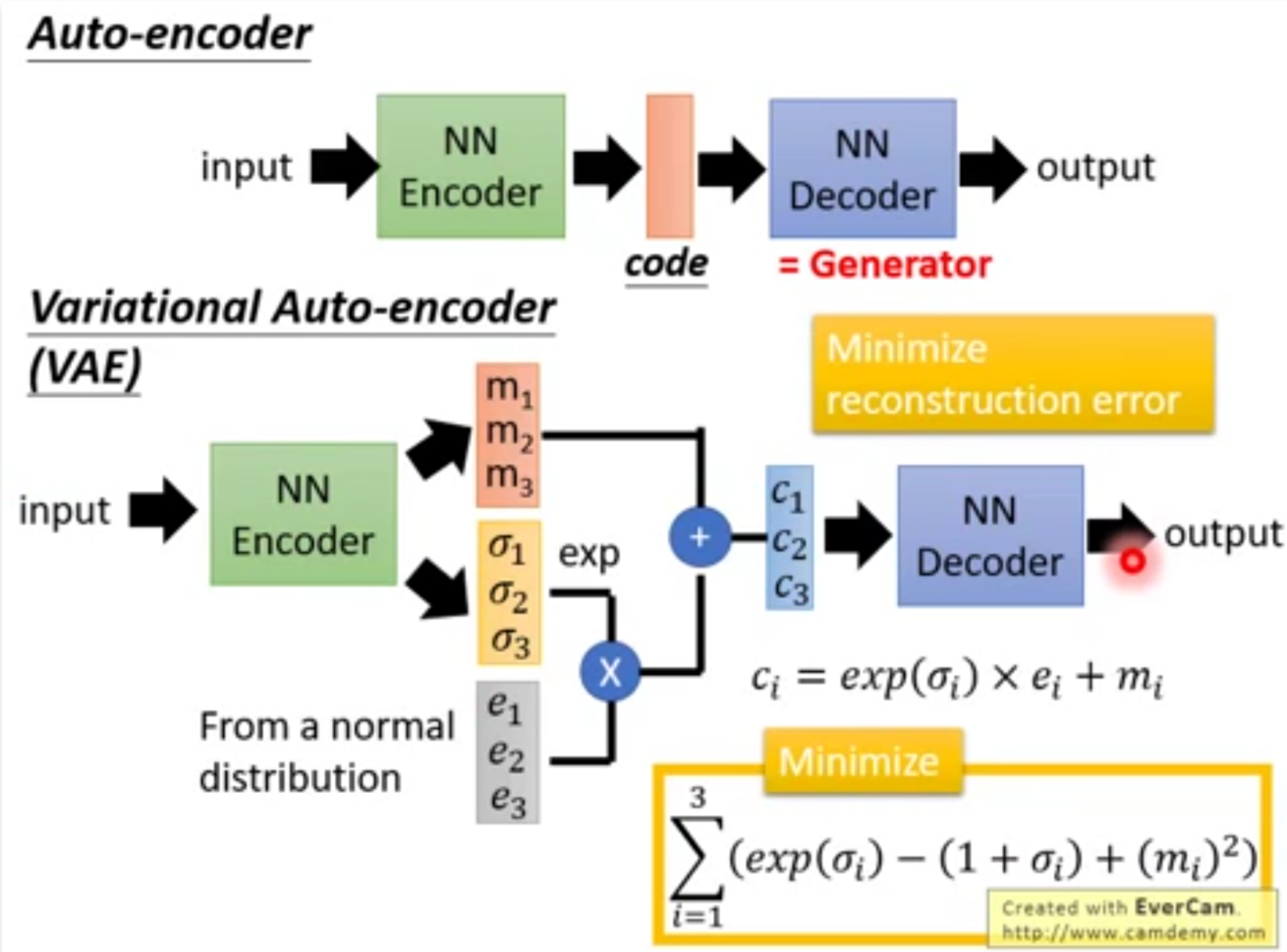
01 GAN

效果不佳

- 普通自编码器通过训练数据学习到的是某个确定的函数；
- 变分自编码器（VAE）通过训练数据学习到的是参数的概率分布。万字长文带你了解变分自编码器VAEs



萌生想法

对抗网络希望通过改变参数 θ ，使生成模型概率分布 $p_{model}(x;\theta)$ 能够逼近真实数据概率分布 $p_{data}(x)$ 。

即 通过 $\theta^* = \arg\max_\theta L = \arg\max_\theta \prod_{i=1}^{m} p_{model}(x^{(i)};\theta)$ 使得 $p_{model}(x;\theta)$ 逼近 $p_{data}(x)$ 。

$$
\begin{aligned}
\theta^* &= \arg\max_{\theta} \prod_{i=1}^{m} p_{model}(x^{(i)};\theta) = \arg\max_{\theta} \log \prod_{i=1}^{m} p_{model}(x^{(i)};\theta) = \arg\max_{\theta} \sum_{i=1}^{m} \log p_{model}(x^{(i)};\theta) \\
&= \arg\max_{\theta} E_{x\sim p_{data}} \log p_{model}(x;\theta) = \arg\max_{\theta} \int p_{data}(x) \log p_{model}(x;\theta)\mathrm{dx} \\
&= \arg\max_{\theta}\left(\int p_{data}(x) \log p_{model}(x;\theta)\mathrm{d}x - \int p_{data}(x) \log p_{data}(x)\mathrm{d}x\right) \\
&= \arg\max_{\theta} \int p_{data}(x)\left(\log p_{model}(x;\theta) - \log p_{data}(x)\right)\mathrm{d}x \\
&= \arg\max_{\theta} \int p_{data}(x) \log \frac{p_{model}(x;\theta)}{p_{data}(x)}\mathrm{d}x = \arg\max_{\theta}\left(-\int p_{data}(x) \log \frac{p_{data}(x)}{p_{model}(x;\theta)}\mathrm{d}x\right) \\
&= \arg\min_{\theta}\left(\int p_{data}(x) \log \frac{p_{data}(x)}{p_{model}(x;\theta)}\mathrm{d}x\right) \\
&= \arg\min_{\theta} KL(p_{data}(x)||p_{model}(x))
\end{aligned} \tag{1}
$$

即 $\boxed{\theta^* = \arg\min_{\theta} KL(p_{data}(x)||p_{model}(x))}$

知识点tip：

KL散度：一种计算概率分布之间相似程度的计算方法。

在假定为连续随机变量的前提下，且两个概率分布分别为 P 和 Q ，则 $KL(P||Q) = \int p(x)\log\frac{p(x)}{q(x)}\mathrm{d}x$ 。

注意：非负性，~~对称性~~。

遇到问题

问题：

生成模型 $x = \mathrm{G}(z)$

输入数据 z 的概率分布函数 已知

但 生成函数 G 是一种神经网络的形式 $\nRightarrow P_{model}(x) \nRightarrow L$

解决：

生成对抗网络中的代价函数

判别器D： $\mathrm{J}^{(D)}(\mathrm{D},\mathrm{G}) = -\frac{1}{2}E_{x\sim p_{data}}[\log \mathrm{D}(\mathrm{x})] - \frac{1}{2}E_{z\sim p_z}[\log(1-\mathrm{D}(\mathrm{G}(z)))]$

生成器G： $\mathrm{J}^{(G)}(\mathrm{D},\mathrm{G}) = \frac{1}{2}E_{x\sim p_{data}}[\log \mathrm{D}(\mathrm{x})] + \frac{1}{2}E_{z\sim p_z}[\log(1-\mathrm{D}(\mathrm{G}(z)))]$

价值函数 $\mathrm{V}(\mathrm{D},\mathrm{G})$ 表示 $\mathrm{J}^{(G)}$ 和 $\mathrm{J}^{(D)}$ 。

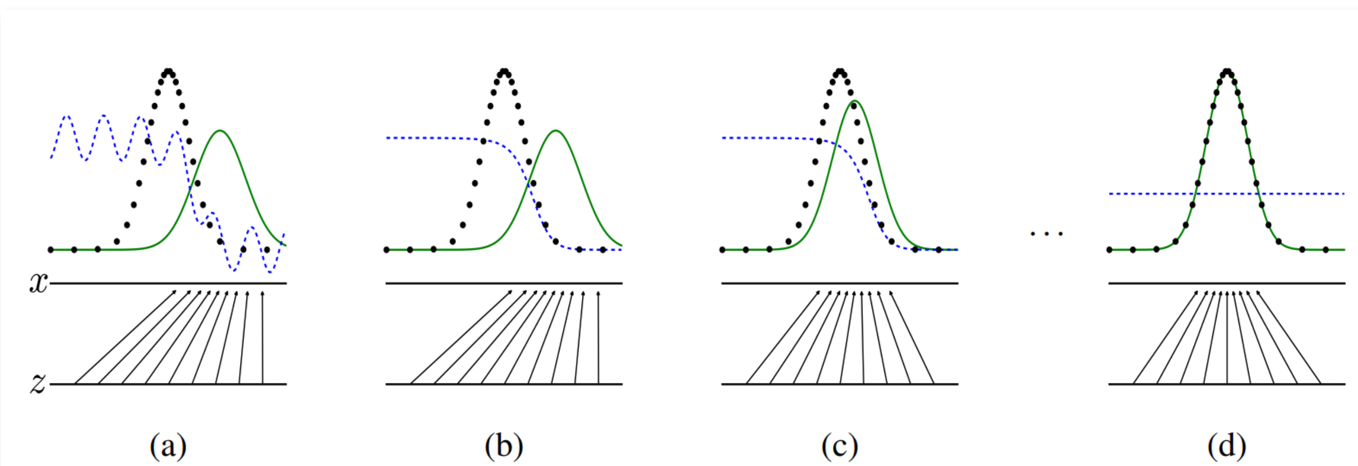
$$\boxed{\mathrm{V}(\mathrm{D},\mathrm{G}) = E_{x\sim p_{data}}[\log \mathrm{D}(x)] + E_{z\sim p_z}[\log(1-\mathrm{D}(\mathrm{G}(z)))]}$$

$$\mathrm{J}^{(D)} = -\frac{1}{2}\mathrm{V}(\mathrm{D},\mathrm{G}) \qquad \mathrm{J}^{(G)} = \frac{1}{2}\mathrm{V}(\mathrm{D},\mathrm{G})$$

在生成对抗网络中，我们要计算的纳什平衡点正是要寻找一个生成器G和判别器D，使得各自的代价函数最小，即希望找到了一个 $\mathrm{V}(\mathrm{D},\mathrm{G})$ 对于生成器来说最小而对判别器来说最大。公式： $\arg\min_G \max_D \mathrm{V}(\mathrm{D},\mathrm{G})$ 。例：鞍点。

可行性证明

当生成数据的分布 $p_g(x)$ 趋近于真实数据分布 $p_{data}(x)$ 时，D网络输出的概率 $\mathrm{D}^*(x)$ 会趋近于 $\frac{1}{2}$ 。这也是最终希望达到的训练结果，这时候G和D网络也就达到了一个平衡状态。



z 是随机噪声， x 是训练数据中的真实数据data。

分别求出理想的判别器 D^* 和生成器 G^* 。

$$\mathrm{D}^* = \arg\max_{\mathrm{D}} \mathrm{V}(\mathrm{D},\mathrm{G}^*) = \arg\min_{\mathrm{G}}\max_{\mathrm{D}} \mathrm{V}(\mathrm{D},\mathrm{G}) = \arg\min_{G} \mathrm{V}(\mathrm{D}^*,\mathrm{G}) = \mathrm{G}^* = \frac{1}{2}$$

- 最优判别器

假设生成器G是固定的，令 $\mathrm{G}(z) = x$ ，

$$\begin{aligned}\mathrm{V}(\mathrm{G},\mathrm{D}) &= E_{x\sim p_{data}}[\log \mathrm{D}(x)] + E_{z\sim p_z}[\log(1-\mathrm{D}(\mathrm{G}(z)))] \\ &= E_{x\sim p_{data}}[\log \mathrm{D}(x)] + E_{x\sim p_g}[\log(1-\mathrm{D}(x))]\end{aligned}$$

则

$$\begin{aligned}&= \int p_{data}(x)\log \mathrm{D}(x)\mathrm{d}x + \int p_g(x)\log(1-\mathrm{D}(\mathrm{x}))\mathrm{d}x \qquad (2) \\ &= \int p_{data}(x)\log \mathrm{D}(x) + p_g(x)\log(1-\mathrm{D}(\mathrm{x}))\mathrm{d}x\end{aligned}$$

希望对 $f(x) = p_{data}(x)\log \mathrm{D}(x) + p_g(x)\log(1-\mathrm{D}(x))$ ，无论 x 取何值都能最大。

$$\frac{\mathrm{d}f(x)}{\mathrm{dD}(x)} = \frac{p_{data}(x)}{\mathrm{D}(x)} - \frac{p_g(x)}{1-\mathrm{D}(\mathrm{x})} = 0 \Rightarrow \boxed{\mathrm{D}^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}} \in (0,1)$$

- 最优生成器

代入 $\mathrm{D}^*(x)$ ，即判别器D是固定的，

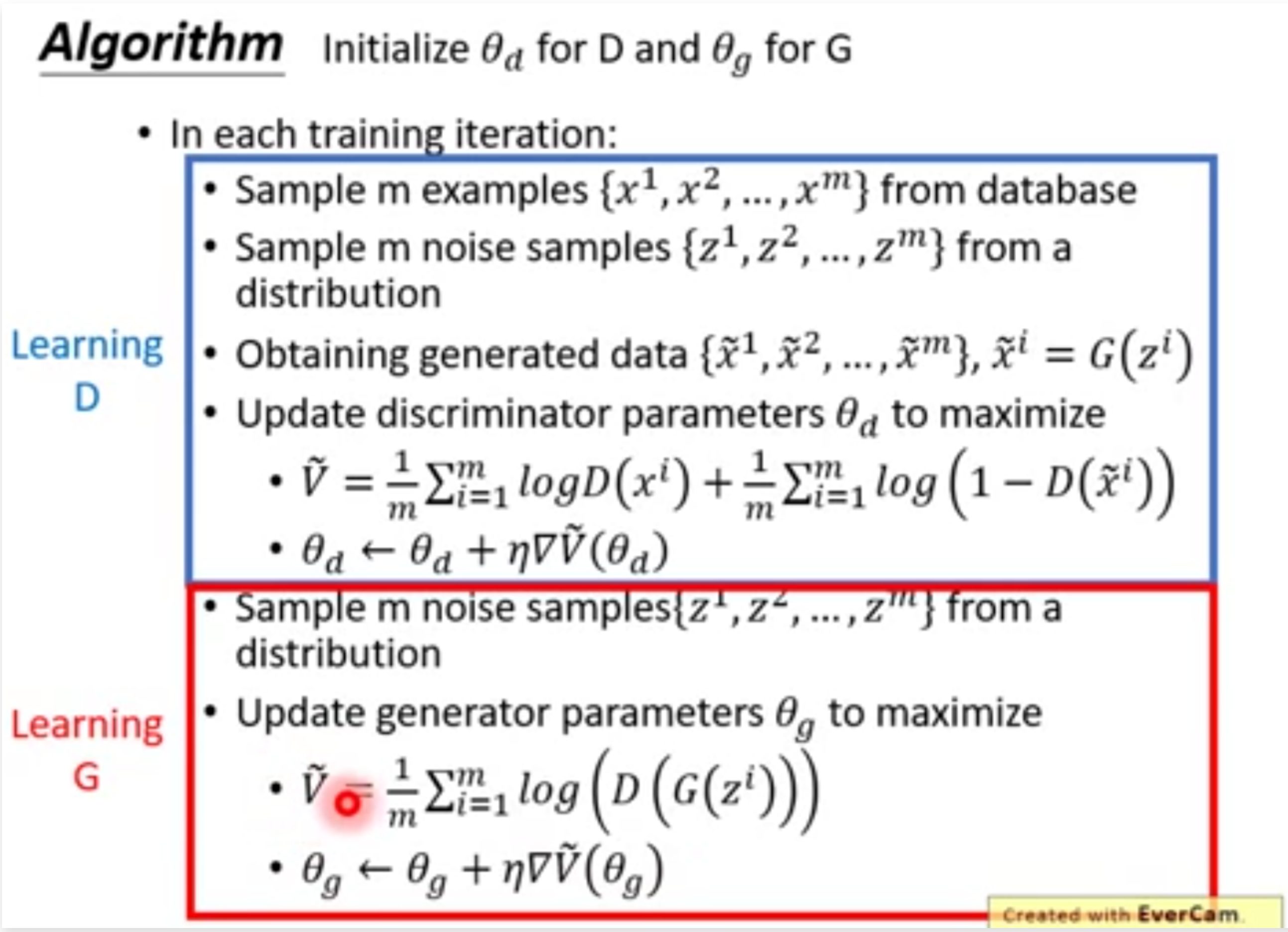
$$\begin{aligned}\mathrm{C}(\mathrm{G}) &= \max_{D} \mathrm{V}(\mathrm{G},\mathrm{D}) = \mathrm{V}(\mathrm{G},\mathrm{D}^*) \\ &= E_{x\sim p_{data}}[\log \mathrm{D}^*(x)] + E_{z\sim p_z}[\log(1-\mathrm{D}^*(\mathrm{G}(z)))] \\ &= E_{x\sim p_{data}}[\log \mathrm{D}^*(x)] + E_{x\sim p_g}[\log(1-\mathrm{D}^*(x))] \\ &= E_{x\sim p_{data}}[\log \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}] + E_{x\sim p_g}[\log \frac{p_g(x)}{p_{data}(x)+p_g(x)}] \\ &= E_{x\sim p_{data}}[\log \frac{p_{data}(x)/2}{(p_{data}(x)+p_g(x))/2}] + E_{x\sim p_g}[\log \frac{p_g(x)/2}{(p_{data}(x)+p_g(x))/2}] \\ &= E_{x\sim p_{data}}[\log \frac{p_{data}(x)}{(p_{data}(x)+p_g(x))/2}] + E_{x\sim p_{data}}[-\log 2] + E_{x\sim p_g}[\log \frac{p_g(x)}{(p_{data}(x)+p_g(x))/2}] + E_{x\sim p_g}[-\log 2] \\ &= \int p_{data}(x)\log \frac{p_{data}(x)}{(p_{data}(x)+p_g(x))/2}\mathrm{d}x + \int p_g(x)\log \frac{p_g(x)}{(p_{data}(x)+p_g(x))/2}\mathrm{d}x - \log(4) \\ &= -\log(4) + KL\left(p_{data}(x) \parallel \frac{p_{data}(x)+p_g(x)}{2}\right) + KL\left(p_g(x) \parallel \frac{p_{data}(x)+p_g(x)}{2}\right) \\ &= -\log(4) + 2\cdot JSD(p_{data} \parallel p_g)\end{aligned}$$

即 $\boxed{\mathrm{C}(\mathrm{G}) = -\log(4) + 2\cdot JSD(p_{data} \parallel p_g)}$

当且仅当 $p_{data} = p_g$ 的时候，上式可以取得全局最小值 $\mathrm{C}^* = -\log(4)$ 。

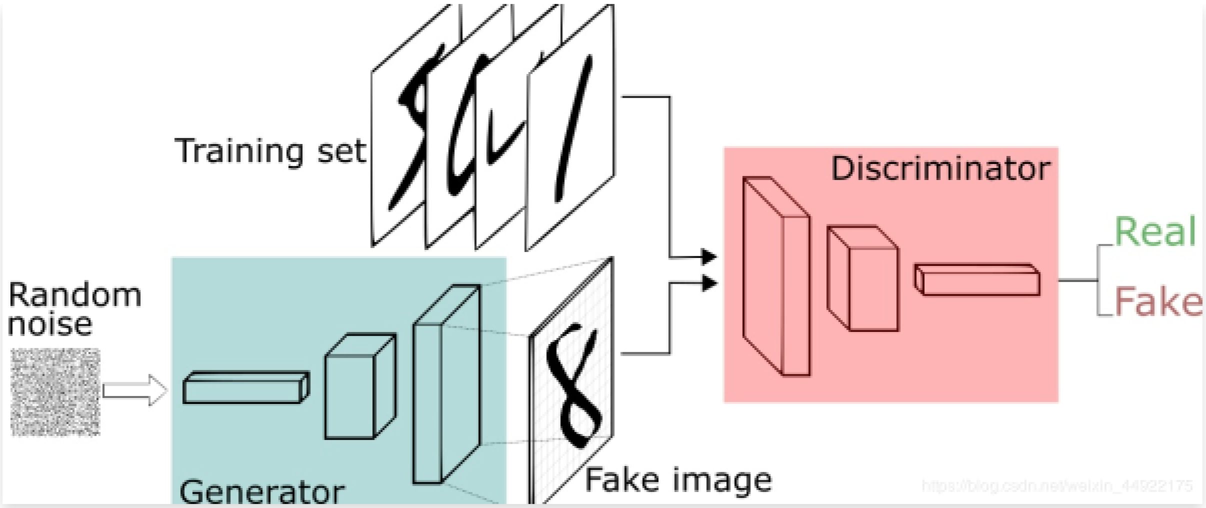
$\Rightarrow \mathrm{G}^* = \arg\min_{G}\max_{D} \mathrm{V}(\mathrm{G},\mathrm{D})$ 。

算法实现流程



针对GAN的认识

1. 由判别模型和生成模型两部分组成。
2. 生成模型负责生成假样本，欺骗判别器。
3. 判别模型来分辨真样本和假样本。
4. 两个模型交替训练，类似对抗博弈的过程。
5. 训练判别器，固定生成器，使用反向传播。
6. 固定判别器，训练生成器，使用反向传播。



论文给出的算法实现：

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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ 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.

训练技巧和方法：

1. 首先G和D是同步训练，但是两者训练次数不一样，通常是D网络训练k次后，G训练一次。主要原因是GAN刚开始训练时候会很不稳定；
2. D的训练是同时输入真实数据和生成数据来计算loss，而不是采用交叉熵（cross entropy）分开计算。不采用cross entropy的原因是这会让 $\mathrm{D}(\mathrm{G}(z))$ 变为0，导致没有梯度提供给G更新，而现在GAN的做法是会收敛到 $\frac{1}{2}$ ；
3. 实际训练的时候，作者是采用 $-\log(\mathrm{D}(\mathrm{G}(z)))$ 来代替 $\log(1-\mathrm{D}(\mathrm{G}(\mathrm{z})))$ ，这是希望在训练初始就可以加大梯度信息，这是因为初始阶段D的分类能力会远大于G生成足够真实数据的能力，但这种修改也将让整个GAN不再是一个完美的零和博弈。