GAN

논문리뷰)

Brief Summary of GAN)

Generative Adversarial Nets

Goodfellow, I. et al., 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pp. 2672–2680.

GAN을 이름 그대로 해석하자면, '적대적 생성 모델' 이다.

Generative model, 즉 ‘그럴듯한 것들을 만드는 것’은, 그야말로 답이 없는 문제를 푸는 것과 같다.

대표적인 예시로, Boltzmann machine, AutoEncoder, 그리고 GAN이 있다.

Generative Model, in GAN:

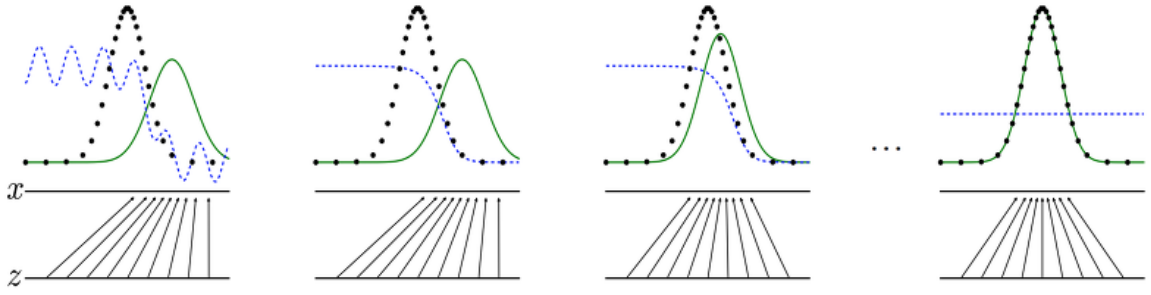
기존의 ML model: Class에 대한 예측 + Continous random variable에 대한 regression

⇒ 이는, softmax 등으로 어떠한 class 등에 대한 ‘확률’, 또는 likelihood를 찾아낸다.

GAN에서 생성하는 것은, ‘데이터의 형태’를 만들어낸다.

데이터의 형태라는 것은, 분포 혹은 분산을 의미하는데, 여기서 이러한 데이터의 분포를 만들어 낸다는 것은,

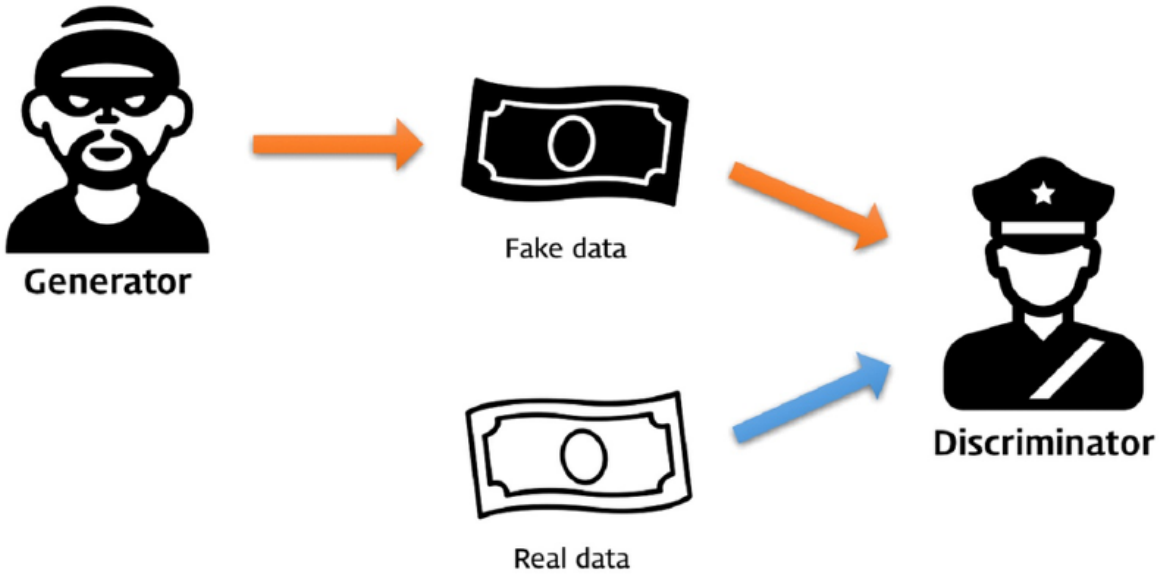
‘실제적인 형태’를 갖춘 데이터를 만들어낸다는 것이다.



명도와 채도와는 상관이 없이, 픽셀들의 분포에 따라 그림이 어떠한 것을 나타내는지를 인식한다.

따라서, 어떠한 '형태'를 이루는 픽셀들의 값 자체 뿐만 아니라, 즉, 데이터의 분포 또는 분산 자체를 만들어내는 것이라고 해석할 수 있다.

그렇가면, 'Adverserial'란 무엇인가?



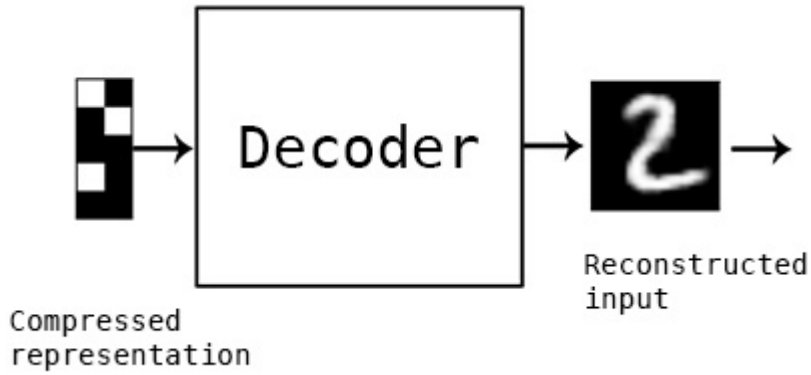
흔히 이러한 '적대적'이라는 개념을 설명하기 위하여 가장 많이 사용되는 지폐위조범-경찰의 그림이다.

지폐위조범은 돈을 벌기 위해, 마치 ‘진짜 지폐처럼 보이는’ 위조 지폐를 만들어낸다 (Generate)

반면, 이 위조지폐가 시장에 떠돌아 다니는 것을 막기 위해,경찰은 진짜 지폐를 가지고 계속 ‘학습’을 진행한다. 진짜 지폐의 분포, 특징, 형태 등을 학습하고, 위조지폐를 감별해내려 노력한다.

한편 지폐위조범은 처음에는 조폐 과정에 대해 익숙하지 않기 때문에, 일단은 생각을 ‘이렇게 하면은 조금 비슷하지 않을까?’ 라는 생각을 한다.

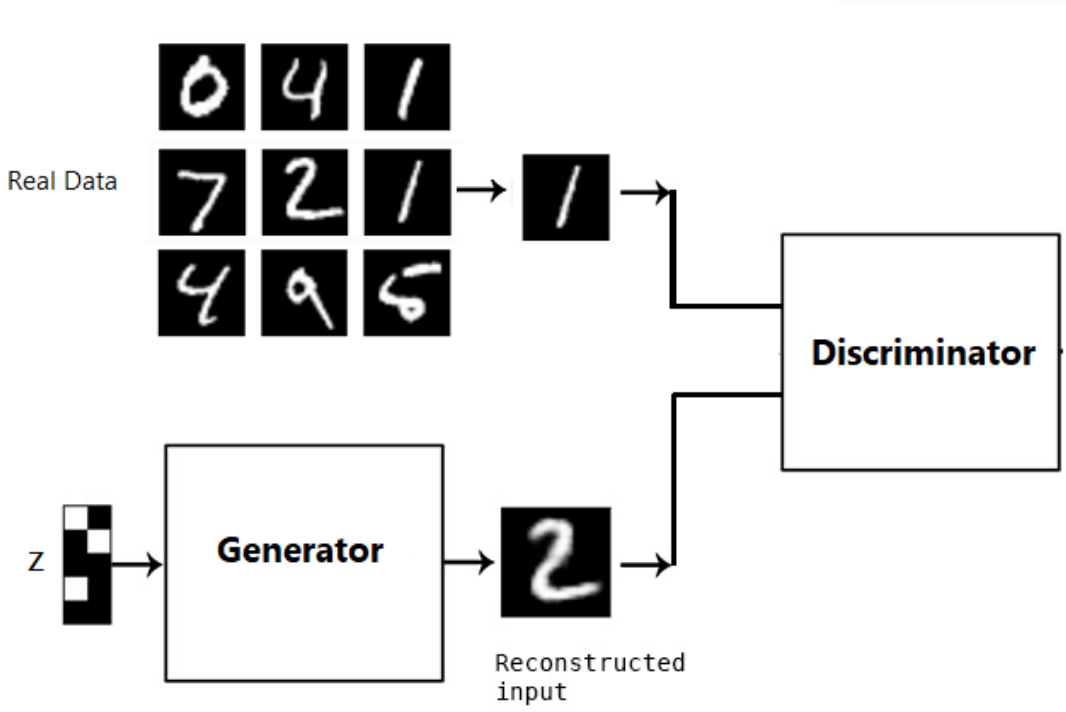
지폐를 감별하는데 있어서 무수히 많은 특징들(실제 분포)이 있는데, 지폐위조범은 이러한 특징들을 처음에는 잘 모르기 때문에 처음에는 계속 체포당한다. 하지만, 옥살이를 하면서, 복기를 하며 조금은 더 나은 지폐를 만들기 위해 노력한다.



GAN의 기본 구조는, AutoEncoder의 뒷부분을 떼어놓은 형태라고 생각해도 좋다.

즉, 어떤 임의의 표현 z로부터, 이미지를 생성 할 수 있는 decoder(Generator)로 이미지를 생성한다.

이 구조만으로는, 원래 데이터의 분포를 알 수 없기 때문에, Discriminator의 개념이 도입된다.



Discriminator는 Classifying task를 수행한다. 실제 데이터로부터 데이터를 뽑고, discriminator은 generator가 ‘생성’한 데이터를 뽑은 후 두 종류의 데이터를 섞어준다. 이 때 discriminator는 주어진 데이터가 실제 데이터인지, 우리가 생성한 데이터인지를 구별한다.

여기서 discriminator와 generator은 objective가 정확히 반대이다. 즉, discriminator은 generator가 ‘생성’한 이미지를 원래 데이터와 최대한 확실히 구분하는 것이 objective인 반면,

generator은 원래 데이터와 비교해도 손색이 없을 정도의 데이터를 ‘생성’하는 것이 objective인 셈이다.

최종적으로는, discrimintator는 어느 순간부터는 구별을 잘 못하게 될 것이고, 이말인 즉슨 generator은 어느 순간부터 실제 데이터와 비교했을 때 손색없는 데이터를 ‘생성’할 수 있게 되었다는 것이다.

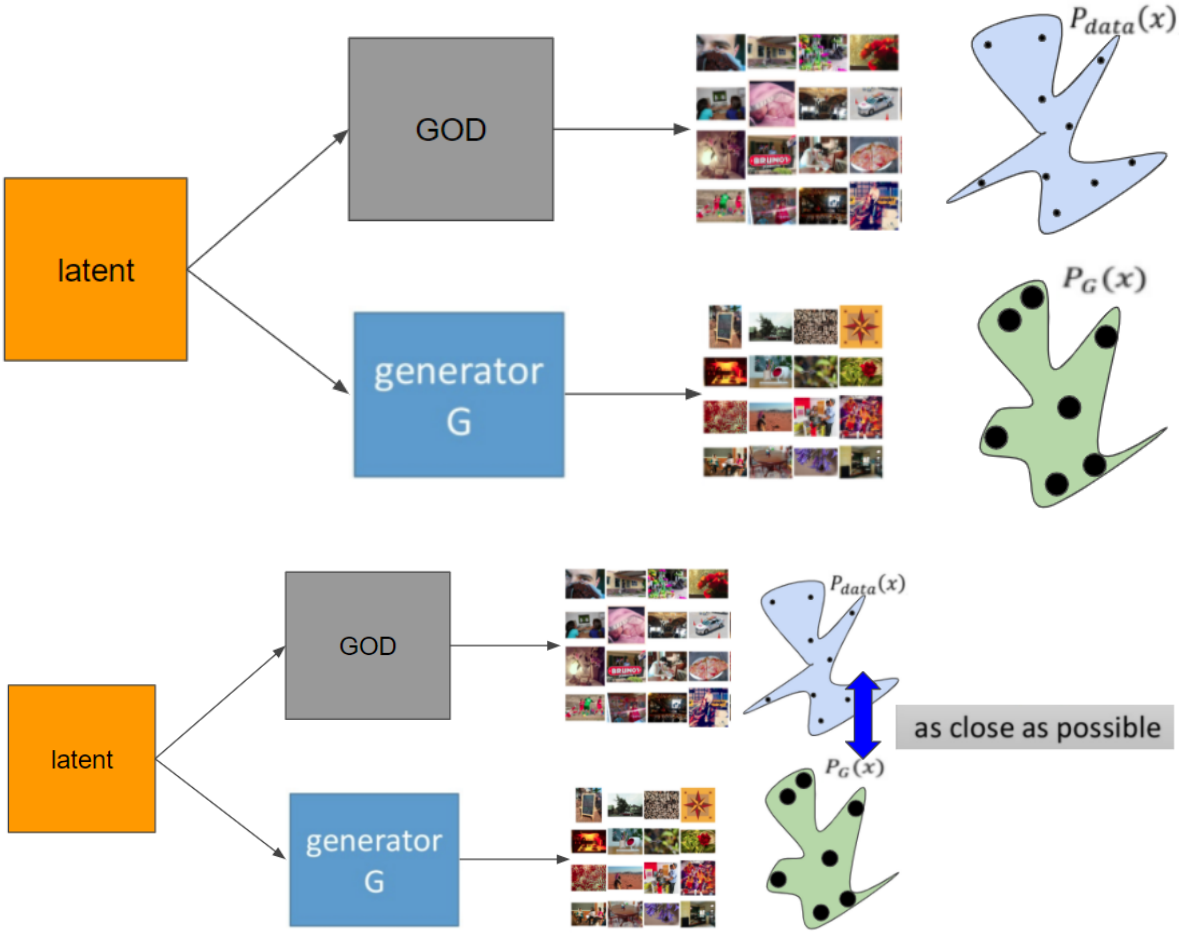
Generator

Latent Space

어떠한 데이터의 '근간', 즉 잠재적인 뜻을 한데 모여 둔 latent space가 있다고 했을 때, 해당 latent space에 있는 정보들이 모여 현재 데이터가 생성(여기서는 generator가 생성한 데이터가 아님)되었다고 가정해보자.

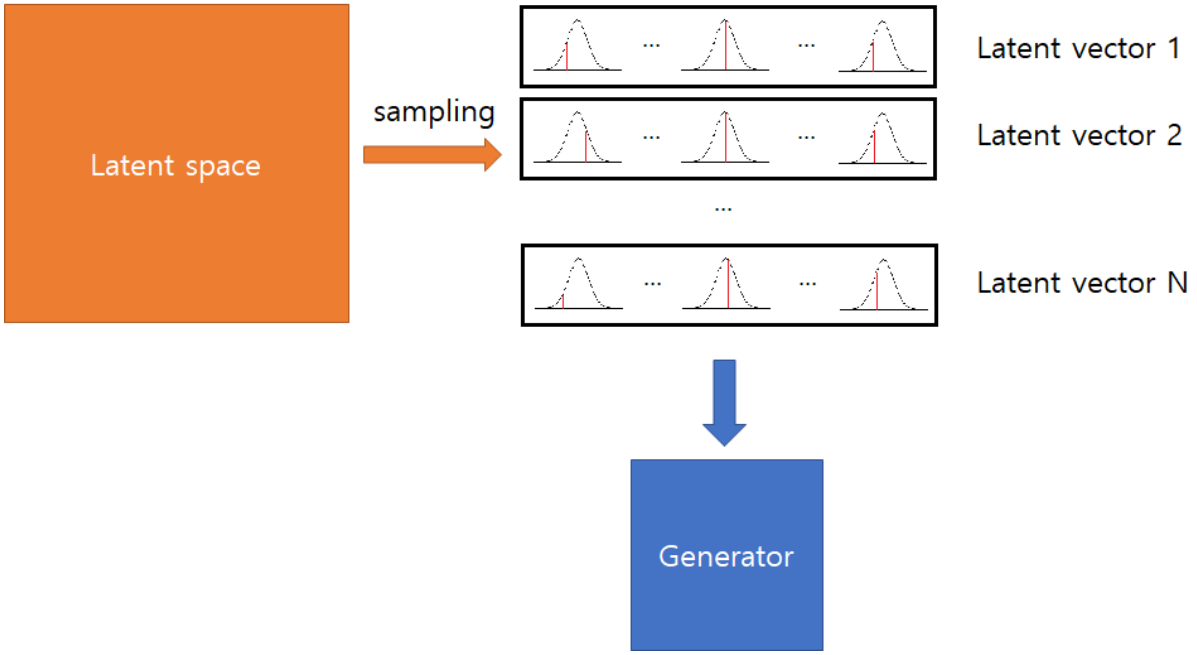
세상에 있는 데이터는 모두 어떠한 latent space에 있는 vector들이 변환되어 만들어진 '결과물'이라고 할 수 있을 것이다.

그렇다면, 이러한 latent space가 '어떻게' 데이터를 만들었는지에 대한 '규칙'을 발견하기 위한 방법으로 Sampling 기법이 존재한다.



즉, latent space에 있는 vector들로서 최대한 세상에 있는 결과물들과 비슷하게 output을 생성하도록 하는 것이다.

GAN에서 sampling을 수행하는 경우, latent space에서의 Normal Distribution으로부터 vector을 몇개 뽑아와 generator의 input으로 주어진다.



Generator은 이 latent space로부터 생성된 latent vector을 input으로 주면, 어떠한 이미지를 생성하도록 학습한다. 즉 어떠한 label이 없는 상황에서의 비지도학습인 셈이다.

초기 latent space은 거의 무의미한 공간이며, 랜덤성을 띈다 . 하지만 학습을 진행하다 보면, generator가 latent space에서 sampling 한 latent vector와 특정 이미지를 mapping하기 시작한다.

이렇게 되면 generator에게 latent space에서 latent vector가 input되었을 때, 해당 데이터를 가지고 training 데이터가 되도록 학습을 진행한다.

즉, 데이터가 흩뿌려져 있는 걸 보고 이 데이터가 특정 training data가 되도록 스스로 변화하며 학습을 진행한다는 것이다.

Discriminator

Discriminator의 주요한 목적은 두개의 분포를 명확히 구분하는 것이다. 이 때 두개의 분포는, 실제 데이터이냐(real-world data), 아니면 생성된 데이터이냐(generated data)이냐를 의미한다.

GAN에서의 loss function은 기존의 Binary Cross Entropy를 다소 수정한 형태를 사용하는데,

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

간략하게 말해 Cross Entropy를 활용해 두 분포의 차이를 구하고, 분포의 차이를 바탕으로 generator을 학습시키는 흐름이다.

Loss에서 Generator의 역할은, Discriminator가 구별할 수 없을 정도로 데이터를 잘 만드는 것입니다. 그리고 Discriminator의 역할은 Generator가 만든 것을 완벽히 가짜라고 구별해 내는 것이다.

허나 이 두가지 함수를 한꺼번에 학습시키기 보다는, 하나를 학습시킬 때는 다른 하나를 고정을 시키고, vice versa의 경우에도 마찬가지이다.

Papers with CODES

```
D = Discriminator(image_size, hidden_size).to(device)
G = Generator(latent_size, hidden_size, image_size).to(device)

criterion = nn.BCELoss()
d_optimizer = torch.optim.Adam(D.parameters(), lr=0.0002)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.0002)
```

(번갈아 가면서 학습 진행)

Fix G, Update D

→ G가 답지를 낸다. 즉 G는 상수처리된다.

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

따라서 loss function 또한 위와 같이 변화한다.

```
# ---- D ----
z = rand_z()
#정규분포에서 샘플링

# real
real_outputs = D(images)

#images는 실제 데이터셋으로부터 뽑은 이미지들
#실제 데이터셋에서 뽑아서 D가 실제 데이터가 실제 데이터라고 (1이라고) 판단했는지 검사
#real_outputs에는 모두 1로된 데이터 -> Ground Truth!

d_loss_real = binary_cross_entropy(real_outputs, real_label)
real_score = real_outputs

# fake
fake_images = G(z)
fake_outputs = D(fake_images)
d_loss_fake = binary_cross_entropy(fake_outputs, fake_label)
fake_score = fake_outputs

#latent space에서 latent vector z를 뽑고 G에게 넘겨줍니다.
#그럼 G는 fake images들을 생성할 것이고, 이를 D에 넘겨주죠. 수식으로 보자면 D(G(Z))
# fake_outputs에는 모두 0로된 데이터

# loss
d_loss = d_loss_real + d_loss_fake

# backprop
reset_grad()
d_loss.backward()
d_optimizer.step()
#전체 Loss는 fake일때와 real일때의 loss값을 d_loss에 더해주고 backward를 수행.

#d_optimizer.step()-> discriminator의 파라미터만 업데이트.
```

Fix D, Update G

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

D가 상수취급된다. 따라서 loss function 또한 위와같이 바뀌게 된다.

```
# ---- G ----
z = rand_z()
fake_images = G(z)
outputs = D(fake_images)
g_loss = binary_cross_entropy(outputs, real_label)

reset_grad()
g_loss.backward()
g_optimizer.step()
```

마찬가지로, g_optimizer를 통해 generator의 gradient만 update.

GAN의 동작 조건

GAN의 목표는 실제 데이터 분포인 P_data 와 Generator 가 만든 가짜 데이터 분포인 P_generator

가 같아지는 것을 목표로 한다.

즉,

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

loss function이 최적일 때, P_data = P_generator을 만족해야 한다.

일단은 최적의 'Generator'을 도출하는 것에 대해 생각을 해 보면,

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

으로서 loss function을 set.

$$\mathbb{E}_{x\sim p}[f] = \int p(x)f(x)dx$$

기댓값 산출의 식은 위와 같다.

이 때,

x: 이미지(high dimensional vector)

G: generator

D: Discriminator

P_data: 실제 데이터셋의 분포

P_g: generator가 생성한 가짜 데이터의 분포

E: 기댓값. 기댓값은 p(x)의 확률로 나타나는 어떤 함수 f(x)의 평균.

현재 loss function을 recall하면,

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

이다.

현재 loss function의 objective는, 그러면 discriminator의 performance에 맞추어져 있다.

따라서, discriminator의 objective를 생각해보았을 때 loss function을 늘려야 한다. 즉, generator가 못 맞추는 정도를 늘려야하는 것이다.

따라서, 현재 이 상황에서는 최적화D*에 대해,

$$D^* = arg\max_D V(D,G)$$

를 구하는 것이고,

$$\mathbb{E}_{x\sim p}[f] = \int p(x)f(x)dx$$

에서,

$$
\begin{aligned}
V &= \mathbb{E}_{x\sim P_{data}}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \\
&= \int_x P_{data}(x)\log D(x)dx + \int_x P_g(x)\log(1-D(x))dx \\
&= \int_x \left[P_{\text{data}}(x)\log D(x) + P_g(x)\log(1-D(x))\right]dx
\end{aligned}
$$

로 전개된다.

이 때 P_data를 a로 치환하고, P_g(x)를 b라 치환하였을 때

$$f(x) = alogx + blog(1-x)$$

즉 위와 같은 함수로서 적분기호 안에 있는 식을 정리할 수 있게 된다.

현재 V를 최대화하기 위해서, 위와 같은 함수는 위로 convex한 형태이며 (0,1)에서 미분이 가능하다.

$$
\begin{aligned}
\frac{f(x)}{dx} = \frac{a}{x} - \frac{b}{1-x} &= 0 \\
a(x-1) + bx &= 0 \\
(b+a)x - a &= 0 \\
\therefore x &= \frac{a}{a+b}
\end{aligned}
$$

즉 x = a/a+b 일 때 최댓값을 가진다는 것이고, 이를 다시 inverse transform 하면

$$optimal\ D^* = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

그렇다면 식은 다음과 같이 전개될 수 있다.

$$\begin{aligned}
&\max_D V(G,D) = V(G,D^*) \\
&= \int_x \left[P_{\text{data}}(x)\log D^*(x) + P_G(x)\log(1-D^*(x))\right] dx \\
&= \int_x [\log(a\frac{a}{a+b}) + \log(b(1-\frac{a}{a+b}))]dx \\
&= \int_x [\log(\frac{1}{2}a\frac{a}{(a+b)/2}) + \log(\frac{1}{2}b\frac{b}{(a+b)/2})]dx \\
&= -2\log 2 + \int_x \log a\frac{a}{(a+b)/2}dx + \int_x \log b\frac{b}{(a+b)/2}dx
\end{aligned}$$

KL Divergence

$$KL(P \parallel Q) = \sum_x P(x) log\frac{P(x)}{Q(x)}$$

KL Divergence란, cross entropy에서 유도될 수 있으며, 두 확률분포의 차이를 계산하는데 사용되는 함수이다.

어떠한 ground-truth 분포에 대해, 해당 분포를 근사하는 분포를 사용해 sampling을 하는 경우 발생할 수 있는 정보 Entropy 차이를 계산하는 것이다.

KL-divergence는 p와 q의 cross entropy에서 p의 엔트로피를 뺀 값으로서,. 결과적으로 두 분포의 차이를 나타낸다.

$$KL(p \parallel q) = \begin{cases} \sum_i p_i \log \frac{p_i}{q_i} \text{ 또는 } -\sum_i p_i \log \frac{q_i}{p_i} & \text{(이산형)} \\ \int p(x) \log \frac{p(x)}{q(x)} dx \text{ 또는 } -\int p(x) \log \frac{q(x)}{p(x)} dx & \text{(연속형)} \end{cases}$$

https://hyunw.kim/blog/2017/10/27/KL_divergence.html

KL Divergence 개념을 차용해 다시 D*를 찾아가는 과정을 식으로서 표현하자면,

$$\begin{aligned} &\max_D V(G,D) = V(G,D^*) \\ &= -2\log 2 + \int_x \log a \frac{a}{(a+b)/2} dx + \int_x \log b \frac{b}{(a+b)/2} dx \\ &= -2\log 2 + \int_x P_{\text{data}}(x) \log \frac{P_{\text{data}}(x)}{(P_{\text{data}}(x) + P_G(x))/2} dx \\ &\quad + \int_x P_G(x) \log \frac{P_G(x)}{(P_{\text{data}}(x) + P_G(x))/2} dx \\ &= -2\log 2 + \mathrm{KL}\left(\mathrm{P_{data}} \| \frac{\mathrm{P_{data}} + \mathrm{P_G}}{2}\right) + \mathrm{KL}\left(\mathrm{P_G} \| \frac{\mathrm{P_{data}} + \mathrm{P_G}}{2}\right) \end{aligned}$$

로서 표현이 될 수 있다.

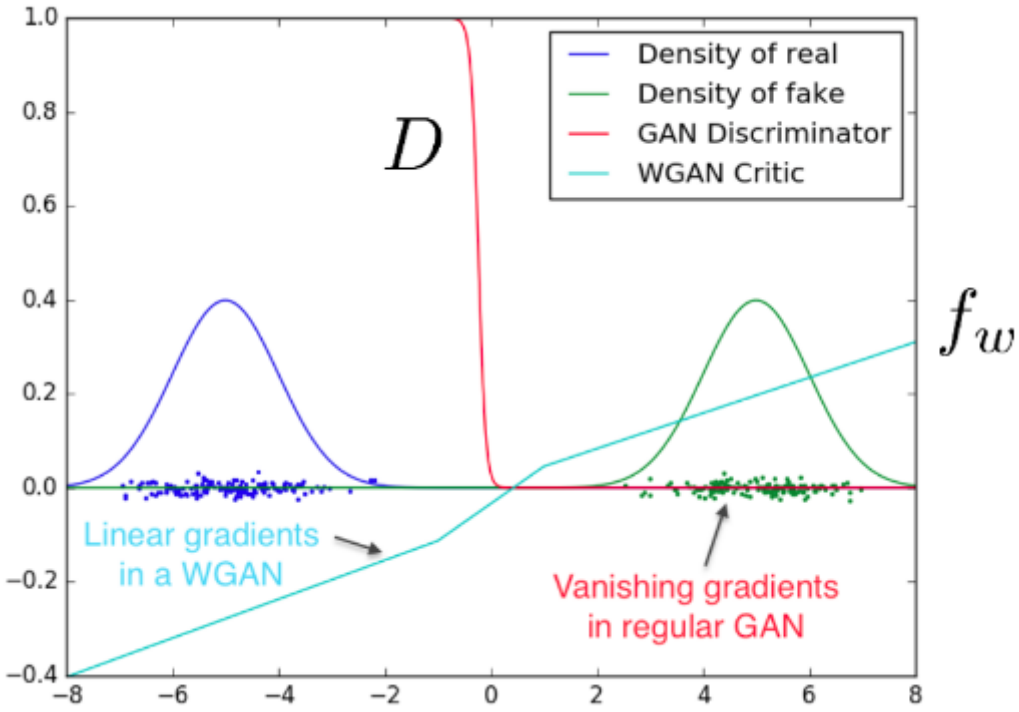
이 때 KL은 assymetric하여 거리개념으로서 사용할 수 없는데, 이 것을 symmetric하게 변형시켜 거리로서 사용할 수 있게 하는 JSD의 개념으로서도 위 loss function을 다시 쓸 수 있다.

$$\begin{aligned} &\max_D V(G,D) = V(G,D^*) \\ &= -2\log 2 + \mathrm{KL}\left(\mathrm{P_{data}} \| \frac{\mathrm{P_{data}} + \mathrm{P_G}}{2}\right) + \mathrm{KL}\left(\mathrm{P_G} \| \frac{\mathrm{P_{data}} + \mathrm{P_G}}{2}\right) \\ &= -2\log 2 + 2JSD(P_{\text{data}} \| P_G) \end{aligned}$$

그리고 JSD는 두 분포사이의 거리로 사용할 수 있기 때문에,

위의 수식을 최소로 만드는 것은 두 분포사이의 거리를 최소로 만드는 것이라는 해석을 할 수 있는 것이다.

허나 JSD의 경우에는 GAN을 '학습'시키는데 큰 문제를 일으킬 수 있다.



oss Function은 가까우면 가깝다고, 멀면 멀다고 명확히 말을 해주어야 이에 따른 gradient를 계산할 수 있다. 하지만 두 분포의 SUPP(지지함수)이 겹치지 않는다면 '두 분포가 완전히 다르다.' 라는 정보만 줄 뿐 어떻게 가깝게 만들지에 대한 정보 즉 gradient를 계산할 수 없다는 뜻이 되기 때문이다.

학습에 있어서 back propagation을 사용하는 만큼, 이러한 vanishing gradinet 문제는 심각한 문제이다.

즉, D가 너무 깐깐하게 두 분포를 판단해서, 만약 두 분포가 겹치지 않는다면 두 분포를 어떻게 가깝게 만들지에 대한 gradient를 계산할 수 없게 된다는 것이다.

이미지 생성과 같은 높은 dimension의 문제를 푸는 GAN에서는 이러한 분포가 겹치지 않는 문제가 더 심하게 발생할 것이기 때문에 GAN의 학습 성능이 떨어지는 등 꽤 심각한 문제로 다가올 것이었습니다.

이를 해결하기 위해,

- 분포를 건드리는 방법
  - Noise를 통한 해결 → Noise를 추가해, 분포를 일부러 겹치게 하여, JSD를 사용해 문제를 해결한다.
  - 허나, 이 방식은 말그대로 ‘noise’를 추가한 것이기 때문에, 선명한 이미지 output을 내는데에는 한계가 있다.
- 분포의 거리 측정 방식을 개선
  - JSD는 두 분포가 겹치지 않을 때 상수가 나와 gradient를 계산하기 힘든 문제가 존재하는데이 근본적인 문제를 해결하기 위해, 분포가 겹치지 않아도 두 분포의 거리를 측정할 수 있는 방법인 Earth Mover Distance를 사용하는 방법이 존재한다.