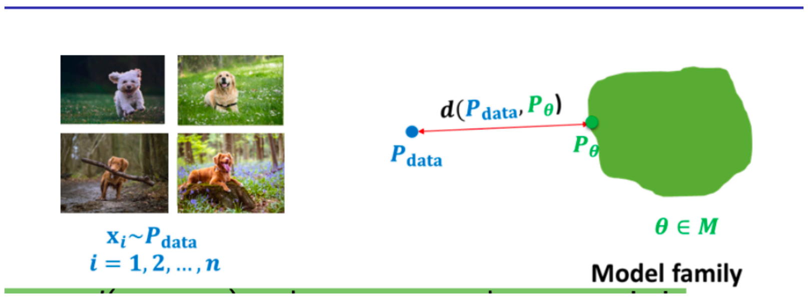
Lecture 10: GAN2

Ki Pyo	완료
선택	GAN



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

- 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

F-Divergence

- 지금까지 p_model들은 p_data와 p_model간의 여러 종류의 두 분포의 간극을 계산하는 척도(여러 종류의 Divergence항)를 최소화하려는 방식으로 업데이트.
- 이러한 Divergence 항이 어떤 학습 지표로 이용됨.

  eg)

  AM, Flow model : KL Divergence

  GAN generator (when D is optimal) : (scaled and shifted) Jensen-Shannon Divergence

- 두 분포의 간극을 계산하는 다른 척도는 없을까? 조금 더 일반화된 방식으로 접근 ⇒ "F-divergence"

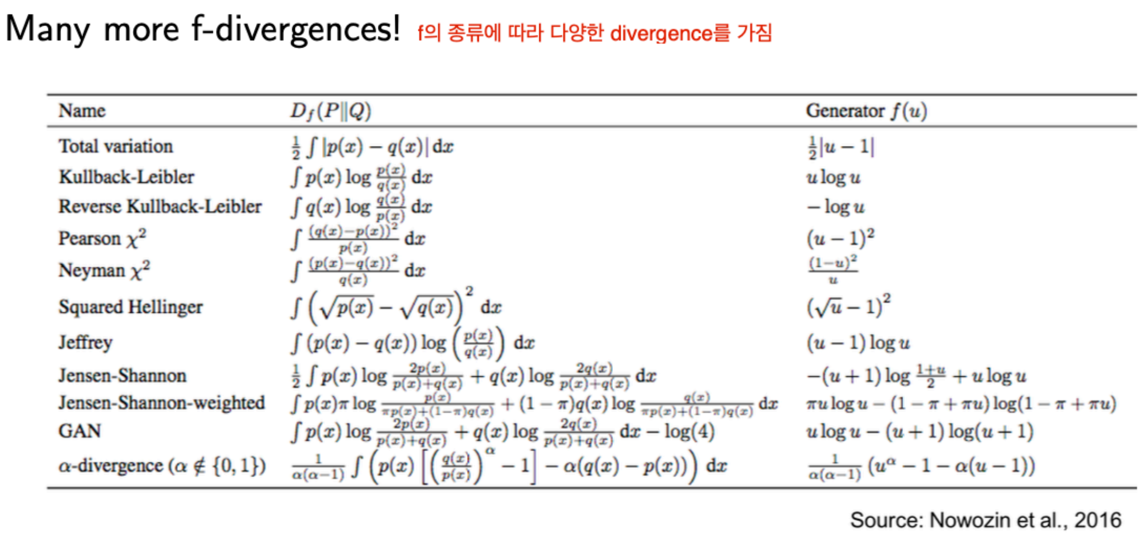
Given two densities p and q, the f-divergence is given by

$$D_f(p, q) = E_{\mathbf{x} \sim q}\left[f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right]$$

- 여기서 f는 다음 3가지를 만족한다면 뭐든 될 수 있음
  - Convex : f Convexity로 인해 f-divergence는 non-negativity 성질을 가짐
  - Lower-Semicontinouos
  - f(1) = 0 : 이 성질로 인해 두 분포가 완전히 같다면 0의 값을 가짐
- 서로 다른 f들은 각각 density ratio가 얼마나 다르냐에 따라 penalize하는 정도가 상이함
- f가 아래와 같다면 f-div는 KL div와 같아짐

Example: KL divergence with $f(u) = u \log u$

F-divergences



Many more f-divergences! f의 종류에 따라 다양한 divergence를 가짐

Name	$D_f(P\|Q)$	Generator $f(u)$
Total variation	$\frac{1}{2}\int \lvert p(x) - q(x)\rvert \,\mathrm{d}x$	$\frac{1}{2}\lvert u-1\rvert$
Kullback-Leibler	$\int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x$	$u\log u$
Reverse Kullback-Leibler	$\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$	$-\log u$
Pearson χ^2	$\int\frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$	$(u-1)^2$
Neyman χ^2	$\int\frac{(p(x)-q(x))^2}{q(x)}\,\mathrm{d}x$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int\left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\mathrm{d}x$	$(\sqrt{u}-1)^2$
Jeffrey	$\int(p(x)-q(x))\log\left(\frac{p(x)}{q(x)}\right)\mathrm{d}x$	$(u-1)\log u$
Jensen-Shannon	$\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$	$-(u+1)\log\frac{1+u}{2}+u\log u$
Jensen-Shannon-weighted	$\int p(x)\pi\log\frac{p(x)}{\pi p(x)+(1-\pi)q(x)}+(1-\pi)q(x)\log\frac{q(x)}{\pi p(x)+(1-\pi)q(x)}\,\mathrm{d}x$	$\pi u\log u-(1-\pi+\pi u)\log(1-\pi+\pi u)$
GAN	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x-\log(4)$	$u\log u-(u+1)\log(u+1)$
α-divergence ($\alpha\notin\{0,1\}$)	$\frac{1}{\alpha(\alpha-1)}\int\left(p(x)\left[\left(\frac{q(x)}{p(x)}\right)^\alpha-1\right]-\alpha(q(x)-p(x))\right)\mathrm{d}x$	$\frac{1}{\alpha(\alpha-1)}\left(u^\alpha-1-\alpha(u-1)\right)$

Source: Nowozin et al., 2016

- f가 무엇이냐에 따라 다양한 종류의 두 분포간의 유사도 척도가 도출됨 ⇒ 일반화된 접근

이렇게 계산된 다양한 척도는 Generator의 학습 objective로 이용될 수 있음 !! 최소화하도록 generator를 훈련시키면 됨.

$$D_f(p_\theta, p_{data}) = \underbrace{E_{\mathbf{x}\sim p_{data}}}_{\text{approx w. samples}}\left[f\underbrace{\left(\frac{p_\theta(\mathbf{x})}{p_{data}(\mathbf{x})}\right)}_{\text{uknown ratio}}\right]$$

$$D_f(p_{data}, p_\theta) = \underbrace{E_{\mathbf{x}\sim p_\theta}}_{\text{approx w. samples}}\left[f\underbrace{\left(\frac{p_{data}(\mathbf{x})}{p_\theta(\mathbf{x})}\right)}_{\text{uknown ratio}}\right]$$

- 학습 척도인 F-divergence 값은 구하는 것부터 쉽지 않음 ㅠㅠ
- 앞의 기댓값은 Monte Carlo로 근사가능.
- 두 확률 분포 간의 ratio를 알기 위해서는 먼저 두 확률 분포에서의 x에 대한 probability density 값을 알아야함.
- p theta는 둘째 치고 p data를 구할 방법이 없음 (모집단을 모르니!)

→ Generator를 훈련시키기 위해 이 f-div를 구할 혹은 근사할 다른 방법이 필요함

→ GAN의 경우에도 D를 optimize하다 보니 Scaled-Shifted Divergence 측도가 나온 것이지 직접 likelihood를 구한 적은 없음!

Fenchel Conjugate For Change of Formulation

$$f^*(t) = \sup_{u\in\mathrm{dom}_f}\left(ut - f(u)\right)$$

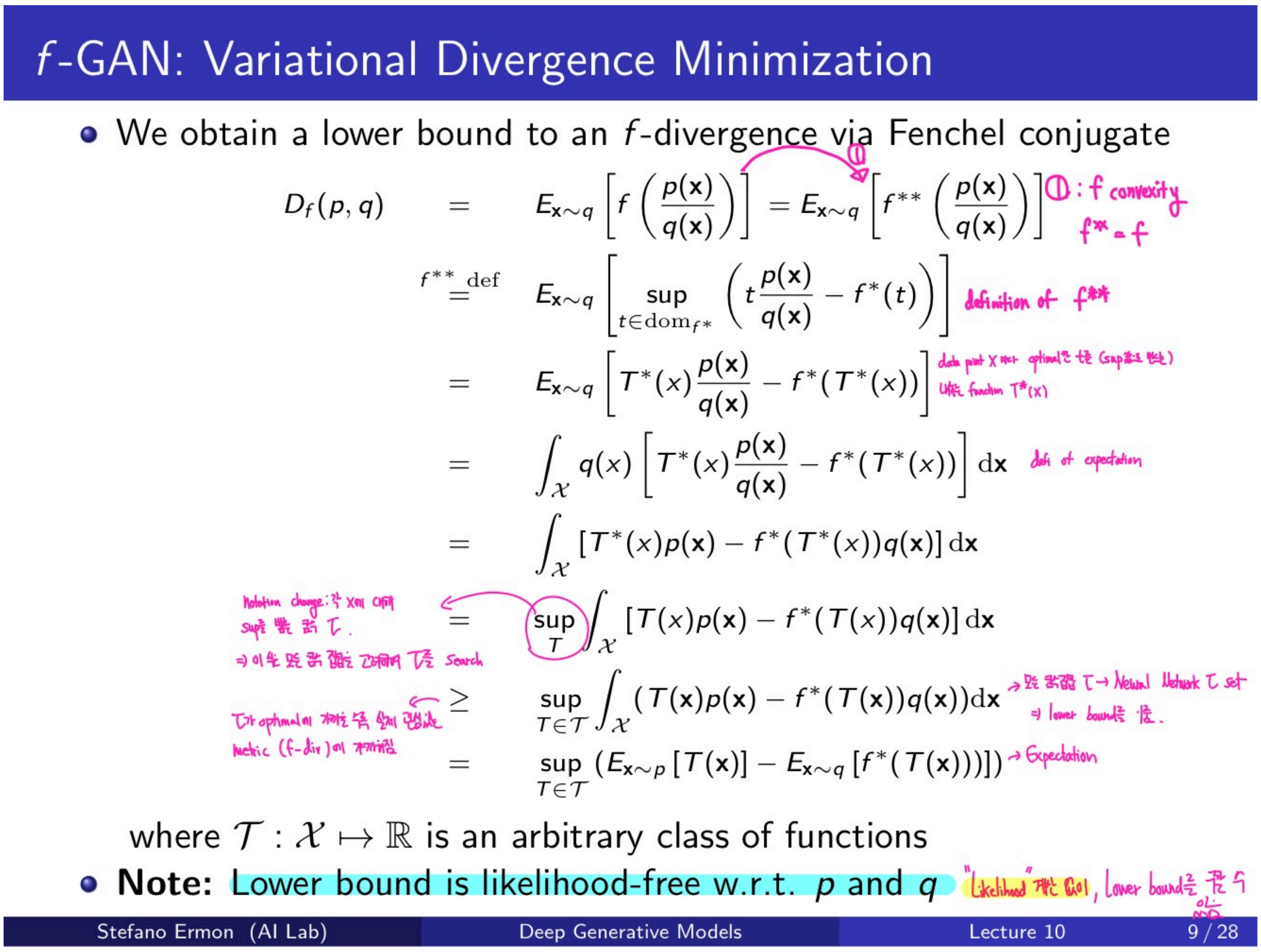
- 어떤 함수 f에 대한 conjugate를 다음과 같이 정의

$$f^{**}(u) = \sup_{t \in \mathrm{dom}_{f^*}} (tu - f^*(t))$$

- Conjugate의 Conjugate는 위와 같음

Strong Duality: $f^{**} = f$ when $f(\cdot)$ is convex, lower semicontinous.

- F가 Convex할 때, f = f** → f-div에서 f는 convex하므로 f div의 f 대신 f**를 이용하여 아래의 식 전개



f-GAN: Variational Divergence Minimization

- We obtain a lower bound to an f-divergence via Fenchel conjugate

$$\begin{aligned} D_f(p,q) &= E_{\mathbf{x}\sim q}\left[f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right] \overset{①}{=} E_{\mathbf{x}\sim q}\left[f^{**}\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right] \\ &\overset{f^{**}\text{def}}{=} E_{\mathbf{x}\sim q}\left[\sup_{t\in\mathrm{dom}_{f^*}}\left(t\frac{p(\mathbf{x})}{q(\mathbf{x})} - f^*(t)\right)\right] \\ &= E_{\mathbf{x}\sim q}\left[T^*(x)\frac{p(\mathbf{x})}{q(\mathbf{x})} - f^*(T^*(x))\right] \\ &= \int_{\mathcal{X}} q(x)\left[T^*(x)\frac{p(\mathbf{x})}{q(\mathbf{x})} - f^*(T^*(x))\right]\mathrm{d}\mathbf{x} \\ &= \int_{\mathcal{X}} [T^*(x)p(\mathbf{x}) - f^*(T^*(x))q(\mathbf{x})]\,\mathrm{d}\mathbf{x} \\ &= \sup_T \int_{\mathcal{X}} [T(x)p(\mathbf{x}) - f^*(T(x))q(\mathbf{x})]\,\mathrm{d}\mathbf{x} \\ &\geq \sup_{T\in\mathcal{T}} \int_{\mathcal{X}} (T(\mathbf{x})p(\mathbf{x}) - f^*(T(\mathbf{x}))q(\mathbf{x}))\mathrm{d}\mathbf{x} \\ &= \sup_{T\in\mathcal{T}} (E_{\mathbf{x}\sim p}[T(\mathbf{x})] - E_{\mathbf{x}\sim q}[f^*(T(\mathbf{x}))]) \end{aligned}$$

① : f convexity
$f^{**} = f$

definition of f^{**}

data point x 마다 optimal한 t를 (sup값을 만드는) 내뱉는 function $T^*(x)$

defi of expectation

notation change: 각 x에 대해 sup을 뽑는 함수 T.
⇒ 이 후 모든 함수 집합을 고려하여 T를 Search

→ 모든 함수집합 T → Neural Network T set
⇒ lower bound를 구함.

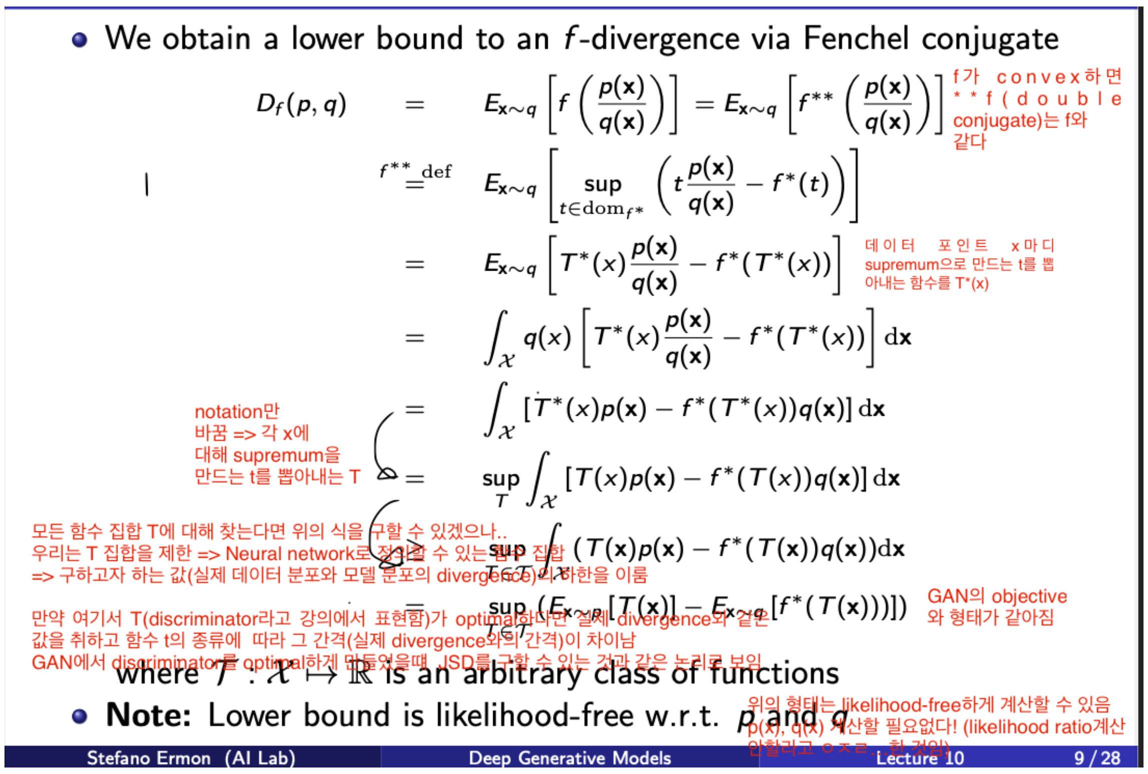
T가 optimal에 가까울수록 실제 관심있는 metric (f-div)에 가까워짐

→ Expectation

where $\mathcal{T} : \mathcal{X} \mapsto \mathbb{R}$ is an arbitrary class of functions

- **Note:** Lower bound is likelihood-free w.r.t. p and q "Likelihood" 계산 없이, lower bound를 구할 수 있음

Stefano Ermon (AI Lab) | Deep Generative Models | Lecture 10 | 9 / 28



- We obtain a lower bound to an f-divergence via Fenchel conjugate

$$\begin{aligned} D_f(p,q) &= E_{\mathbf{x}\sim q}\left[f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right] = E_{\mathbf{x}\sim q}\left[f^{**}\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right] \\ &\overset{f^{**}\text{def}}{=} E_{\mathbf{x}\sim q}\left[\sup_{t\in\mathrm{dom}_{f^*}}\left(t\frac{p(\mathbf{x})}{q(\mathbf{x})} - f^*(t)\right)\right] \\ &= E_{\mathbf{x}\sim q}\left[T^*(x)\frac{p(\mathbf{x})}{q(\mathbf{x})} - f^*(T^*(x))\right] \\ &= \int_{\mathcal{X}} q(x)\left[T^*(x)\frac{p(\mathbf{x})}{q(\mathbf{x})} - f^*(T^*(x))\right]\mathrm{d}\mathbf{x} \\ &= \int_{\mathcal{X}} [T^*(x)p(\mathbf{x}) - f^*(T^*(x))q(\mathbf{x})]\,\mathrm{d}\mathbf{x} \\ &= \sup_T \int_{\mathcal{X}} [T(x)p(\mathbf{x}) - f^*(T(x))q(\mathbf{x})]\,\mathrm{d}\mathbf{x} \\ &\geq \sup_{T\in\mathcal{T}} \int_{\mathcal{X}} (T(\mathbf{x})p(\mathbf{x}) - f^*(T(\mathbf{x}))q(\mathbf{x}))\mathrm{d}\mathbf{x} \\ &= \sup_{T\in\mathcal{T}} (E_{\mathbf{x}\sim p}[T(\mathbf{x})] - E_{\mathbf{x}\sim q}[f^*(T(\mathbf{x}))]) \end{aligned}$$

f가 convex하면 **f(double conjugate)는 f와 같다

데이터 포인트 x마다 supremum으로 만드는 t를 뽑아내는 함수를 T*(x)

notation만 바꿈 => 각 x에 대해 supremum을 만드는 t를 뽑아내는 T

모든 함수 집합 T에 대해 찾는다면 위의 식을 구할 수 있겠으나..
우리는 T 집합을 제한 => Neural network로 정의할 수 있는 함수 집합
=> 구하고자 하는 값(실제 데이터 분포와 모델 분포의 divergence)의 하한을 이룸

만약 여기서 T(discriminator라고 강의에서 표현함)가 optimal하다면 실제 divergence와 같은 값을 취하고 함수 t의 종류에 따라 그 간격(실제 divergence와의 간격)이 차이남
GAN에서 discriminator를 optimal하게 만들었을때 JSD를 구할 수 있는 것과 같은 논리로 보임

GAN의 objective 와 형태가 같아짐

where $\mathcal{T} : \mathcal{X} \mapsto \mathbb{R}$ is an arbitrary class of functions

- **Note:** Lower bound is likelihood-free w.r.t. p and q

위의 형태는 likelihood-free하게 계산할 수 있음
p(x), q(x) 계산할 필요없다! (likelihood ratio계산 안할려고 ㅇㅈㄹ...한 것임)

Stefano Ermon (AI Lab) | Deep Generative Models | Lecture 10 | 9 / 28

- 최종적으로 아래와 같은, 실제 divergence에 대한 어떤 lower bound를 계산하고 싶음.

- 다시 한번 lower bound인 이유는 함수 T를 전체 함수 집합에 대해 sup을 만족하는 함수를 찾는 것이 아니라, Neural Network로 제한된 집합에 대해 최대 또는 sup를 만족하는 함수를 찾는 것이기에 lower bound가 생김.
- 제한된 set내에서 optimal한 T를 찾을 수록, 실제 metric과의 간극이 줄어 들음.

Variational lower bound

$$D_f(p, q) \geq \sup_{T \in \mathcal{T}} \left(E_{\mathbf{x} \sim p} \left[T(\mathbf{x}) \right] - E_{\mathbf{x} \sim q} \left[f^*(T(\mathbf{x})) \right] \right)$$

Training Process with F-div objective training function

1. p를 p_data라 하고, q를 p_generator라 했을 때
2. T를 파라미터화(phi)한 neural network로 정의 G, generator network도 파라미터화(theta) 하여 정의

$$\min_{\theta} \max_{\phi} F(\theta, \phi) = E_{\mathbf{x} \sim p_{\text{data}}} \left[T_{\phi}(\mathbf{x}) \right] - E_{\mathbf{x} \sim p_{G_{\theta}}} \left[f^*(T_{\phi}(\mathbf{x})) \right]$$

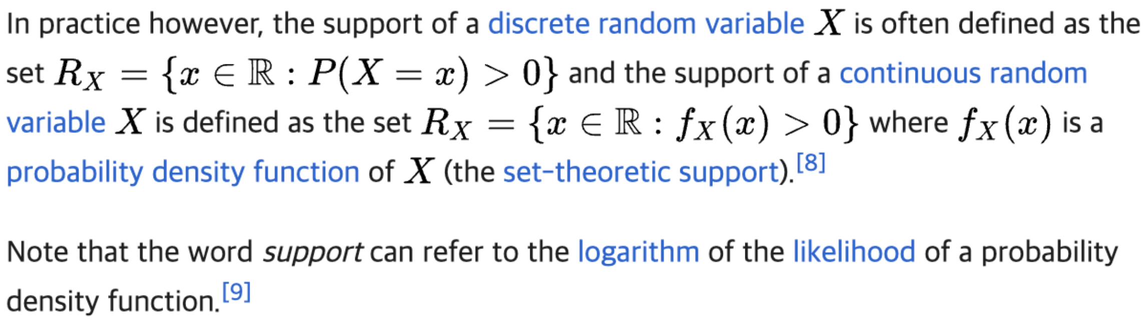
3. max phi를 통해 f-divergence의 good approximation 도달(하한이니까) ⇒ GAN의 discriminator가 optimal하기 바꾸는 것과 같음 (Tϕ tries to tighten the lower bound)
4. min theta를 통해 generator가 phi가 만들어낸 good approximation값(p data와 p generator)을 최소화 ⇒ 잘 근사한 데이터 분포와 모델 분포 사이의 metric을 최소화, GAN의 generator update과정과 같음 (Generator Gθ tries to minimize the divergence estimate)

→ 이외에도 다른 모든 "조건을 만족하는 f"를 이용하여 generative model을 훈련시킬 수 있음 : GAN도 그 특수한 경우중 하나

→ f를 어떤 것으로 하냐에 따라 generator가 최적화 해야할 divergence항(metric)이 달라짐

Limitations of F-divergence?

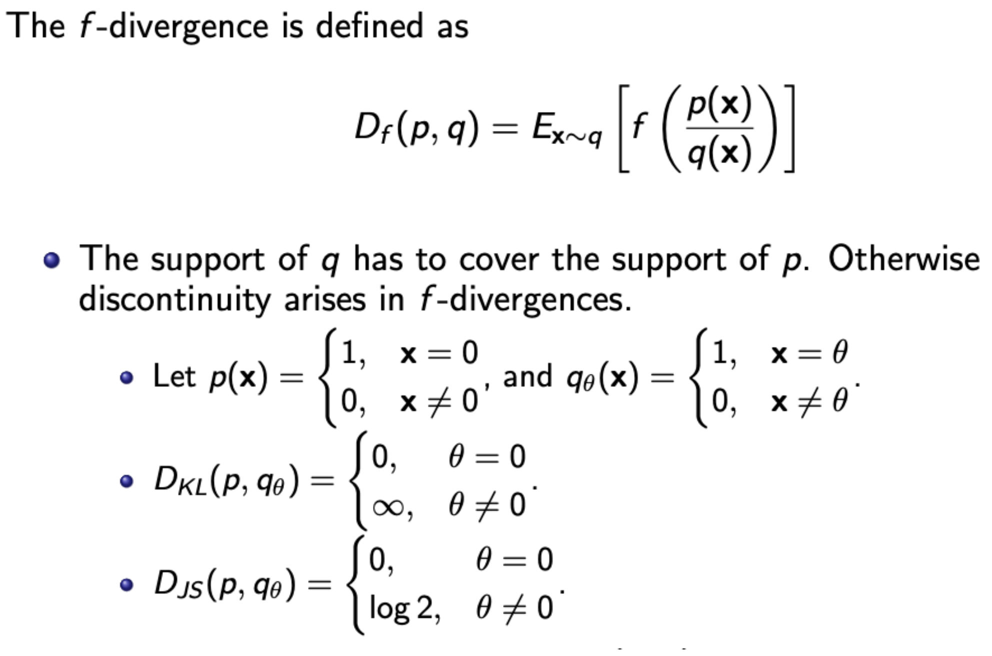
Support in Probability Distribution



In practice however, the support of a discrete random variable X is often defined as the set $R_X = \{x \in \mathbb{R} : P(X = x) > 0\}$ and the support of a continuous random variable X is defined as the set $R_X = \{x \in \mathbb{R} : f_X(x) > 0\}$ where $f_X(x)$ is a probability density function of X (the set-theoretic support).[8]

Note that the word *support* can refer to the logarithm of the likelihood of a probability density function.[9]

⇒ probability density function f에 대해 0을 제외한 양의 density를 가지는 모든 variable들의 집합



The f-divergence is defined as

$$D_f(p,q) = E_{\mathbf{x}\sim q}\left[f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right]$$

- The support of q has to cover the support of p. Otherwise discontinuity arises in f-divergences.
  - Let $p(\mathbf{x}) = \begin{cases} 1, & \mathbf{x}=0 \\ 0, & \mathbf{x}\neq 0 \end{cases}$, and $q_\theta(\mathbf{x}) = \begin{cases} 1, & \mathbf{x}=\theta \\ 0, & \mathbf{x}\neq \theta \end{cases}$.
  - $D_{KL}(p, q_\theta) = \begin{cases} 0, & \theta = 0 \\ \infty, & \theta \neq 0 \end{cases}$.
  - $D_{JS}(p, q_\theta) = \begin{cases} 0, & \theta = 0 \\ \log 2, & \theta \neq 0 \end{cases}$.

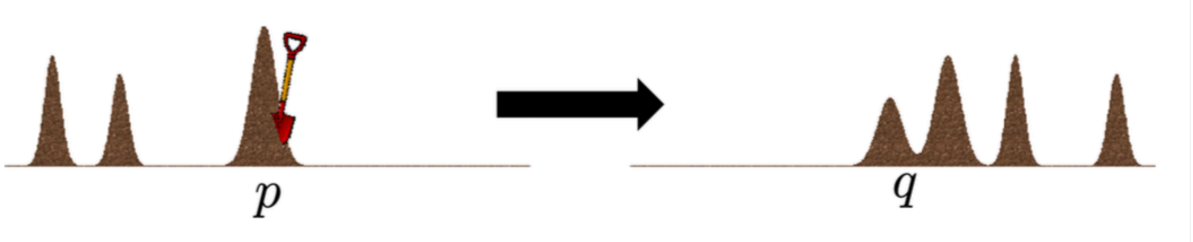
<확률 분포 P와 Q의 support가 disjoint, 두 분포의 support 교집합이 공집합이라면>

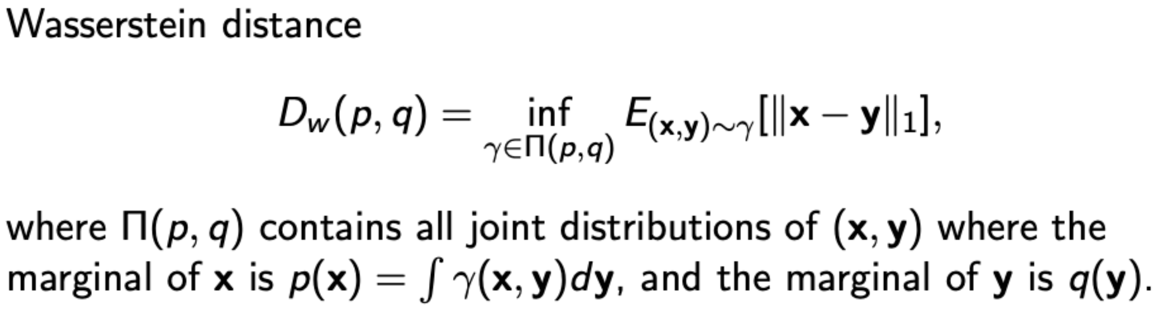
학습 신호로 사용되는 KL, JSD와 같은 divergence항들을 사용했을 때 두 확률 분포가 완전히 같지 않은 이상 발산하거나 두 분포가 얼마나 다르던 간에 항상 일정한 값을 내놓는 등 모델이 잘 학습을 할 수 있게 하지 않음.

→ 전자의 경우 무한대 값에 대해 gradient를 구할 수 없고, 후자의 경우 상수값에 대한 gradient는 0이니 학습 불가

Wasserstein GAN

⇒ 분포 간의 density ratio 기반이 아닌 분포 안의 샘플들 간의 거리를 기반의 Probability Similarity Metric을 이용하여 P_data와 P_Generator간의 간극을 줄이는 방향으로 generator를 학습



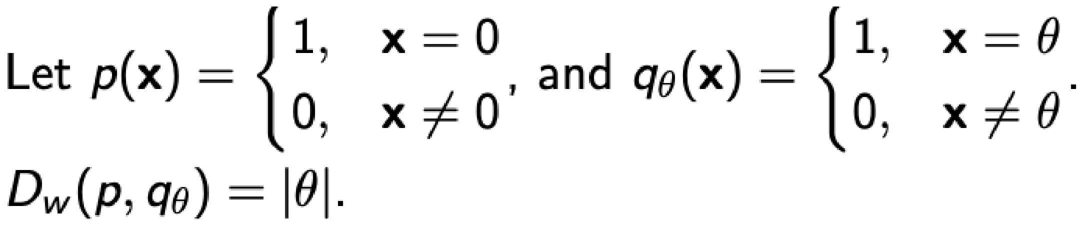


Wasserstein distance

$$D_w(p,q) = \inf_{\gamma\in\Pi(p,q)} E_{(\mathbf{x},\mathbf{y})\sim\gamma}[\|\mathbf{x}-\mathbf{y}\|_1],$$

where $\Pi(p,q)$ contains all joint distributions of $(\mathbf{x},\mathbf{y})$ where the marginal of $\mathbf{x}$ is $p(\mathbf{x}) = \int \gamma(\mathbf{x},\mathbf{y})d\mathbf{y}$, and the marginal of $\mathbf{y}$ is $q(\mathbf{y})$.

- Cost for transporting p to q (한 분포를 다른 분포로 이동시키는데에 드는 최소한의 비용)
- 두 분포의 확률 변수 X,Y에 대해 최적의 X, Y coupling(쌍)을 찾아(최적의 X, Y joint distribution r. 이때 r joint distribution은 marginalize x of r = q(y), marginalize y of r = p(x)를 만족해야함) 나타낸 최소한의 거리 이동 비용을 나타낸다. 다시 말해 r분포에 대해 x - y 백터의 L1 norm의 기댓값을 최소화하는 r을 찾고, 그때의 기댓값을 Wasserstein Distance라고 함



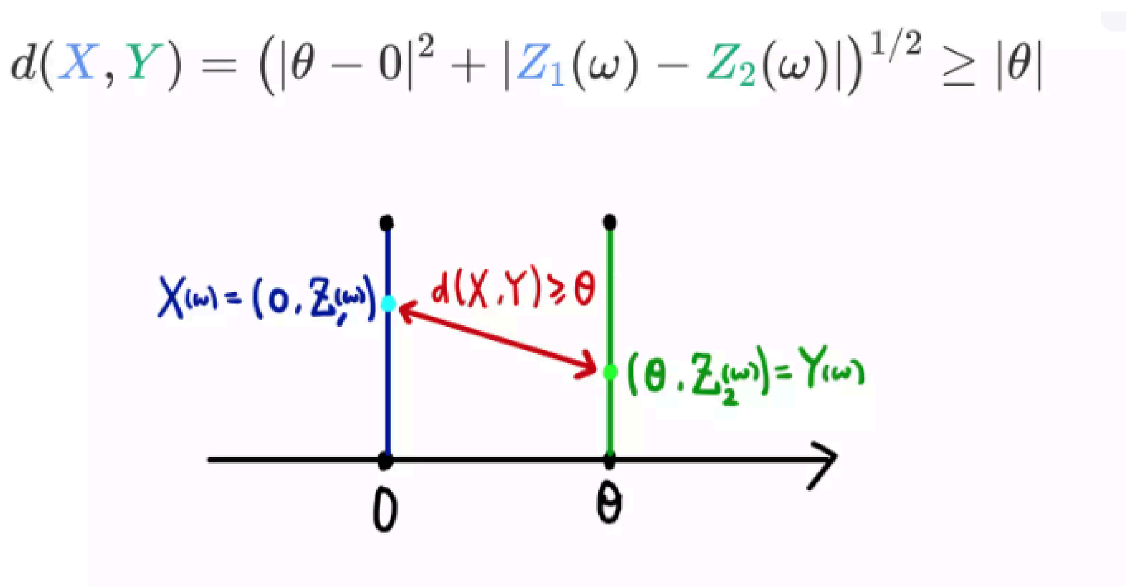
Let $p(\mathbf{x}) = \begin{cases} 1, & \mathbf{x}=0 \\ 0, & \mathbf{x}\neq 0 \end{cases}$, and $q_\theta(\mathbf{x}) = \begin{cases} 1, & \mathbf{x}=\theta \\ 0, & \mathbf{x}\neq \theta \end{cases}$.

$D_w(p, q_\theta) = |\theta|$.

- 위와 같이 두 확률 분포 사이의 거리가 멀어질 수록 (선형적으로) 더 멀다(다르다)라는 신호를 줌. ⇒ 이만큼 달라요! 라는 학습신호를 줌 (앞의 KL, JSD와 같이 발산 하거나 상수값을 내놓지 않음)

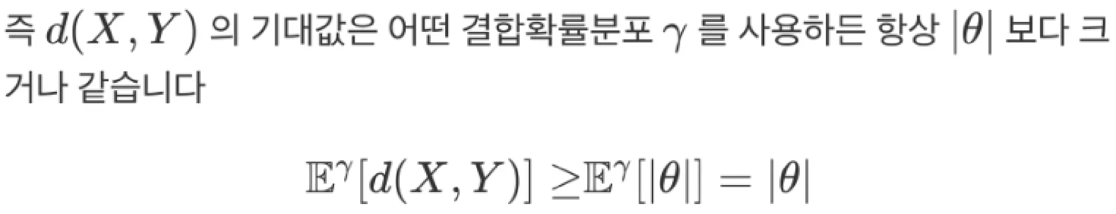
Detail

EX)



$$d(X,Y) = \left(|\theta - 0|^2 + |Z_1(\omega) - Z_2(\omega)|\right)^{1/2} \geq |\theta|$$

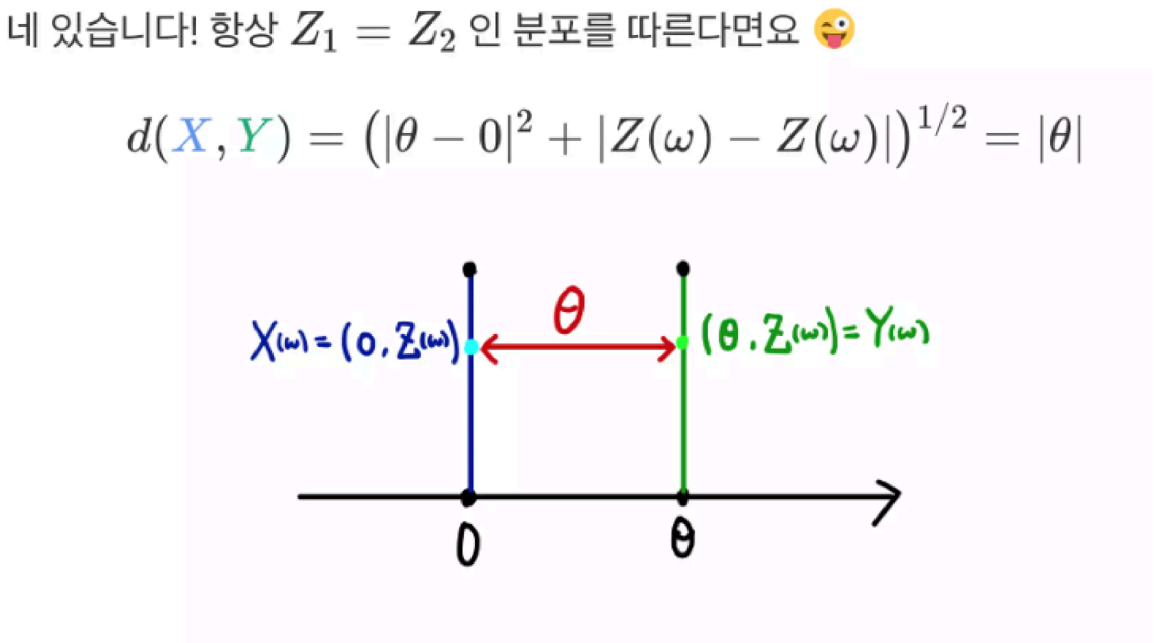
- 다음과 같이 disjoint support를 가지는 두 확률분포에 대해 (각 확률 분포에서의 샘플은 2-dim vector이고 각각 확률 변수를 X, Y)



즉 $d(X,Y)$ 의 기대값은 어떤 결합확률분포 γ 를 사용하든 항상 $|\theta|$ 보다 크거나 같습니다

$$\mathbb{E}^{\gamma}[d(X,Y)] \geq \mathbb{E}^{\gamma}[|\theta|] = |\theta|$$

- 어떤 두 임의의 (X, Y)쌍에 대해 D(X,Y)는 무조건 | theta | 이상의 값을 가짐

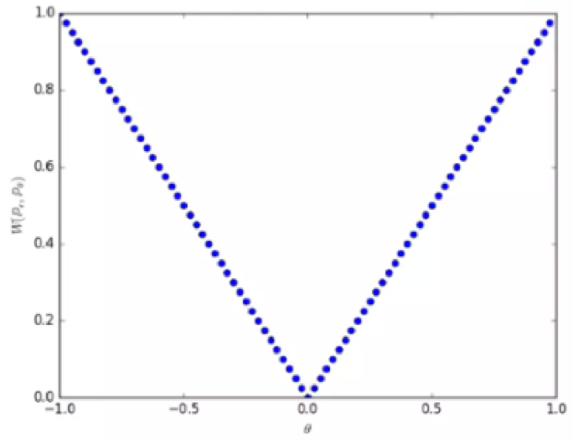


네 있습니다! 항상 $Z_1 = Z_2$ 인 분포를 따른다면요 😜

$$d(X,Y) = \left(|\theta - 0|^2 + |Z(\omega) - Z(\omega)|\right)^{1/2} = |\theta|$$

- 이 거리의 기댓 값을 최소로하는 Coupling은 모든 q에 대해 {(0,q),(theta,q)}를 만족하는 Coupling을 맺는 r joint distribution이고 그 때의 거리는 모두 | theta | 이고 기댓값도 역시 | theta |
- 그리고 이 때의 기댓값을 Wasserstein Distance라고 함

$$\therefore \quad W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$$

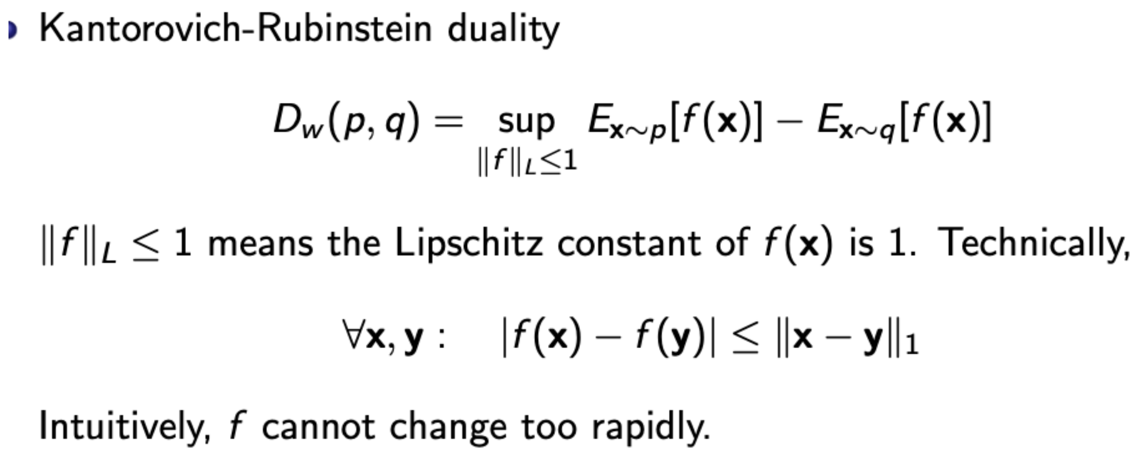


- 이 Wasserstein Distance는 두 분포의 거리가 멀어짐에 따라 최소 이동 비용 또한 증가함으로 선형적 증가
- 두 분포가 멀어질 수록 더 큰 신호를 줌

Optimization Problem in Wasserstein Distance

- 그러나 이런 최적의 joint distribution을 찾는 것은 연속하는 X, Y에 대해 모든 매칭 경우의 수를 판단하는 것과 같아 매우 어려움.

  ⇒ 최적화 문제의 어려움 (= 최소화할 대상인 Wasserstein Distance 자체를 구하는 것이 어려움)



- Kantorovich-Rubinstein duality

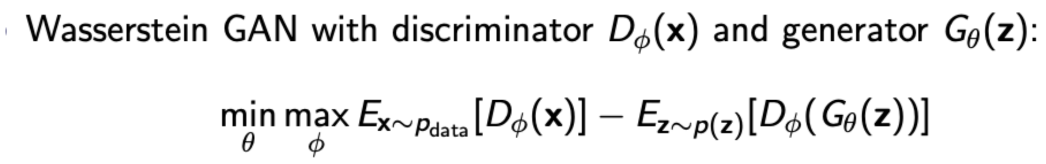
$$D_w(p,q) = \sup_{\|f\|_L \le 1} E_{\mathbf{x}\sim p}[f(\mathbf{x})] - E_{\mathbf{x}\sim q}[f(\mathbf{x})]$$

$\|f\|_L \le 1$ means the Lipschitz constant of $f(\mathbf{x})$ is 1. Technically,

$$\forall \mathbf{x}, \mathbf{y}: \quad |f(\mathbf{x}) - f(\mathbf{y})| \le \|\mathbf{x} - \mathbf{y}\|_1$$

Intuitively, f cannot change too rapidly.

- r 분포에 대한 제약사항 r(x,y)에서 x marginalization은 q(y) & vice versa임을 이용하여 lagrangian multiplier를 사용하여 Kantorovich-Rubinstein duality 유도하여 Wassetstein 식을 다른 최적화 문제로 바꿈 (GAN training objective와 동일한 형태의 식을 띔)
- 그래서 Kantorovich-Rubinstein duality를 이용하여 더욱 최적화 하기 쉬운 식으로 변환 (f에 대한 제약이 있음 -> Lipscitz constant가 1이어야한다. => 너무 급격히 변하는 함수가 아니어야 한다)



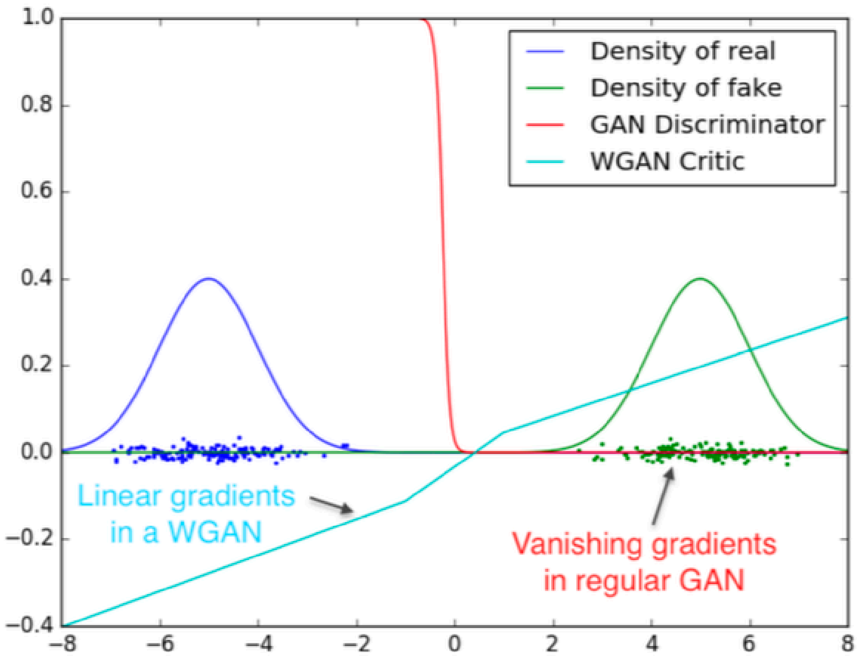
Wasserstein GAN with discriminator $D_\phi(\mathbf{x})$ and generator $G_\theta(\mathbf{z})$:

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

- 해당 제약을 만족하면서 식을 최대화 시키는 f(discriminator)를 찾기
- 실제로 training 시킬 때는 discriminator에 대한 이론적인 제약을 weight clipping이나 gradient penalty를 줌으로써 실현한다
- 똑같은 방식으로 최적의 f를 구해 식의 최댓값을 구하여 Similarity Metric인 Wasserstein Distance를 구하고
- 해당 metric에 대해 최소화하는 방향으로 generator를 훈련하여 generator가 실제 분포를 따라가도록함

⇒ 이렇게 f에 대해 최적화를 하여 p generator와 p data간의 wasserstein distance를 구하고, 이 metric에 대해 generator를 최소화하도록 업데이트 함으로써 generator가 만들어내는 분포가 실제 분포와 match하도록 한다.

Advantage of W-GAN: Stable Training



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

- 일반 GAN의 Discriminator는 분포가 겹치지 않으면 gradient가 사라져 generator 학습이 어려움.
- W-GAN은 GAN에 비해 discriminator(critic)가 generator에게 유의미한 gradient를 제공하여 training objective가 쉽게 최적화되도록 함. ⇒ 학습의 안정성

Inferring Latent Representations with GAN: BiGAN

„Unsupervised representation learning“

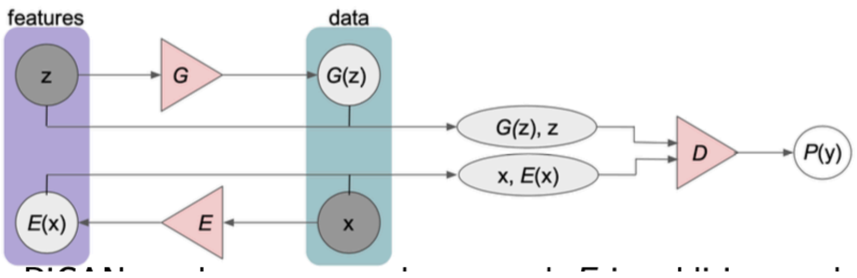
⇒ GAN에서는 어떻게 X에 대한 latent feature를 뽑아낼 수 있을까?

- x→z mapping : Flow model과 같이 generator를 역변환할 수도 없고, VAE와 같이 인코더도 없음

두가지 solution

1. discriminator를 활용하여 x에 대한 representation을 뽑기 : x가 fake/true 인지 구별하는 과정에서, x에 대해 좋은 representation을 뽑아냈을 것이다.
2. x를 만들어낸 z latent가 무엇인지를 direct하게 inference하는 방법 : generator가 생성한 x에 대한 z는 알 수 있으나, p_data로부터 나온 관측값 x에 대한 z는 알 수 없음 관측값 x를 z로 mapping 시킬 모델 구조가 필요함
   => “BiGAN의 encoder”

Training BiGAN



- Discriminator D는 1) p data의 x와 그에 대응되는, encoder가 뽑아낸 latent 쌍과, p generator의 x와 그에 대응되는 prior로부터 나온 latent variable z쌍이 서로 다른 분포에서 나온 것을 구분하도록 학습 (maximize the two-sample test objective)
- 쌍 안의 데이터가 concat되어 Discriminator로 들어간다고 함
- 학습이 완료된 후에는 1) Generator부터 새로운 데이터 X를 생성할 수 있고, 2) Encoder로부터 X에 대한 direct latent representation을 추출할 수 있다고 함(어떤 z로부터 추출된 x지?를 유추)