How does GAN work?

最后之作

2022 年 8 月 7 日

1 从一个简单的例子说起

首先假设手上有 m 张图片，大小为 32x32, 每张图片表示成一个长为 1024 的向量，分别为 $y_1, y_2, \ldots, y_m$.

想法：设计一个函数 g，输入为 0，1 之间的一个随机数字 z，输出为一个长为 1024 的向量 $x = g(z)$, 并且 g(z) 是一张图片，不是乱七八糟的东西.

开始建模–待定系数法

现在考虑一个简单的做法：不妨设 $g = w_2(w_1 x + b_1) + b_2 \overset{\text{记为}}{=} g(x;\theta)$

1. 随机采样：$z_1, z_2, \ldots, z_m$，并分别代入 g，得到 $x_1, x_2, \ldots, x_m$，其中,$x_i = g(z_i;\theta)$

2. 定义距离：$L = \dfrac{1}{m}\sum_{i=1}^{m}[g(z_i;\theta) - y_i]^2$

为了让 $x_1, x_2, \ldots, x_m$ 这组数据更接近 $y_1, y_2, \ldots, y_m$,

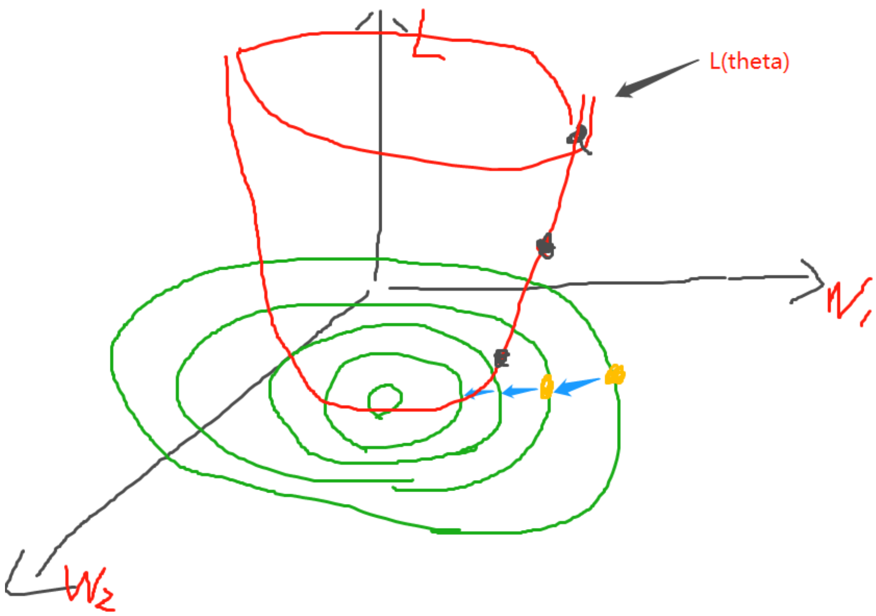
解析法（人工求解）:

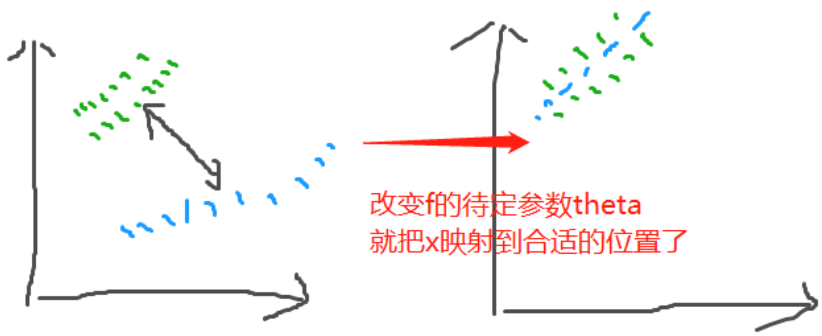
由多元函数极值条件，令： $\dfrac{\partial L}{\partial w_1} = 0, \dfrac{\partial L}{\partial w_2} = 0, \dfrac{\partial L}{\partial b_1} = 0, \dfrac{\partial L}{\partial b_2} = 0$

此时求得的参数值，可以使得 L 最小，也就是说明 $z_1, z_2, \ldots, z_m$ 通过 g 后，输出产生的 $g(z_1), g(z_2), ..., g(z_m)$, 整体上在平方距离的意义下靠近

$y_1, y_2, \dots, y_m.$，此时的输出就不是一堆噪音图片，而是比较接近手中的这堆真实图片了。

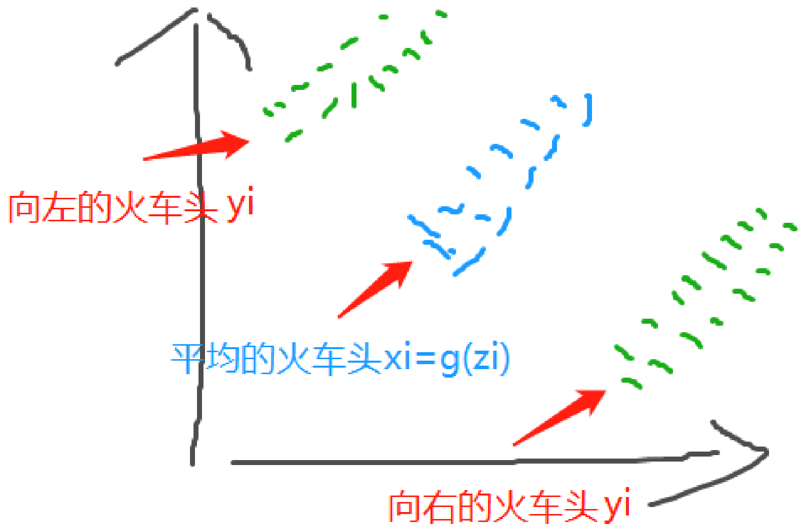
梯度下降法（机器求解）：

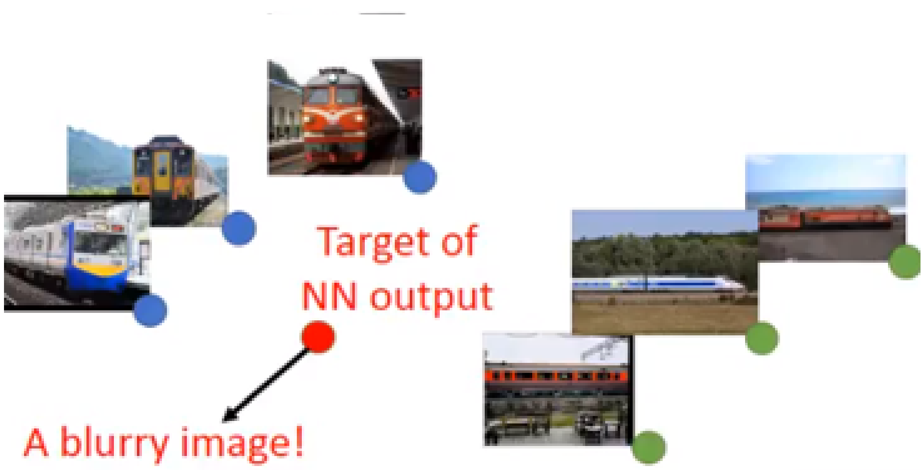




通过改变参数 w_1, w_2 的值，一步一步把 L 的值变小，L 值变小，意味着随机点 $z_1, z_2, \dots, z_m$ 通过 g 映过去的点 $x_1, x_2, \dots, x_m$，离 $y_1, y_2, \dots, y_m$ 很近，也就是说 g 把 z 变成图片了

缺点





这样学到的 g：输出的图片，是原有图片的一种平均，输出的图片可能四不像，比如向左的火车头，和像右的火车头的平均，是平均情况的火车头，可能并不是真的火车头了。

Question：

能否找到一个新的度量这两堆点距离的方法，通过改变参数缩小这个距离 L，能使 g 产生的图片更加真实？

2 寻找新的度量两堆点之间距离的方法

梳理一下我们的目标：

第一步：寻找一个函数 L，只要把我们手上的两堆点 $x_1, x_2, \ldots, x_m$ 和 $y_1, y_2, \ldots, y_m$ 丢进去，就能测量这两堆点的距离

第二步：在距离 L 找到的情况下，通过改变 $g(z;\theta)$ 的参数 θ 的值来缩小距离 L，进而达到让两堆点距离足够近的目的

开始实现：

已知条件

显式条件：

$$x_1, x_2, \ldots, x_m; y_1, y_2, \ldots, y_m$$

隐性条件：

$$p_1(x), p_2(y)$$

那么存在 x 与 y 的联合分布 p（x,y），满足 p(x,y) 的边缘分布分别是 $p_1(x), p_2(y)$。而这样的 p(x,y) 有无穷多个，把这些联合分布放入到一个集合中，记为：

$$\prod(p_1, p_2)$$

从已有知识中（Optimal transport, old and new (Cédric Villani)），找到了如下一个可以度量分布 $p_1(x), p_2(y)$ 之间距离的公式，

$$w(p_1, p_2) = \inf_{\gamma \in \prod(p_1, p_2)} E_{(x,y)\sim\gamma} ||x - y|| \quad (Wasserstein\ distance)$$

$$\xRightarrow{\text{由某定理}} \quad = \frac{1}{K} \sup_{||d||_L \le K} E_{x\sim p_1}[d(x)] - E_{y\sim p_2}[d(y)]$$

其中，K 是任意给定的常数，d 是 K-Lipschitz 连续函数，即满足

$$|d(x_1) - d(x_2)| \le K|x_1 - x_2|$$

的函数.

若记

$$A = \{d \ : \ ||d||_L \le K\}$$

那么

$$\begin{aligned} w(p_1,p_2) &= \frac{1}{K}\sup_{d\in A} E_{x\sim p_1}[d(x)] - E_{y\sim p_2}[d(y)] \\ &\approx \frac{1}{K}\sup_{d\in A}\frac{1}{m}\sum_{i=1}^{m} d(x_i) - \frac{1}{m}\sum_{i=1}^{m} d(y_i) \quad (\text{大数定律}) \end{aligned}$$

对于 $d \in A = \{d \ : \ ||d||_L \le K\}$，由于满足这种条件的 d 无穷无尽，没办法具体表示出来，而我们的神经网络正好有着强大的表示能力，不妨设

$$d = w_3[w_2(w_1 x + b_1) + b_2] + b_3$$

且满足

$$|w_1| \le c, |w_2| \le c, |w_3| \le c$$

容易知道 $||d||_L \le K$

设

$$B = \{d \ : \ d = w_3[w_2(w_1 x + b_1) + b_2] + b_3,\ |w_1| \le c, |w_2| \le c, |w_3| \le c\}$$

那么有 $B \subseteq A$

我们知道，d 的网络层数越多，其表示能力越强，假设上面我们的网络层数调到了合适的位置，此时 $B \approx A$

那么

$$\begin{aligned} w(p_1,p_2) &= \frac{1}{K}\sup_{d\in A}\frac{1}{m}\sum_{i=1}^{m} d(x_i) - \frac{1}{m}\sum_{i=1}^{m} d(y_i) \\ &= \frac{1}{K}\sup_{d\in B}\frac{1}{m}\sum_{i=1}^{m} d(x_i;\omega) - \frac{1}{m}\sum_{i=1}^{m} d(y_i;\omega) \\ &= \frac{1}{K}\arg\max_{\omega}\frac{1}{m}\sum_{i=1}^{m} d(x_i;\omega) - \frac{1}{m}\sum_{i=1}^{m} d(y_i;\omega) \end{aligned}$$

记

$$L = \frac{1}{m}\sum_{i=1}^{m} d(x_i;\omega) - \frac{1}{m}\sum_{i=1}^{m} d(y_i;\omega)$$

回顾已知显式条件：

$$x_1, x_2, \ldots, x_m; y_1, y_2, \ldots, y_m$$

那么将我的这两堆点丢进刚刚设计的神经网络 d 中去，把输出的值丢进 L 中，通过梯度下降法求解 $\arg\max\limits_{\omega} L$, 不妨设此时的参数为 ω_0, 固定下来，在这组参数下的 L 函数，正是这两堆点之间的距离（W-distance）.

此时

$$\begin{aligned} L &= \frac{1}{m}\sum_{i=1}^{m} d(x_i;\omega_0) - \frac{1}{m}\sum_{i=1}^{m} d(y_i;\omega_0) \\ &= \frac{1}{m}\sum_{i=1}^{m} d(g(z_i;\theta);\omega_0) - \frac{1}{m}\sum_{i=1}^{m} d(y_i;\omega_0) \\ &= \frac{1}{m}\sum_{i=1}^{m} d(g(z_i;\theta)) - \frac{1}{m}\sum_{i=1}^{m} d(y_i) \end{aligned}$$

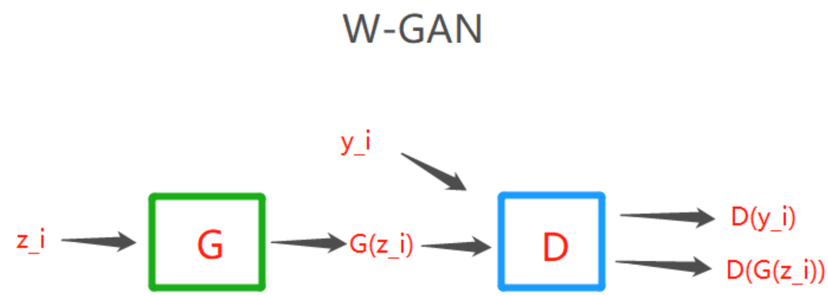
我们知道 $x_i = g(z_i;\theta)$,x_i 是由神经网络 g 生成的，此时我们可以改变网络参数 θ 的值，减小 L 的值，也就是缩小生成的点与给定的点之间的距离，这一步所做的事情就是求解 $\arg\min\limits_{\theta} L$.

综合两步，我们做的事情就是：

$$\arg\min_{\theta}\max_{\omega} L$$

注：第一步是泛函求极值，第二步是普通函数求极值。如果有真正懂的大佬可以在评论区进行补充说明

接下来，请看 W-GAN 的网络框架和算法：



$$\arg\min_{\theta}\max_{\omega} L = \arg\min_{\theta}\max_{\omega} V(G,D) = \arg\min_{G}\max_{D} \frac{1}{m}\sum_{i=1}^{m} D(G(z_i;\theta)) - \frac{1}{m}\sum_{i=1}^{m} D(y_i)$$

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

上面算法的符号说明：

D 函数的参数为 w，G 函数的参数为 θ

算法的第 2-7 行：

固定 G 不动，也就是产生的点 $x_1, x_2, \ldots, x_m$ 是固定的，然后通过梯度下降法更新 D 的参数 ω, 去完成 $\arg\max_D L$ 这件事，最后 ω 到达了一个合适的值，能使此时 ω 下的函数 D 能够让 $L \approx w(p_1, p_2)$.

注：这里 D 必须要满足 K-Lipschitz 连续条件，这里通过第 7 行来近似满足这个条件.

算法的第 9-11 行：

固定 D 不动，此时 L 就是两堆点之间的距离，通过梯度下降法改变 θ 缩小距离 L，从而使 G 产生的点离真实的点更近，这步在前面已经说过了。

3 重新回顾原始的 GAN

损失函数:

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

先看内层:

$$\begin{aligned}\arg\max_D L &= \arg\max_D E_{x\sim P_{data}}[logD(x)] + E_{x\sim P_G}[log(1-D(x))] \\ &= \arg\max_D \int_x P_{data}(x)logD(x)dx + \int_x P_G(x)log(1-D(x))dx \\ &= \arg\max_D \int_x [P_{data}(x)logD(x;\omega)dx + P_G(x)log(1-D(x;\omega))]dx \\ &= \int_x \arg\max_\omega [P_{data}(x)logD(x;\omega)dx + P_G(x)log(1-D(x;\omega))]dx\end{aligned}$$

设 $P_{data}(x) = a, P_G(x) = b, D(x;\omega) = D, f(D) = alogD + blog(1-D)$

$$当D = \frac{a}{a+b} = \frac{P_{data}(x)}{P_{data}(x)+P_G(x)}时, f(D)最大$$

.

接着上面的步骤:

$$\begin{aligned}\arg\max_D L &= \int_x \arg\max_\omega [P_{data}(x)logD(x;\omega)dx + P_G(x)log(1-D(x;\omega))]dx \\ &= \int_x [P_{data}(x)log\frac{P_{data}(x)}{P_{data}(x)+P_G(x)}dx + P_G(x)log(1-\frac{P_{data}(x)}{P_{data}(x)+P_G(x)})]dx \\ &= -2log2 + 2JS(P_{data}(x)||P_G(x))\end{aligned}$$

由此可见，这里的第一步，原来是为了得到 $P_{data}(x)$ 与 $P_G(x)$ 的 JS 散度，和前面 w-GAN 的第一步完全一样，就是找一个函数，可以度量这两堆点分布的距离，后面的分析同上，略.

一个铺垫概念:

这里求解 $\sup_{D\in Vspace} f(D) = \sup_{D\in Vspace} alogD + blog(1-D)$ 的时候, $a, b, D \in Vspace$, 它们都是同一个空间的元素，所以 D 实质也是一个分布函数.

4 更多的距离 f-divergence

直接参见李宏毅 2021 视频–（选修）To learn more-General Framework

另外补充一个实例（https://poloclub.github.io/ganlab/）

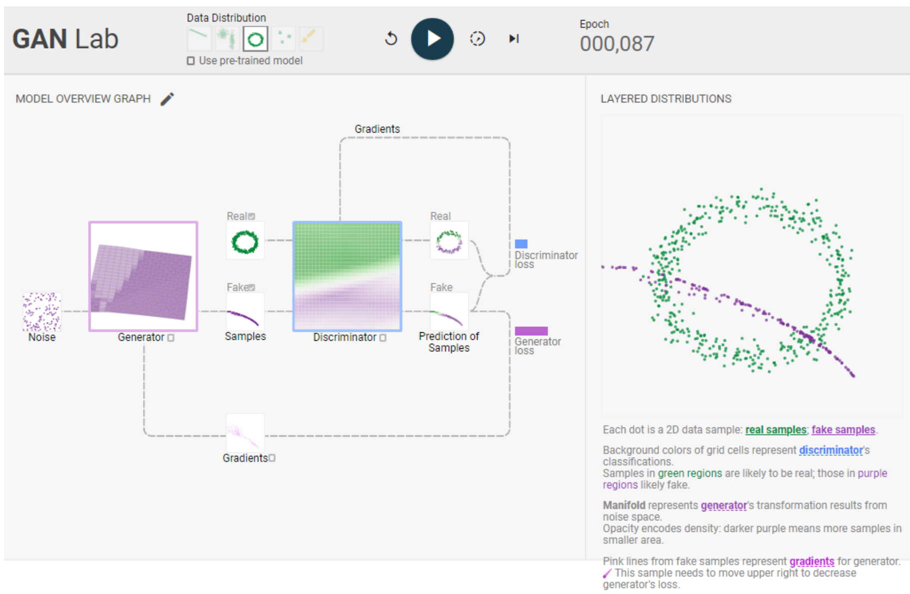


图 1: 随机生成一堆红色的点

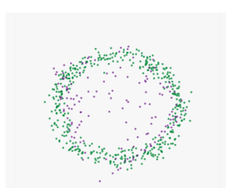


图 2: 训练几次后，红点往绿点偏移

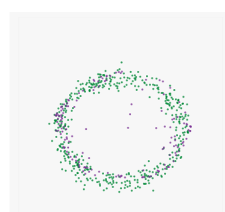


图 3: 多次训练后，整体接近绿色点的分布

5 Conditional GAN

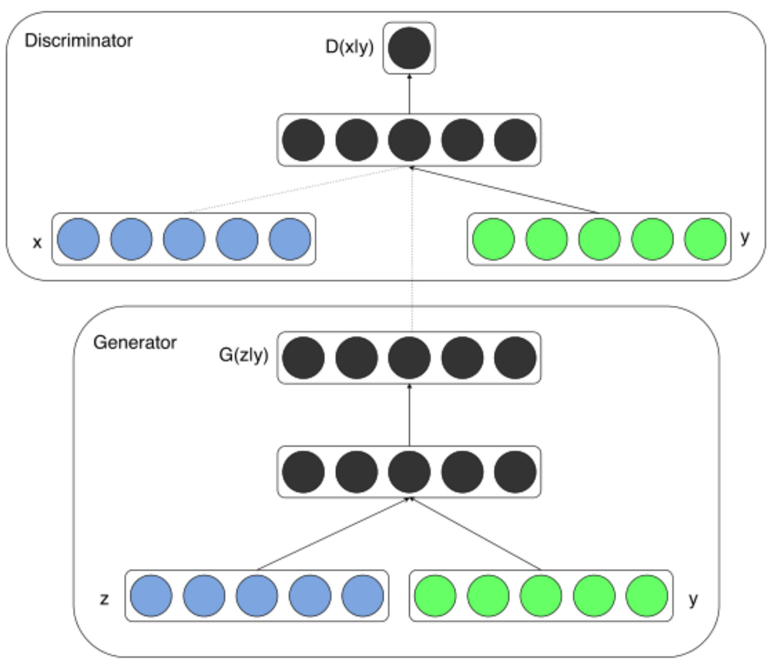
Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y. y could be any kind of auxiliary information,such as class labels or data from other modalities. We can perform the conditioning by feeding y into the both the discriminator and generator as additional input layer.

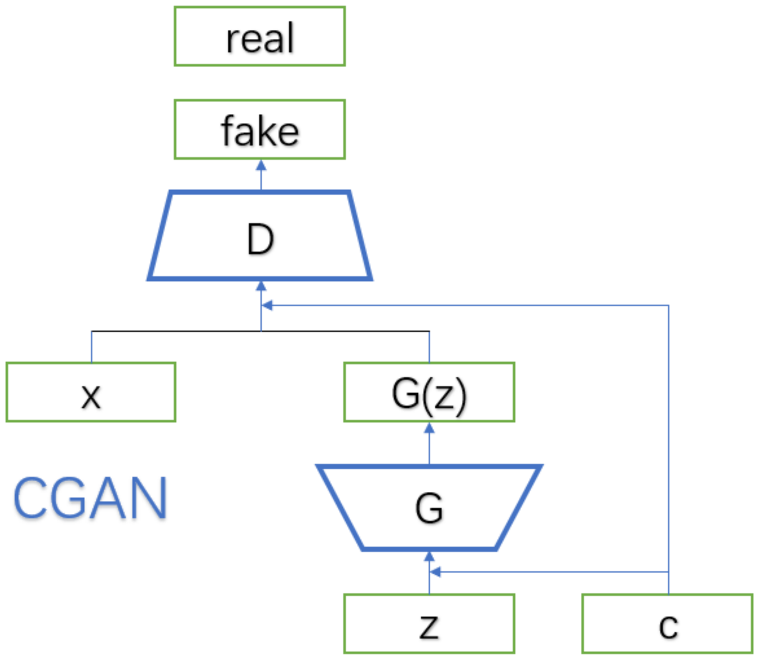
In the generator the prior input noise $P_z(z)$, and y are combined in joint hidden representation, and the adversarial training framework allows for considerable flexibility in how this hidden representation is composed.

In the discriminator x and y are presented as inputs and to a discriminative function (embodied again by a MLP in this case)

The objective function of a two-player minimax game would be as follow:

$$\arg\min_G \max_D E_{x \sim P_{data}(x)}[logD(x|y)] + E_{z \sim P_z(z)}[log(1 - D(G(z|y)))]$$





根据网络框架结构，稍微修改一下式子如下：

$$\arg\min_G \max_D E_{x \sim P_{data}(x)}[logD(x|y)] + E_{z \sim P_z(z)}[log(1-D(G(z|y)|y))]$$

实际训练的时候是一堆离散的点，公式如下：

$$\arg\min_G \max_D \frac{1}{m}\sum_{i=1}^{m}[logD(x_i|c_i)] + \frac{1}{m}\sum_{i=1}^{m}[log(1-D(G(z_i|c_i)|c_i))]$$

理论分析：

第一步：获得衡量 $P_{data}(x|c)$和$P_G(x|c)$ 这两个条件概率分布差距的公式

第二步：通过缩小上述差距，使得 $P_G(x|c)$接近$P_{data}(x|c)$

对于如何做到公式和网络框架的对应，必须回答以下两个问题：

Question1: 条件概率分布该如何理解？请对应实例进行解释。

Question2: 为什么上述以条件概率呈现的式子，网络结构是上述形状呢？也就是为什么把两个向量拼接输入，就认为得到了 $G(z_i|c_i)$和$D(x_i|c_i)$呢？