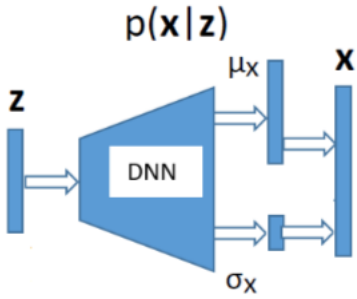
4.2 Generative Adversarial Networks

回顾：变分自动编码器 Variational Autoencoder/VAE



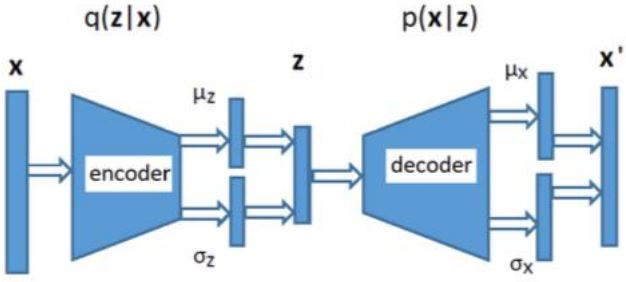
1. 其目的在于学习一个解码器，可将一个低维隐向量通过深度网络映射成一个高维数据空间当中的概率分布$p_\theta(\boldsymbol{x}|\boldsymbol{z})$。
2. VAE 解码器可用于生成与训练集相似但不同的新数据：

$$\boldsymbol{z} \sim p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

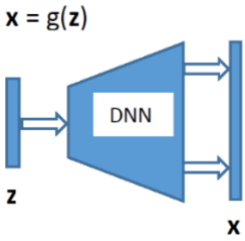
$$\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|\boldsymbol{z})$$

过程为：在$\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$当中为$\boldsymbol{z}$采样，输入解码器，解码器通过训练好的网络得到$\mu_x$和$\sigma_x$，进而得到一个概率分布$p_\theta(\boldsymbol{x}|\boldsymbol{z})$，之后在$p_\theta(\boldsymbol{x}|\boldsymbol{z})$当中为$\boldsymbol{x}$采样，可以得到一些$\boldsymbol{x}$。

3. 解码器同样明确定义了一个概率分布$p(x) = \int p(x|z)p(z)dz$，可用于近似计算点$\boldsymbol{x}$处的概率密度函数$p(\boldsymbol{x})$。
4. 编码器/识别模型可以用于提取高维数据的低维特征。
5. VAE 的学习过程是通过最小化真实数据的分布和模型的数据分布之间的 KL 散度，同时训练编码器和解码器的神经网络模型参数$\boldsymbol{\theta}, \boldsymbol{\phi}$,
6. 以上的优化问题难以解决，因此引入了编码器/识别模型，进而得到实际似然函数的变分下确界；从而实际上，VAE 通过最大化变分下确界来最大化似然函数，得到相应的模型参数$\boldsymbol{\theta}, \boldsymbol{\phi}$。



生成对抗网络 Generative Adversarial Networks/GAN

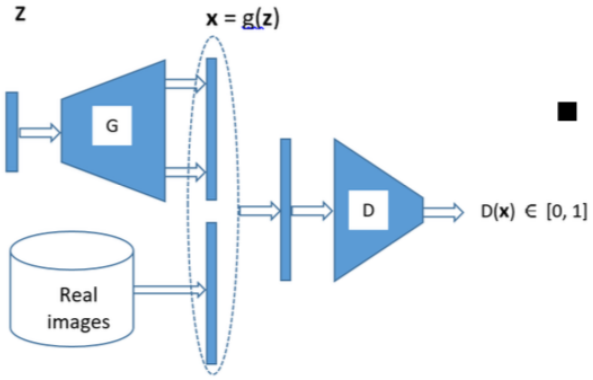


1. 其目的在于学习一个生成器 generator，其可以将一个低维隐向量通过深度网络映射成一个数据空间当中的高维向量，即$\boldsymbol{x} = g(\boldsymbol{z})$。
2. 生成器可用于生成与训练集相似但不同的新数据：

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = g(\boldsymbol{z})$$

过程为：在$p(\boldsymbol{z})$当中为$\boldsymbol{z}$采样，输入解码器，解码器通过训练好的网络直接得到$\boldsymbol{x}$。

3. 生成器并没有定义$\boldsymbol{x}$在数据空间上的概率分布，从而无法计算点$\boldsymbol{x}$处的概率密度函数$p(\boldsymbol{x})$。
4. GAN 无法用于提取高维数据的低维特征。
5. GAN 的学习过程是通过最小化真实数据的分布p_r和模型的数据分布p_θ之间的 JS 散度$JS(p_r||p_\theta)$，来得到相对应的模型参数$\boldsymbol{\theta}$
6. 和 VAE 遇到的问题一样，因为 JS 散度的梯度无法追踪，从而引入一个鉴别器 discriminator D，用于近似 JS 散度的梯度；其同样为一个深度神经网络，作用是将一个向量$\boldsymbol{x}$映射至区间[0,1]。输入$\boldsymbol{x}$可以为一个真实的图像 real example，或者是一个伪造的图像 fake example，输出的$D(\boldsymbol{x})$是输入的图像为真实图像的概率。



7. 不同于 VAE 当中编码器和解码器是同时进行训练的，GAN 中的生成器和鉴别器是不同步训练的：

训练鉴别器时，目标任务是让鉴别器有效分别出 real 和 fake example，即找到参数θ_d，使得鉴别器在输入为真实图像时，$D(\boldsymbol{x}) = 1$或者$D(\boldsymbol{x})$近似于 1；

输入为伪造图像时，$D(\boldsymbol{x})=0$或者$D(\boldsymbol{x})$近似于 0；

训练生成器时，目标任务是找到参数θ_g，使得生成器生成一系列令鉴别器无法判别真伪的图像；

从而相互对抗互相学习，因而称之为生成对抗网络。

8. 具体的算法迭代细节如下：

Each iteration of GAN learning proceeds as follows:

1. Improve the θ_d so that the discriminator becomes better at distinguishing between fake and real examples:Run the following k times:
   - Sample m **real examples** $\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(m)}$ from training data.
   - Generate m **fake examples** $g(\mathbf{z}^{(1)}),\ldots,g(\mathbf{z}^{(m)})$ using the current generator g, where $\mathbf{z}^{(i)}\sim p(\mathbf{z})$.
   - Improve θ_d so that the discriminator can better distinguish between **those** fake examples and **those** real examples.
2. Improve generator θ_g so as to generate examples the discriminator would find hard to tell fake or real:
   - Generate m **new fake examples** $g(\mathbf{z}^{(1)}),\ldots,g(\mathbf{z}^{(m)})$ using the current generator g. (Those are examples that the discriminator can tell as fake with high confidence because of the training in step 1.)
   - Improve θ_g to generate examples that would be more like real images than **those** fake images.

9. 鉴别器的损失函数是常用的交叉熵：

$$J(\boldsymbol{\theta}_g,\boldsymbol{\theta}_d)=-\frac{1}{2}\sum_{i=1}^{m}\log D(\boldsymbol{x}^{(i)})-\frac{1}{2}\sum_{i=1}^{m}\log(1-D(g(\boldsymbol{z}^{(i)}))$$

其中$\boldsymbol{x}^{(i)}$是指来自训练集的真实图像，$g(\boldsymbol{z}^{(i)})$是指来自生成器的伪造图像.

$J(\boldsymbol{\theta}_g,\boldsymbol{\theta}_d)$的最小值为 0，当且仅当其在所有真实图像都被标上 1 同时所有伪造图像都被标成 0 时取得。因此，鉴别器的模型参数$\boldsymbol{\theta}_d$应该通过最小化以上函数得到，问题等价于最大化以下函数，可通过梯度上升法实现：

$$V(\boldsymbol{\theta}_g,\boldsymbol{\theta}_d)=\sum_{i=1}^{m}\log D(\boldsymbol{x}^{(i)})+\sum_{i=1}^{m}\log(1-D(g(\boldsymbol{z}^{(i)}))$$

10. 生成器的目标是使得鉴别器的识别错误率尽可能的高，从而我们把生成器的损失函数定义为$-V(\boldsymbol{\theta}_g,\boldsymbol{\theta}_d)$，又因为上式当中的第一项与$\boldsymbol{\theta}_g$无关，从而生成器的损失函数定义为：找到合适的$\boldsymbol{\theta}_g$，最小化以下式子

$$\frac{1}{m}\sum_{i=1}^{m}\log(1-D(g(\boldsymbol{z}^{(i)}))$$

实际工程当中常用

$$\frac{1}{m}\sum_{i=1}^{m}\log(-D(g(\boldsymbol{z}^{(i)}))$$

使用随机梯度下降法对以上式子求最小值的具体算法如下：

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

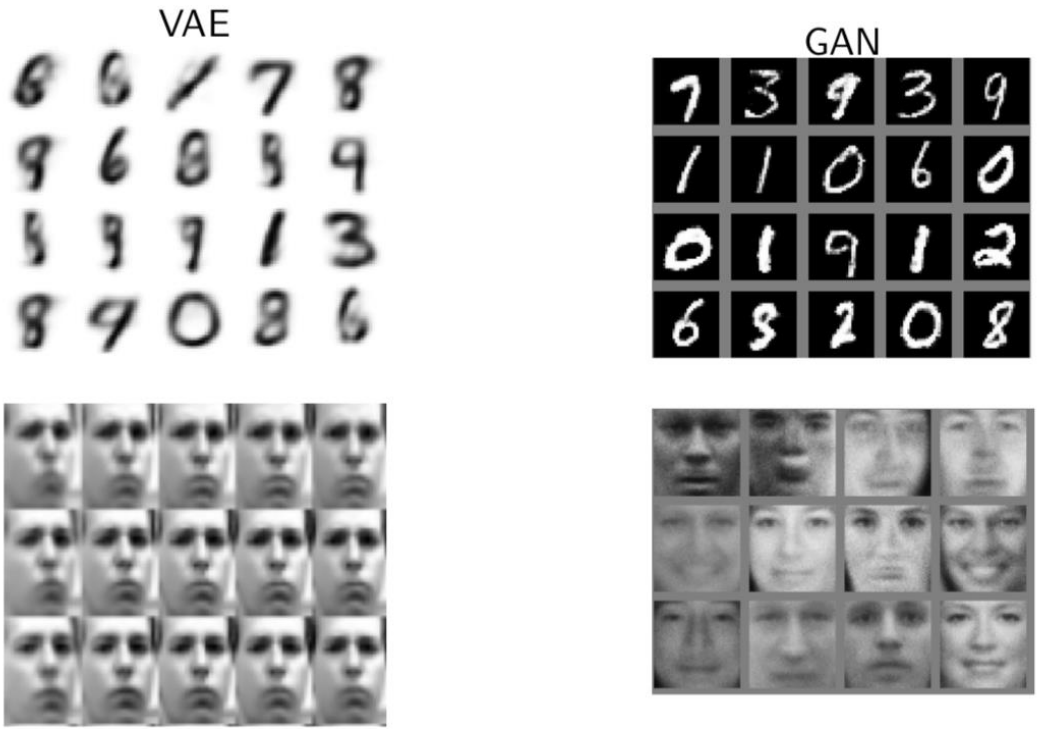
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

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

end for

GAN vs VAE

1. GAN 生成的图像往往比 VAE 锐利的多，种类也更多：



2. 以上现象的原因在于，VAE 所采用的 KL 散度：

$$KL(p_r||p_g) = \int p_r(\boldsymbol{x}) \log \frac{p_r(\boldsymbol{x})}{p_g(\boldsymbol{x})} d\boldsymbol{x}$$

当$p_r(\boldsymbol{x})$很大而$p_g(\boldsymbol{x})$很小时，意即模型没有 cover 到大部分数据时，KL 散度

很大，最小化 KL 散度避免了这个问题；而$p_r(\boldsymbol{x})$很小而$p_g(\boldsymbol{x})$很大时，意即生产模型产出的样例已经远远大于原来的数据空间，导致部分样例已经非常 fake 时，KL 散度很小，最小化 KL 散度并没有解决这个问题，从而导致其产生 fake 样例的 cost 非常低，因而样例都比较 fake。如果将以上的 KL 散度从$KL(p_r||p_g)$变为$KL(p_g||p_r)$，则有相反的效果。

同时，GAN 采用的 JS 散度：

$$JS(p_r||p_g) = \frac{1}{2}KL(p_r||p_a) + \frac{1}{2}KL(p_g||p_a)$$

其中

$$p_a = \frac{1}{2}(p_r + p_g)$$

从而实际上 JS 散度可以同时避免以上两个问题，所以生成的图像较为锐利。

GAN 的问题与改进

上面我们提到 GAN 生成器的两个损失函数：找到合适的$\boldsymbol{\theta}_g$，最小化以下式子

$$\frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g(\boldsymbol{z}^{(i)})) = E_{\boldsymbol{x}\sim p_g(\boldsymbol{x})}[\log(1 - D(\boldsymbol{x}))]$$

以上式子表示鉴别器鉴别正确的似然函数的期望。

我们对以上式子求梯度：

$$\nabla_{\boldsymbol{\theta}_g}E_{\boldsymbol{x}\sim p_g(\boldsymbol{x})}[\log(1 - D(\boldsymbol{x}))] = \nabla_{\boldsymbol{\theta}_g}\int p_g(\boldsymbol{x})\log(1 - D(\boldsymbol{x}))\,d\boldsymbol{x}$$

$$= \int \log(1 - D(\boldsymbol{x}))\nabla_{\boldsymbol{\theta}_g}p_g(\boldsymbol{x})\,d\boldsymbol{x}$$

我们看到，当一开始训练时，生成器生成的效果非常差，鉴别器可以非常简单的判别出效果的好坏。最极端的情况是，所有的生成器生成的样例都被判为 0，即$D(\boldsymbol{x}) = 0, \forall \boldsymbol{x} \sim p_g(\boldsymbol{x})$, 因而梯度等于 0，进而生成器无法得知应该往哪个方向训练模型参数。这是梯度消失问题。

因而为了避免以上问题，实际工程当中常用

$$\frac{1}{m}\sum_{i=1}^{m}\log(-D(g(\boldsymbol{z}^{(i)})) = -E_{\boldsymbol{x}\sim p_g(\boldsymbol{x})}[\log D(\boldsymbol{x})]$$

以上式子表示鉴别器鉴别错误的似然函数的期望的负数。

我们对以上式子求梯度：

$$\nabla_{\boldsymbol{\theta}_g}E_{\boldsymbol{x}\sim p_g(\boldsymbol{x})}[-\log D(\boldsymbol{x})] = \nabla_{\boldsymbol{\theta}_g}\int p_g(\boldsymbol{x})[-\log D(\boldsymbol{x})\,d\boldsymbol{x}$$

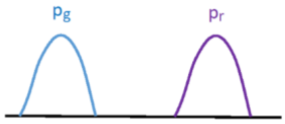
$$= \int [-\log D(\boldsymbol{x})] \nabla_{\boldsymbol{\theta}_g} p_g(\boldsymbol{x})\, d\boldsymbol{x}$$

虽然这样避免了一开始训练遇到的梯度为 0 的问题，但是当鉴别器表现较好时，最极端的情况是，所有的生成器生成的样例都被判为 0，即$D(\boldsymbol{x}) \approx 0, \forall \boldsymbol{x} \sim p_g(\boldsymbol{x})$，梯度变为无穷大，使得训练过程不稳定。

因而如果选第一个损失函数，则生成器的训练速度过慢；若选择第二个，则生成器的训练过程不稳定，所以在训练生成器的过程当中，避免以上问题的方法就是，不要将生成器训练到最优，例如设定 epoch = 1。

Wasserstein GAN (WGAN)

我们以上对 VAE 中 KL 散度和 GAN 中 JS 散度的讨论都是基于模型分布和实际数据分布是有部分重合的。然而实际上，对于一个较大的图像，两个分布不重合的可能性非常大，如下所示。



此时计算其 JS 散度：

$$JS(p_r||p_g) = \frac{1}{2}KL(p_r||p_a) + \frac{1}{2}KL(p_g||p_a)$$

其中

$$p_a = \frac{1}{2}(p_r + p_g)$$

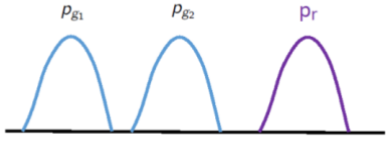
我们有

$$KL(p_g||p_a) = \int p_r(x) \log \frac{p_r(x)}{p_a(x)} dx = \int p_r(x) \log \frac{p_r(x)}{\frac{p_r(x)}{2}} dx = \log 2$$

同理$KL(p_g||p_a) = \log 2$

从而$JS(p_r||p_g) = \log 2$，同时$\nabla_{\boldsymbol{\theta}_g} JS = 0$。

同时，另外一个情况是：如下图，我们认为 Pg2 更接近真实的变换，但是 JS 散度计算出来是一样的。

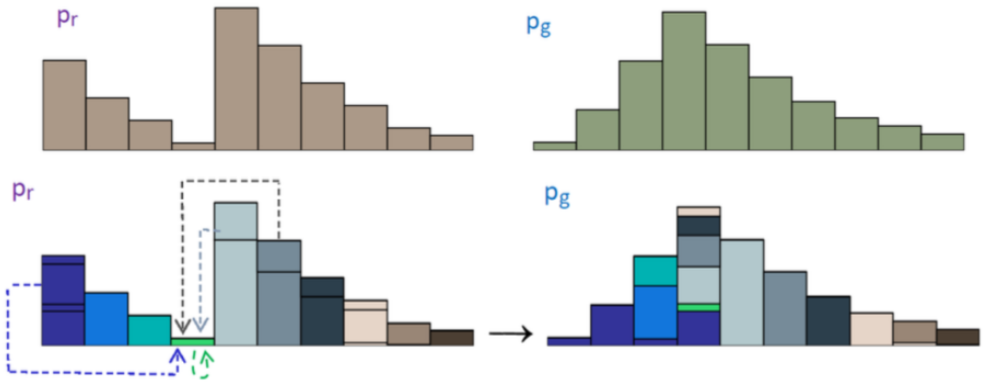


所以我们可以看到 JS 散度在此时并不是一个描述模型分布和实际数据分布的好方法。

为了解决以上问题，我们引入一个新的方法：**Wasserstein GAN (WGAN)**。

相较于采用 JS 散度来描述两个分布的差异，WGAN 采用了另外一种描述差异的方法：Earth Movers Distance/EMD or Wasserstein Distance.

在离散分布当中，一个分布可以描述为一个直方图。EMD 表示的就是从一个分布直方图变换到另外一个直方图的最小步数。



一个运输过程可以用一个函数$\gamma(\boldsymbol{x},\boldsymbol{y})$来表示，其表示从一个位置$\boldsymbol{x}$搬运到另外一个位置$\boldsymbol{y}$的“尘土”/dirt 量。该函数有以下性质：

1. $\gamma(\boldsymbol{x},\boldsymbol{y}) \geq 0$;
2. $\sum_y \gamma(\boldsymbol{x},\boldsymbol{y}) = p_r(\boldsymbol{x})$: 所有在位置 x 的 dirt 可能被全部搬运至其他地方，也可以是这个位置 x 本身；
3. $\sum_x \gamma(\boldsymbol{x},\boldsymbol{y}) = p_g(\boldsymbol{x})$: 所有在位置 y 的 dirt 可能被全部搬运至其他地方，也可以是这个位置 y 本身；
4. $\sum_{x,y} \gamma(\boldsymbol{x},\boldsymbol{y}) = 1$.

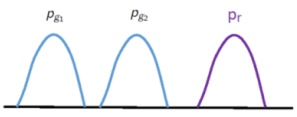
从而对于一个运输计划γ而言，有

$$\sum_{\boldsymbol{x},\boldsymbol{y}} \gamma(\boldsymbol{x},\boldsymbol{y})||\boldsymbol{x}-\boldsymbol{y}|| = E_{(\boldsymbol{x},\boldsymbol{y})\sim\gamma}||\boldsymbol{x}-\boldsymbol{y}||$$

进而 **EMD** 定义为

$$EMD(p_r,p_g) = \min_{\gamma} E_{(\boldsymbol{x},\boldsymbol{y})\sim\gamma}||\boldsymbol{x}-\boldsymbol{y}||$$

从而回到一开始提到的问题：如下图，我们认为 Pg2 更接近真实的变换，但是 JS 散度计算出来是一样的。



此时根据 EMD 的定义，我们可以得到

$$EMD(p_{g2}, p_r) \leq EMD(p_{g1}, p_r)$$

相较于这种情况下的 JS 散度

$$D_{JS}(p_{g2}||p_r) = D_{JS}(p_{g1}||p_r)$$

显然 EMD 是一个更好的度量方法。采用以上度量方式的 GAN 称为 WGAN。

WGAN 通过最小化 EMD 距离来训练生成器：即找到θ_g，满足

$$\arg\min_{\theta} EMD(p_r, p_g)$$

计算 EMD 是一个复杂的优化问题，我们采用一个称为 critic 的神经网络来计算 EMD 的近似值：求 EMD 的似然函数，并用梯度下井方法求解，过程涉及 Lipschitz 函数，KR 对偶问题等较为复杂。此（you）处（kong）省（zai）略（bu）。

WGAN 的具体算法如下：

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

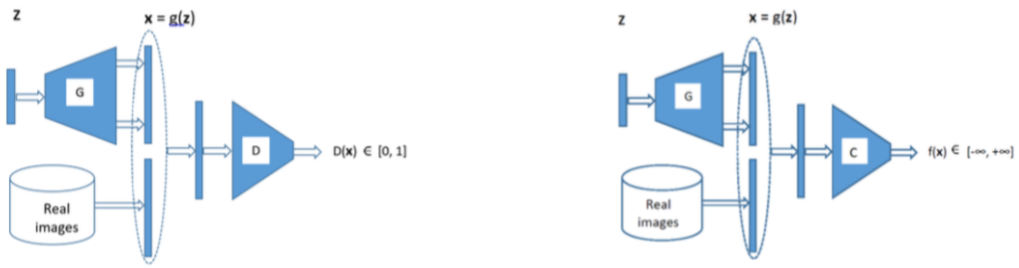
Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

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

Adam found not stable. So, RMSProp is used.

GAN vs WGAN



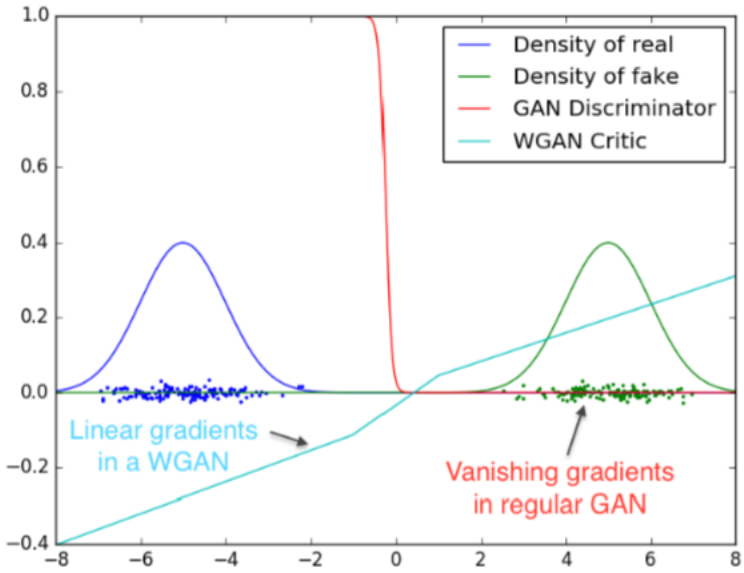
在 GAN 当中，我们通过最大化鉴别器的似然函数来训练鉴别器：

$$E_{\boldsymbol{x}\sim p_r(\boldsymbol{x})} \log D(\boldsymbol{x}) + E_{\boldsymbol{x}\sim p_r(\boldsymbol{x})} \log[1 - D(\boldsymbol{x})]$$

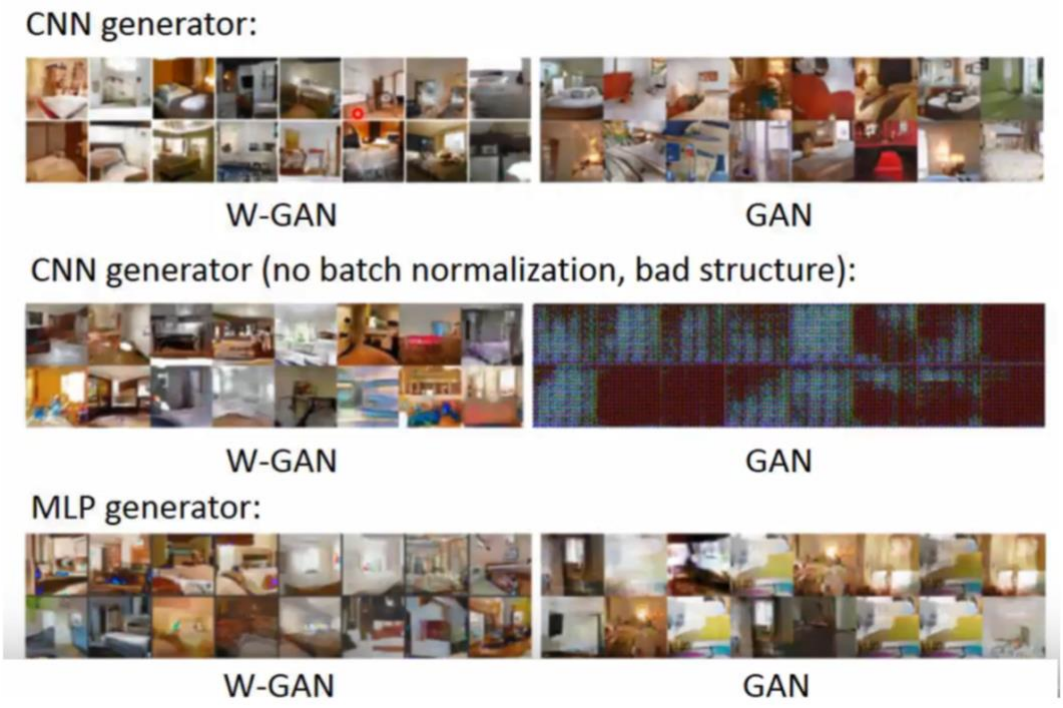
我们希望该函数可以给真实样例高概率，给到伪造样例低概率，从而正确分类不同的样例。

在 WGAN 中，我们令 critic 网络给真实样例较高的值，给到伪造样例较低的值。这是一个估值问题，而非一个分类问题。

对于我们之前提到的实际数据分布与模型概率分布完全不同的情况：



我们可以看到，WGAN 的梯度（图上应该是把浅蓝线 flipped 过来）近乎线性平滑，而 GAN 存在着梯度消失问题。虽然 GAN 正确识别出绿色的点是伪造的，但是却没有提供改进方向，而 WGAN 不仅做到了 GAN 的工作，还告诉了这些点离正确的分布还有多远，以及改进的方向。以下是一些 GAN 和 WGAN 生成的样例：



同时 GAN 存在着模型崩溃 Model Collapse 的问题，即生成的样例种类很少；然而 WGAN 并没有这个问题。

WGAN 的改进：WGAN with Gradient Penalty

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while