本文第一部分详解 GAN 的原理，然后描述 GAN 不稳定的两个难点以及各自的解决方案。

一．GAN 的原理：

GAN 是一种生成模型，包括两个部分：生成器和判别器。

考虑我们需要一个生成模型，令 $P_G(x;\theta)$是一个以 θ 为参数的分布，则生成模型的目标是找到一个 θ*使 $P_G(x;\theta)$与 $P_{data}(x)$最接近。

所以需要衡量这两个分布的距离，如果从最大似然的角度考虑，从 $P_{data}(x)$中 sample 一组数据点（即真实数据集），则这些数据点的似然概率为：

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

最大这个似然概率：

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

$$=\arg\max_\theta\sum_i^m\log P_G(x^i;\theta)$$

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

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

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

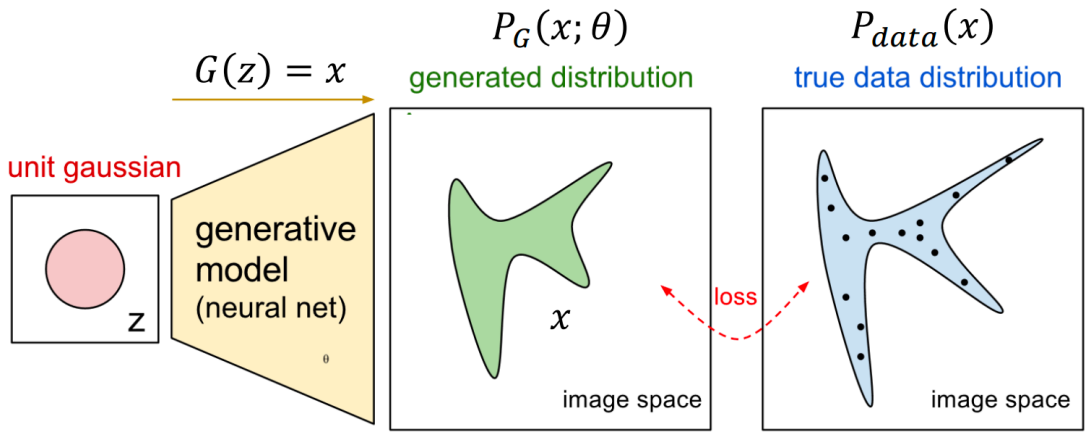
$$=\arg\min_\theta KL(P_{data}(x)||P_G(x;\theta))$$

发现求最大似然相当于使这两个分布的 KL 散度最小，θ*= $arg\min_\theta KL\left(P_{data}(x)||P_G(x;\theta)\right)$。为了使 $P_G(x;\theta)$足够灵活可以逼近真实数据分布，GAN 用 NN 来构造这个生成器。

生成器 $G(z,\theta_g)$的输入是从某个固定分布（比如正态分布）中采样的一个向量，输出一副生成的图片（这个图片也可以看做是向量），希望它尽可能 realistic，也就是说，生成器把正态分布扭转为一个接近真实数据分布的分布。

这样 $P_G(x)$可以写成：$$P_G(x)=\int_z P_{prior}(z)I_{[G(z)=x]}dz$$，但是这个 $P_G(x)$无法显式的写出来，所以无法用解最大似然的方法求 θ。此外，使用 KL 散度来衡量 P_G 与 P_{data} 的距离本身也存在问题[4]。由于 KL 散度的不对称性，如果最小化 $KL(P_{data}||P_G)$，如果在一些区域，P_{data} 的值很大，P_G 趋近于 0，kl 散度趋近于正无穷，也就是说生成器没有覆盖这些数据的情况会被赋予很高的 cost；而反过来如果生成器产生的数据不逼真，P_{data} 趋近于 0，kl 散度趋于 0，

则赋予的 cost 很低。对两种情况的惩罚相差很多，所以用 kl 散度衡量生成器的效果来训练生成器是不适合的。



由于无法直接衡量两个分布是否接近，GAN 又构造了一个判别器 $D(x, \theta_d)$，输入一个数据，输出一个描述概率的标量，描述这个数据有多大的可能来自真实数据集。构造 value function V(G,D)：

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

$\max_D V(G,D)$ 的作用类似于 kl 散度，给定生成器 G，$\max_D V(G,D)$ 可以衡量 $P_G(x)$ 与 $P_{data}(x)$ 的差距，则使这个差距最小的 G 即为所求：

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

1. 为什么 $\max_D V(G,D)$ 可以衡量 $P_G(x)$ 与 $P_{data}(x)$ 的差距？

如果 G 固定，寻找最优判别器 D*:

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

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

$$= \int \left[P_{data}(x)logD(x) + P_G(x)log\left(1-D(x)\right)\right]dx$$

要使这个积分最大，需要寻找 D* 对给定的 x，$P_{data}(x)logD(x) + P_G(x)log\left(1-D(x)\right)$ 最大。

$$f(D) = a\log(D) + b\log(1-D)$$

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

$$a \times \frac{1}{D^*} = b \times \frac{1}{1-D^*}$$

$$\Leftrightarrow a \times (1-D^*) = b \times D^*$$

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

找到最优的判别器 D*后，带入 V(G,D)，得到 $\max_D V(G,D)$：

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

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)$$

不同于 kl 散度是非对称的，js 散度是对称的，范围在 0~log2 定义如下：

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

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

可以看到 $\max_D V(G,D)$ 等同于 js 散度加上常数，最优生成器 G*使 $\max_D V(G,D)$ 最小，则等同于使 $P_G(x) = P_{data}(x)$

2. 为什么设置这样一个 value function V(G,D)？

先解释一下熵的概念，熵的本质是香农信息量 log(1/p)的期望。即一个事件发生的概率为 p，如果它发生了，携带的信息量为 log(1/p)，发生概率越小的事件，发生后提供越多信息，在哈弗曼编码中，出现概率越低的字母需要的编码长度越长。如果有关于样本集的 2 个概率分布 p 和 q，其中 p 为真实分布，q 非真实分布。按照真实分布 p 来衡量识别一个样本的所需要的编码长度的期望(即平均编码长度)为：

$$H(p) = \sum_i p(i) * log\frac{1}{p(i)}$$

如果使用错误分布 q 来表示来自真实分布 p 的平均编码长度，则应该是：

$$H(p,q) = \sum_i p(i) * log\frac{1}{q(i)}$$

因为用 q 来编码的样本来自分布 p，所以期望 H(p,q)中概率是 p(i)。H(p,q)我们称之为“交叉熵”。H(p,q)恒大于等于 H(p)，两者之差为相对熵也就是 kl 散度。交叉熵可以用作损失函数，p 表示真实标记的分布，q 则为训练后的模型的预测标记分布，交叉熵损失函数可以衡量 p 与 q 的相似性。

对于 $V = E_{x\sim P_{data}}[logD(x)] + E_{x\sim P_G}[log(1-D(x))]$，由于无法用积分计算期望，所以需要从真实数据中和生成分布中采样近似：

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

这个 value function 其实来自于二分类问题优化的时候常用的交叉熵误差。类似于二分类逻辑回归（y = sigmoid(w x)，而 D(x) = sigmoid(f(x)),f(x)为 NN），y = D(x)是 0~1 之间的值，表示 x 属于第一类的概率。对于数据集（x_n，t_n），似然函数可以写成：$p(t_n) = \prod_{n=1}^{N} y_n^{t_n}\{1-y_n\}^{1-t_n}$

负似然对数 $V = -\ln p(t_n) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$

可以看到这个 loss function 就是交叉熵误差函数。

当 t_n 为正例时，最小化 $\ln y_n$；当 t_n 为负例时，最小化 $\ln(1-y_n)$。对于判别器 D，来自真实数据集的都是正例，来自生成器 G 的为负例。最小化交叉熵误差相当于最大化下式：

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

3. 如何用 GAN 进行训练？

训练 GAN 时，轮流训练 G 和 D，对于给定的初始 G_0，找到 D_0*，使 $V(G_0,D)$最大，$V(G_0, D_0^*)$就是 $P_{G0}(x)$与 $P_{data}(x)$的 JS 散度。然后固定 D_0*，训练 G 来最小化 $V(G, D_0^*)$，使 JS 散度最小。

而这里存在一个问题，在 G 改变的过程中，$V(G, D_0^*)$其实不会保持是 $P_G(x)$与 $P_{data}(x)$的 js 散度，因为 G 改变了，比如 G 在梯度下降的过程中从 G_0 下降了一步为 G_1，这时 $P_G(x)$与 $P_{data}(x)$的 js 散度应该是 $V(G, D_1^*)$，但是继续梯度下降时仍然以 $V(G, D_0^*)$作为 loss。所以不应该让 G 一直训练到找到 $V(G, D_0^*)$最小值为止，而是 G 训练一次（优化 G 时 D_0^*与 D_1^*可看做相等），然后重新训练 D，训练 D 时可以重复多次，因为要找到局部最大值。（理论上是这样，实际操作时，如果判别器训练到收敛，可以正确区分所有样本，会导致梯度消失问题，详见第三部分）

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:

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

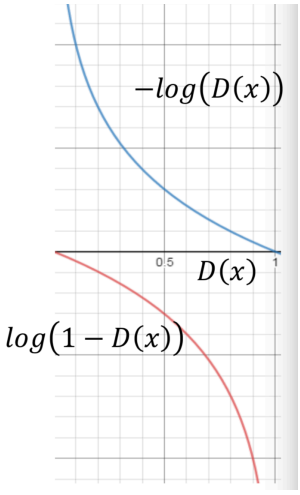
end for

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

4. 生成器 loss function 的替代函数 $E_{x\sim P_G}[-log(D(x))]$

训练初期，生成器生成的结果很差，判别器很容易区分出生成数据和真实数据。此时生成数据 x 的 D(x)在 0 附近，而 log(1-D(x))在此处梯度很小，非常平滑。生成器希望可以生成数据使 D(x)增大，但是通过这个 function 进行训练时生成器训练的很慢。

在原始论文中[5]，作者提出了一个生成器替代的 loss function：$E_{x\sim P_G}[-log(D(x))]$，这个函数在 D(x)在 0 附近时非常陡峭，当生成器生成的数据越来越好时，训练的速度会越来越慢。直观上看，原始的函数是要最小化 D 判断生成数据是假的概率，替代函数是要最大化 D 判断自己是真的概率，函数的单调性上是一致的。

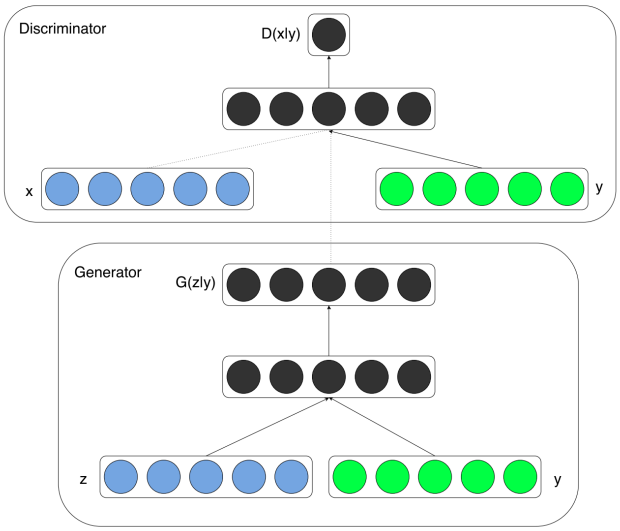


二．训练难点：神经网络本身不稳定导致的 GAN 过于自由

目前 GAN 的一般都选择 NN 作为生成器和判别器，所以生成的结果很不稳定，一种改良的方案是给 GAN 加入一些外部信息进行指导。

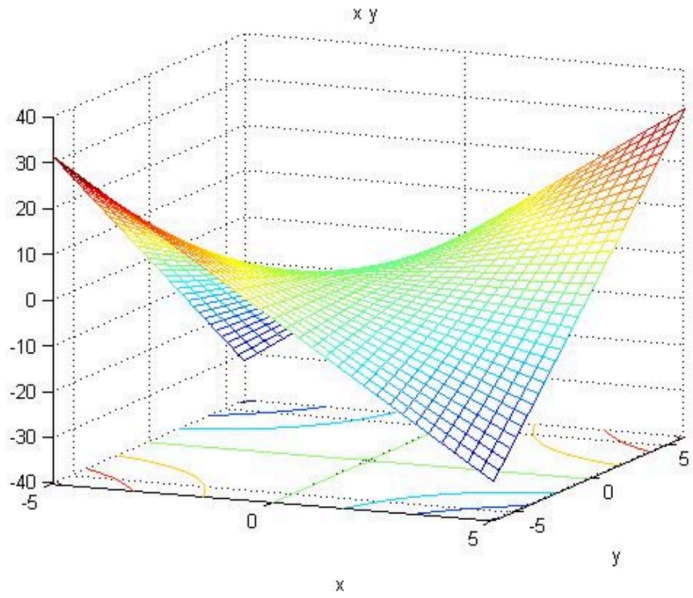
Conditional GAN[1]，是一种由 GAN 拓展出的 conditional 版本，CGAN 在生成器和判别器上都加上了一个额外的信息 y：在生成器中，将随机噪音 z 与 y 在隐藏层组合起来，在判别器中，将 x 和 y 作为输入。

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



CGAN 应用在基于 label 进行生成的场景中，比如在 mnist 数据集中，将表示数字的 one hot vector 作为 y 输入，这样可以产生指定数字的图片。Y 也可以是图片，可以基于一个图片生成文字或图片，比如为一个图片打标签（一个图片可以对应多个词），可以将图片的 vector 作为 y，来生成能描述该图片的标签 x。

Improved GAN[2]是针对 GAN 存在的难以收敛的问题提出的一些改良的技术。GAN 可以形式化表示为零和博弈，判别器控制 θd，将函数 $v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$ 作为收益，生成器控制 θg，将函数 $-v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$ 作为收益，两个玩家都想要最大化各自的收益。如果可以达到纳什均衡点，生成器的收益基于 θg 达到了最大，判别器亦然。可以证明如果 $\max_d v(g,d)$ 在 θg 中是凸的，那么这个过程最终可以收敛，而由 NN 构造的生成器和判别器往往使该函数非凸，所以可能不能达到平衡。举例来说[3]，value function 为 v(x,y) = xy，两个玩家分别控制 x、y 来减小 cost：xy、-xy。



能达到平衡的点（鞍点）是(0,0)，由于此时 v=0，这个点相对于第一个玩家是最小值，对于第二个玩家来说是最大值。如果两个玩家以固定的步长进行梯度下降，每个玩家以另一个玩家为代价降低自己的 cost，可能不会收敛到原点，而是进入稳定的圆形轨迹。比如从(2,3)开始，步长为 1 两个玩家轮流进行梯度下降，最终会回到（2,3）。

Improved GAN 为了鼓励 GAN 收敛提出了几个 trick，重点有两个：其一是 Feature matching，由于常规的 GAN 往往 G 比 D 要弱，Improved GAN 为 GAN 设置了新的目标函数，鼓励生成器生成的数据符合真实数据的统计规律，而真实数据的规律由判别器提供，也就是说，既然判别器往往比较强，那么判别器将它学到的真实数据的特征告诉生成器。具体就是以真实数据在判别器的中间层输出作为目标，让生成器产生的数据来匹配。生成器的目标为：$||\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\mathbf{f}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\mathbf{f}(G(\boldsymbol{z}))||_2^2$，其中 f(x)是 D 的中间层；判别器的目标和普通 GAN 相同。

其二是 Minibatch discrimination，GAN 模型最常见的一种失效情况就是训练出来的参数使生成器只能生成一种数据，也就是 G 收敛到了同一个点。出现这种情况是因为判别器是单独对每一个样本做判断，而不能感知样本之间是否相似。Minibatch discrimination 的概念就是使判别器对样本以组合的观点做判断而不是单独的一个样本。具体方法是在判别器的中间层，除了得到的样本特征 f(x)本身，再加入一个 side information：对于输入 xi，$\mathbf{f}(\boldsymbol{x}_i) \in \mathbb{R}^A$，乘以一个 A*B*C 的张量，得到矩阵 $M_i \in \mathbb{R}^{B\times C}$.

$$c_b(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-||M_{i,b} - M_{j,b}||_{L_1}) \in \mathbb{R}.$$

$$o(\boldsymbol{x}_i)_b = \sum_{j=1}^{n} c_b(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathbb{R}$$

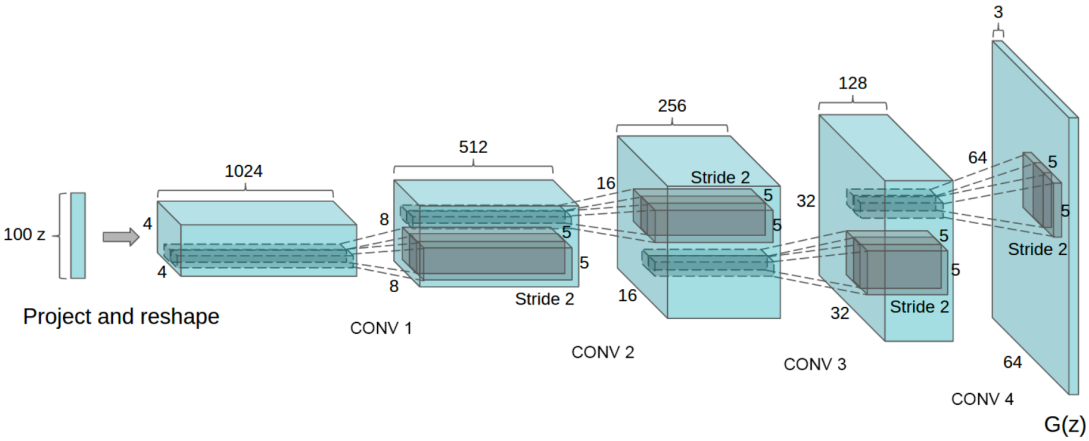
$$o(\boldsymbol{x}_i) = \left[o(\boldsymbol{x}_i)_1, o(\boldsymbol{x}_i)_2, \ldots, o(\boldsymbol{x}_i)_B\right] \in \mathbb{R}^B$$

$$o(\mathbf{X}) \in \mathbb{R}^{n \times B}$$

将 $o(x_i)$与 $f(x_i)$组装到一起，输入判别器的下一层，这样就能够使得判别器知道该样本是否与同批次其他样本过于相似。这样鼓励 D 学习到多样性的样本，从而 G 能生成多样性的样本，而不是仅仅生成逼真的样本。

第二种改良方案则对生成器和判别器使用的网络结构进行创新，比如 DCGAN[7]，pix2pix[8]。

DCGAN[7]为 GAN 找到了一套可以稳定训练的卷积网络结构，对 CNN 进行进行了移除全连接层、将 pooling 层用 convolution 代替、生成器和判别器都使用 batchnorm 等变化：



pix2pix[8]是一种 conditional GAN 的结构。针对于 image-to-image translation 任务，使用 cGAN 不只能学习一个从输入图片到输出图片的映射（生成器），还能学习一个 loss function 来训练这个映射（判别器），所以不同上文描述的 cGAN，判别器的输入从一张图片变成了一对图片，判断从输入图片到输出图片是否是一个真实的，好的转换。

由于判别器的输入时原始图片和目标图片\生成图片，cGAN 的目标变为：

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{x,y}[\log D(x, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))],\end{aligned}$$

除了 GAN 的 loss function 外，pix2pix 还加入了图片翻译任务使用的传统的 loss：l1 loss（由于 l2 loss 更容易产生模糊的图片）。

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

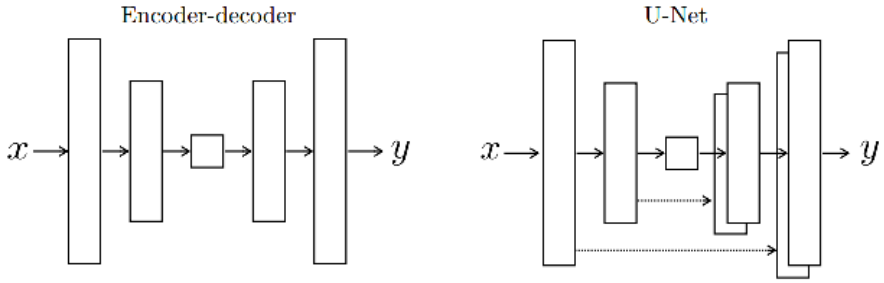
最终的目标函数则变为：$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

1. 如果没有噪音 z，生成器仍然可以学到输入图片到输出图片的映射，但是得到的只有 y 单个点，而不是 y 的分布（类似于 AE）。而直接将 z 作为生成器的输入，会导致生成器试图忽略 z，因为 z 没有提供有用的信息。所以 pix2pix 将 z 以 dropout 的形式，加入到了生成器

的多个层。

这种操作实际上已经不同于 GAN，生成器学习一个映射 f 从一个随机的分布比如正态分布扭转到真实数据分布，判别器学习生成的分布是否符合真实分布；在本任务中，生成器学习到的映射是从一种数据的分布扭转到另外一种数据的分布，噪音 z 起到的作用只是防止过拟合，判别器学习这两种数据的转换是否良好。这是因为本任务是一对一的。

2. pix2pix 的第二点改进也是它最大的贡献在于对 GAN 使用的网络结构进行了改良：不同于 DCGAN 使用的卷积神经网络，pix2pix 使用了一种 U-Net 的结构，在 AE 的基础上，为以 bottleneck 为中心对称的层增加了 skip connection，将 encoder 的所有通道直接连通给了 decoder 对应的层。如果没有 skip connection，图片会压缩到非常抽象的层，再还原出来会损失一些信息，而这样做一定程度上保留了高维的特征。



3. 判别器使用了 PatchGAN 的技术，即将图片切分为固定大小的 patch 输入到 D 中进行判断，再对所有判断结果求均值得到最终结果。作者认为使用了 L1 loss 已经可以学到“低频”的特征，对于“高频”的特征，使用图片的一部分就足够学到了。而这样会减少网络的参数，加速训练。同时也可以适用于任意大小的图片。

三．训练难点：选择的目标函数本身不合理

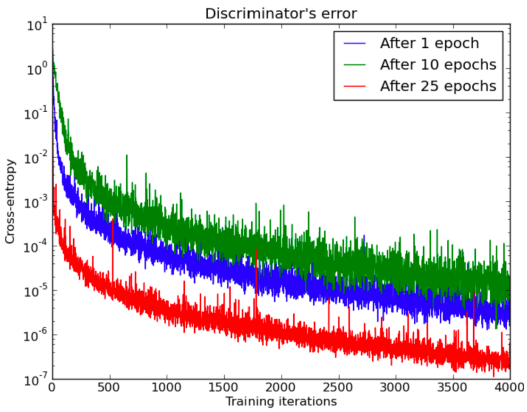
在本文第一部分的提到了一个训练的 trick，训练判别器时不能训练的太好，否则会导致生成器梯度消失问题；这个现象在[4]中给出了具体的理论解释。

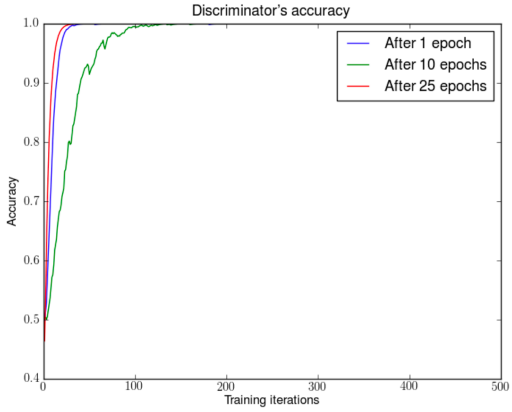
已知当判别器达到最优时，生成器的优化目标$\max_D V(G,D)$相当于 $P_G(x)$与 $P_{data}(x)$的 js 散度。而 js 散度，在两个分布没有重叠或者重叠可以忽略时，一直是 log2，而不会随着两个分布的改变而改变。（分布不重叠意味着对于 x，或者 $P_G(x)=0$，$P_{data}(x)\neq 0$，或者 $P_G(x)\neq 0$，$P_{data}(x)=0$，这两种情况下 js 散度都是 log2。）这样导致了生成器的目标函数为常数，梯度下降时梯度为 0，梯度消失的问题出现了。

如果判别器达到最优（训练到收敛）时，能够完全区分来自真实数据（D(x)=1）和生成数据（D(x)=0）的样本，那么将这个 D* 代入 $V = E_{x\sim P_{data}}[logD(x)] + E_{x\sim P_G}[log(1-D(x))]$ 等于 0，由于 $\max_D V(G,D)$ 等于 $2JSD(P_G(x)||P_{data}(x)) - 2log2$，可知 js 散度为 log2，也就是说认为这两个分布

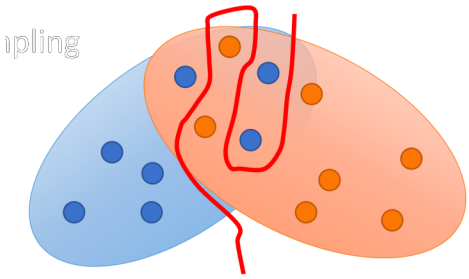
不重叠。

而在实验中发现，训练判别器时准确率总是会趋近 100%，loss 都趋近于 0，也就是说，如果判别器训练到收敛，梯度消失总是会发生。





出现这样的现象有两个原因，一个是我们并不是真的得到了两个分布进行计算的，而是通过采样进行计算的，有可能分布本身是有重叠的，但是对于两种样本，判别器仍然能够找到一个分割面进行区分：



另一个原因则是通过生成器生成的分布本身与真实分布不会重叠，因为当 $P_G(x)$与 $P_{data}(x)$的支撑集（support）是高维空间中的低维流形（manifold）时，$P_G(x)$与 $P_{data}(x)$重叠部分测度（measure）为 0 的概率为 1。

支撑集（support）：函数的非零部分子集，比如 relu 函数的支撑集就是(0,+∞)，一个概率分布的支撑集就是所有概率密度非零部分的集合。

流形（manifold）：高维空间中曲线曲面的拓广，三维空间中的一个曲面是一个二维流形，因为它的本质维度（intrinsic dimension）只有 2，一个点在这个二维流形上移动只有两个方向的自由度。同理，三维空间或者二维空间中的一条曲线都是一个一维流形。

测度（measure）：是高维空间中长度、面积、体积概念的拓广。

生成器是从一个低维空间的采样 z，再经过一个映射 g(z)得到高维数据 x，P_{data} 的支撑集被 g(z)限定，在高维空间不连续，是一个低维(z 的维度)的流形。在高维空间中，两个低维流形重叠的概率为 0。就好像在二维空间中，两条线之间有重叠线段的概率为 0，而两条线可能存在交点，但是点比线低一维度，长度（测度）为 0，可以忽略。所以 js 散度总为 log2，产生了梯度消失现象。

而原始 GAN 论文[5]提到生成器的替代 loss function $E_{x\sim P_G}[-log(D(x))]$，同样存在问题。将生成数据和真实数据 kl 散度转化成包含 D*的形式：

$$
\begin{aligned}
KL(P_g||P_r) &= \mathbb{E}_{x\sim P_g}[\log\frac{P_g(x)}{P_r(x)}] \\
&= \mathbb{E}_{x\sim P_g}[\log\frac{P_g(x)/(P_r(x)+P_g(x))}{P_r(x)/(P_r(x)+P_g(x))}] \\
&= \mathbb{E}_{x\sim P_g}[\log\frac{1-D^*(x)}{D^*(x)}] \\
&= \mathbb{E}_{x\sim P_g}\log[1-D^*(x)] - \mathbb{E}_{x\sim P_g}\log D^*(x)
\end{aligned}
$$

则目标函数可以化成：

$$
\begin{aligned}
\mathbb{E}_{x\sim P_g}[-\log D^*(x)] &= KL(P_g||P_r) - \mathbb{E}_{x\sim P_g}\log[1-D^*(x)] \\
&= KL(P_g||P_r) - 2JS(P_r||P_g) + 2\log 2 + \mathbb{E}_{x\sim P_r}[\log D^*(x)]
\end{aligned}
$$

后两项与生成器无关，所以最小化这个 loss function 相当于最小化 $KL(P_g||P_r) - 2JS(P_r||P_g)$。但是 kl 散度和 js 散度是同向的，最小化一个的同时最大化另外一个，会导致训练方向不稳定。此外，在本文第一部分论述了，kl 散度由于它的不对称性，对生成了不真实的样本惩罚非常大，对生成不了真实样本惩罚几乎为 0，这样就会导致生成器丧失多样性，只能生成某一种逼真的样本，也就是 mode collapse 的情况。

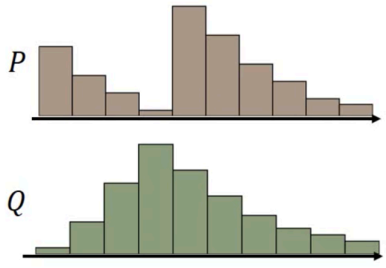
梯度消失问题有两个主要的解决方法，一个是为真实样本和生成样本增加噪音，另一个是抛弃 JS 散度、KL 散度，使用其他的距离，WGAN[6]就是后者。

WGAN 使用了 wasseratein 距离，也称为 Earth-Mover（推土机）距离，定义如下：

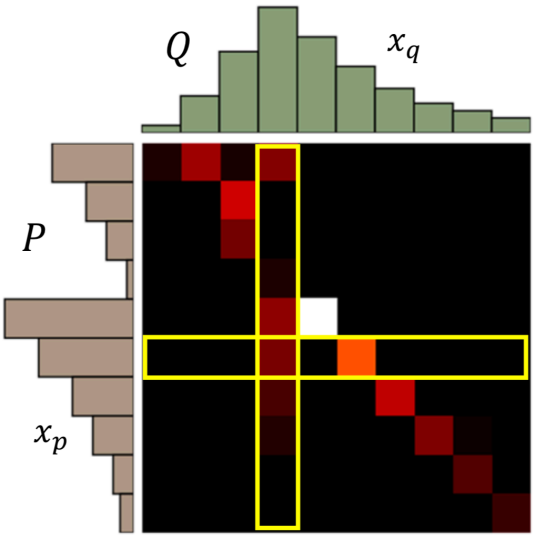
$$
W(\mathbb{P}_r,\mathbb{P}_g) = \inf_{\gamma\in\Pi(\mathbb{P}_r,\mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\,\|x-y\|\,\right]
$$

其中$\prod(P_r, P_g)$代表 P_r、P_g 所有可能的联合概率分布的集合。 $\gamma(x,y)$代表了在 P_r 中出现 x 同时在 P_g 中出现 y 的概率，γ 的边缘分布分别为 P_r 和 P_g。在这个联合分布下可以求得所有 x 与 y 距离的期望，存在某个联合分布使这个期望最小，这个期望的下确界（infimum）就是 P_r、P_g 的 EM 距离。

直观上看，如果两个分布是两堆土，希望把其中的一堆土移成另一堆土的位置和形状，有很多种可能的方案。



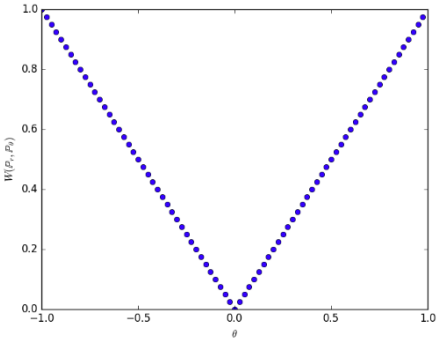
每一种方案可以对应于两个分布的一种联合概率分布，$\gamma(x,y)$代表了在 P_r 中从 x 的位置移动 $\gamma(x,y)$的土量到 P_g 中的 y 位置，对所有的 x 按 γ 移动，则可将分布 P_r 转化成 P_g。在所有的方案中，存在一种推土代价最小的方案，这个代价就称为两个分布推土机距离。

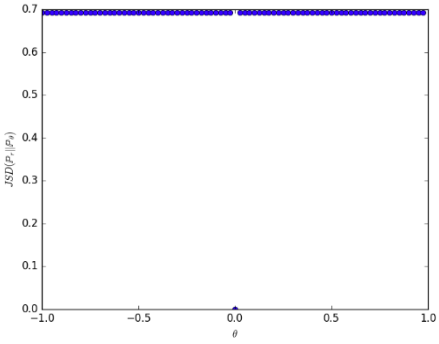


EM 距离与 JS 散度，KL 散度的区别在于，这个距离在两个分布不重叠的时候也是连续的。在二维空间中存在两个分布，P0 是在（0，1）上的均匀分布，P1 是在（θ,1）上的均匀分布，那么根据定义可得：

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$

可以看到当 θ 趋近于 0 时，在 EM 距离衡量下，P1 会趋近于 P0。而其他距离在两个分布不重叠的时候是常量，在两个分布重叠时有一个突变。所以在低维流形上学习概率分布时，可以通过 EM 距离进行梯度下降，而不能通过 JS 距离，因为它作为 loss function 是不连续的。





如何利用 EM 距离来训练生成模型呢，由于无法找出所有的 γ 从中找最优，文章给出了下面一个代替的公式：

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

L<1 代表 f 是一个 1-Lipsschitz 函数。K-Lipsschitz 函数定义为对于 K >0，||f(x1) − f(x2)||≤K||x1 − x2||。在这里 K=1，也就是说 f 的导数不能超过 1（如果 K 取 k 的话，求得的距离则是 k 倍的 EM 距离。），限制了 f 不能改变的太快，在这个前提下，找到 f 使 $\mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)$最大，这个上确

界（supremum）即为 EM 距离。f(x)就可以用 nn 来拟合了，类似于普通 GAN 的判别器一样。

要限制 f 是一个 k-Lipsschitz 函数，WGAN 给出的方案是做 weight clipping，即限制构成 f 的 NN 的参数 w 的范围在[-c,c]之间，经过梯度下降更新后，一旦某个 w 超过了这个范围，则把 w 修剪为-c 或 c。这样因为 w 的范围固定，那么随着输入的改变，输出的改变一定在一个有限的范围内，存在 k 使 f 满足 K-Lipsschitz。

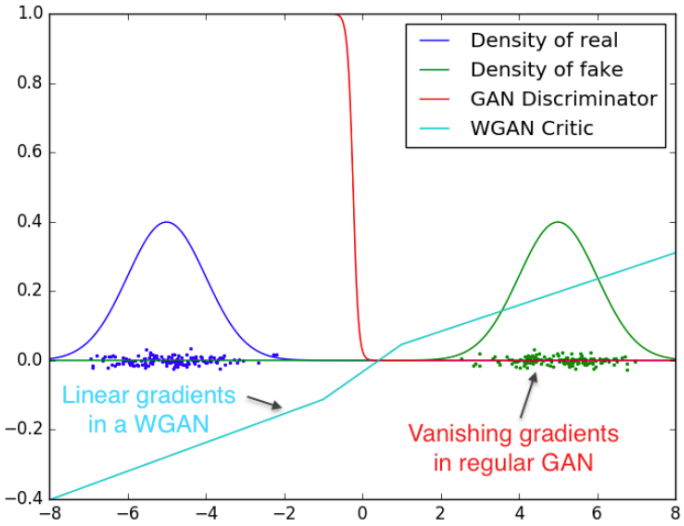
WGAN 的算法如下，与普通 GAN 对比，判别器和生成器的 loss function 都去掉了 log，判别器的最后一层去掉了 sigmoid（因为不用限制 D 的输出是 0~1），以及为判别器网络增加了 clip

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

使用 WGAN 训练出来的最优判别器没有梯度消失的问题，下图是两个正态分布，可以看到 GAN 的最优判别器得到了一个 sigmoid function，完全区分开了两种分布的样本，基于本章开头的理论分析，梯度消失已经发生了，从图上也可以直观的看到，D(x)在取样处的梯度为常数，没有有效的梯度信息传给生成器；而 WGAN 限制了判别器不能太陡峭，找到了一条接近直线的线，梯度是线性的。



Reference：

[1].Mirza M, Osindero S. Conditional generative adversarial nets[J]. arXiv preprint arXiv:1411.1784, 2014.

[2].Salimans T, Goodfellow I, Zaremba W, et al. Improved techniques for training gans[C]//Advances in Neural Information Processing Systems. 2016: 2234-2242.

[3].LeCun Y, Bengio Y, Hinton G. Deep learning[J]. nature, 2015, 521(7553): 436.

[4].Arjovsky M, Bottou L. Towards principled methods for training generative adversarial networks[J]. arXiv preprint arXiv:1701.04862, 2017.

[5].Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.

[6].Arjovsky M, Chintala S, Bottou L. Wasserstein gan[J]. arXiv preprint arXiv:1701.07875, 2017.

[7].Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks[J]. arXiv preprint arXiv:1511.06434, 2015.

[8].Isola P, Zhu J Y, Zhou T, et al. Image-to-image translation with conditional adversarial networks[J]. arXiv preprint, 2017.