

Theory behind GAN

Generation

Using Generative
Adversarial
Network (GAN)

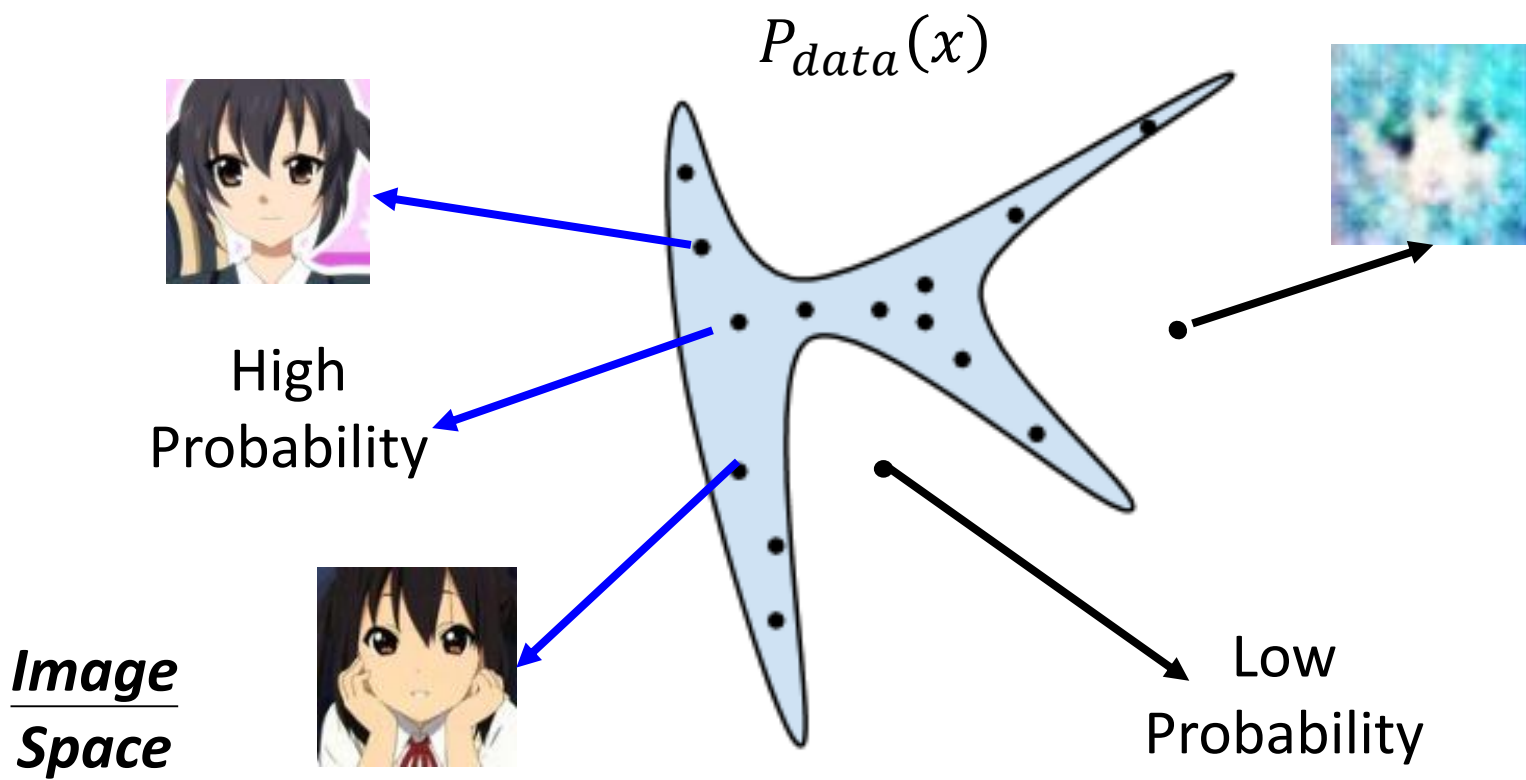


Drawing?

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 $P_{data}(x)$

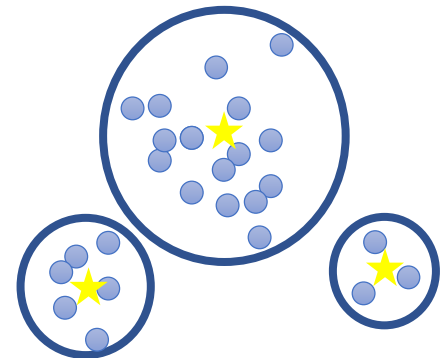
We can compute $P_G(x^i; \theta)$ 假設給定參數 θ ，則可以知道 x^i 的機率

Likelihood of generating the samples

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

找出 θ 使得likelihood最大

Find θ^* maximizing the likelihood



Maximum Likelihood Estimation = Minimize KL Divergence

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

如果是很複雜的distribution，則很難去估測它的probability

$$= \arg \max_{\theta} \sum_{i=1}^m \log P_G(x^i; \theta) \quad \{x^1, x^2, \dots, x^m\} \text{ from } P_{data}(x)$$

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

外加一項，由於與PG無關，因此不影響結果

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

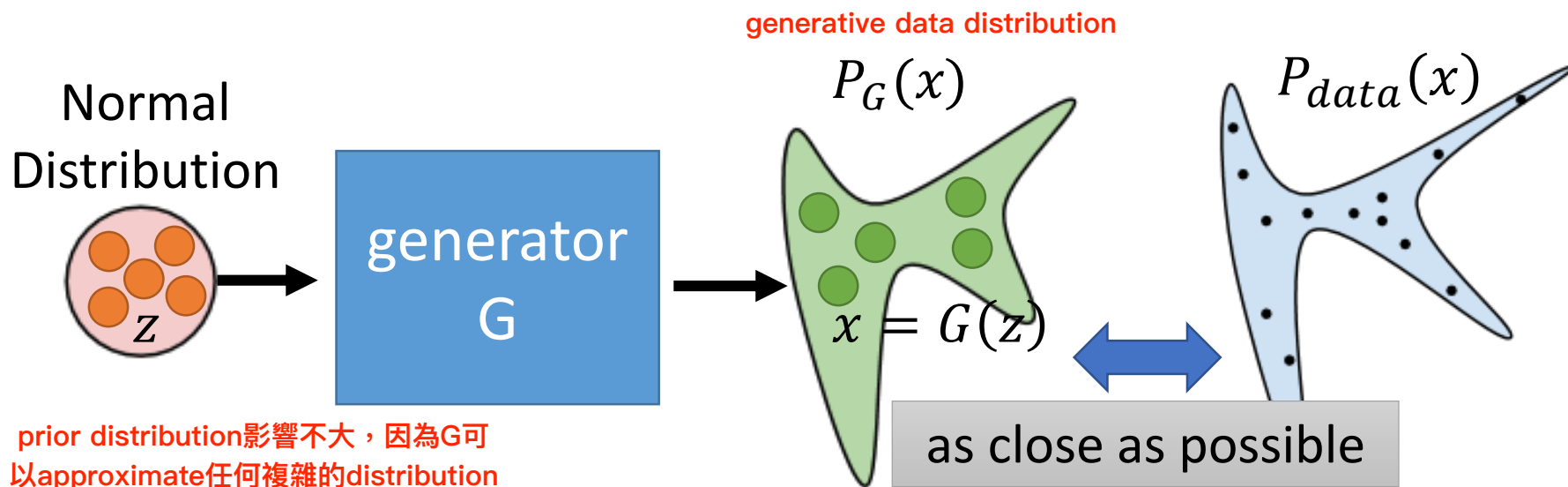
$$= \arg \min_{\theta} KL(P_{data} || P_G)$$

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



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

怎麼計算出這個 divergence? P_{data} , P_G 的 formulation 都是不知道呀

Divergence between distributions P_G and P_{data}

How to compute the divergence?

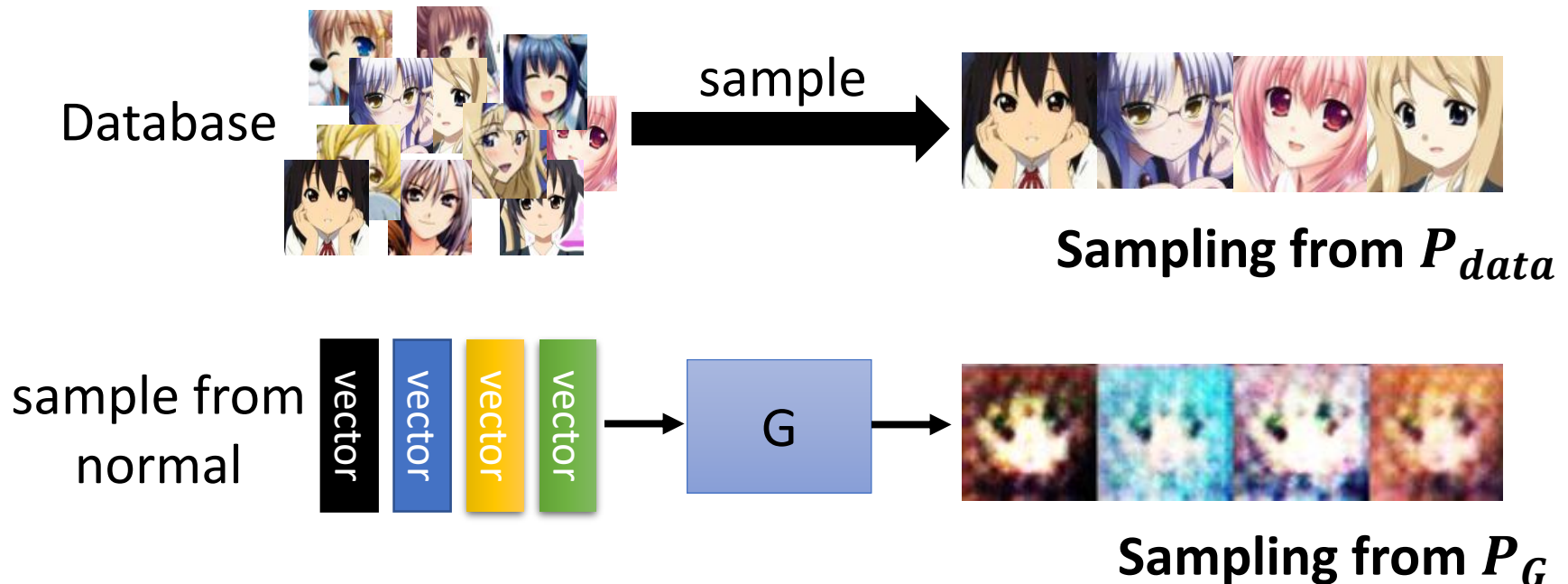
前提是不知道PG, Pdata的distribution長什麼樣子，但是我們可以sample他們的data

Discriminator

如何根據sample計算divergence呢

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

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



假設現在sample的data靠得很近，對binary classifier來說很難區別兩種data的不同
意味著兩種data很靠近，divergence應該是要小的

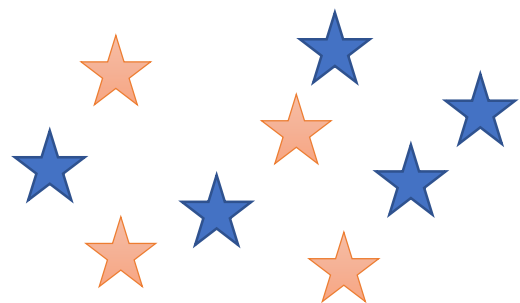
Discriminator $G^* = \arg \min_G \text{Div}(P_G, P_{data})$

★ : data sampled from P_{data}

★ : data sampled from P_G

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

就像是classifier minimize cross-entropy



Discriminator

Sigmoid Output

Example Objective Function for D

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

(G is fixed)

JS divergence

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

The maximum objective value is related to JS divergence.

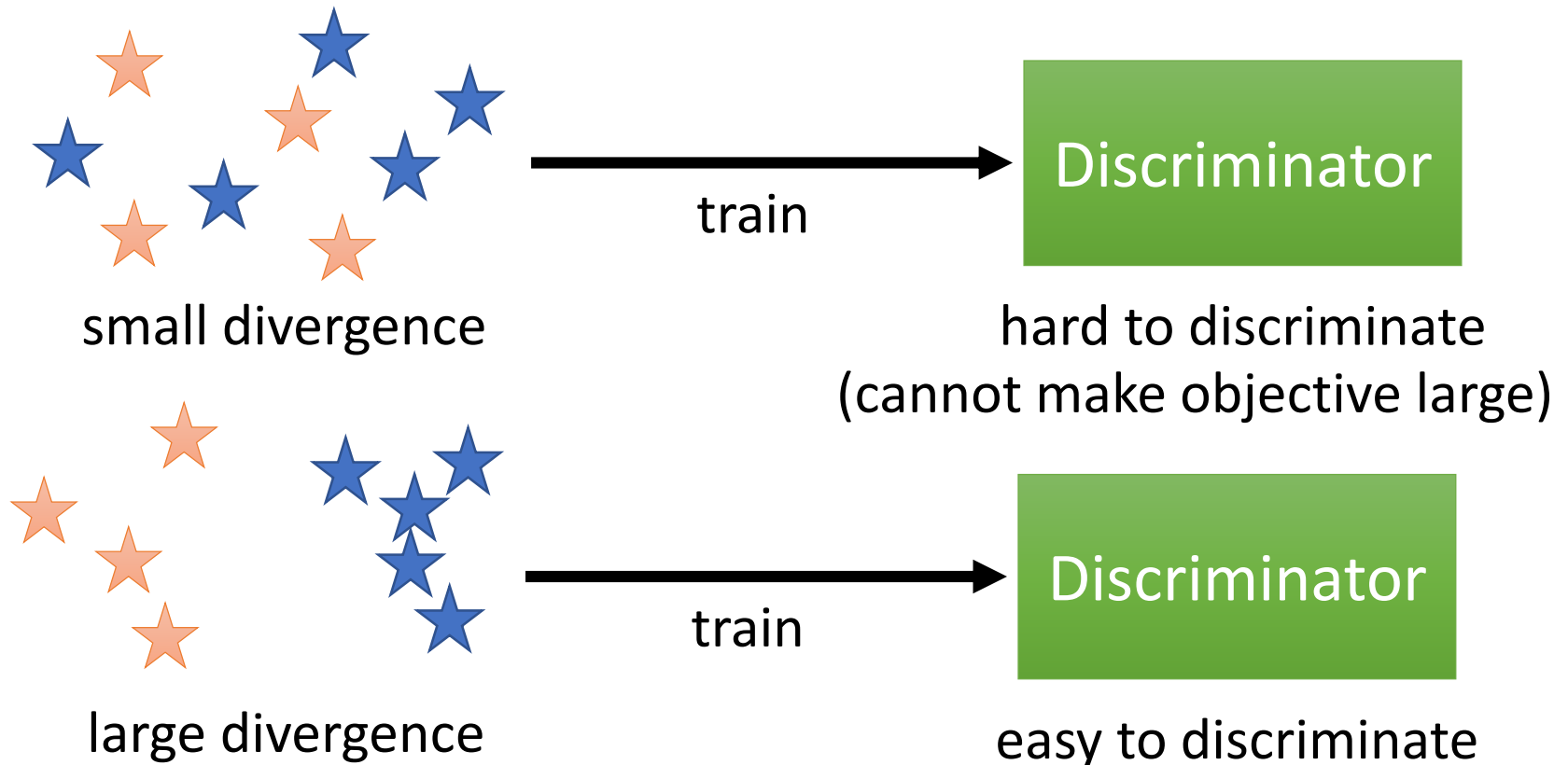
Discriminator $G^* = \arg \min_G \text{Div}(P_G, P_{data})$

★ : data sampled from P_{data}

★ : data sampled from P_G

Training:

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



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

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

- Given G, what is the optimal D^* maximizing

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

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

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

Assume that $D(x)$ can be any function

前提是neuron是無窮多個

- Given x , the optimal D^* maximizing

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

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

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

- Given x , the optimal D^* maximizing

$$\underbrace{P_{data}(x)}_a \log \underbrace{D(x)}_D + \underbrace{P_G(x)}_b \log \underbrace{(1 - D(x))}_D$$

- Find D^* maximizing: $f(D) = a \log(D) + b \log(1 - D)$

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

首先找critical point

$$a \times \frac{1}{D^*} = b \times \frac{1}{1 - D^*} \quad a \times (1 - D^*) = b \times D^* \\ a - aD^* = bD^* \quad a = (a + b)D^*$$

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



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

local maximum

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

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

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

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

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

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

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

整理成JS divergence，這邊不影響結果

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

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

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

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

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

$$\begin{aligned} = -2\log 2 + \int_x P_{data}(x) \log \frac{P_{data}(x)}{(P_{data}(x) + P_G(x))/2} dx \\ + \int_x P_G(x) \log \frac{P_G(x)}{(P_{data}(x) + P_G(x))/2} dx \end{aligned}$$

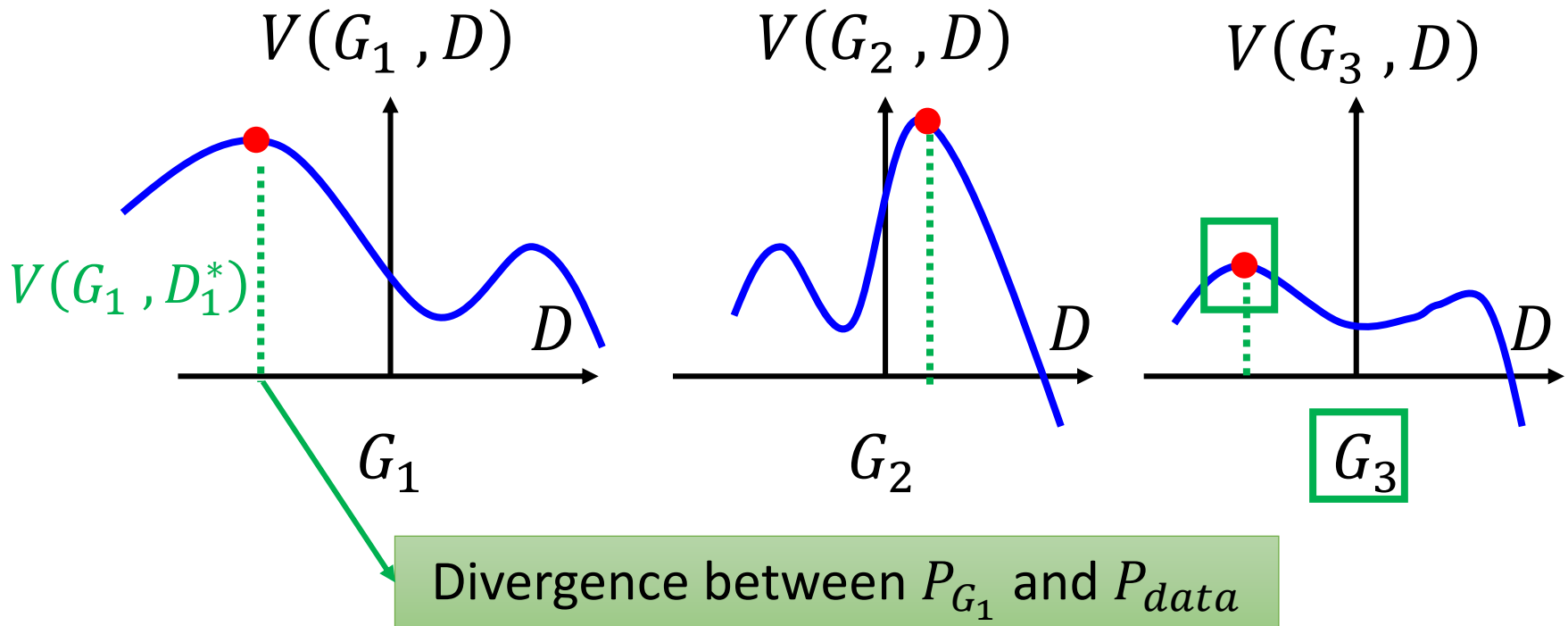
$$= -2\log 2 + \text{KL}\left(P_{data} \parallel \frac{P_{data} + P_G}{2}\right) + \text{KL}\left(P_G \parallel \frac{P_{data} + P_G}{2}\right)$$

$$= -2\log 2 + 2\text{JSD}(P_{data} \parallel P_G) \quad \text{Jensen-Shannon divergence}$$

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

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

The maximum objective value is related to JS divergence.



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

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

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

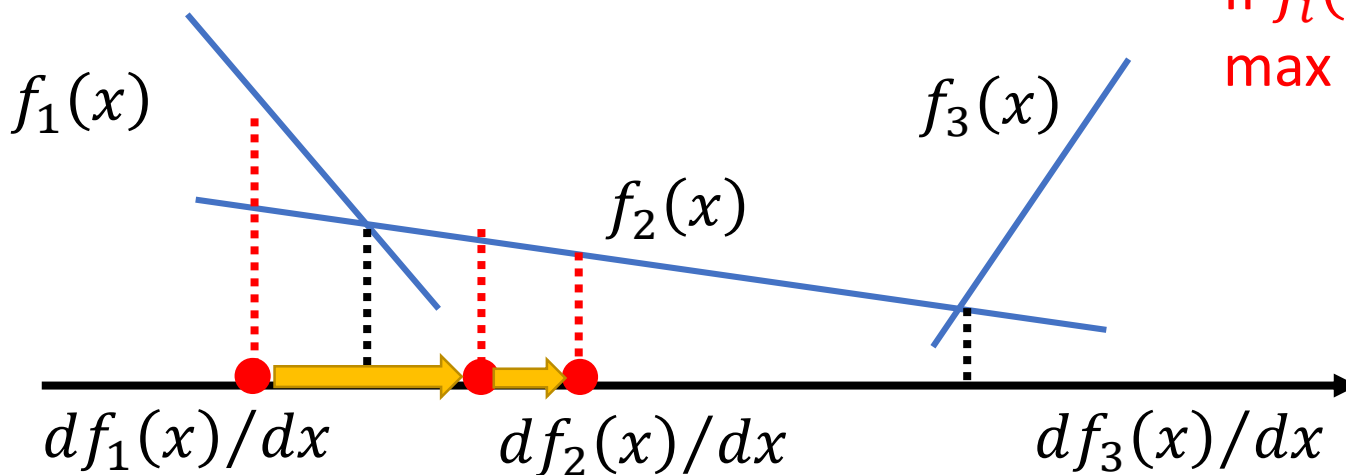
max可不可以微分？回想maxout network

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

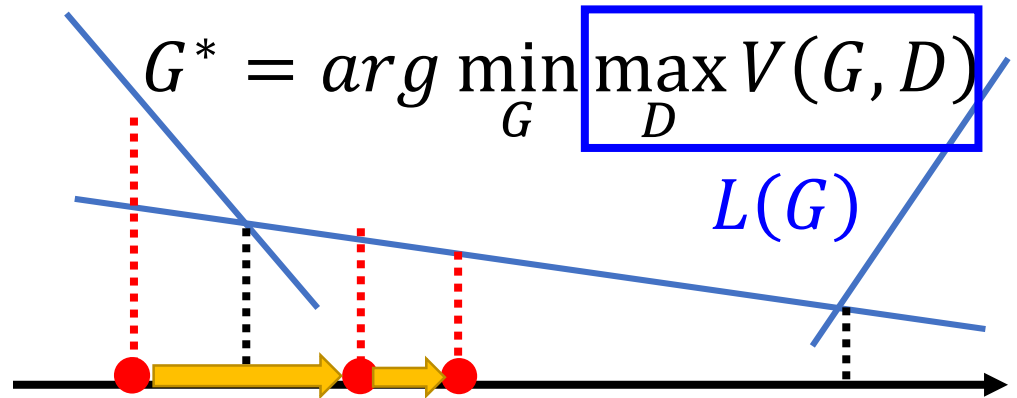
一樣是可以微分的

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

If $f_i(x)$ is the
max one



Algorithm



- Given G_0

- Find D_0^* maximizing $V(G_0, D)$ find JS divergence **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 $V(G_1, D)$ find JS divergence

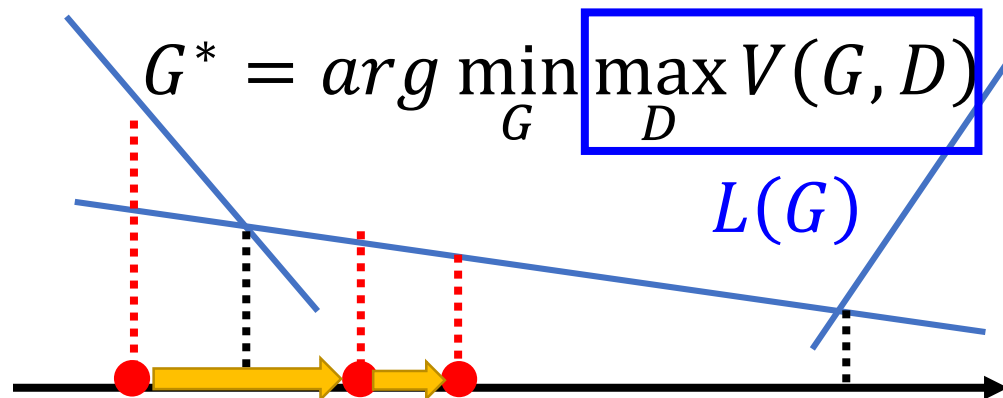
$V(G_1, D_1^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_1}(x)$

- $\theta_G \leftarrow \theta_G - \eta \partial V(G, D_1^*) / \partial \theta_G$ **Obtain G_2** **Decrease JS divergence(?)**
-

Algorithm

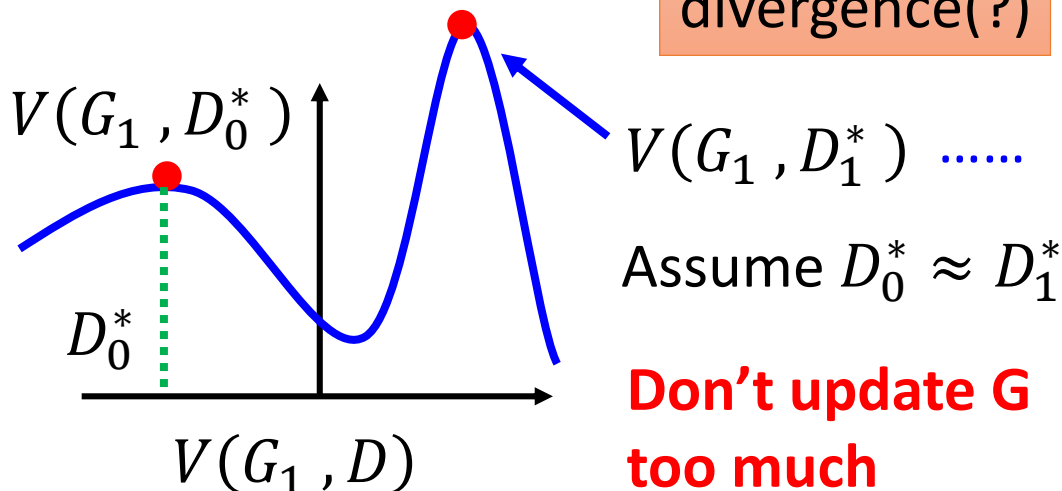
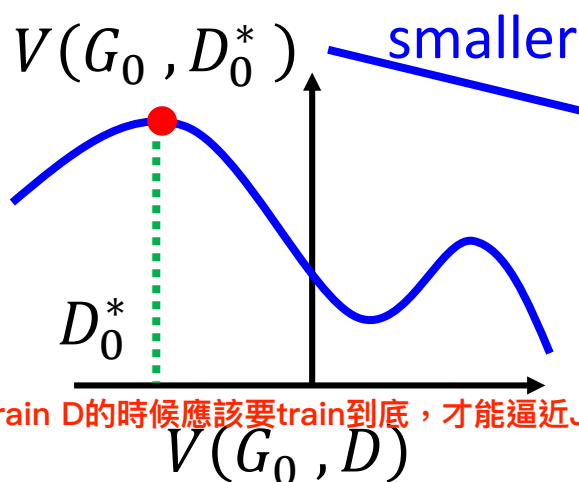
- Given G_0
- Find D_0^* maximizing $V(G_0, D)$

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



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

Decrease JS divergence(?)



在train D的時候應該要train到底，才能逼近JS

Don't update G too much

G update後 maxD可能會有變化，因此G不要update太多

假設：update G後objective function差異不大

In practice ...

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


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

II

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 bound of

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

- In each training iteration:

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
 - $\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i))$
因為要考慮到model capacity, 可能會卡在local minimum
 - $\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$
這邊沒辦法真實算出JS divergence, 頂多只能找到lower bound

Learning
G

Only
Once

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

Objective Function for Generator in Real Implementation

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

Slow at the beginning

Minimax GAN (MMGAN)

收斂較快

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

Real implementation:

label x from P_G as positive

Non-saturating GAN (NSGAN)

