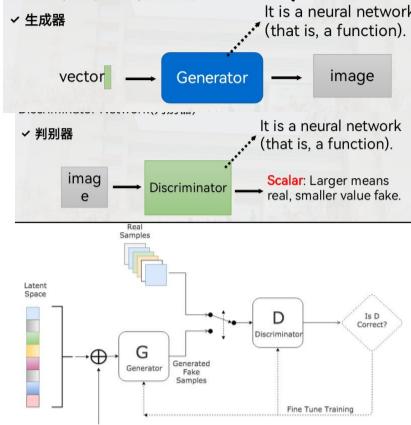


生成对抗网络(Generative adversarial network, GAN)

由两个主要网络构成：生成器；判别器

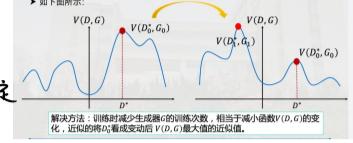


GAN的训练过程及梯度下降



训练过程简述如下

- 1) 固定生成器 G , 训练判别器 D , 获得 $\max V(D, G)$
- 2) 固定判别器 D , 训练生成器 G , 通过梯度下降求更新参数:
 $\theta_g = \theta_g - \eta \frac{\partial L(G)}{\partial \theta_g} = \theta_g - \eta \nabla L(\theta_g) \rightarrow$ 训练后 $V(D, G)$ 度化而判别器固定
此时 D^* 点不是JS散度最大点
- 3) 重复循环上面两步直到模型收敛。



进阶总结

- 1) 通过抽样方式获得样本，真实样本 $x_1, x_2, \dots, x_N \in P_{\text{data}}(x)$, 噪声样本 $z_1, z_2, \dots, z_N \in P_z(x)$, 生成样本 $x'_1, x'_2, \dots, x'_N \in P_g(x)$
- 2) 固定生成器 G , 训练判别器 D , 以样本分布来近似表示真实分布与生成分布:
 $\max V'(D, G) = \frac{1}{N} \sum_{i=1}^N \log D(x_i; \theta_D) + \frac{1}{N} \sum_{i=1}^N \log(1 - D(x'_i; \theta_D))$
更新参数:
 $\theta_d = \theta_d - \eta \nabla V'(D_g)$ 训练多次才能梯度上升
- 3) 固定判别器 D , 训练生成器 G :
 $\min V'(D^*, G) = \frac{1}{N} \sum_{i=1}^N \log(1 - D(x'_i; \theta_g))$
更新参数:
 $\theta_g = \theta_g - \eta \nabla V'(G_g)$ 梯度下降
- 4) 重复循环上面两步直到模型收敛。

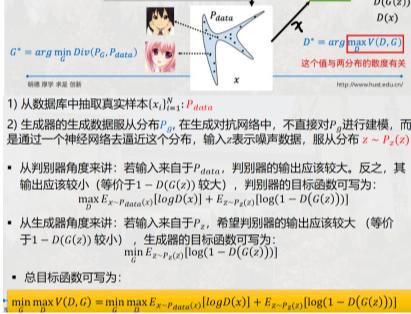
统一框架 F-GAN \Rightarrow 使用F散度/JS散度形式一般化GAN目标函数

$$\text{DfCPPIQ} = \int_x q(x) f(-\frac{p(x)}{q(x)}) dx$$

满足① f 为 $-$ 凸函数 ② $f(1)=0$ 时可用 DfCPPIQ 衡量两种概率分布之间差异

函数名	表达式
KL散度	$\frac{1}{2} \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_g(x)} dx$
JS散度	$\frac{1}{2} \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{avg}}(x)} dx$
Wasserstein	$\int_x p_{\text{data}}(x) - p_g(x) dx$
Pearson	$\int_x \frac{(p_{\text{data}}(x) - p_g(x))^2}{p_{\text{data}}(x)} dx$
余弦相似度	$\int_x \frac{(p_{\text{data}}(x) \cdot p_g(x))}{\sqrt{p_{\text{data}}(x)p_g(x)}} dx$
Hausdorff	$\int_x (\sqrt{p_{\text{data}}(x)} - \sqrt{p_g(x)})^2 dx$
JSD散度	$\int_x p_{\text{data}}(x) - p_g(x) \log \frac{p_{\text{data}}(x)}{p_g(x)} dx$
逆散度	$\frac{1}{2} \int_x p_{\text{data}}(x) \log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} dx + \frac{1}{2} \int_x p_g(x) \log \frac{2p_g(x)}{p_{\text{data}}(x) + p_g(x)} dx$
alpha散度	$\frac{1}{\alpha(\alpha-1)} \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)^{\alpha-1}}{p_g(x)^{\alpha-1}} dx - \alpha(p_g) - \frac{1}{\alpha(\alpha-1)} \int_x p_g(x) \log \frac{p_g(x)^{\alpha-1}}{p_{\text{data}}(x)^{\alpha-1}} dx - \alpha(p_{\text{data}})$

目标函数构造



1) 从数据库中抽取真实样本 $\{x_i\}_{i=1}^N \sim P_{\text{data}}$

2) 生成器的生成数据服从分布 P_g , 在生成对抗网络中, 不直接对 P_g 进行建模, 而是通过一个神经网络去逼近这个分布, 输入 z 表示噪声数据, 服从分布 $z \sim P_z(x)$

• 从判别器角度来讲: 若输入 x 来自 P_{data} , 判别器的输出应该较大. 反之, 其输出应该较小 (等价于 $1 - D(G(x))$ 较大). 判别器的目标函数可写为:
 $\max_D E_{x \sim P_{\text{data}}(x)} [\log D(x)] + E_{x \sim P_g(x)} [\log(1 - D(x))]$

• 从生成器角度来讲: 若输入 x 来自 P_g , 希望判别器的输出应该较大 (等价于 $1 - D(G(x))$ 较小). 生成器的目标函数可写为:
 $\min_G \min_{x \sim P_g(x)} [\log(1 - D(x))]$

• 总目标函数可写为:

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

全局最优解

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

等价于 $E_{x \sim P_{\text{data}}} [\log D(x)] + E_{x \sim P_g} [\log(1 - D(x))]$

第一步希望判别器输出最大, 那么此时固定 G 的参数, 只训练 D , 将判别器的目标函数转换成积分形式:

$$\max_D V(D, G) = \int_D [P_{\text{data}}(x) \log D(x) + P_g(x) \log(1 - D(x))] dx$$

要找到一个 D 使得 $V(D, G)$ 最大, 对其求偏导:

$$\frac{\partial V(D, G)}{\partial D} = \int \frac{\partial}{\partial D} [P_{\text{data}}(x) \log D(x) + P_g(x) \log(1 - D(x))] dx = 0$$

$$\int [P_{\text{data}}(x) \frac{1}{D(x)} + P_g(x) \frac{-1}{1 - D(x)}] dx = 0$$

将上式进行变化可得:

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

将推导出的公式带入目标函数中:

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

$$= \min_G \int_{P_{\text{data}}} [E_{x \sim P_{\text{data}}} [\log D^*(x)] + E_{x \sim P_g} [\log(1 - D^*(x))]] dx$$

$$= \min_G \int_{P_{\text{data}}} [E_{x \sim P_{\text{data}}} [\log \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_g(x)}] + E_{x \sim P_g} [\log \frac{P_g(x)}{P_{\text{data}}(x) + P_g(x)}]] dx$$

做一些简单的变换:

$$= \min_G \int_{P_{\text{data}}} [E_{x \sim P_{\text{data}}} [\log \frac{1}{P_{\text{data}}(x) + P_g(x)}] + E_{x \sim P_g} [\log \frac{1}{P_{\text{data}}(x) + P_g(x)}]] dx$$

$$= \min_G -2 \log 2 + E_{x \sim P_{\text{data}}} [\log \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_g(x)}] + E_{x \sim P_g} [\log \frac{P_g(x)}{P_{\text{data}}(x) + P_g(x)}]$$

$$= \min_G -2 \log 2 + E_{x \sim P_{\text{data}}} [\log \frac{P_{\text{data}}(x)}{2}] + E_{x \sim P_g} [\log \frac{P_g(x)}{2}] = \min_G -\log 2 + \frac{1}{2} \text{JS}(P_{\text{data}} || \frac{P_{\text{data}} + P_g(x)}{2})$$

$\geq -\log 2$

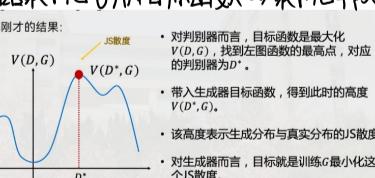
当 $P_{\text{data}} = \frac{P_{\text{data}}(x) + P_g(x)}{2} = P_g$ 时成立:

$$\text{最优 } P_g^* = P_{\text{data}}, \text{ 此时 } D^* = \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_g(x)} = \frac{1}{2}$$

KL散度: $\text{KLCPPIQ} = \int P(x) \log \frac{P(x)}{Q(x)} dx$ JS散度: $\text{JS}(P||Q) = \frac{1}{2} \text{KL}(P_1 || \frac{P_1 + P_2}{2}) + \frac{1}{2} \text{KL}(P_2 || \frac{P_1 + P_2}{2})$

反生成器最小化 GAN目标函数 \Rightarrow 最小化真实分布与生成分布之间的JS散度/相对熵(拟似度)

图像化理解刚才的结果:



- 对判别器而言, 目标函数是最大化 $V(D, G)$, 找到左图函数的最高点, 对应的判别器为 D^* 。
- 带入生成器目标函数, 得到此时的高度 $V(D^*, G)$ 。
- 该高度表示生成分布与真实分布的JS散度
- 对生成器而言, 目标就是训练 G 最小化这个JS散度。

解决方法: ①不把判别器训练得太好 ②给生成数据和真实数据加噪声, 让生成数据与真实数据在高位产生重叠, 此时JS散度

$$\min_D E_{x \sim P_{\text{data}}} [\log D^*(x)] + E_{x \sim P_g} [\log(1 - D^*(x))]$$

2. 有问题: 模式崩溃

生成图像相同太多, 继续训练会生成更多相同/相近图像, 造成生成数据多样性不足

解决方法:

- 1) 从目标函数考虑: 经验发现, 当GAN出现模式崩溃问题时, 通常判别器在真实样本附近更新参数时, 其梯度值非常大。可对判别器在真实样本附近施加梯度惩罚项。试图在真实样本附近构建线性函数, 因为线性函数具有全局最优解。

(DRAGAN)

- 2) 从网络架构考虑: 即使单个生成器会产生模式崩溃的问题, 但是如果同时构造多个生成器, 且让每个生成器产生的不同模式, 则这样的多生成器结合起来也可以保证产生的样本具有多样性。(MAD (Multi-agent diverse)-GAN)

EM距离/推土距离

Wasserstein GAN: 采用Wasserstein距离衡量两个分布之间的距离。

将数据分布 P 移到分布 Q 的代价:

$$B(Y) = \sum_{y \in Y} \|y - x_q\| \|x_p - x_q\|$$

$y \sim x_q$ 表示随机行为, 即随机批次 Q 将分布 P 的数据放到分布 Q 的附近 $|x_p - x_q|$ 。

EM距离:

$$W(P, Q) = \min_B B(Y)$$

WGAN中, 采取Weight clipping的方式让判别器目标函数更平滑。

$$\max_{D \in \text{Lipschitz}} E_{x \sim P_{\text{data}}} [D(x)] - E_{x \sim P_g} [D(x)]$$

满足1-Lipschitz的约束条件。Lipschitz函数定义如下:

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

即要求输入的變化量 K 大于输出的變化量, 使得变化不快, 从而保证函数是平滑的。当 $K=1$ 时, 此Upsilon函数满足1-Lipschitz。

判别器为了实现返回的损失可以更好的区分真实样本和生成样本, 需要参数是取到最大值或最小值。假设设置范围[-0.01, 0.01], 如下图所示会集中在最大值0.01与最小值-0.01上。



WGAN-GP (gradient penalty): 更改 weight clipping

WGAN-GP: 判别器D服从1-Lipschitz约束时, 等价于判别器D在任意地方的梯度都小

$$D \in 1-\text{Lipschitz} \Leftrightarrow \|\nabla_x D(x)\| \leq 1$$

判别器的目标函数可改写为:

$$\max_D E_{x \sim P_{\text{data}}} [D(x)] - E_{x \sim P_g} [D(x)] - \lambda \int_x \max(0, \|\nabla_x D(x)\| - 1) dx$$

梯度最后一项, 即惩罚样本抽取数据的空间分布 P_{penalty} , 通过实验发现, P_{penalty} 设置成生成数据空间分布与真实数据空间分布之间的效果比较好,

上式可改写为:

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

从生成样本和真实数据空间抽取一些样本点, 多个样本之间连线构成的空间就是 P_{penalty}