Theory behind GAN

Generation

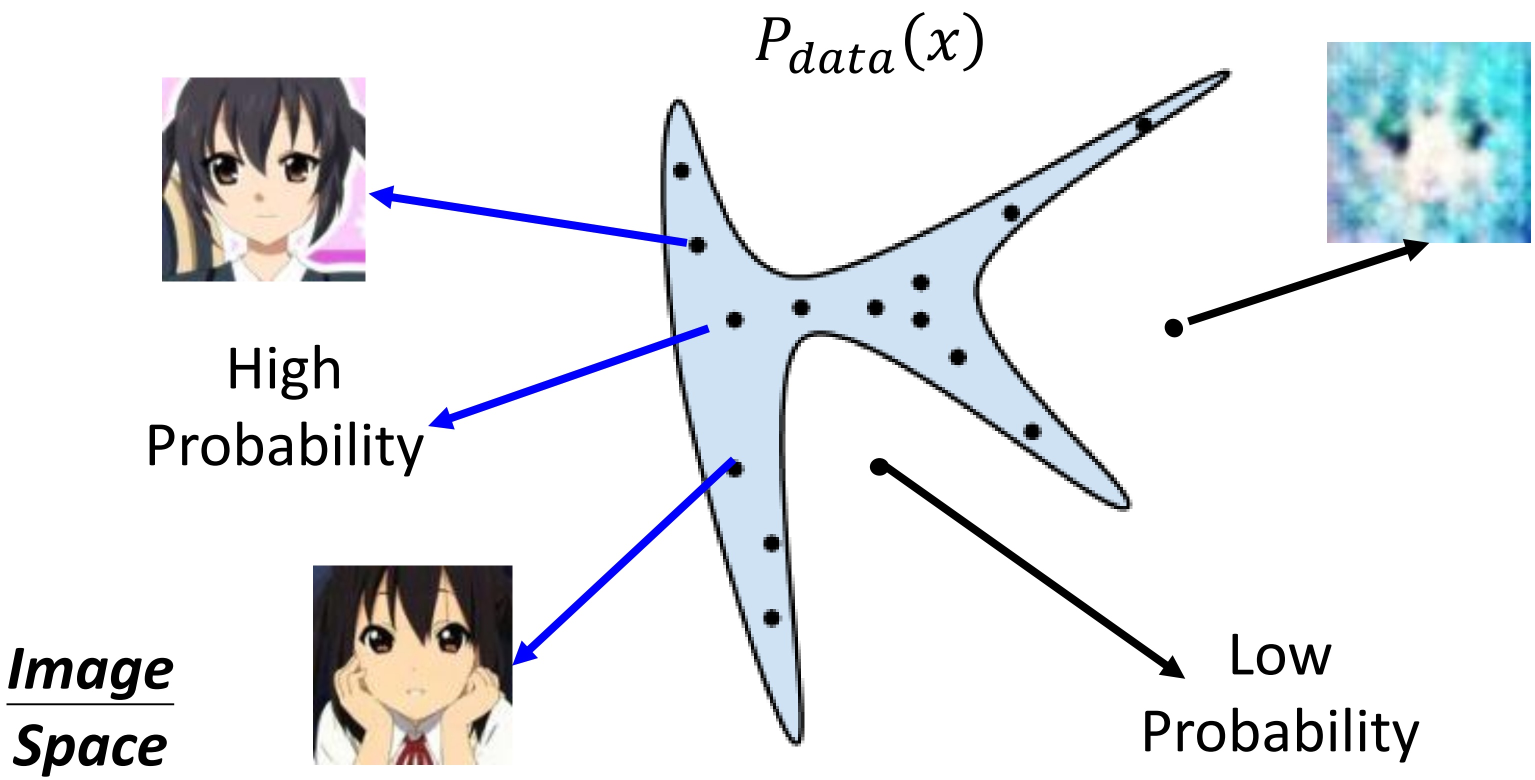
Using Generative Adversarial Network (GAN)

Drawing?

Generation

x: an image (a high-dimensional vector)

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



Maximum Likelihood Estimation

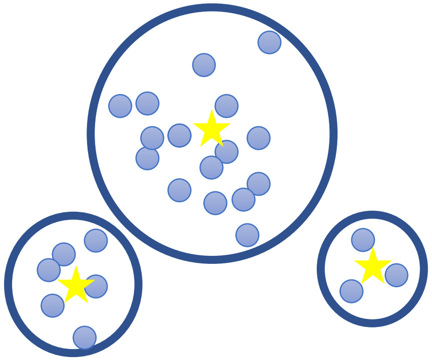
- Given a data distribution $P_{data}(x)$ (We can sample from it.)
- We have a distribution $P_G(x;\theta)$ parameterized by θ
  - We want to find θ such that $P_G(x;\theta)$ close to $P_{data}(x)$
  - E.g. $P_G(x;\theta)$ is a Gaussian Mixture Model, θ are means and variances of the Gaussians

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

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

Likelihood of generating the samples

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



Find θ^* maximizing the likelihood

Maximum Likelihood Estimation = Minimize KL Divergence

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

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

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

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

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

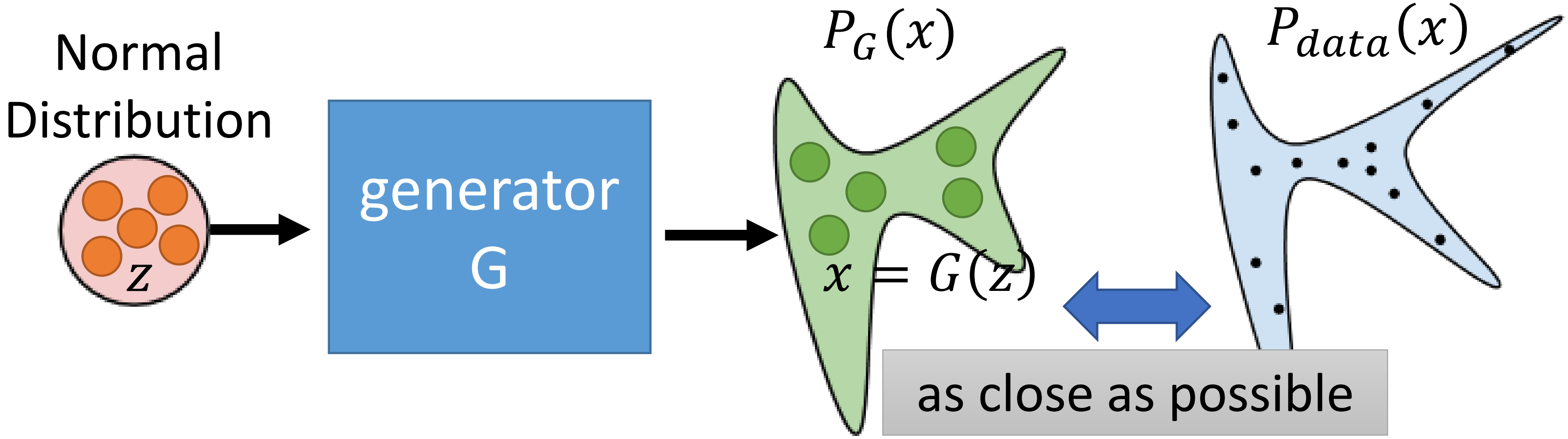
$$= arg \min_\theta KL(P_{data}||P_G)$$

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



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

Divergence between distributions P_G and P_{data}

How to compute the divergence?

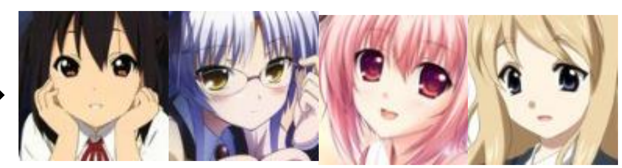
Discriminator

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

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

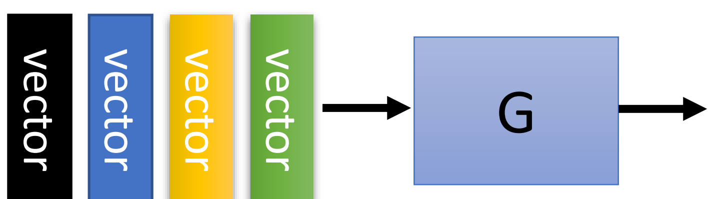
Database

sample



Sampling from P_{data}

sample from normal



Sampling from P_G

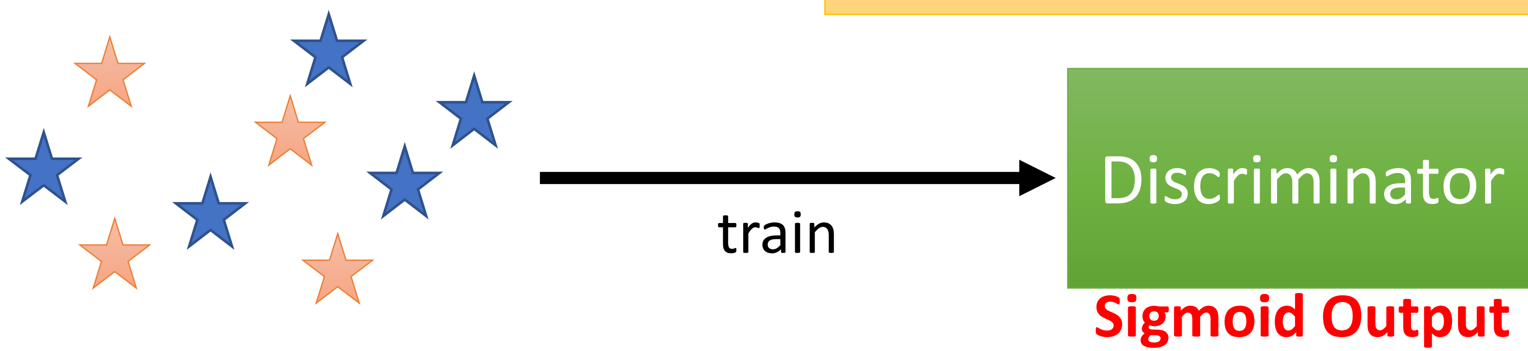
Discriminator

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

★ : data sampled from P_{data}

☆ : data sampled from P_G

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



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

The maximum objective value is related to JS divergence.

[Goodfellow, et al., NIPS, 2014]

Discriminator

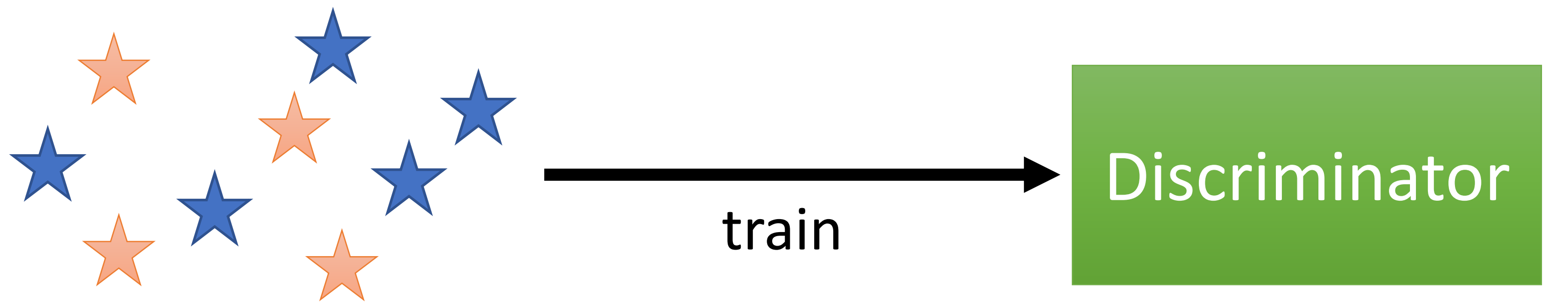
$$G^* = arg \min_G Div(P_G, P_{data})$$

 : data sampled from P_{data}

 : data sampled from P_G

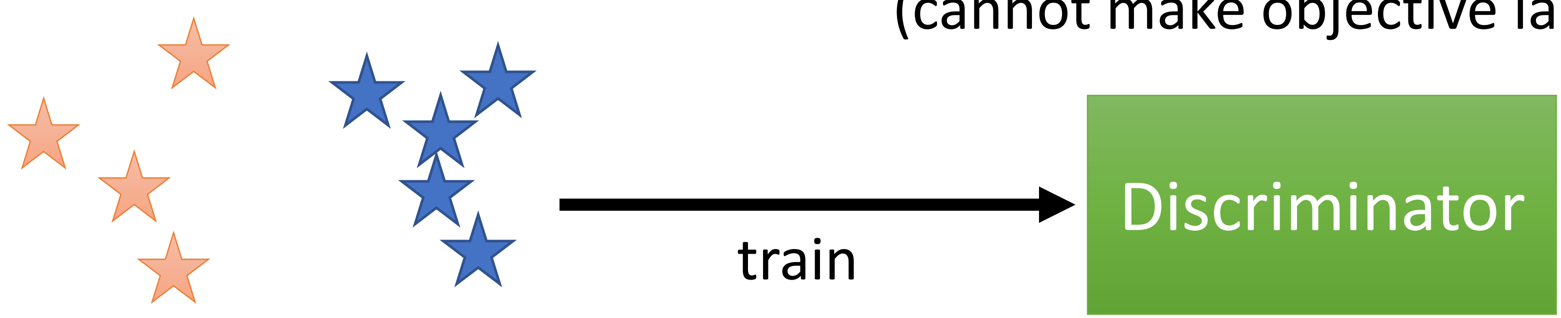
Training:

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



small divergence

hard to discriminate
(cannot make objective large)



large divergence

easy to discriminate

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

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

- Given 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

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

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

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

- Given x, the optimal D* maximizing

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

a D b D

- Find D* maximizing: $f(D) = alog(D) + blog(1-D)$

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

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

$$a \times (1-D^*) = b \times D^*$$

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

$$D^* = \frac{a}{a+b}$$

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

$0 <$ ... < 1

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

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

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

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

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

$$+2log\frac{1}{2} \quad -2log2$$

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

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

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

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

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

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

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

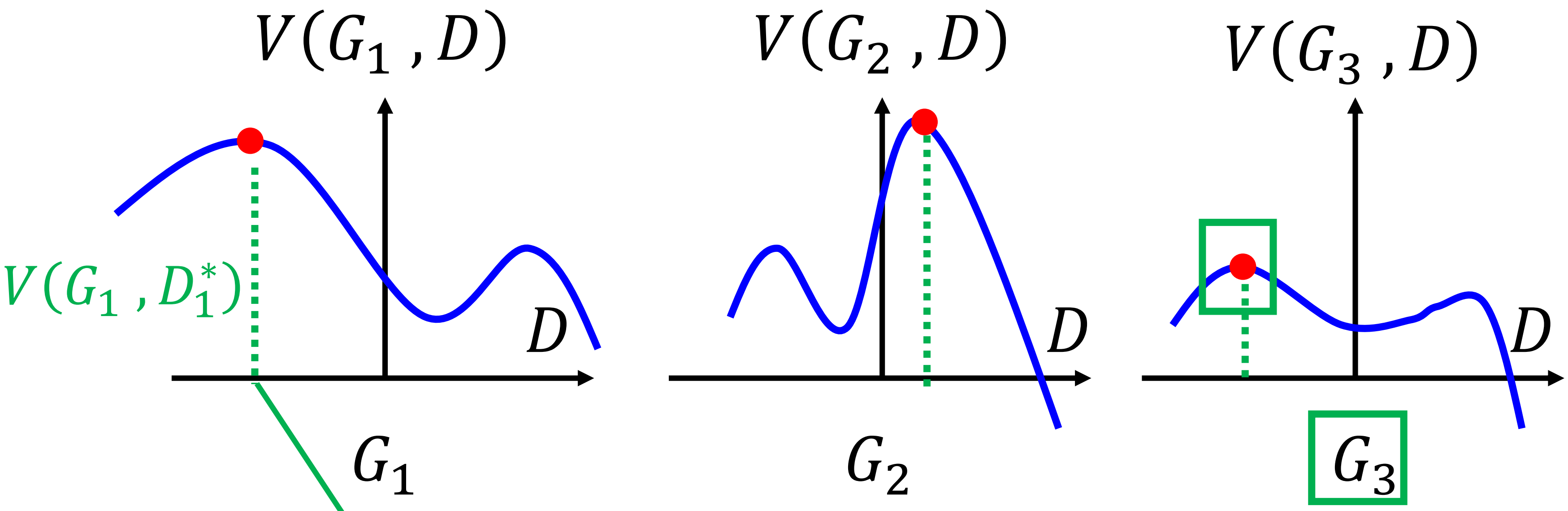
$$= -2log2 + 2JSD(P_{data}||P_G)$$

Jensen-Shannon divergence

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

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

The maximum objective value is related to JS divergence.



Divergence between P_{G_1} and P_{data}

[Goodfellow, et al., NIPS, 2014]

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

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

The maximum objective value is related to JS divergence.

- Initialize generator and discriminator
- In each training iteration:

Step 1: Fix generator G, and update discriminator D

Step 2: Fix discriminator D, and update generator G

Algorithm

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

$L(G)$

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

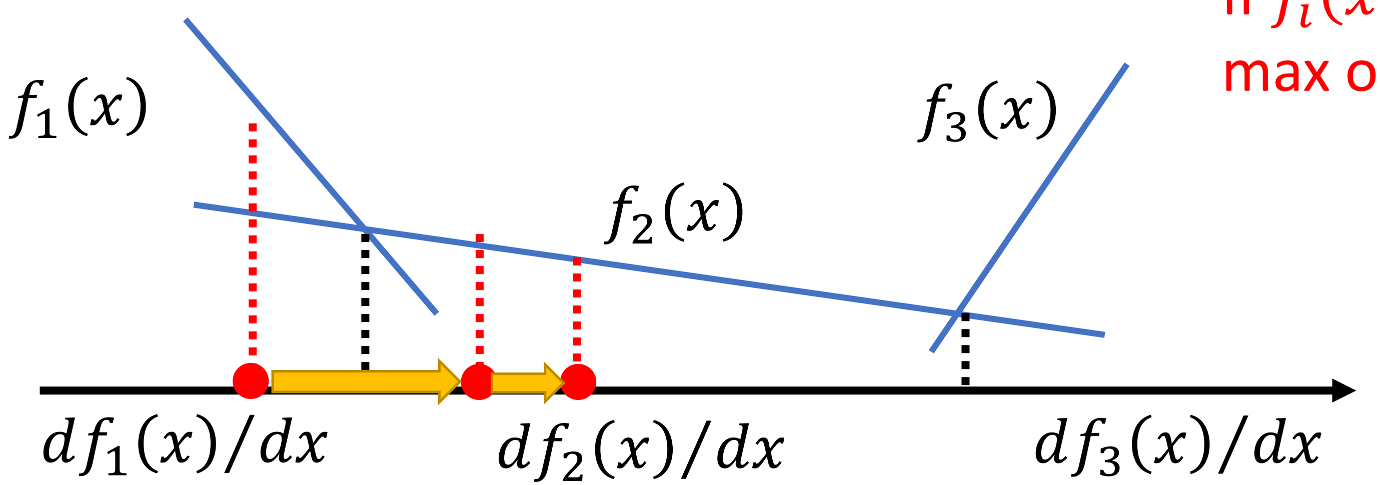
$$\theta_G \leftarrow \theta_G - \eta \, \partial L(G) / \partial \theta_G$$

θ_G defines G

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

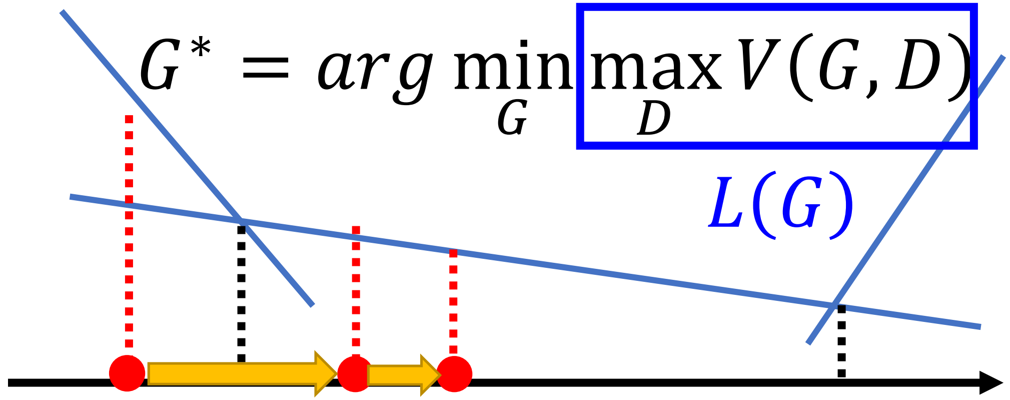
$$\frac{df(x)}{dx} =? \; df_i(x)/dx$$

If $f_i(x)$ is the max one



Algorithm

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



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

Decrease JS divergence(?)

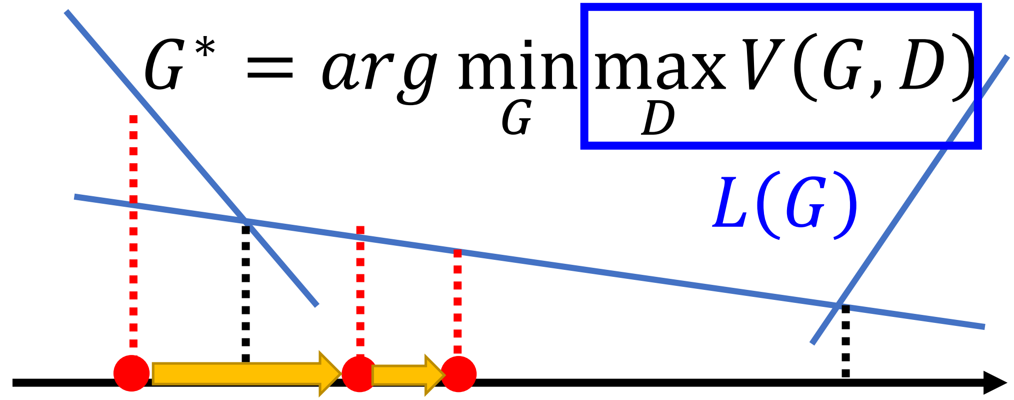
$V(G_1, D_1^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_1}(x)$

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

Decrease JS divergence(?)

Algorithm

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

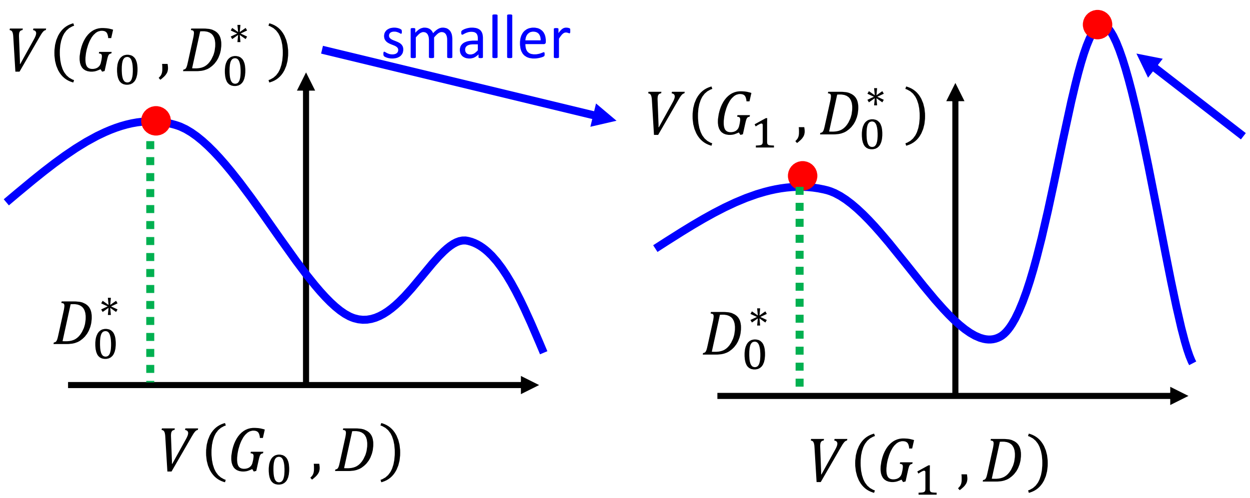


- Given G_0
- Find D_0^* maximizing $V(G_0, D)$

$V(G_0, D_0^*)$ 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

- In each training iteration:

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

Learning D

Repeat k times

- Sample m examples $\{x^1, x^2, \ldots, x^m\}$ from data distribution $P_{data}(x)$
- Sample m noise samples $\{z^1, z^2, \ldots, z^m\}$ from the prior $P_{prior}(z)$
- Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
- 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

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
  - $\tilde{V} = ~~\frac{1}{m}\sum_{i=1}^{m} logD(x^i)~~ + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D\left(G(z^i)\right)\right)$
  - $\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$

Objective Function for Generator in Real Implementation

$$V = E_{x \sim P_{data}}[log D(x)]$$

$$+E_{x \sim P_G}\left[log\left(1 - D(x)\right)\right]$$

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)