ from p_z
- Update the discriminator parameters ϕ by stochastic gradient **ascent**

$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m}\nabla_\phi \sum_{i=1}^{m}[\log D_\phi(\mathbf{x}^{(i)}) + \log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))]$$

- Update the generator parameters θ by stochastic gradient **descent**

$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m}\nabla_\theta \sum_{i=1}^{m}\log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))$$

- Repeat for fixed number of epochs

```
# Hyperparameters
num_epochs = ...
```

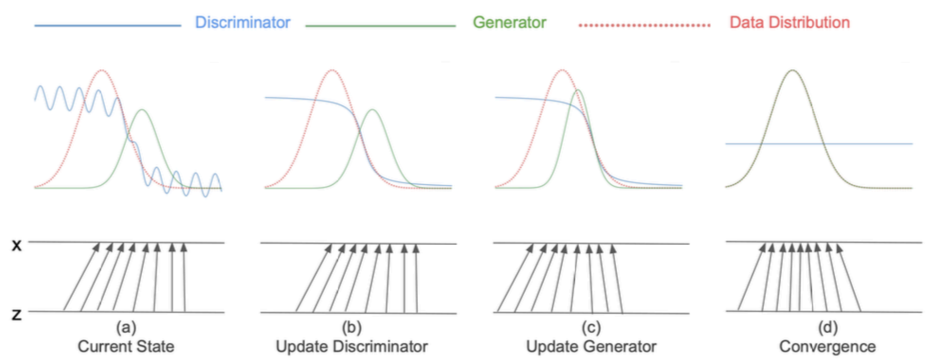
```
batch_size = ...
lr = ...
k = 1   # number of D updates per G update

Initialize θ_d (Discriminator parameters)
Initialize θ_g (Generator parameters)

for epoch in range(num_epochs):
    for each minibatch of real data {x1, ..., xm}:

        # 1. Train Discriminator k times
        for i in range(k):
            # Sample noise {z1, ..., zm} ~ p(z) (e.g. Gaussian)
            z = sample_noise(batch_size)

            # Generate fake samples
            x_fake = G(z; θ_g)

            # Compute loss for D
            L_D = - [ log D(x_real; θ_d) + log(1 - D(x_fake; θ_d)) ]

            # Update Discriminator parameters
            θ_d ← θ_d - lr * ∇_θd L_D

        # 2. Train Generator once
        # Sample new noise
        z = sample_noise(batch_size)

        # Generate fake samples
        x_fake = G(z; θ_g)

        # Compute loss for G (want D(x_fake) → 1)
        L_G = - log D(x_fake; θ_d)

        # Update Generator parameters
        θ_g ← θ_g - lr * ∇_θg L_G
```

→ 매 스탭마다 Discriminator에 대해서는 K번 Update하는 경우도 있음

→ 실제 구현에서는 D, G가 모두 각자의 loss를 최소화하는 방식으로 구현됨

Training Procedure

$$\min_{\theta} \max_{\phi} V(G_\theta, D_\phi) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + E_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$



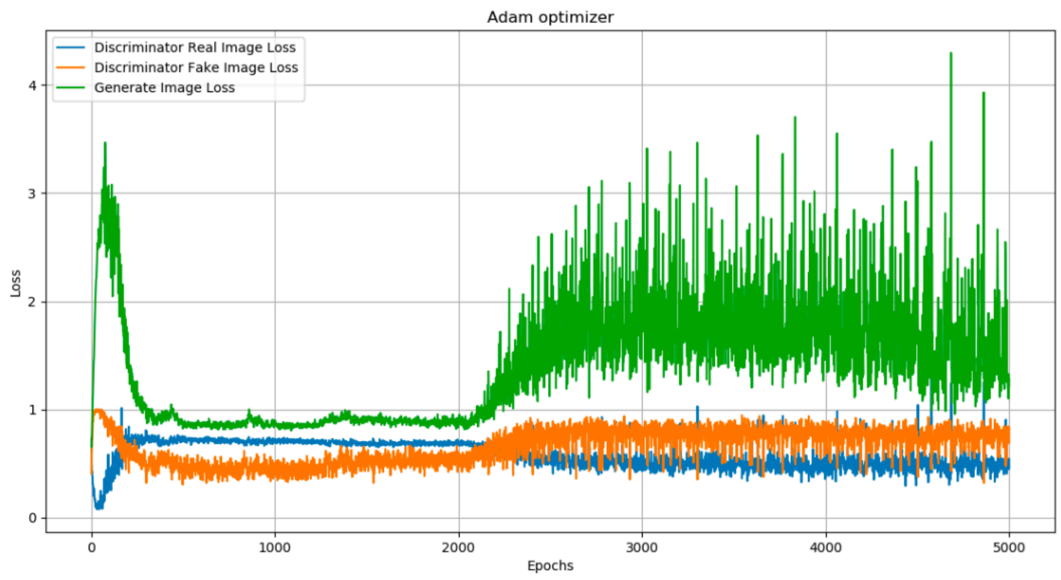
1. a → b : Discriminator가 P_data와 P_generator의 데이터를 잘 구분하도록 업데이트 됨

2. b → c : Discriminator가 업데이트됨으로써 생기는 좋은 P_Data, P_G의 분포 차이 metric을 기반으로 그 차이가 최소가 되도록 Generator를 업데이트

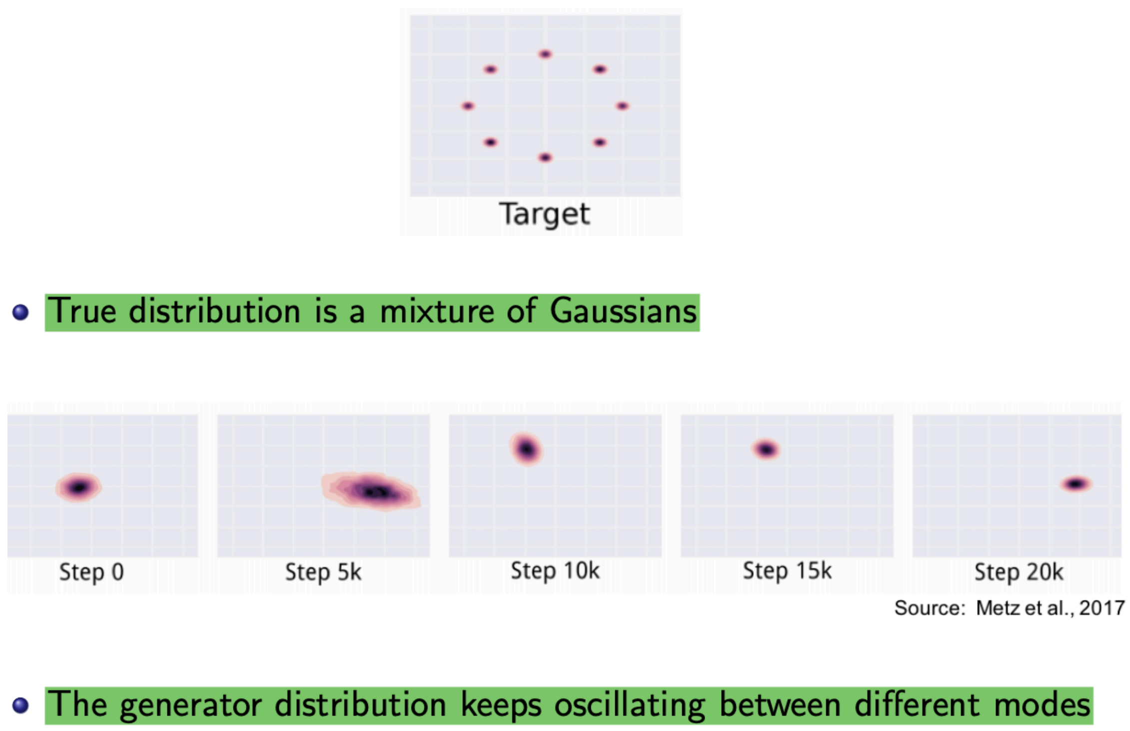
3. c → d : 1, 2의 과정을 반복하여 D와 G가 Optimal한 상태에 가까워지면 P_D = P_G에 가까워지고 Optimal한 D는 어떤 data이던 1/2(P_data일 확률이 1/2입니다)의 값을 내놓게됨.

Limits of GAN

- Unstable하여 최적점 도달의 어려움 : generator와 discriminator의 loss가 oscillation, 매 스텝마다 discriminator 최적점에 도달하여 optimal한 상태에 대한 보장이 없음 → generator에 좋은 학습 신호(분포 간의 정확한 차이 metric)를 보내준다는 보장이 없음



- Mode Collapse : generator가 한가지 유형의 데이터(Mode)만 생성하도록 유도됨



예) MNIST 숫자 0~9 중에서 0만 반복 생성하거나, 특정 스타일 이미지만 생성