Theory behind GAN

Generation

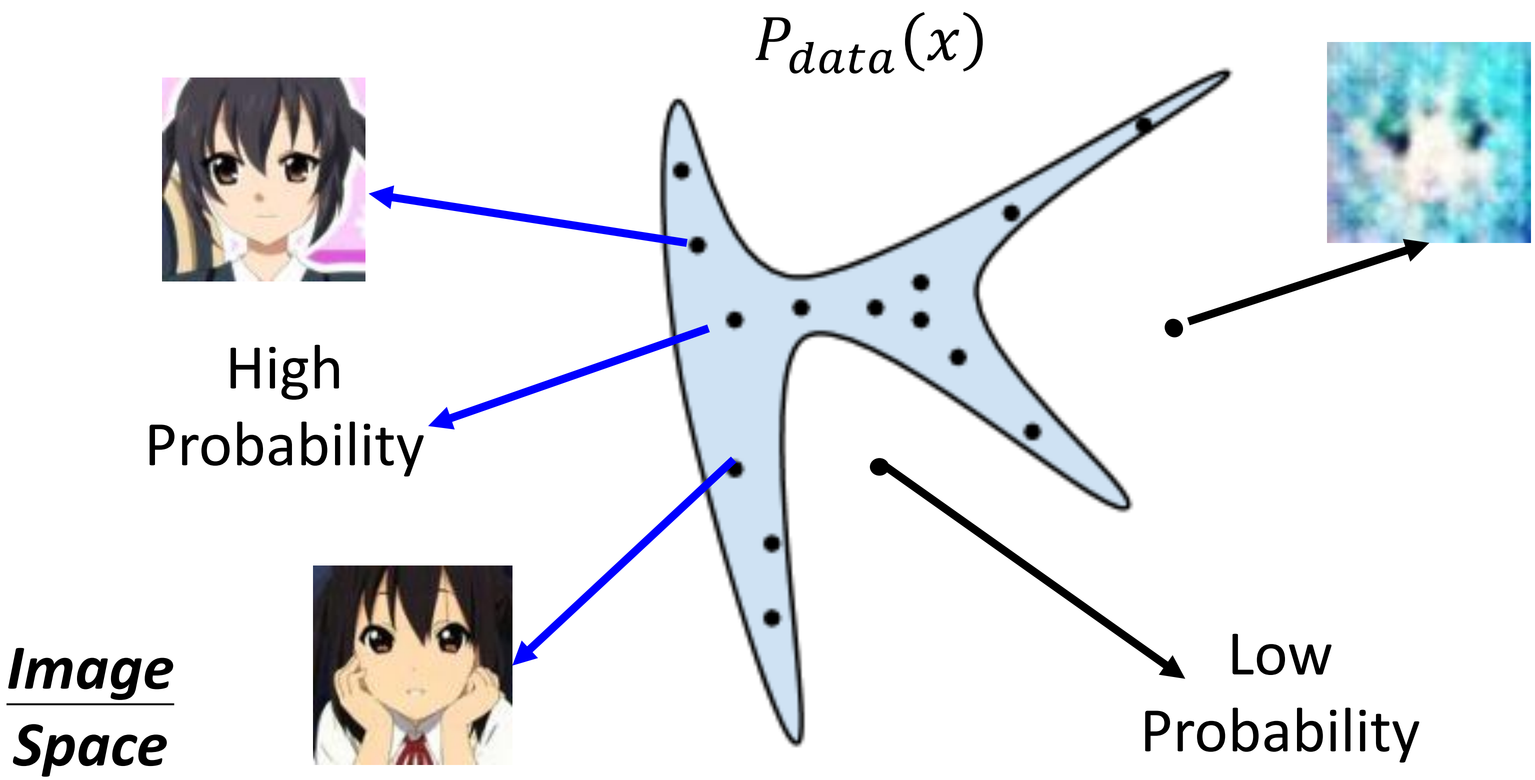
Using Generative Adversarial Network (GAN)

Drawing?

Generation

x: an image (a high-dimensional vector)

- We want to find data distribution $P_{data}(x)$



Maximum Likelihood Estimation

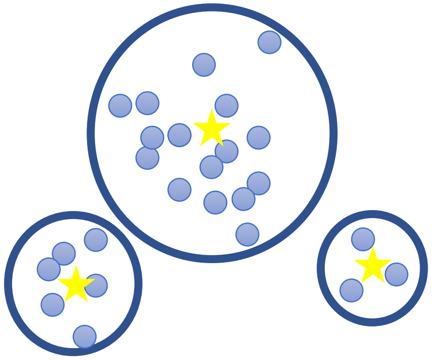
- Given a data distribution $P_{data}(x)$ (We can sample from it.)
- We have a distribution $P_G(x;\theta)$ parameterized by θ
  - We want to find θ such that $P_G(x;\theta)$ close to $P_{data}(x)$
  - E.g. $P_G(x;\theta)$ is a Gaussian Mixture Model, θ are means and variances of the Gaussians

Sample $\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$

We can compute $P_G\left(x^i;\theta\right)$

Likelihood of generating the samples

$$L = \prod_{i=1}^{m} P_G\left(x^i;\theta\right)$$



Find θ^* maximizing the likelihood

Maximum Likelihood Estimation = Minimize KL Divergence

$$\theta^* = arg \max_\theta \prod_{i=1}^{m} P_G(x^i;\theta) = arg \max_\theta log \prod_{i=1}^{m} P_G(x^i;\theta)$$

$$= arg \max_\theta \sum_{i=1}^{m} log P_G(x^i;\theta)$$

$\{x^1, x^2, \ldots, x^m\}$ from $P_{data}(x)$

$$\approx arg \max_\theta E_{x \sim P_{data}}[log P_G(x;\theta)]$$

$$= arg \max_\theta \int_x P_{data}(x) log P_G(x;\theta) dx - \int_x P_{data}(x) log P_{data}(x) dx$$

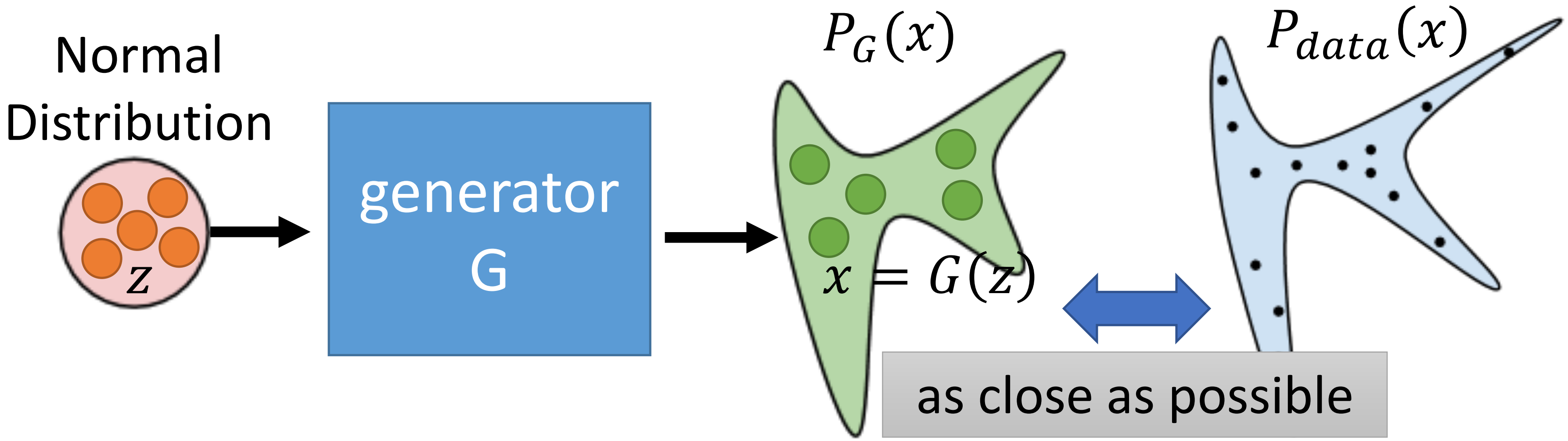
$$= arg \min_\theta KL(P_{data}||P_G)$$

How to define a general P_G?

Generator

x: an image (a high-dimensional vector)

- A generator G is a network. The network defines a probability distribution P_G



$$G^* = arg \min_G \underline{Div(P_G, P_{data})}$$

Divergence between distributions P_G and P_{data}

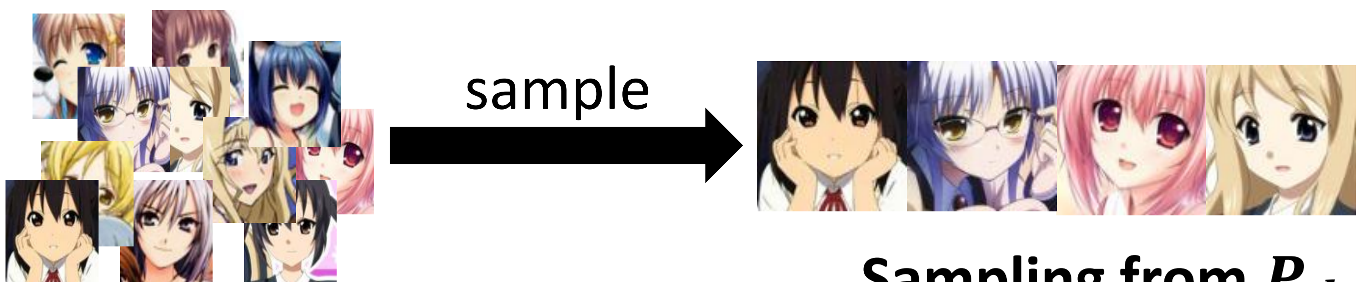
How to compute the divergence?

Discriminator

$$G^* = arg\min_G Div(P_G, P_{data})$$

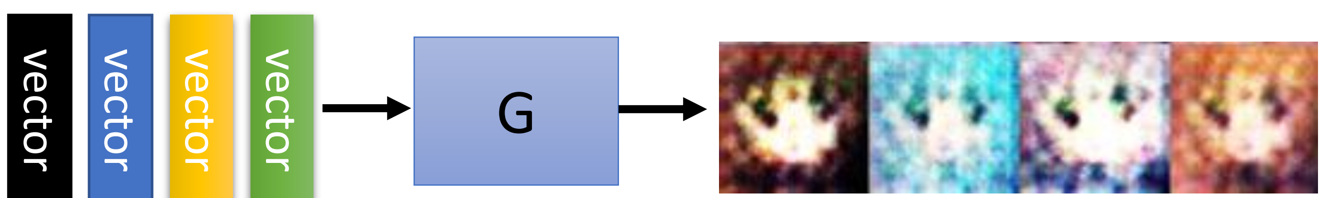
Although we do not know the distributions of P_G and P_{data}, we can sample from them.

Database



Sampling from P_{data}

sample from normal



Sampling from P_G

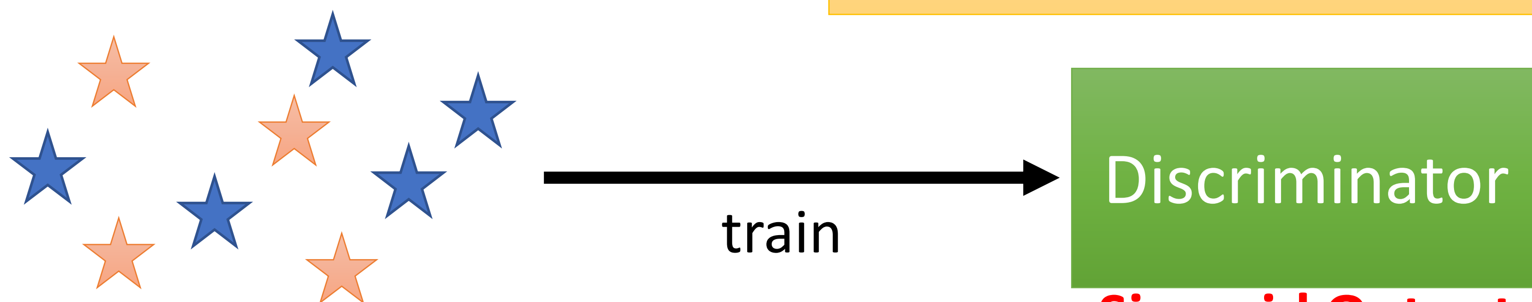
Discriminator

$$G^* = arg \min_G Div(P_G, P_{data})$$

: data sampled from P_{data}

: data sampled from P_G

Using the example objective function is exactly the same as training a binary classifier.



Sigmoid Output

Example Objective Function for D

$$V(G,D) = E_{x \sim P_{data}}[logD(x)] + E_{x \sim P_G}[log(1 - D(x))]$$

(G is fixed)

Training: $D^* = arg \boxed{\max_D V(D,G)}$

The maximum objective value is related to JS divergence.

[Goodfellow, et al., NIPS, 2014]

Discriminator

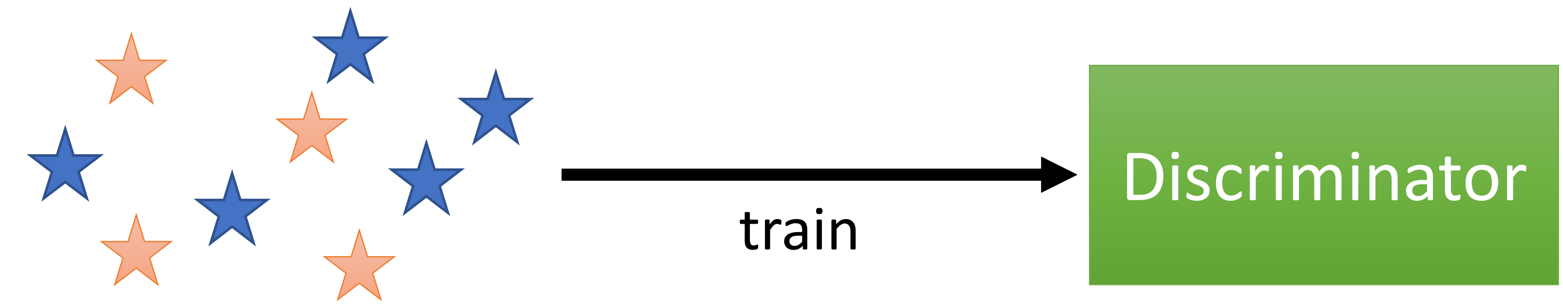
$$G^* = arg \min_G Div(P_G, P_{data})$$

★ : data sampled from P_{data}

★ : data sampled from P_G

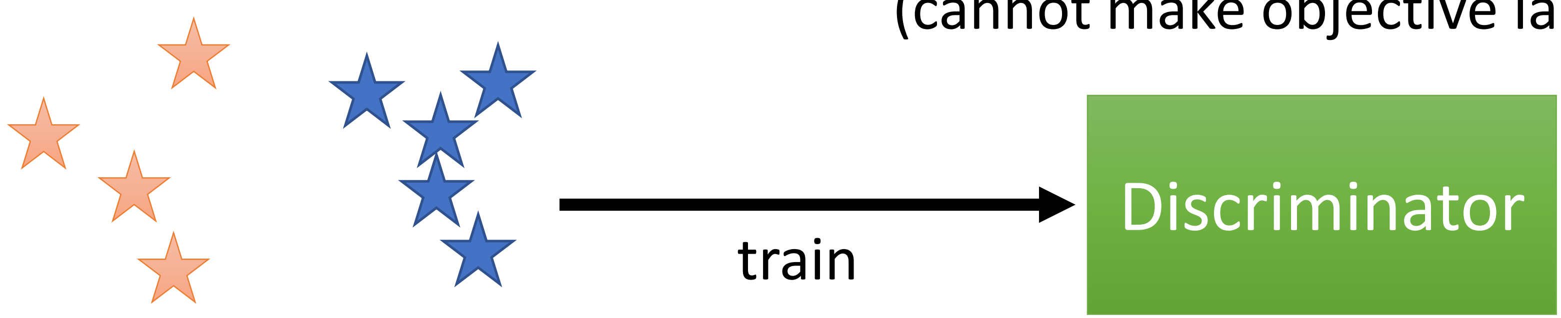
Training:

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



small divergence

hard to discriminate
(cannot make objective large)



large divergence

easy to discriminate

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

$$\boxed{\begin{aligned} V = &E_{x\sim P_{data}}[logD(x)] \\ &+E_{x\sim P_G}\left[log\left(1-D(x)\right)\right]\end{aligned}}$$

- Given G, what is the optimal D* maximizing

$$V = E_{x\sim P_{data}}[logD(x)] + E_{x\sim P_G}\left[log\left(1-D(x)\right)\right]$$

$$= \int_x P_{data}(x)logD(x)\,dx + \int_x P_G(x)log\left(1-D(x)\right)dx$$

$$= \int_x \left[P_{data}(x)logD(x) + P_G(x)log\left(1-D(x)\right)\right]dx$$

Assume that D(x) can be any function

- Given x, the optimal D* maximizing

$$P_{data}(x)logD(x) + P_G(x)log\left(1-D(x)\right)$$

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

$$V = E_{x\sim P_{data}}[logD(x)] + E_{x\sim P_G}\left[log\left(1 - D(x)\right)\right]$$

- Given x, the optimal D* maximizing

$$\underset{a}{P_{data}}(x)log\underset{D}{D}(x) + \underset{b}{P_G}(x)log\left(1 - \underset{D}{D}(x)\right)$$

- Find D* maximizing: $\mathrm{f}(D) = alog(D) + blog(1 - D)$

$$\frac{d\mathrm{f}(D)}{dD} = a \times \frac{1}{D} + b \times \frac{1}{1-D} \times (-1) = 0$$

$$a \times \frac{1}{D^*} = b \times \frac{1}{1-D^*} \qquad a \times (1 - D^*) = b \times D^*$$

$$a - aD^* = bD^* \qquad a = (a+b)D^*$$

$$D^* = \frac{a}{a+b} \quad \Rightarrow \quad 0 < D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} < 1$$

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

$$\boxed{\begin{aligned} V = {} & E_{x\sim P_{data}}[\log D(x)] \\ & + E_{x\sim P_G}[\log(1-D(x))] \end{aligned}}$$

$$\max_D V(G,D) = V(G,D^*) \qquad D^*(x) = \frac{P_{data}(x)}{P_{data}(x)+P_G(x)}$$

$$= E_{x\sim P_{data}}\left[\log\frac{P_{data}(x)}{P_{data}(x)+P_G(x)}\right]$$

$$+E_{x\sim P_G}\left[\log\frac{P_G(x)}{P_{data}(x)+P_G(x)}\right]$$

$$= \int_x P_{data}(x)\log\frac{\frac{1}{2}P_{data}(x)}{\frac{P_{data}(x)+P_G(x)}{2}}dx$$

$$+\int_x P_G(x)\log\frac{\frac{1}{2}P_G(x)}{\frac{P_{data}(x)+P_G(x)}{2}}dx$$

$$+2\log\frac{1}{2} \quad -2\log 2$$

$$\max_{D} V(G,D)$$

$$\mathrm{JSD}(P \parallel Q) = \frac{1}{2} D(P \parallel M) + \frac{1}{2} D(Q \parallel M)$$

$$M = \frac{1}{2}(P + Q)$$

$$\max_{D} V(G,D) \ = V(G, D^*) \qquad D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

$$= -2log2 + \int_x P_{data}(x) log \frac{P_{data}(x)}{\left(P_{data}(x) + P_G(x)\right)/2} dx$$

$$+ \int_x P_G(x) log \frac{P_G(x)}{\left(P_{data}(x) + P_G(x)\right)/2} dx$$

$$= -2\log 2 + \mathrm{KL}\left(\mathrm{P_{data}} || \frac{\mathrm{P_{data}} + \mathrm{P_G}}{2}\right) + \mathrm{KL}\left(\mathrm{P_G} || \frac{\mathrm{P_{data}} + \mathrm{P_G}}{2}\right)$$

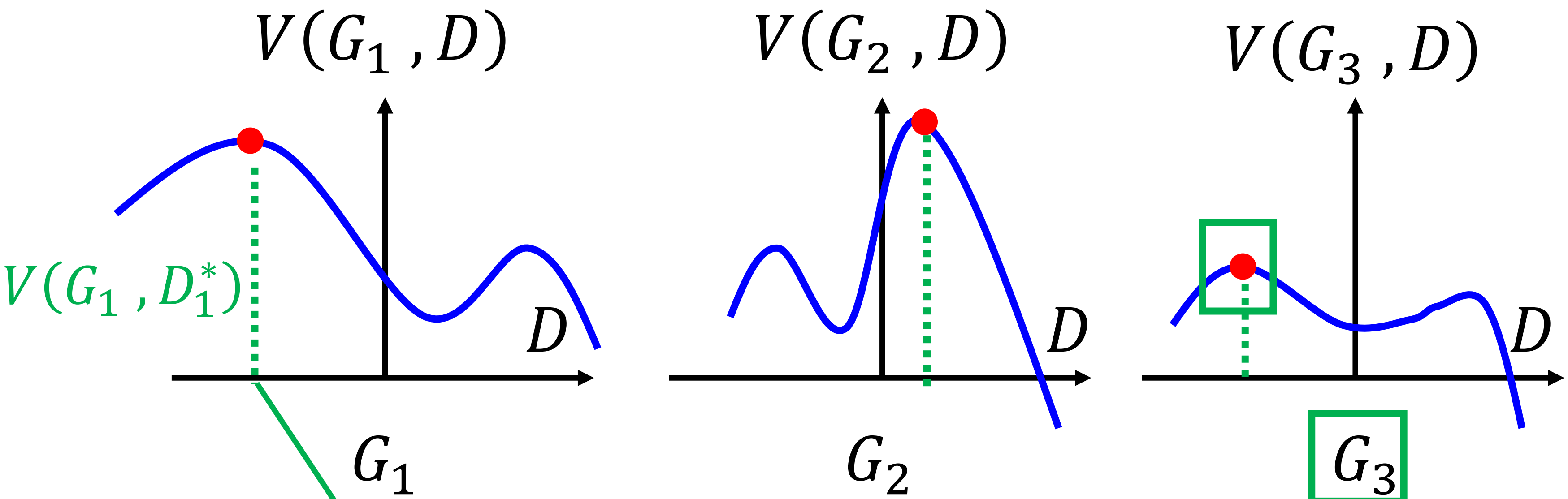
$$= -2log2 + 2JSD(P_{data} || P_G)$$

Jensen-Shannon divergence

$$\boxed{G^*} = arg \min_G \boxed{\max_D V(G,D)}$$

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

The maximum objective value is related to JS divergence.



Divergence between P_{G_1} and P_{data}

[Goodfellow, et al., NIPS, 2014]

$$G^* = arg \min_G \boxed{\max_D V(G, D)}$$

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

The maximum objective value is related to JS divergence.

- Initialize generator and discriminator
- In each training iteration:

<u>Step 1</u>: Fix generator G, and update discriminator D

<u>Step 2</u>: Fix discriminator D, and update generator G

Algorithm

$$G^* = arg \min_G \boxed{\max_D V(G,D)}$$

$L(G)$

- To find the best G minimizing the loss function $L(G)$,

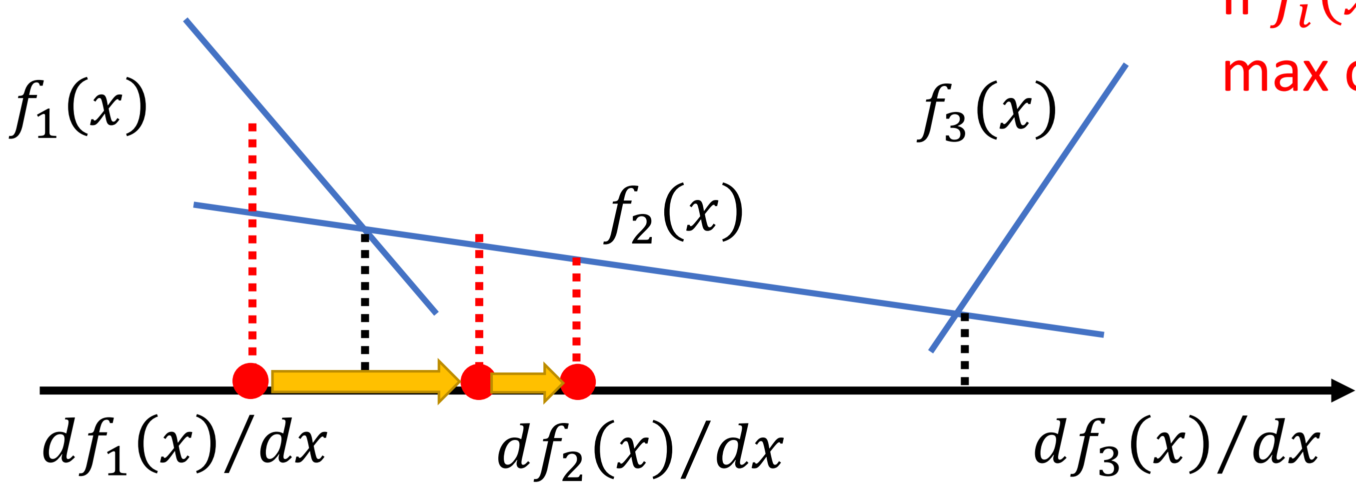
$$\theta_G \leftarrow \theta_G - \eta \, \partial L(G) / \partial \theta_G$$

θ_G defines G

$$f(x) = \max\{f_1(x), f_2(x), f_3(x)\}$$

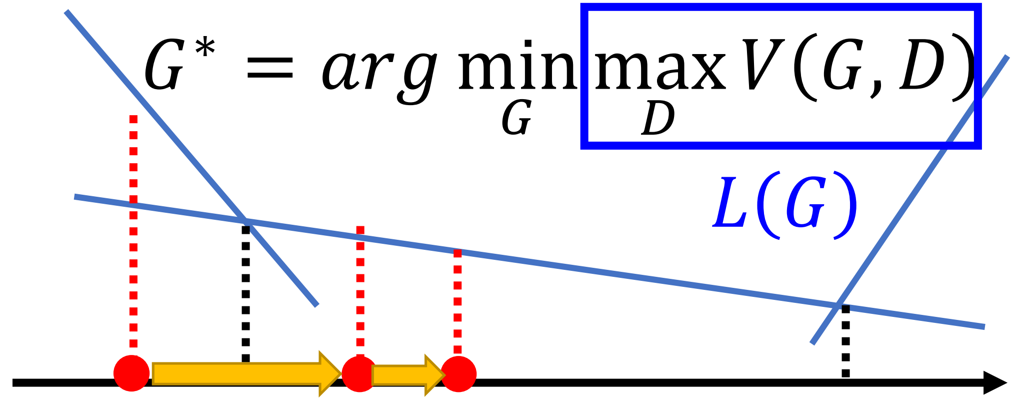
$$\frac{df(x)}{dx} =? \; df_i(x)/dx$$

If $f_i(x)$ is the max one



Algorithm

$$G^* = arg \min_G \boxed{\max_D V(G,D)}$$



- Given G_0
- Find D_0^* maximizing $V(G_0, D)$

Using Gradient Ascent

$V(G_0, D_0^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_0}(x)$

- $\theta_G \leftarrow \theta_G - \eta \, \partial V(G, D_0^*)/\partial \theta_G$ ⇒ Obtain G_1

Decrease JS divergence(?)

- Find D_1^* maximizing $V(G_1, D)$

$V(G_1, D_1^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_1}(x)$

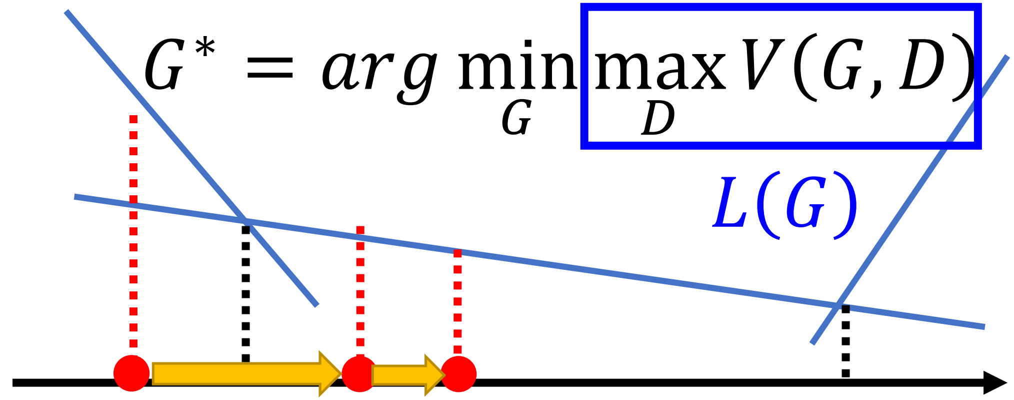
- $\theta_G \leftarrow \theta_G - \eta \, \partial V(G, D_1^*)/\partial \theta_G$ ⇒ Obtain G_2

Decrease JS divergence(?)

- ……

Algorithm

$$G^* = arg \min_G \boxed{\max_D V(G,D)}$$

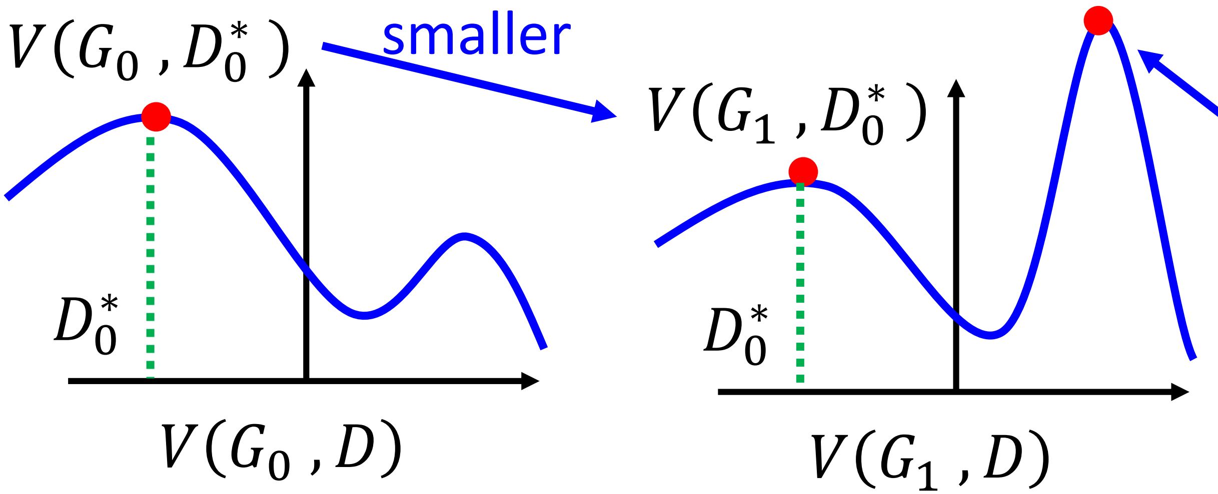


- Given G_0
- Find D_0^* maximizing $V(G_0, D)$

$V(G_0, D_0^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_0}(x)$

- $\theta_G \leftarrow \theta_G - \eta \, \partial V(G, D_0^*)/\partial \theta_G$ → Obtain G_1

Decrease JS divergence(?)



$V(G_1, D_1^*)$ ……

Assume $D_0^* \approx D_1^*$

Don't update G too much

In practice …

$$V = E_{x\sim P_{data}}[logD(x)] + E_{x\sim P_G}[log(1 - D(x))]$$

- Given G, how to compute $\max_D V(G,D)$
  - Sample $\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$, sample $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$ from generator $P_G(x)$

Maximize $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} logD(x^i) + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D(\tilde{x}^i)\right)$

=

Binary Classifier

D is a binary classifier with sigmoid output (can be deep)

$\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$ ➡ Positive examples

$\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$ from $P_G(x)$ ➡ Negative examples

Minimize Cross-entropy

Algorithm

Initialize θ_d for D and θ_g for G

Can only find lower found of $\max_D V(G, D)$

- In each training iteration:

Learning D

$\boxed{\text{Repeat k times}}$

- Sample m examples $\{x^1, x^2, \dots, x^m\}$ from data distribution $P_{data}(x)$
- Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$
- Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
- Update discriminator parameters θ_d to maximize
  - $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} logD(x^i) + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D(\tilde{x}^i)\right)$
  - $\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$

Learning G

$\boxed{\text{Only Once}}$

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

Objective Function for Generator in Real Implementation

$$V = \cancel{E_{x\sim P_{data}}[logD(x)]}$$
$$+E_{x\sim P_G}\left[log\left(1 - D(x)\right)\right]$$

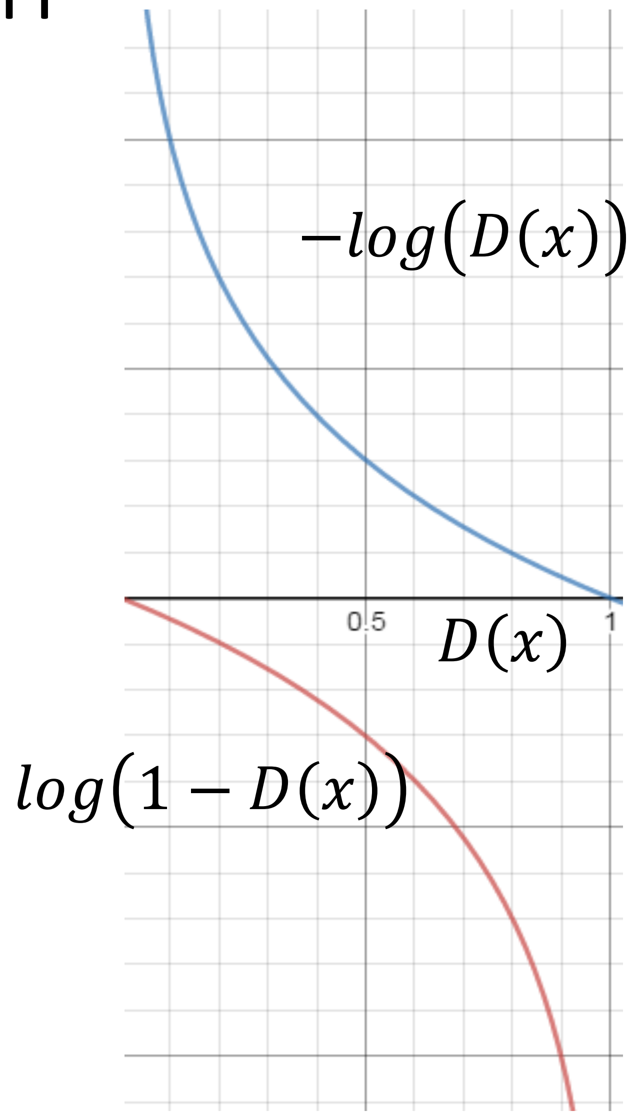
Slow at the beginning

Minimax GAN (MMGAN)

$$V = E_{x\sim P_G}\left[-log\left(D(x)\right)\right]$$

Real implementation:
label x from P_G as positive

Non-saturating GAN (NSGAN)

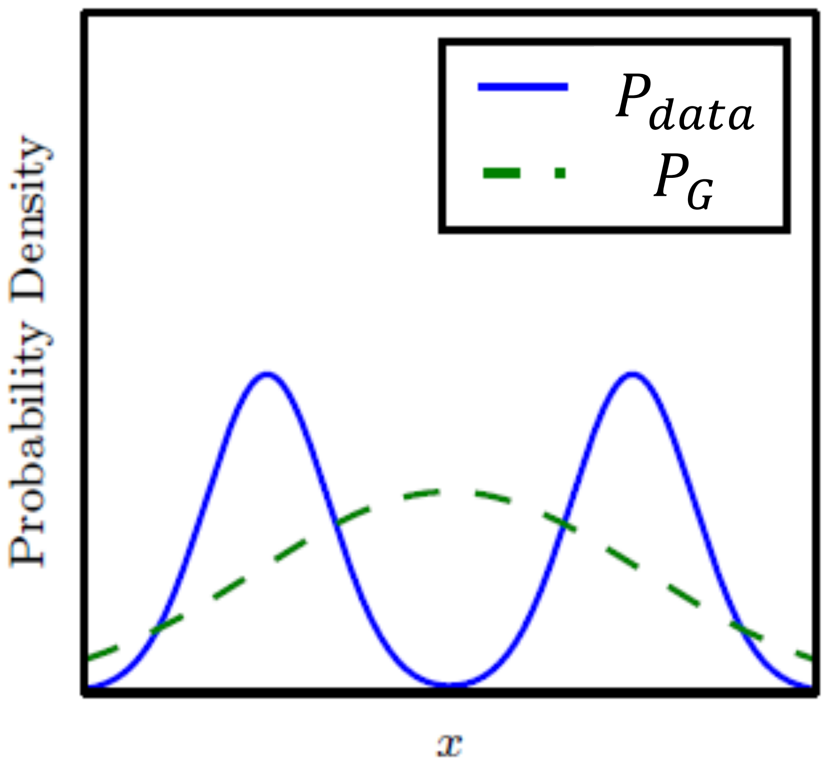


fGAN: General Framework of GAN

Modified from Ian Goodfellow's tutorial

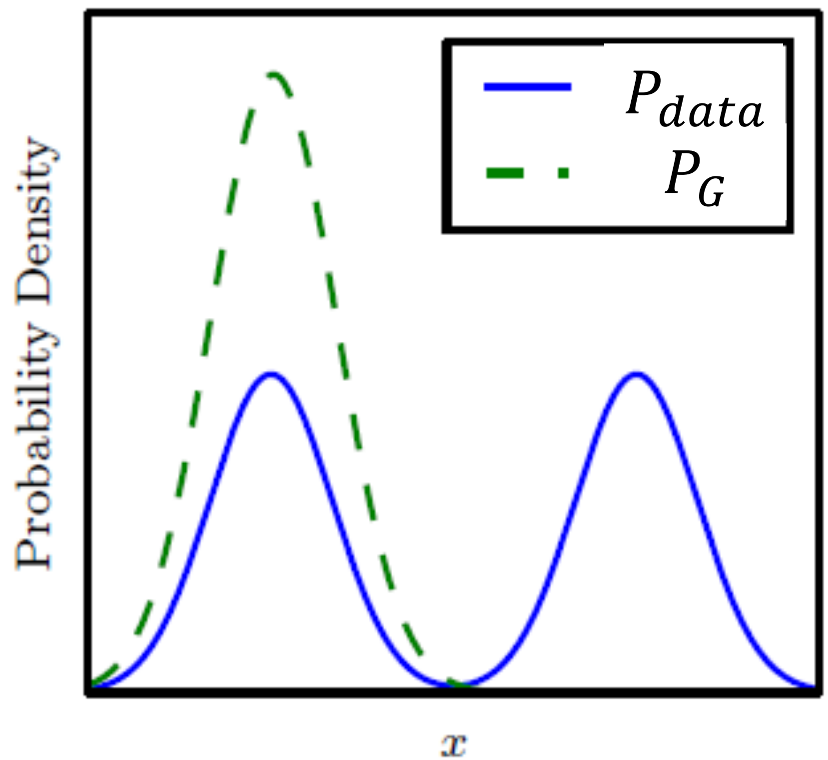
Flaw in Optimization?

$$KL = \int P_{data} log \frac{P_{data}}{P_G} dx$$



Maximum likelihood
(minimize KL($P_{data}||P_G$))

Reverse $KL = \int P_G log \frac{P_G}{P_{data}} dx$



Minimize KL($P_G||P_{data}$)
(reverse KL)

f-divergence

P and Q are two distributions. $p(x)$ and q(x) are the probability of sampling x.

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex

f(1) = 0

$D_f(P||Q)$ evaluates the difference of P and Q

If $p(x) = q(x)$ for all x

smallest

$$D_f(P||Q) = \int_x q(x) \underbrace{f\left(\overset{=1}{\boxed{\frac{p(x)}{q(x)}}}\right)}_{=0} dx = 0$$

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

Because f is convex

$$\geq f\left(\int_x \cancel{q(x)} \frac{p(x)}{\cancel{q(x)}} dx\right)$$

$$= f(1) = 0$$

If P and Q are the same distributions,

$D_f(P||Q)$ has the smallest value, which is 0

f-divergence

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex

f(1) = 0

$f(x) = x log x$

KL

$$D_f(P||Q) = \int_x q(x) \frac{p(x)}{q(x)} log\left(\frac{p(x)}{q(x)}\right) dx = \int_x p(x) log\left(\frac{p(x)}{q(x)}\right) dx$$

$f(x) = -log x$

Reverse KL

$$D_f(P||Q) = \int_x q(x)\left(-log\left(\frac{p(x)}{q(x)}\right)\right) dx = \int_x q(x) log\left(\frac{q(x)}{p(x)}\right) dx$$

$f(x) = (x-1)^2$

Chi Square

$$D_f(P||Q) = \int_x q(x)\left(\frac{p(x)}{q(x)} - 1\right)^2 dx = \int_x \frac{\left(p(x) - q(x)\right)^2}{q(x)} dx$$

Fenchel Conjugate

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex, f(1) = 0

- Every convex function f has a conjugate function f*

$$f^*(t) = \max_{x \in dom(f)} \{xt - f(x)\}$$

$$f^*(t_1) = \max_{x \in dom(f)} \{x t_1 - f(x)\}$$

$$f^*(t_2) = \max_{x \in dom(f)} \{x t_2 - f(x)\}$$

$x_1 t_1 - f(x_1)$ $f^*(t_1)$

$x_2 t_1 - f(x_2)$

$x_3 t_1 - f(x_3)$

$x_3 t_2 - f(x_3)$ $f^*(t_2)$

$x_2 t_2 - f(x_2)$

$x_1 t_2 - f(x_1)$

t_1 t_2 t

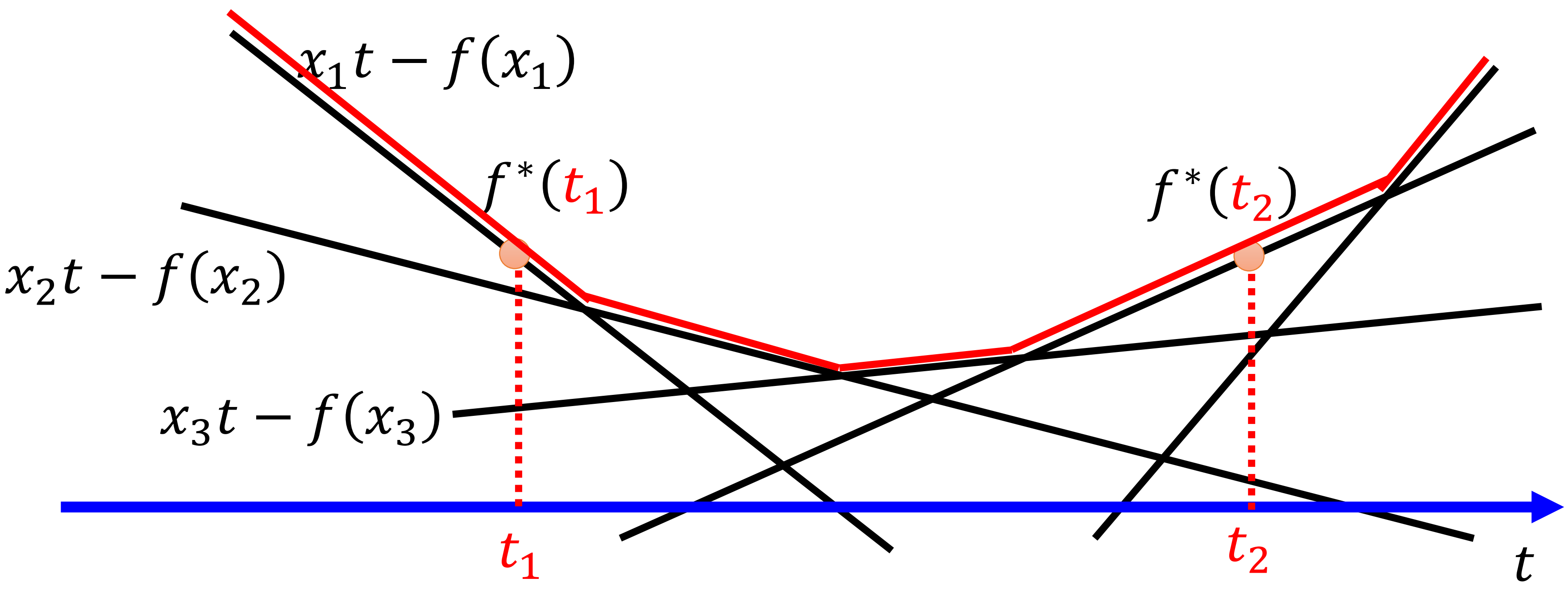
Fenchel Conjugate

$$D_f(P||Q) = \int\limits_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex, f(1) = 0

- Every convex function f has a conjugate function f*

$$f^*(t) = \boxed{\max_{x \in dom(f)} \{\boxed{xt - f(x)}\}}$$



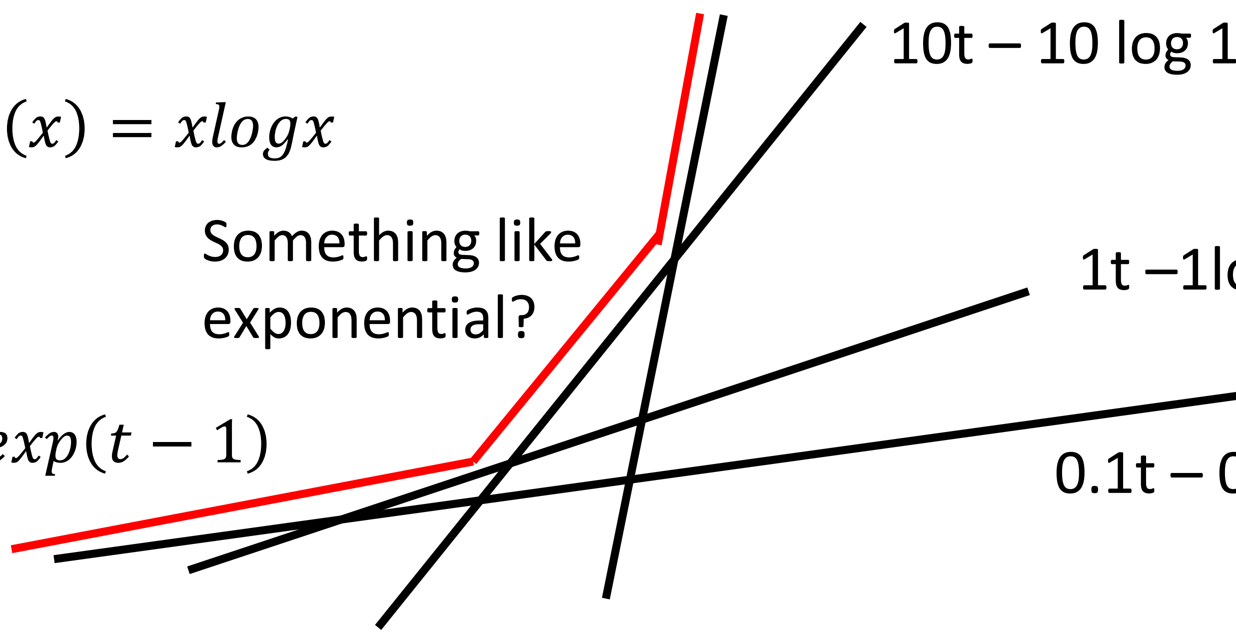
Fenchel Conjugate

- Every convex function f has a conjugate function f*

$$f^*(t) = \max_{x \in dom(f)} \{xt - f(x)\}$$

$$f(x) = xlogx$$

$$f^*(t) = exp(t-1)$$



Fenchel Conjugate

- Every convex function f has a conjugate function f*
- (f*)* = f

$$f^*(t) = \max_{x \in dom(f)} \{xt - f(x)\}$$

$$f(x) = xlogx \longleftrightarrow f^*(t) = exp(t-1)$$

$$f^*(t) = \max_{x \in dom(f)} \{xt - xlogx\}$$

$g(x) = xt - xlogx$ Given t, find x maximizing $g(x)$

$t - logx - 1 = 0$ $x = exp(t-1)$

$$f^*(t) = exp(t-1) \times t - exp(t-1) \times (t-1) = exp(t-1)$$

Connection with GAN

$$f^*(t) = \max_{x \in dom(f)} \{xt - f(x)\} \longleftrightarrow f(\underline{x}) = \max_{t \in dom(f^*)} \{\underline{x}t - f^*(t)\}$$

$$\boxed{\frac{p(x)}{q(x)}} \qquad \boxed{\frac{p(x)}{q(x)}}$$

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

$$= \int_x q(x) \left(\max_{t \in dom(f^*)} \left\{ \frac{p(x)}{q(x)} \underline{t} - f^*(\underline{t}) \right\} \right) dx$$

$$\approx \max_D \int_x p(x) D(x) dx - \int_x q(x) f^*(D(x)) dx$$

D is a function whose input is x, and output is t

$$D_f(P||Q) \geq \int_x q(x) \left(\frac{p(x)}{q(x)} \underline{D(x)} - f^*(\underline{D(x)}) \right) dx$$

$$= \int_x p(x) D(x) dx - \int_x q(x) f^*(D(x)) dx$$

Connection with GAN

$$D_f(P||Q) \approx \max_{\mathrm{D}} \int_x p(x)D(x)dx - \int_x q(x)f^*\big(D(x)\big)dx$$

$$= \max_{\mathrm{D}}\{E_{x\sim P}[D(x)] - E_{x\sim Q}\big[f^*\big(D(x)\big)\big]\}$$

Samples from P

Samples from Q

$$D_f(P_{data}||P_G) = \max_{\mathrm{D}}\{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}\big[f^*\big(D(x)\big)\big]\}$$

$$G^* = arg\min_G D_f(P_{data}||P_G)$$

Original GAN has different V(G,D)

$$= arg\min_G \max_D \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}\big[f^*\big(D(x)\big)\big]\}$$

$$= arg\min_G \max_D V(G,D)$$ familiar? ☺

$$D_f(P_{data}||P_G) = \max_{D}\{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[f^*(D(x))]\}$$

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

Name	Conjugate $f^*(t)$
Total variation	t
Kullback-Leibler (KL)	$\exp(t-1)$
Reverse KL	$-1-\log(-t)$
Pearson χ^2	$\frac{1}{4}t^2+t$
Neyman χ^2	$2-2\sqrt{1-t}$
Squared Hellinger	$\frac{t}{1-t}$
Jeffrey	$W(e^{1-t}) + \frac{1}{W(e^{1-t})} + t - 2$
Jensen-Shannon	$-\log(2-\exp(t))$
Jensen-Shannon-weighted	$(1-\pi)\log\frac{1-\pi}{1-\pi e^{t/\pi}}$
GAN	$-\log(1-\exp(t))$

Using the f-divergence you like ☺

https://arxiv.org/pdf/1606.00709.pdf

Tips for Improving GAN

Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, arXiv prepring, 2017

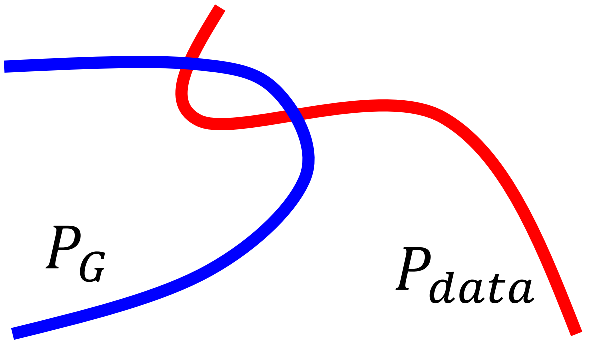
Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville, “Improved Training of Wasserstein GANs”, arXiv prepring, 2017

JS divergence is not suitable

- In most cases, P_G and P_{data} are not overlapped.
- 1. The nature of data

Both P_{data} and P_G are low-dim manifold in high-dim space.

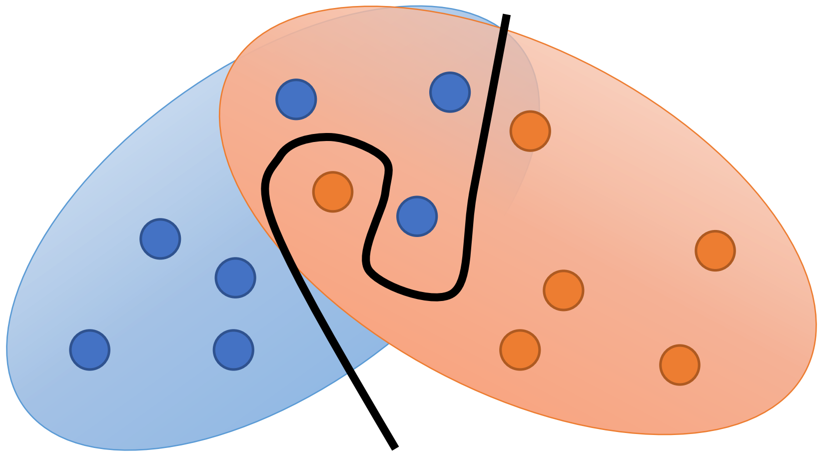
The overlap can be ignored.



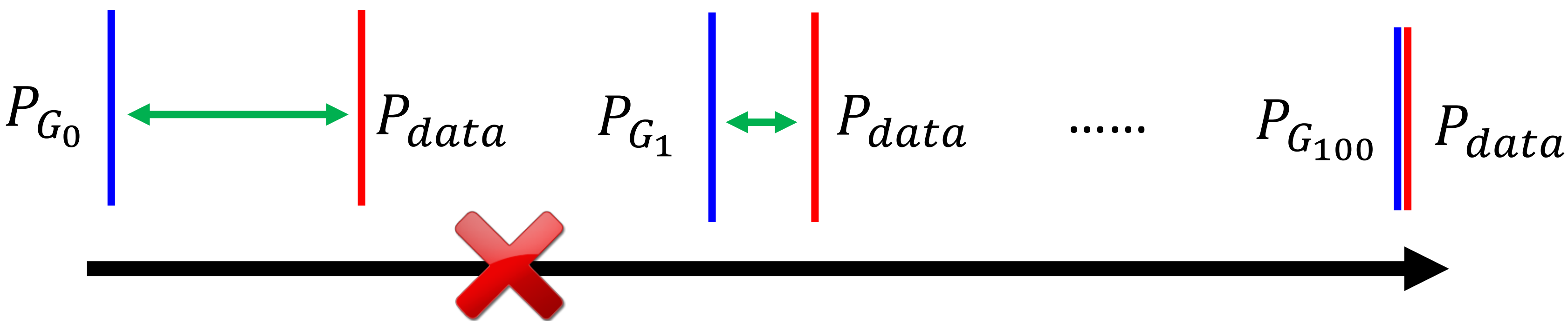
- 2. Sampling

Even though P_{data} and P_G have overlap.

If you do not have enough sampling ……



What is the problem of JS divergence?



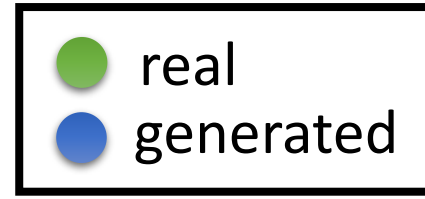
Equally bad

$$JS(P_{G_0}, P_{data}) = log2$$ $$JS(P_{G_1}, P_{data}) = log2$$ …… $$JS(P_{G_{100}}, P_{data}) = 0$$

JS divergence is log2 if two distributions do not overlap.

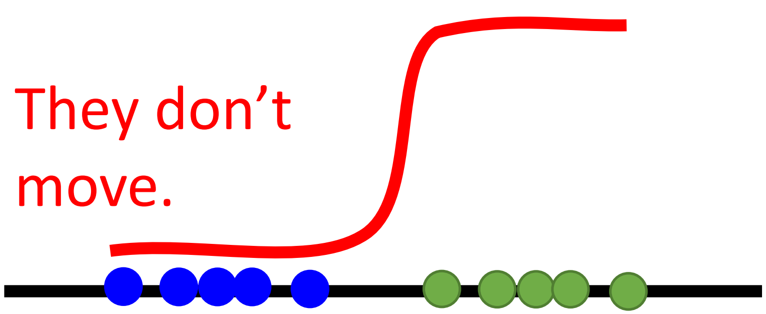
Intuition: If two distributions do not overlap, binary classifier achieves 100% accuracy

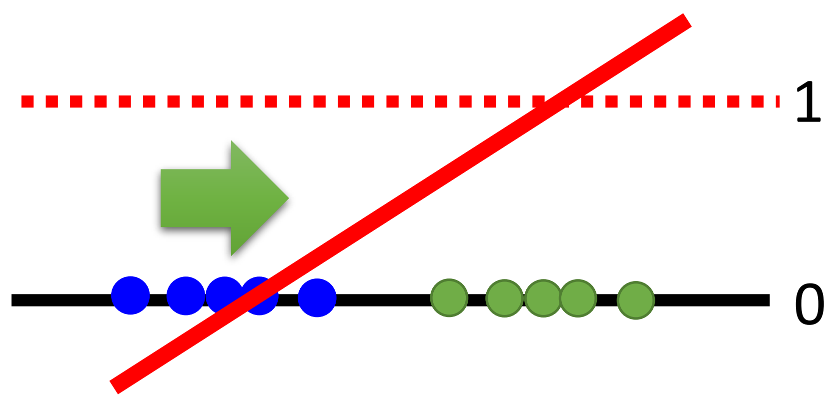
➡ Same objective value is obtained. ➡ Same divergence



Least Square GAN (LSGAN)

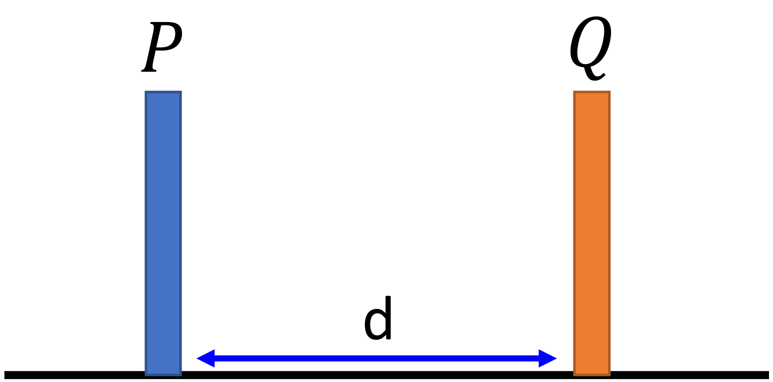
- Replace sigmoid with linear (replace classification with regression)





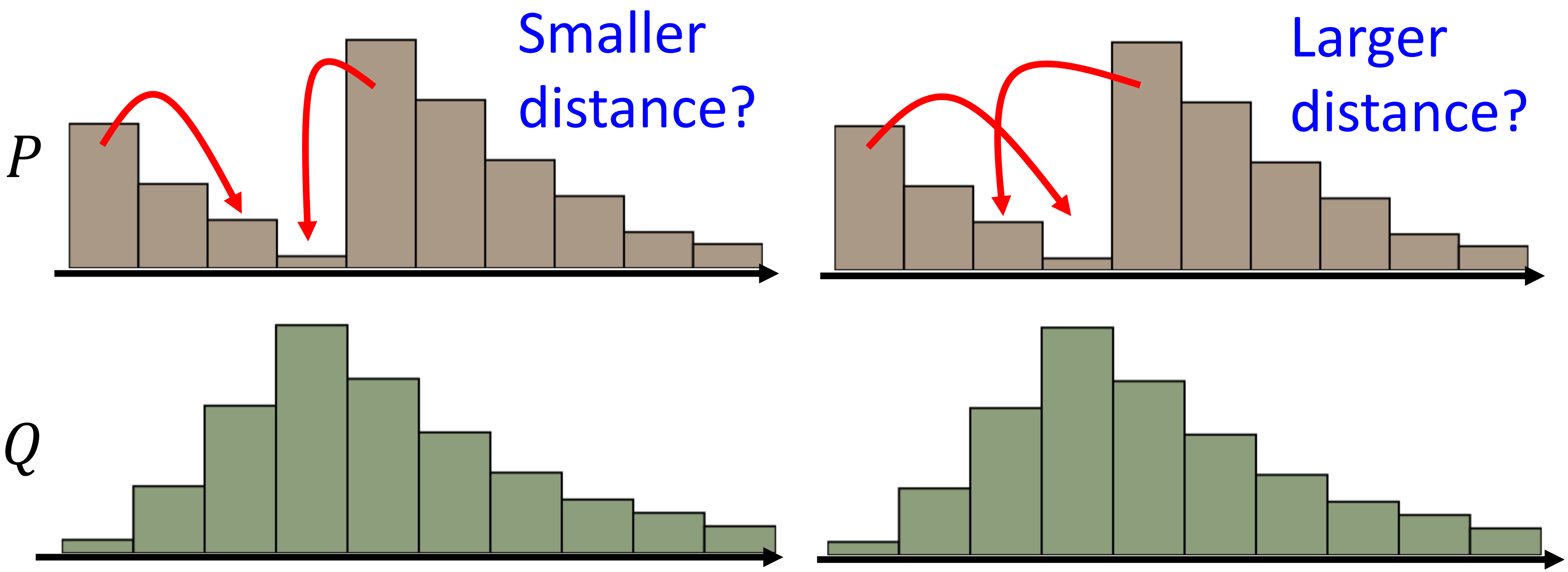
Wasserstein GAN (WGAN): Earth Mover's Distance

- Considering one distribution P as a pile of earth, and another distribution Q as the target
- The average distance the earth mover has to move the earth.



$$W(P, Q) = d$$

WGAN: Earth Mover's Distance

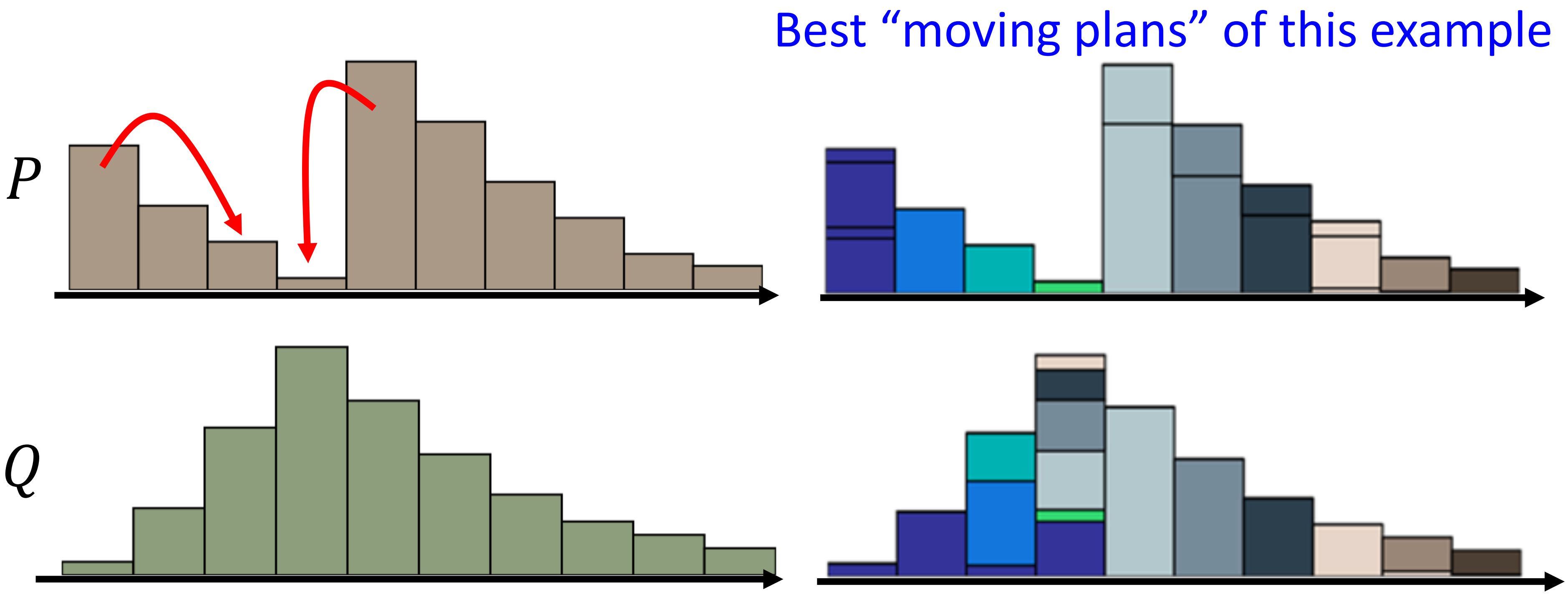


There many possible "moving plans".

Using the "moving plan" with the smallest average distance to define the earth mover's distance.

Source of image: https://vincentherrmann.github.io/blog/wasserstein/

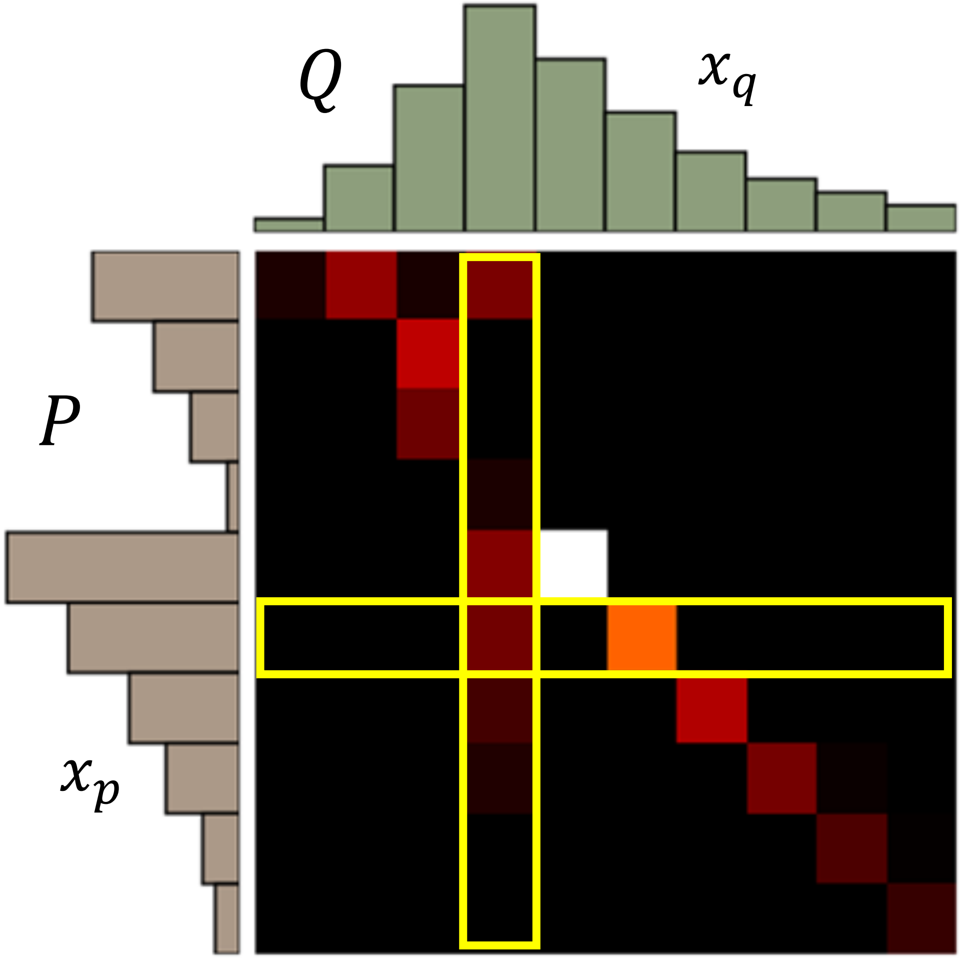
WGAN: Earth Mover's Distance



There many possible "moving plans".

Using the "moving plan" with the smallest average distance to define the earth mover's distance.

Source of image: https://vincentherrmann.github.io/blog/wasserstein/



moving plan γ

All possible plan Π

A "moving plan" is a matrix

The value of the element is the amount of earth from one position to another.

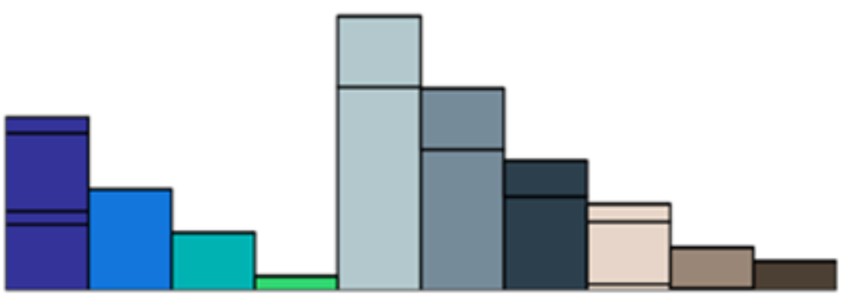
Average distance of a plan γ:

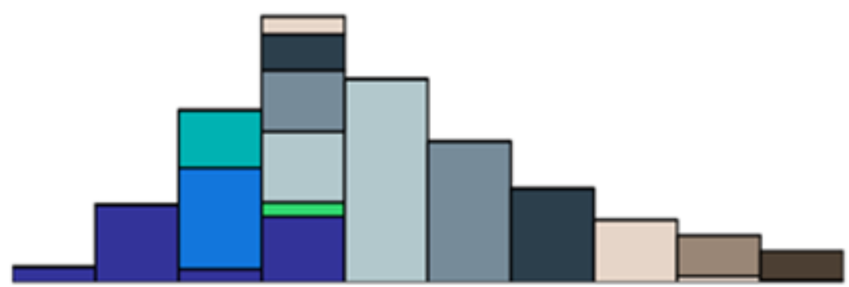
$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \| x_p - x_q \|$$

Earth Mover's Distance:

$$W(P, Q) = \min_{\gamma \in \Pi} B(\gamma)$$

The best plan

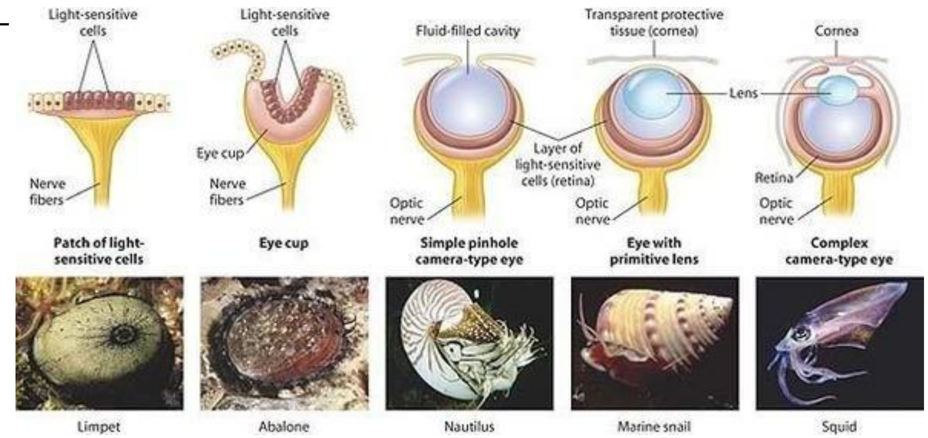


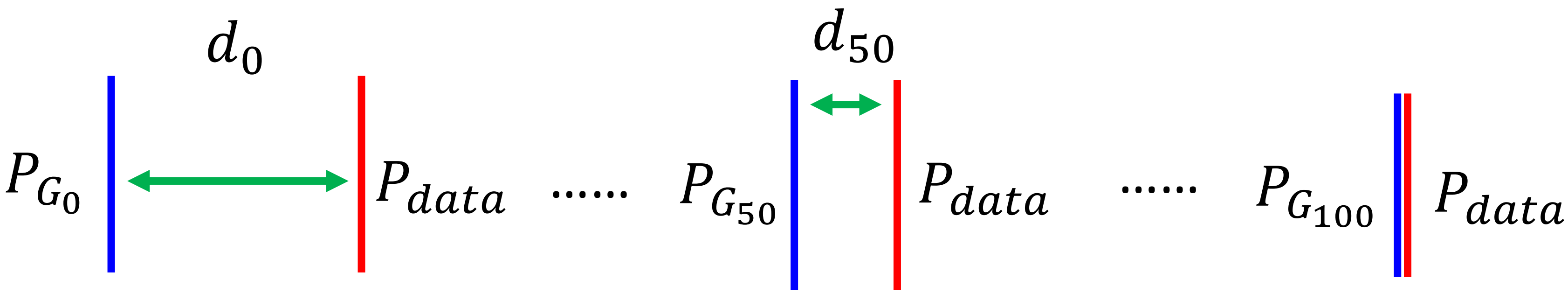


Why Earth Mover's Distance?

$$D_f(P_{data}||P_G)$$

$$W(P_{data}, P_G)$$





$$JS(P_{G_0}, P_{data}) = log2$$

$$JS(P_{G_{50}}, P_{data}) = log2$$

$$JS(P_{G_{100}}, P_{data}) = 0$$

$$W(P_{G_0}, P_{data}) = d_0$$

$$W(P_{G_{50}}, P_{data}) = d_{50}$$

$$W(P_{G_{100}}, P_{data}) = 0$$

[Martin Arjovsky, et al., arXiv, 2017]

WGAN

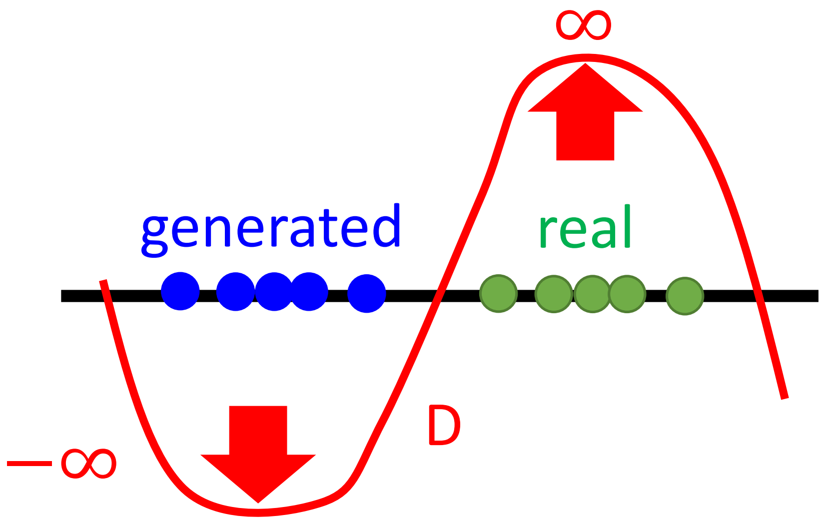
Evaluate wasserstein distance between P_{data} and P_G

$$V(G,D) = \max_{D \in 1-Lipschitz} \left\{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)] \right\}$$

D has to be smooth enough.

Without the constraint, the training of D will not converge.

Keeping the D smooth forces D(x) become ∞ and $-\infty$



WGAN

Weight Clipping [Martin Arjovsky, et al., arXiv, 2017]

Force the parameters w between c and -c

After parameter update, if w > c, w = c;

if w < -c, w = -c

Evaluate wasserstein distance between P_{data} and P_G

$$V(G,D) = \max_{D \in 1-Lipschitz} \{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)] \}$$

D has to be smooth enough. How to fulfill this constraint?

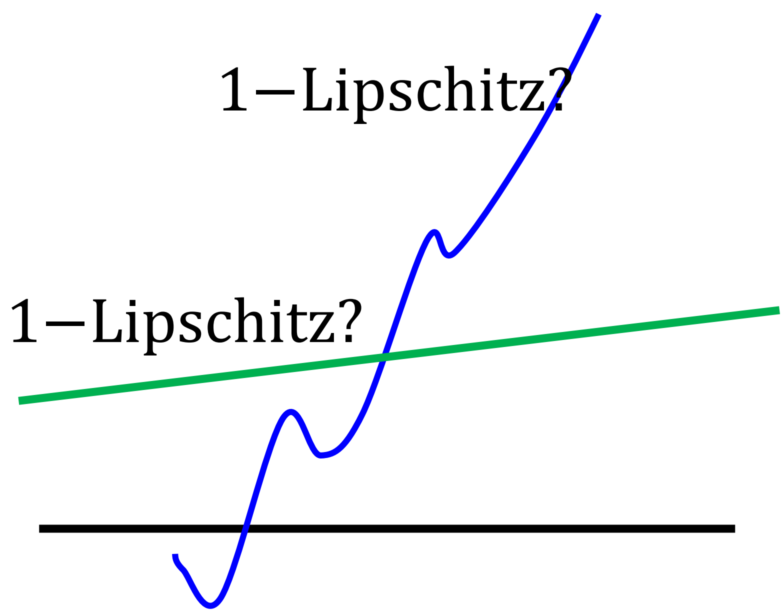
Lipschitz Function

$$\|f(x_1) - f(x_2)\| \leq K \|x_1 - x_2\|$$

Output change — Input change

K=1 for "$1 - Lipschitz$"

Do not change fast



Improved WGAN (WGAN-GP)

$$V(G,D) = \max_{D \in 1-Lipschitz} \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)]\}$$

A differentiable function is 1-Lipschitz if and only if it has gradients with norm less than or equal to 1 everywhere.

$D \in 1 - Lipschitz$ $\Leftrightarrow$ $\|\nabla_x D(x)\| \leq 1$ for all x

$$V(G,D) \approx \max_D \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)]$$

$$\cancel{-\lambda \int_x max(0, \|\nabla_x D(x)\| - 1)dx\}}$$

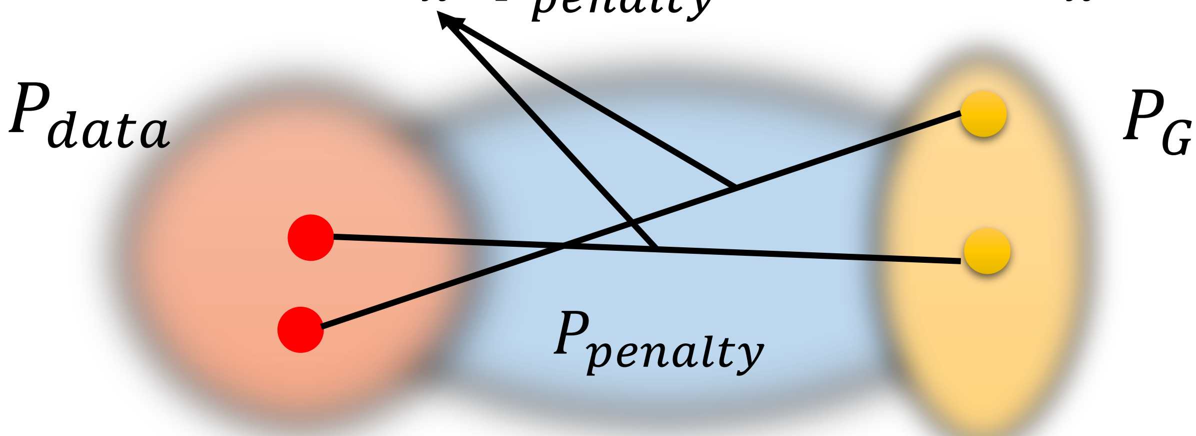
Prefer $\|\nabla_x D(x)\| \leq 1$ for all x

$$-\lambda E_{x\sim P_{penalty}}[max(0, \|\nabla_x D(x)\| - 1)]\}$$

Prefer $\|\nabla_x D(x)\| \leq 1$ for x sampling from $x \sim P_{penalty}$

Improved WGAN (WGAN-GP)

$$V(G,D) \approx \max_{D}\{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)] \\ -\lambda E_{x\sim P_{penalty}}[max(0, \|\nabla_x D(x)\| - 1)]\}$$



"Given that enforcing the Lipschitz constraint everywhere is intractable, enforcing it ***only along these straight lines*** seems sufficient and experimentally results in good performance."

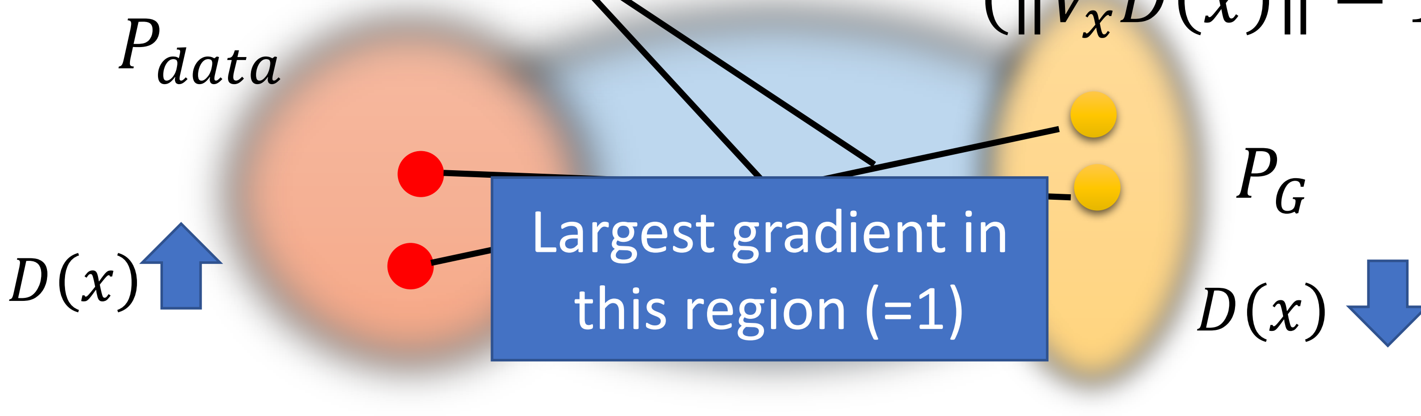
Only give gradient constraint to the region between P_{data} and P_G because they influence how P_G moves to P_{data}

Improved WGAN (WGAN-GP)

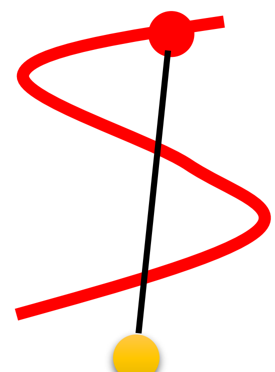
$$V(G,D) \approx \max_D \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)]$$

$$-\lambda E_{x\sim P_{penalty}}[\cancel{max(0, \|\nabla_x D(x)\| - 1)}]\}$$

$$(\|\nabla_x D(x)\| - 1)^2$$



"Simply penalizing overly large gradients also works in theory, but experimentally we found that this approach converged faster and to better optima."



Spectrum Norm

Spectral Normalization → Keep gradient norm smaller than 1 everywhere [Miyato, et al., ICLR, 2018]

Algorithm of WGAN

- In each training iteration:

No sigmoid for the output of D

Learning D

Repeat k times

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

Weight clipping / Gradient Penalty …

Learning G

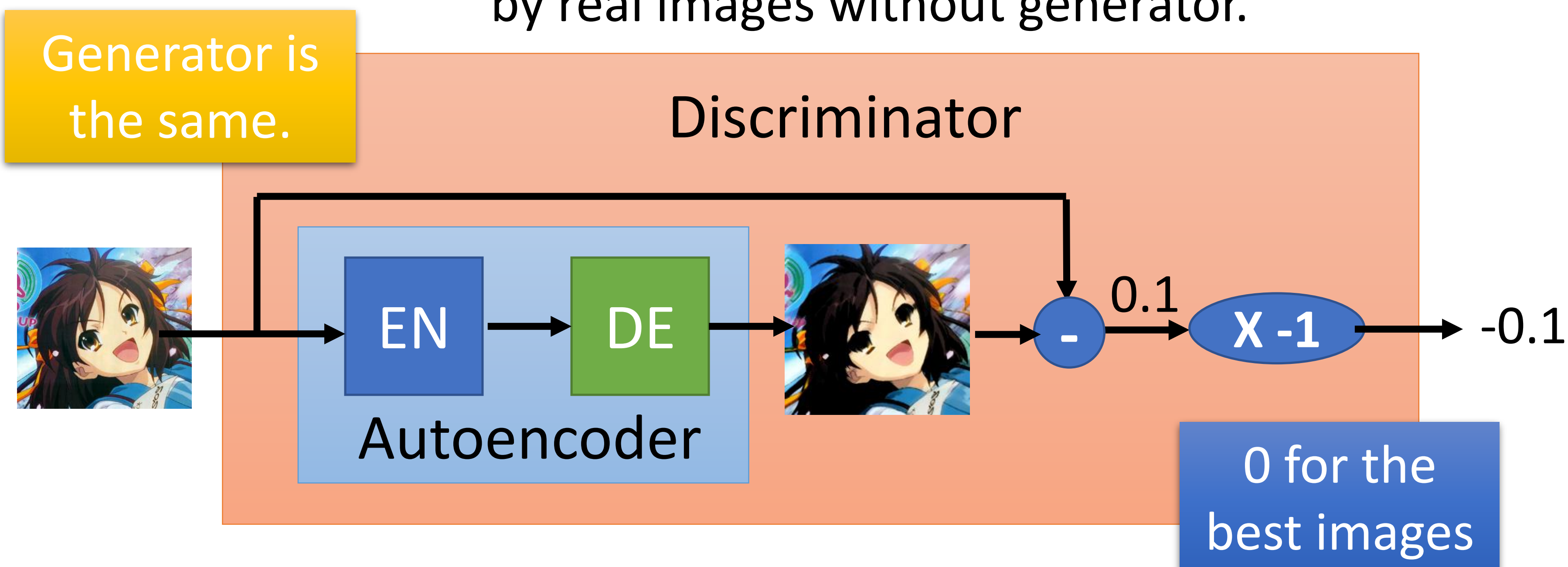
Only Once

- Sample another m noise s } from the prior $P_{prior}(z)$
- Update generator parameters θ_g to minimize
  - $\tilde{V} = \cancel{\frac{1}{m}\sum_{i=1}^{m} log D(x^i}$ $- \frac{1}{m}\sum_{i=1}^{m} D\left(G(z^i)\right)$
  - $\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$

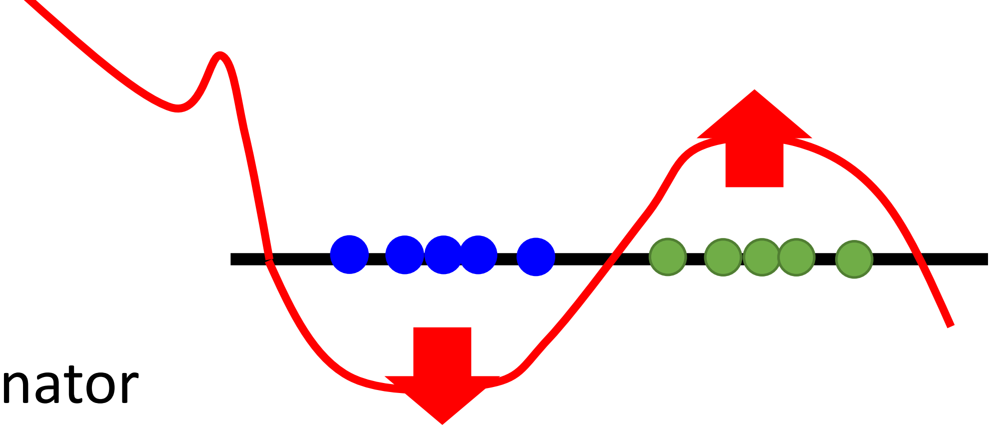
[Junbo Zhao, et al., arXiv, 2016]

Energy-based GAN (EBGAN)

- Using an autoencoder as discriminator D
  - ➢ Using the negative reconstruction error of auto-encoder to determine the goodness
  - ➢ **Benefit**: The auto-encoder can be pre-train by real images without generator.

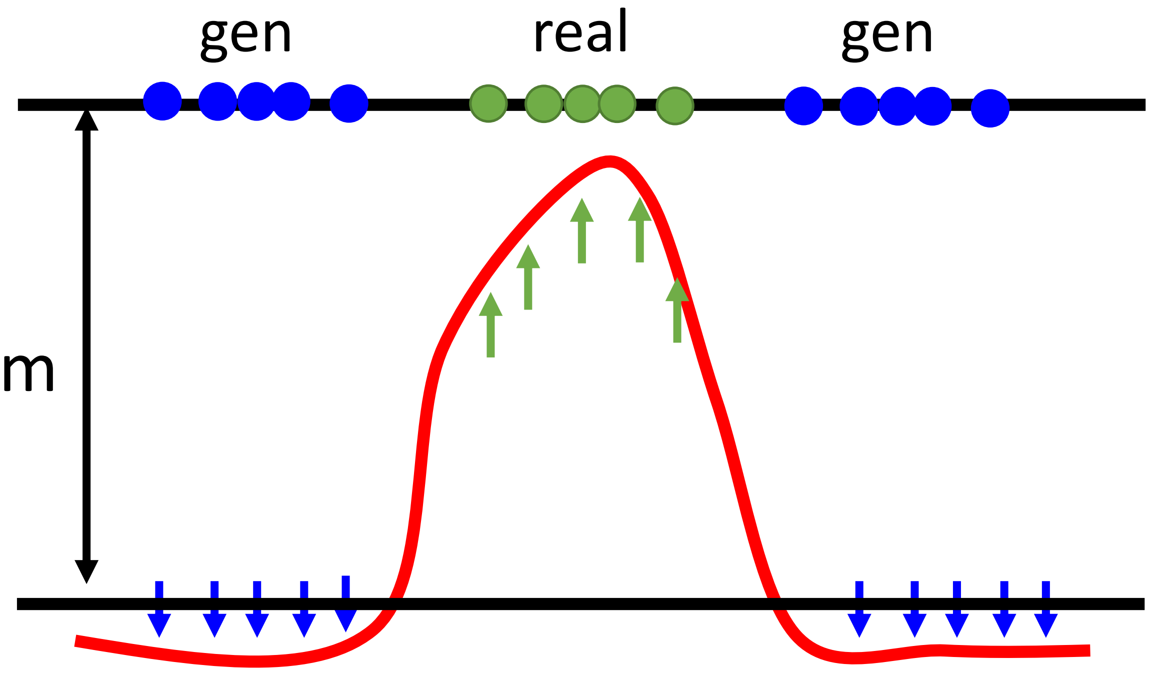


EBGAN



Auto-encoder based discriminator only gives limited region large value.

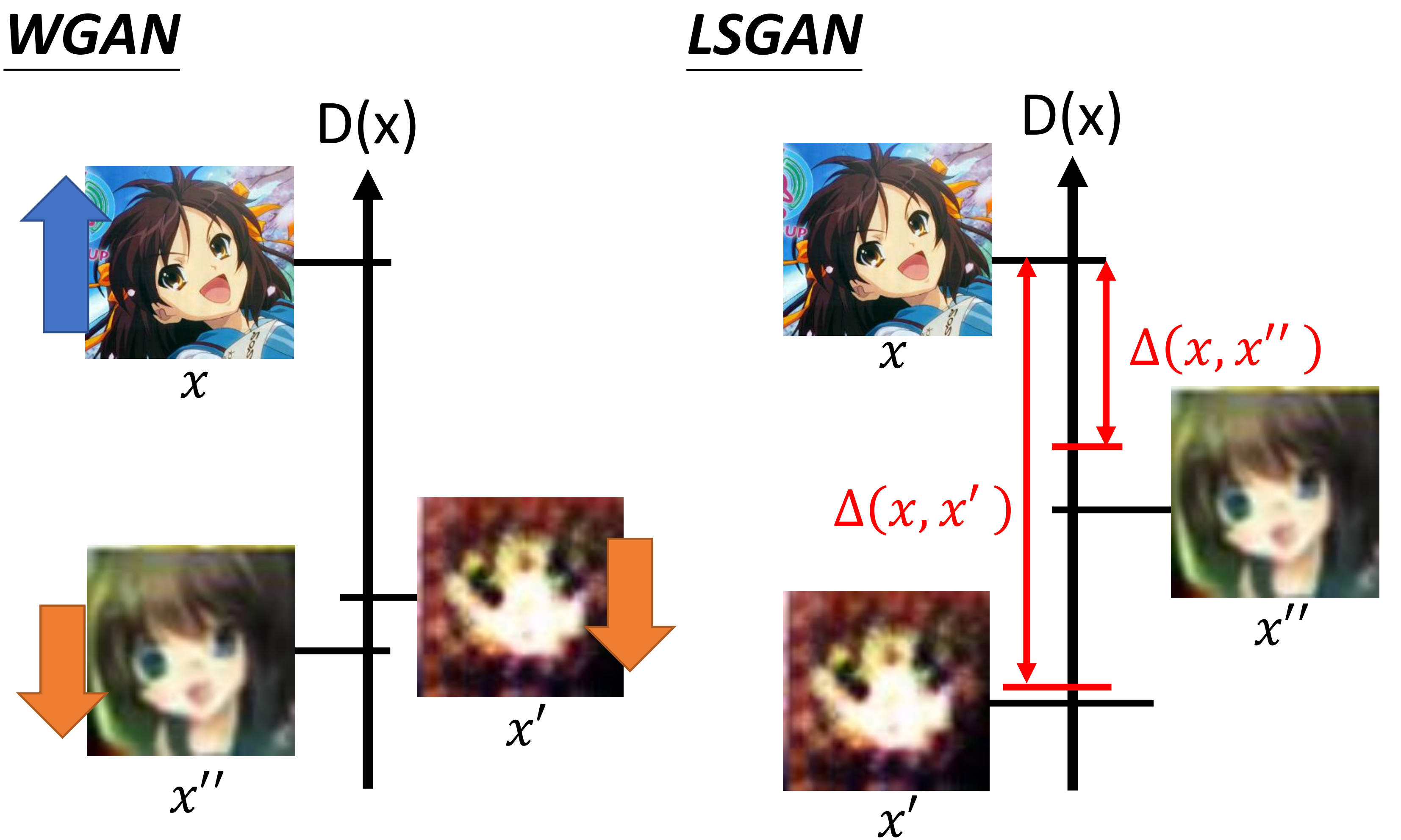
0 is for the best.



Do not have to be very negative

Hard to reconstruct, easy to destroy

Outlook:
Loss-sensitive GAN (LSGAN)



Reference

- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative Adversarial Networks, NIPS, 2014
- Sebastian Nowozin, Botond Cseke, Ryota Tomioka, “f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization”, NIPS, 2016
- Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, arXiv, 2017
- Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville, Improved Training of Wasserstein GANs, NIPS, 2017
- Junbo Zhao, Michael Mathieu, Yann LeCun, Energy-based Generative Adversarial Network, arXiv, 2016
- Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet, “Are GANs Created Equal? A Large-Scale Study”, arXiv, 2017
- Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen Improved Techniques for Training GANs, NIPS, 2016

Reference

- Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, NIPS, 2017
- Naveen Kodali, Jacob Abernethy, James Hays, Zsolt Kira, “On Convergence and Stability of GANs”, arXiv, 2017
- Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, Liqiang Wang, Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect, ICLR, 2018
- Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida, Spectral Normalization for Generative Adversarial Networks, ICLR, 2018