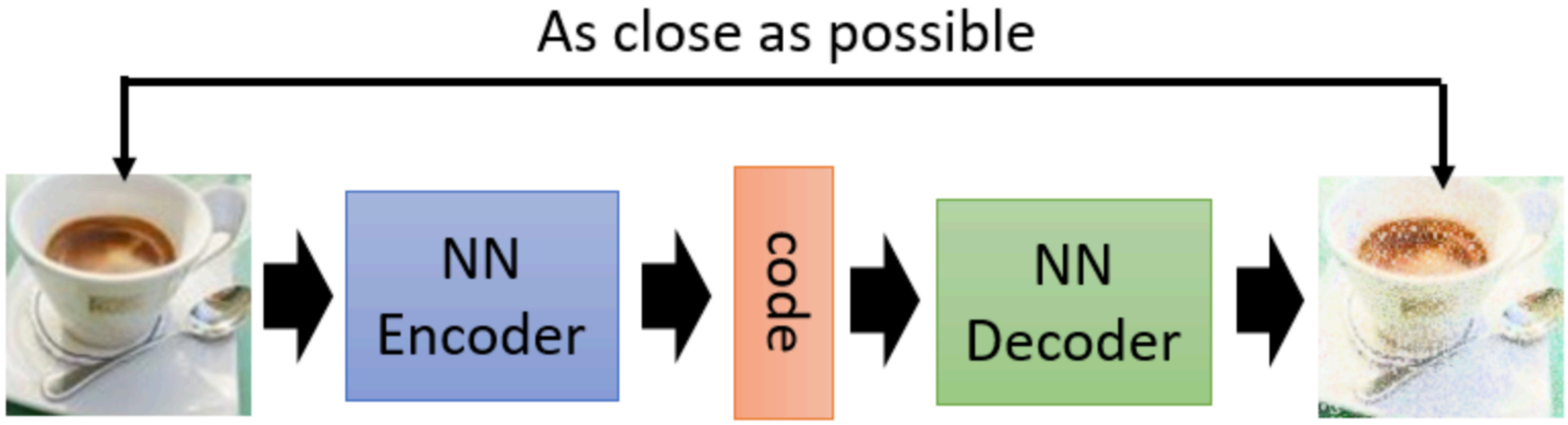
Generative Adversarial Networks (GANs)

宋宇

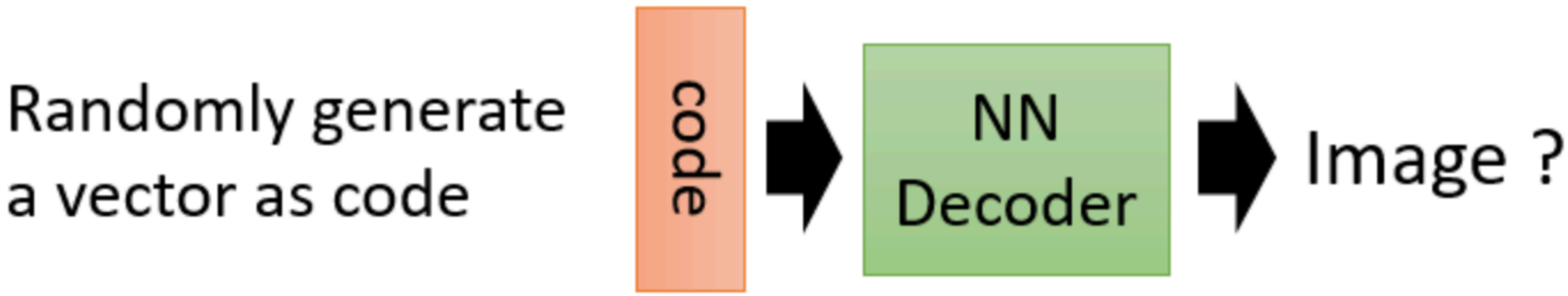
Road Map

- Why GAN?
- How do GANs work?
- When training GANs.
- GANs with other tricks.

Auto Encoder



“Generation”就像是创作，通过学习输入数据内部的Pattern，然后再利用这个模型去产生新的数据。所产生出来的内容应该能够“模仿”数据，而不是“抄袭”数据。



如果我们真的学到了编码空间，那么任何编码，只要能正确地输入到Decoder中，都应该能生成出“逼真”的数据。

http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS17.html

Maximum Likelihood Estimation

假设给定一个分布$P_{data}(x)$。我们现在想要用另一个由θ控制的分布$P_G(x;\theta)$来逼近$P_{data}(x)$。

这时候用极大似然估计就是个很intuitive的想法。形式化流程：

1. 从$P_{data}(x)$中采样得到一组数据$\{x^1, x^2, \dots, x^m\}$
2. 计算生成的数据对$P_G(x^i;\theta)$的似然函数：

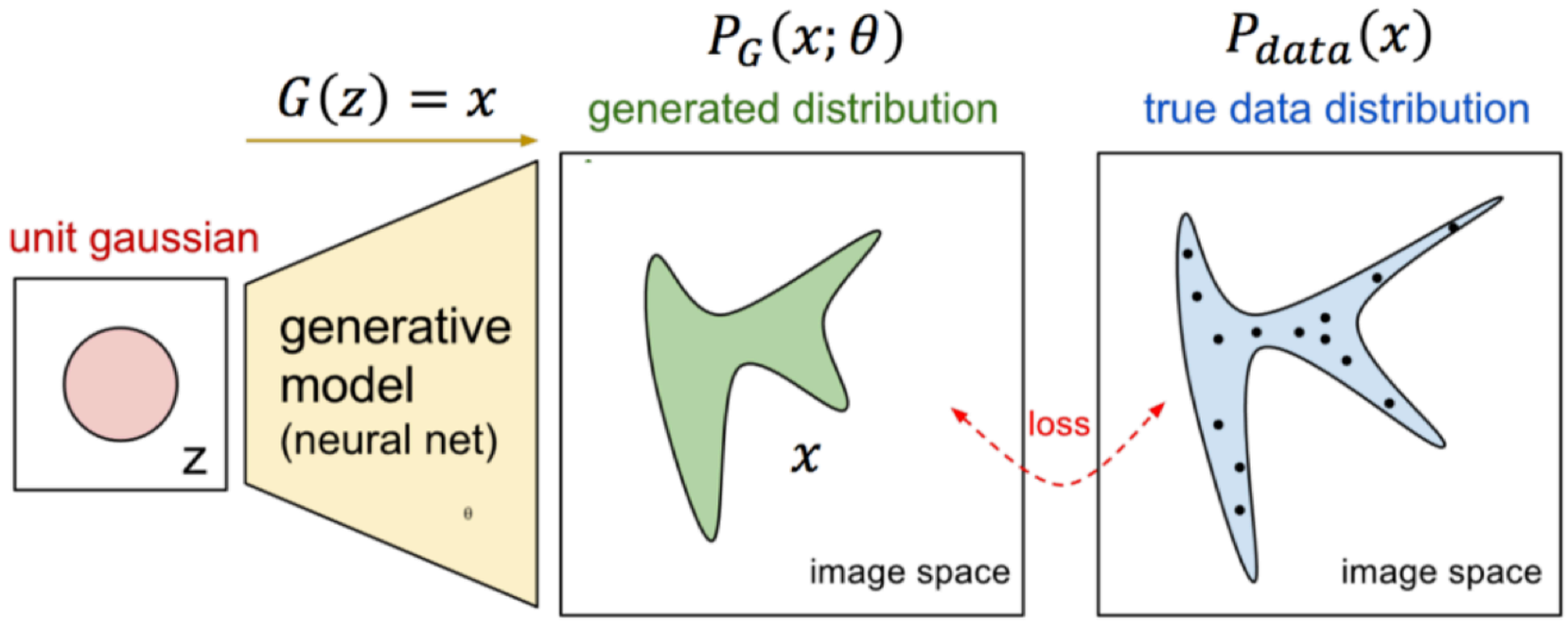
$$L = \prod_{i=1}^{m} P_G(x^i;\theta)$$

3. 求解能够让似然函数最大的θ^*

那我们要如何得到这个θ^*呢?

$$\begin{aligned}
\theta^* &= \arg\max_{\theta} \prod_{i=1}^{m} P_G(x^i;\theta) \\
&= \arg\max_{\theta} \log \prod_{i=1}^{m} P_G(x^i;\theta) \\
&= \arg\max_{\theta} \sum_{i=1}^{m} \log P_G(x^i;\theta) \\
&\approx \arg\max_{\theta} E_{x\sim P_{data}}[\log P_G(x;\theta)]
\end{aligned}$$

Now $P_G(x;\theta)$ is a NN



$$P_G(x) = \int_z P_{prior}(z) I_{[G(z)=x]} dz$$

It is difficult to compute the likelihood.

https://blog.openai.com/generative-models/

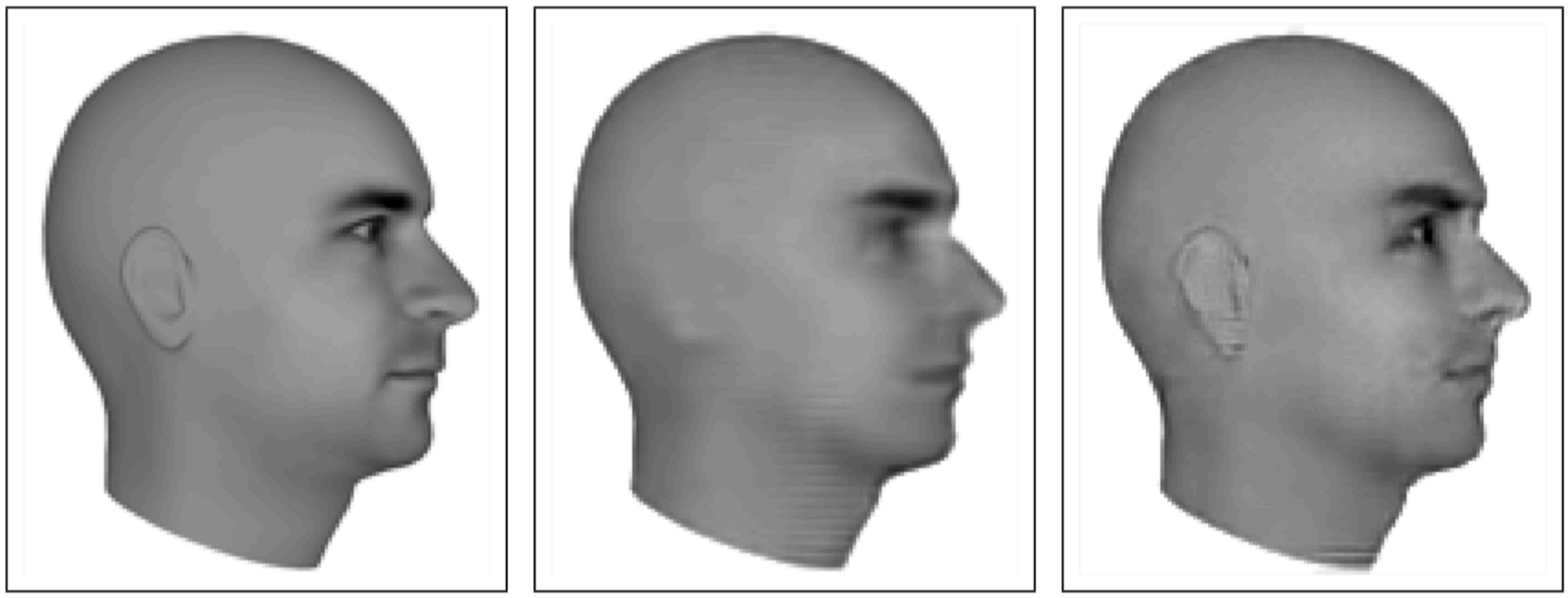
这个神经网络将一个prior分布映射到$P_G(x;\theta)$，并通过loss function使得$P_G(x;\theta)$与$P_{data}(x)$尽可能相似。

但是这样虽然能直接定义出$P_G(x)$的解析式，但是由于其中的$G(z)$是用各种复杂结构的神经网络所定义的，所以它很难用来计算似然函数。

那么，如何在没法计算似然函数的情况下通过调整模型参数θ来让P_G接近P_{data}呢？这时候我们就可以使用GAN了！

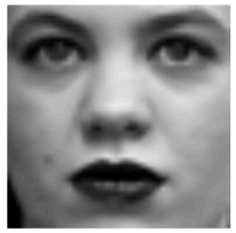
Next Video Frame Prediction

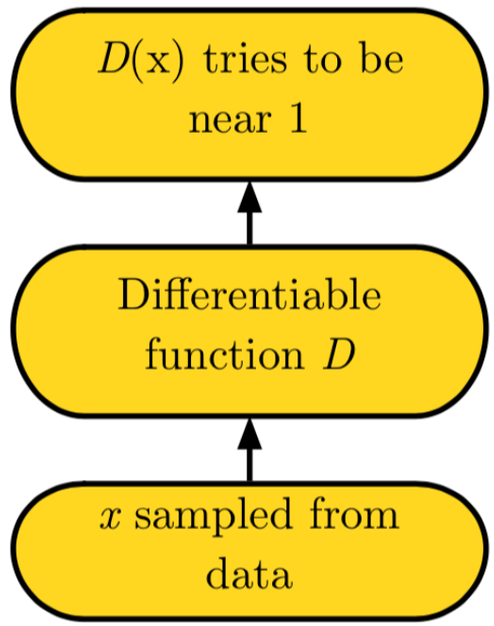
Ground Truth | MSE | Adversarial

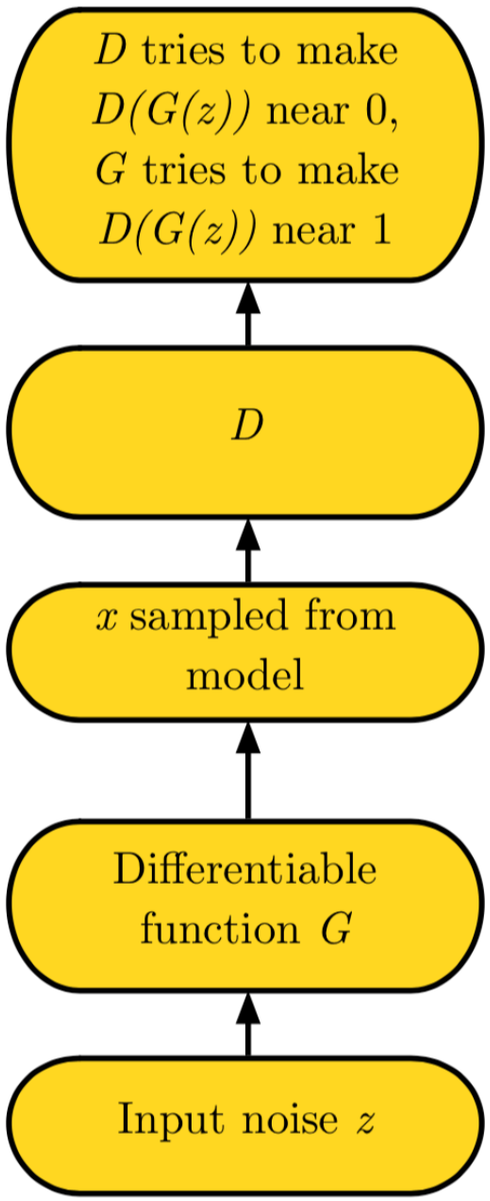


(Lotter et al 2016)

How does GAN work?







(Goodfellow 2016)

In other words, D and G play the following two-player minimax game with value function $V(G, D)$:

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

How to Train GAN?

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

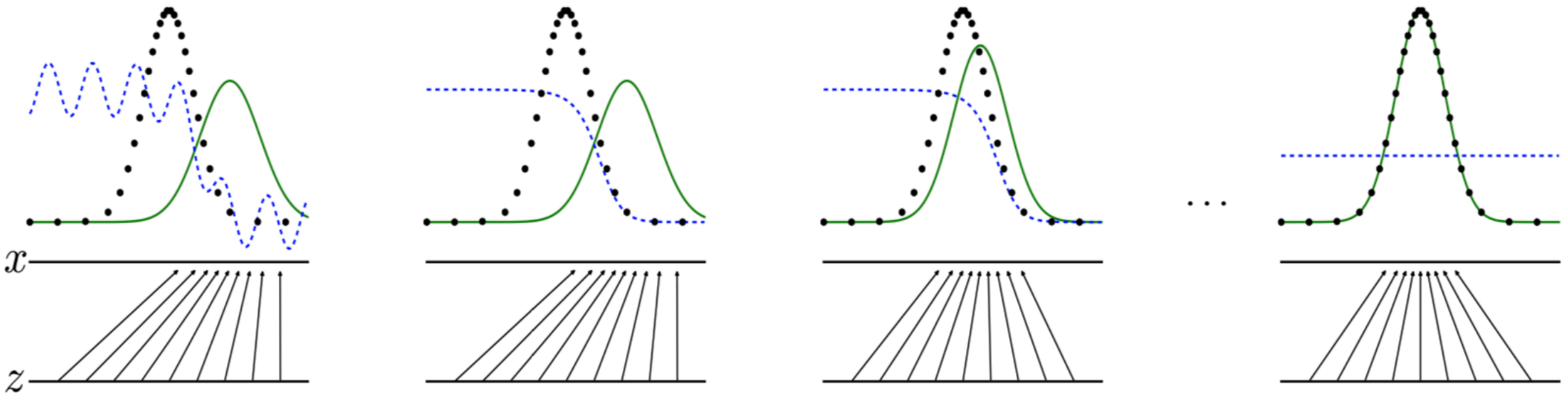
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

‘Generative Adversarial Networks’

https://arxiv.org/abs/1406.2661

Basic Idea of GAN



(a) (b) (c) (d)

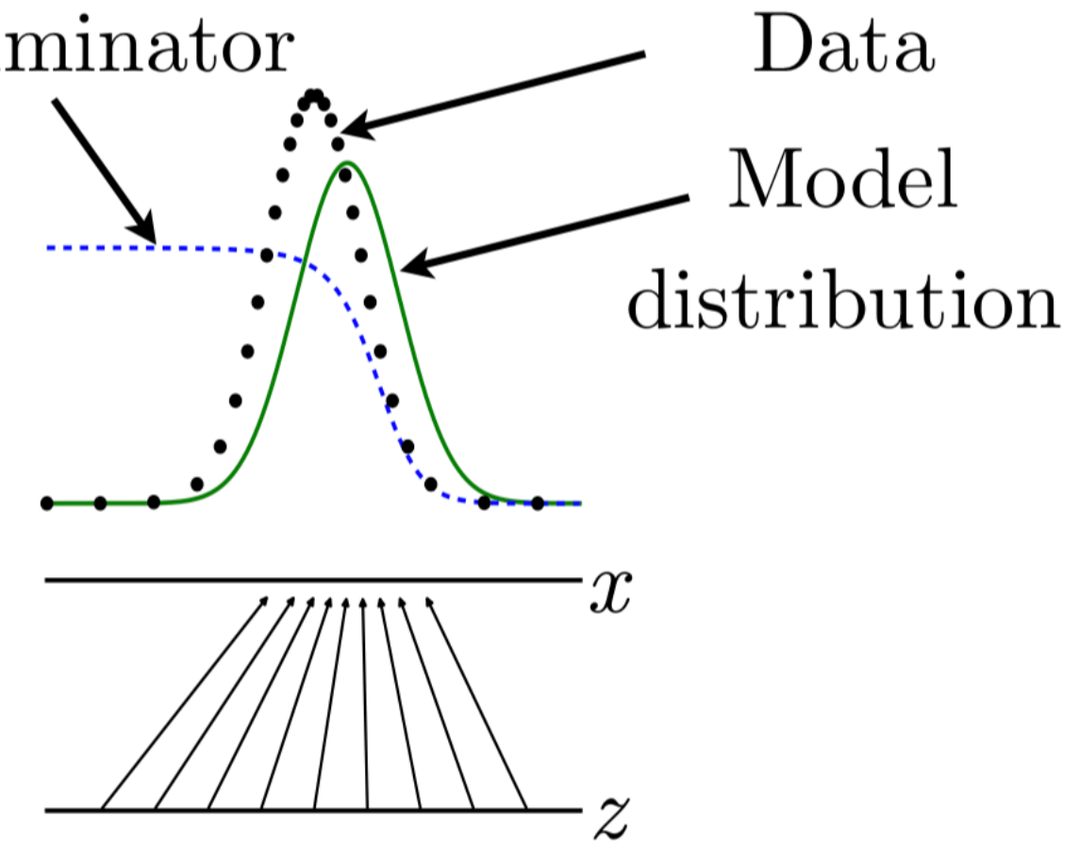
Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

Discriminator Strategy

Optimal $D(\boldsymbol{x})$ for any $p_{\text{data}}(\boldsymbol{x})$ and $p_{\text{model}}(\boldsymbol{x})$ is always

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

Estimating this ratio using supervised learning is the key approximation mechanism used by GANs



http://www.iangoodfellow.com/slides/2016-12-04-NIPS.pdf

http://www.iangoodfellow.com/slides/2016-12-04-NIPS.key

The minimax game can be reformulated as

$$
\begin{aligned}
C(G) =& \max_D V(G, D) \\
=& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} [\log(1 - D_G^*(G(\boldsymbol{z})))] \\
=& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [\log(1 - D_G^*(\boldsymbol{x}))] \\
=& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right] + \mathbb{E}_{\boldsymbol{x} \sim p_g} \left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right]
\end{aligned}
\tag{4}
$$

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x} \sim p_{data}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [\log(1 - D_G^*(\boldsymbol{x}))]$$

then p_g converges to p_{data}

Any problems about GANs?

1. How to determine step parameter k?
   G>>D or D>>G?

2. Can training loss indicate the training process?
   Assume that G and D can be infinite capacity.

原始GAN究竟出了什么问题?

1.判别器越好，生成器梯度消失越严重

回顾一下，原始GAN中判别器要最小化如下损失函数，尽可能把真实样本分为正例，生成样本分为负例：

$$-\mathbb{E}_{x\sim P_r}[\log D(x)] - \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \quad \text{（公式1）}$$

其中 P_r 是真实样本分布， P_g 是由生成器产生的样本分布。对于生成器，Goodfellow一开始提出来一个损失函数，后来又提出了一个改进的损失函数，分别是

$$\mathbb{E}_{x\sim P_g}[\log(1-D(x))] \quad \text{（公式2）}$$

$$\mathbb{E}_{x\sim P_g}[-\log D(x)] \quad \text{（公式3）}$$

固定生成器G，最优的判别器应该是：

令其关于 $D(x)$ 的导数为0，得

$$-\frac{P_r(x)}{D(x)} + \frac{P_g(x)}{1-D(x)} = 0$$

化简得最优判别器为：

$$D^*(x) = \frac{P_r(x)}{P_r(x)+P_g(x)} \quad \text{（公式4）}$$

‘Towards Principled Methods for Training Generative Adversarial Networks’ https://arxiv.org/abs/1701.04862

https://zhuanlan.zhihu.com/p/25071913

给公式2加上一个不依赖于生成器的项，使之变成

$$\mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1-D(x))]$$

注意，最小化这个损失函数等价于最小化公式2，而且它刚好是判别器损失函数的反。代入最优判别器即公式4，再进行简单的变换可以得到

$$\mathbb{E}_{x\sim P_r}\log\frac{P_r(x)}{\frac{1}{2}[P_r(x)+P_g(x)]} + \mathbb{E}_{x\sim P_g}\log\frac{P_g(x)}{\frac{1}{2}[P_r(x)+P_g(x)]} - 2\log 2 \quad \text{（公式5）}$$

变换成这个样子是为了引入Kullback–Leibler divergence（KL散度）和Jensen-Shannon divergence（JS散度）

$$KL(P_1||P_2) = \mathbb{E}_{x\sim P_1}\log\frac{P_1}{P_2} \quad \text{（公式6）}$$

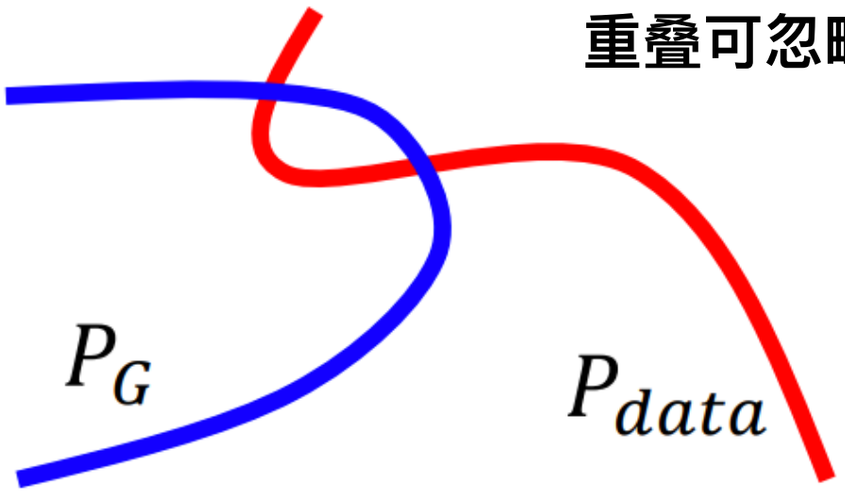
$$JS(P_1||P_2) = \frac{1}{2}KL(P_1||\frac{P_1+P_2}{2}) + \frac{1}{2}KL(P_2||\frac{P_1+P_2}{2}) \quad \text{（公式7）}$$

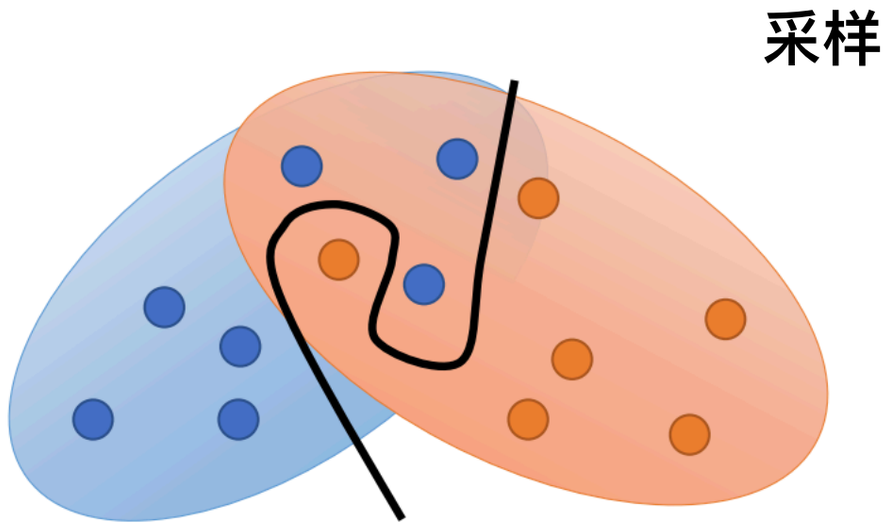
于是公式5就可以继续写成

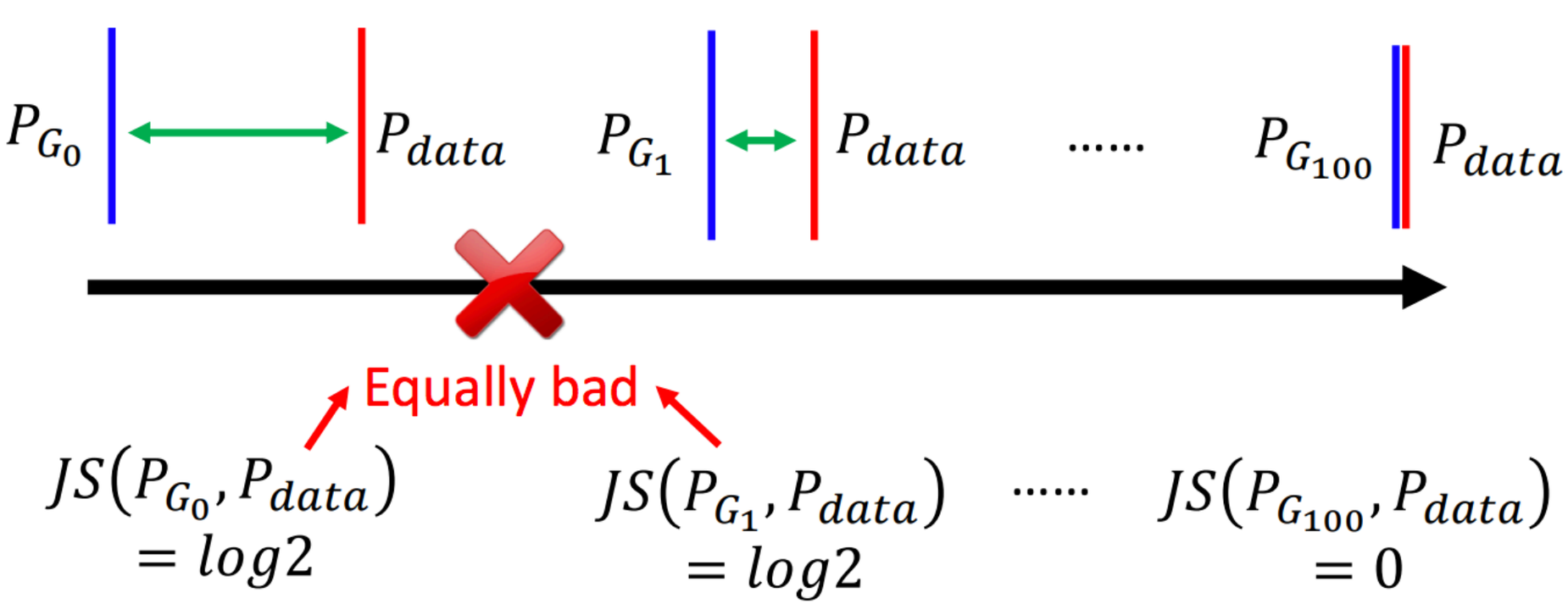
$$2JS(P_r||P_g) - 2\log 2 \quad \text{（公式8）}$$

在最优判别器的下，我们可以把原始GAN定义的生成器loss等价变换为最小化真实分布与生成分布之间的JS散度。

- In most cases, P_G and P_{data} are not overlapped.







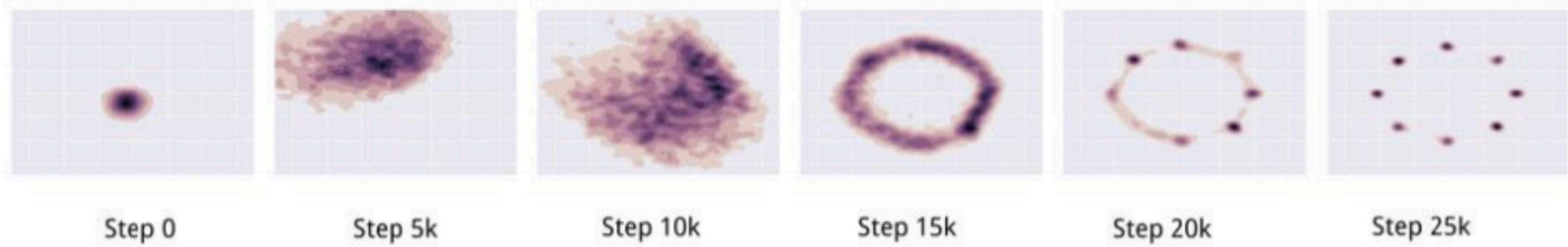
JS divergence is log2 if two distributions do not overlap.

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

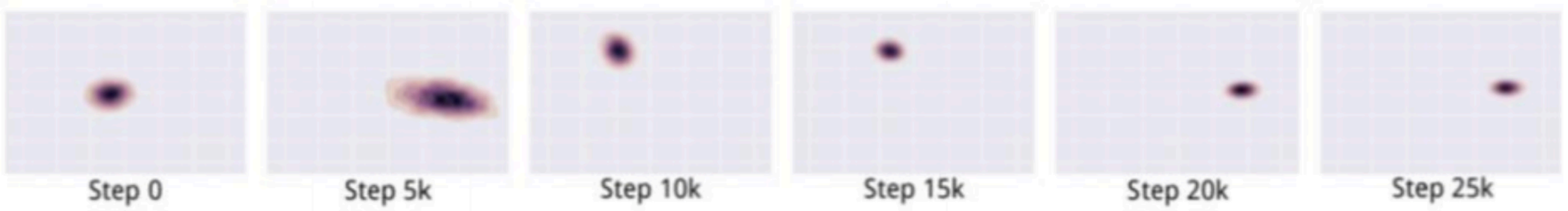
2. collapse mode即多样性不足

Mode Collapse

What we want ...



In reality ...



KL散度不是一个对称的衡量

- 当 $P_g(x) \to 0$ 而 $P_r(x) \to 1$ 时，$P_g(x) \log \frac{P_g(x)}{P_r(x)} \to 0$，对 $KL(P_g||P_r)$ 贡献趋近0
- 当 $P_g(x) \to 1$ 而 $P_r(x) \to 0$ 时，$P_g(x) \log \frac{P_g(x)}{P_r(x)} \to +\infty$，对 $KL(P_g||P_r)$ 贡献趋近正无穷

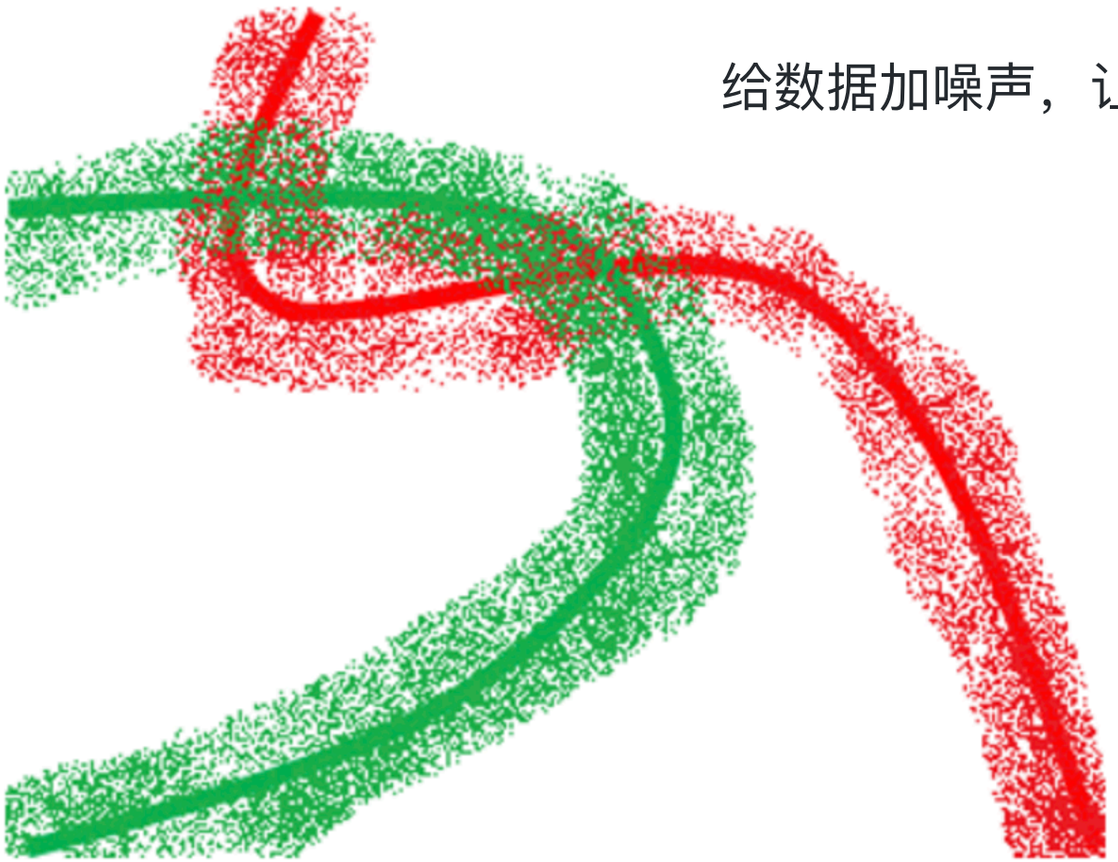
①第一种错误对应的是“生成器没能生成真实的样本”，惩罚微小；

②第二种错误对应的是“生成器生成了不真实的样本”，惩罚巨大；

③第一种错误对应的是缺乏多样性，第二种错误对应的是缺乏准确性。

为了保证最小化损失，它会宁可永远输出一样但是肯定正确的输出，也不愿意尝试其他不同但可能错误的输出。
我们的生成器有时可能无法兼顾数据分布的所有内部模式，只会保守地挑选出一个肯定正确的模式。

GANs with other tricks



给数据加噪声，让生成器和真实数据分布“更容易”重叠在一起

- 加入噪声势必会影响我们生成数据的质量。
- 一个简单的做法是让噪声的幅度随着时间缩小(退火）。
- 两个低维流形“本体”都已经有重叠时，就算把噪声完全拿掉，JS散度也能照样发挥作用，

WGAN

Wasserstein距离的优越性质

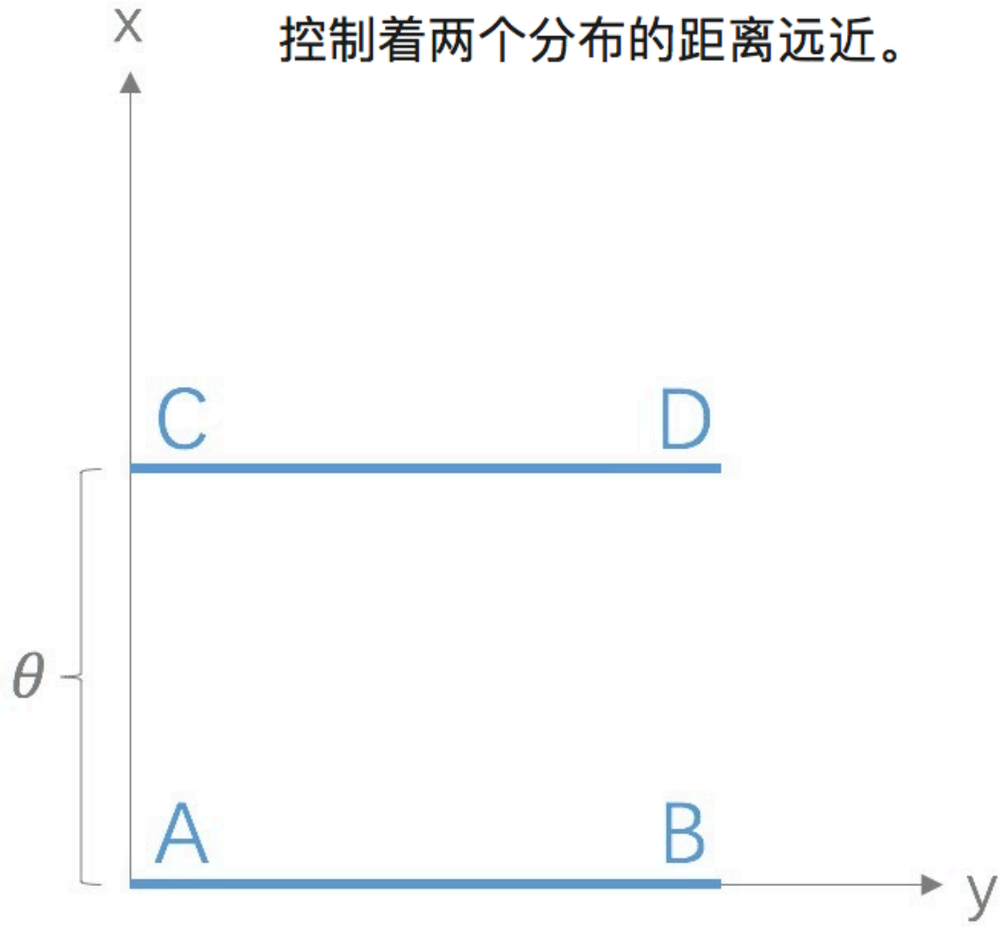
Wasserstein距离又叫Earth-Mover（EM）距离，定义如下：

$$W(P_r, P_g) = \inf_{\gamma \sim \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \quad \text{(公式12)}$$

解释如下：$\Pi(P_r, P_g)$ 是 P_r 和 P_g 组合起来的所有可能的联合分布的集合，反过来说，$\Pi(P_r, P_g)$ 中每一个分布的边缘分布都是 P_r 和 P_g 。对于每一个可能的联合分布 γ 而言，可以从中采样 $(x, y) \sim \gamma$ 得到一个真实样本 x 和一个生成样本 y，并算出这对样本的距离 $||x - y||$，所以可以计算该联合分布 γ 下样本对距离的期望值 $\mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$ 。在所有可能的联合分布中能够对这个期望值取到的下界 $\inf_{\gamma \sim \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$，就定义为Wasserstein距离。

即便两个分布没有重叠，Wasserstein距离仍然能够反映它们的远近。

两个分布 P_1 和 P_2 ， P_1 在线段AB上均匀分布， P_2 在线段CD上均匀分布，通过控制参数 θ 可以控制着两个分布的距离远近。



$$KL(P_1||P_2) = KL(P_1||P_2) = \begin{cases} +\infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases} \quad \text{（突变）}$$

$$JS(P_1||P_2) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta - 0 \end{cases} \quad \text{（突变）}$$

$$W(P_0, P_1) = |\theta| \quad \text{（平滑）}$$

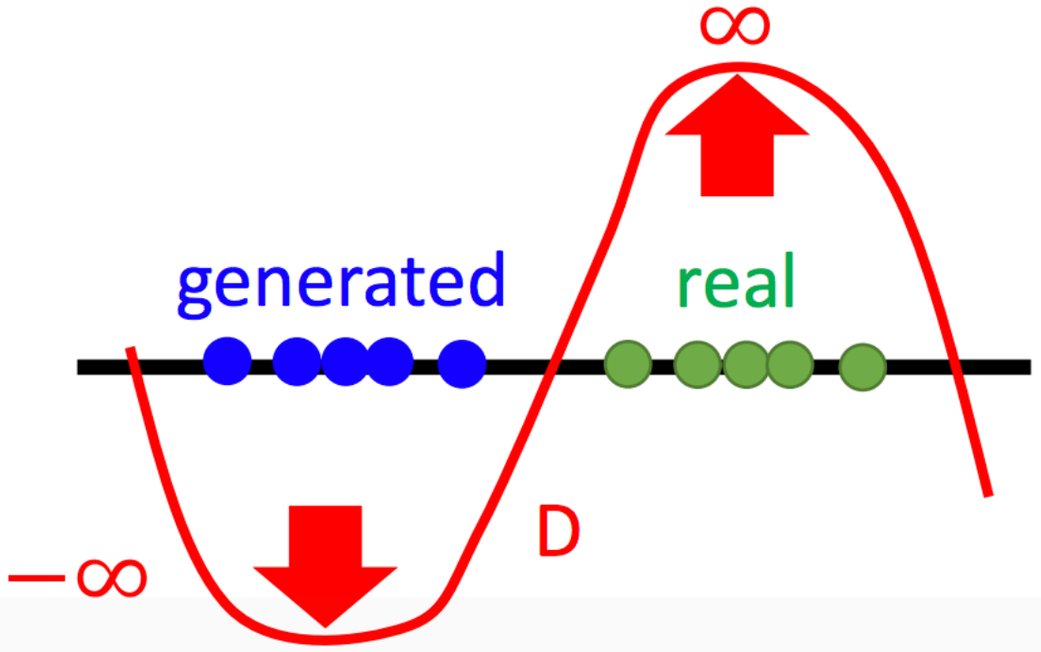
Evaluate wasserstein distance between P_{data} and P_G

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

D has to be smooth enough.

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

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

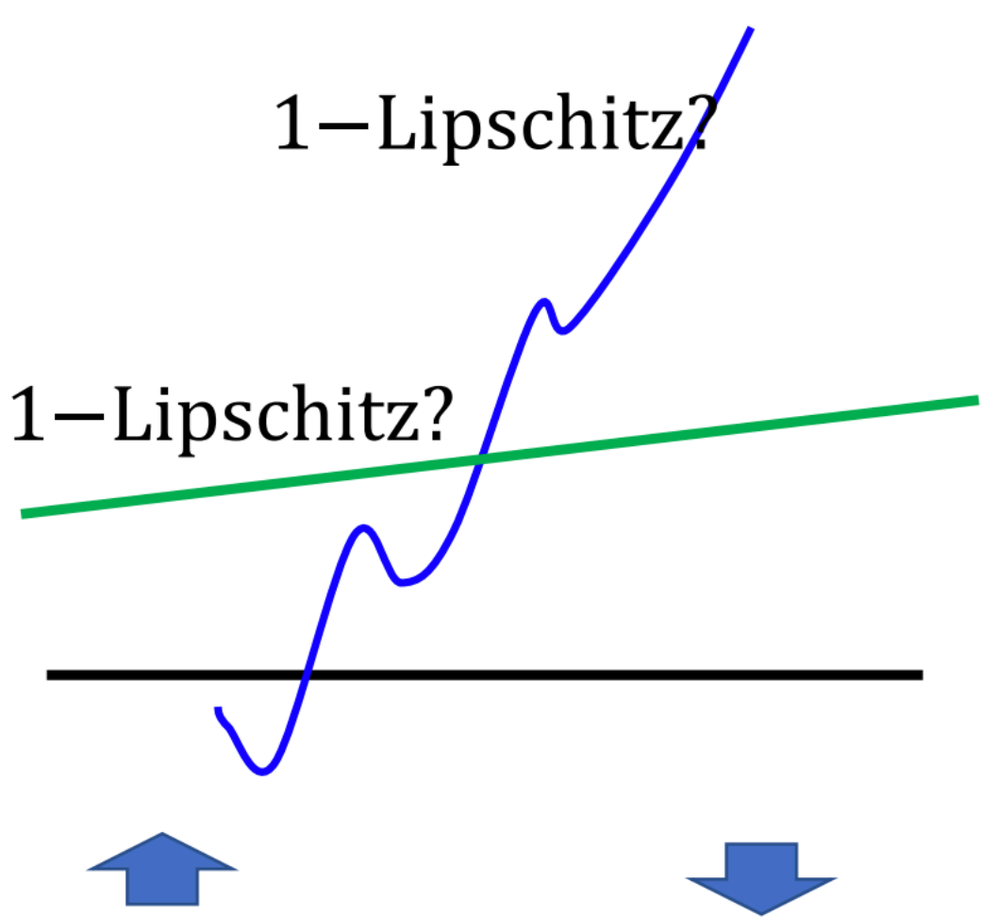


Lipschitz Function

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

Output change / Input change

K=1 for "$1-Lipschitz$"



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

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

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

- 判别器最后一层去掉sigmoid
- 生成器和判别器的loss不取log
- 每次更新判别器的参数之后把它们的绝对值截断到不超过一个固定常数c
- 不要用基于动量的优化算法（包括momentum和Adam），推荐RMSProp，SGD也行

Other GANs

https://github.com/hindupuravinash/the-gan-zoo

Python and tensor-flow Implementation of GANs

https://github.com/YadiraF/GAN

Thanks