Generative Adversarial Networks 生成对抗网络的简单理解

1. 引言

在对抗网络中，生成模型与判别相竞争，判别模型通过学习确定样本是来自生成模型分布还是原始数据分布。生成模型可以被认为是类似于一组伪造者，试图产生假币并在没有检测的情况下使用它，而判别模型类似于警察，试图检测假币。 在这个游戏中的竞争驱动两个团队改进他们的方法，直到假冒与真正的物品难以分别。

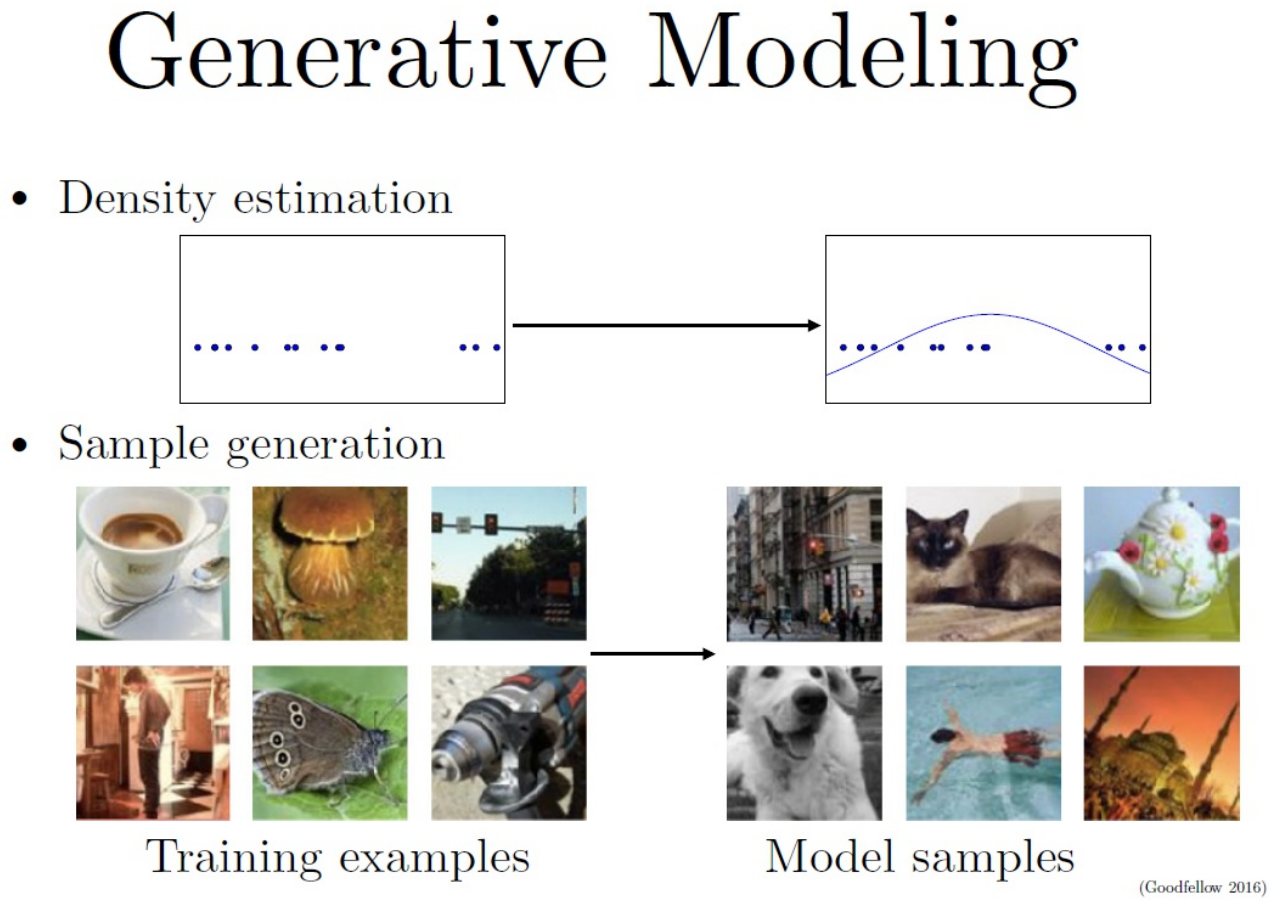
生成模型和判别模型对抗的结果是，两者都提升了自己的能力。生成模型提高了模拟原始数据分布的能力，判别模型提高了分辨的能力。

GAN之所以叫生成对抗网络，是因为判别模型和生成模型都是用的神经网络模型，如果选择其他模型的话，可能就要叫GAXX了

---

2. 生成模型

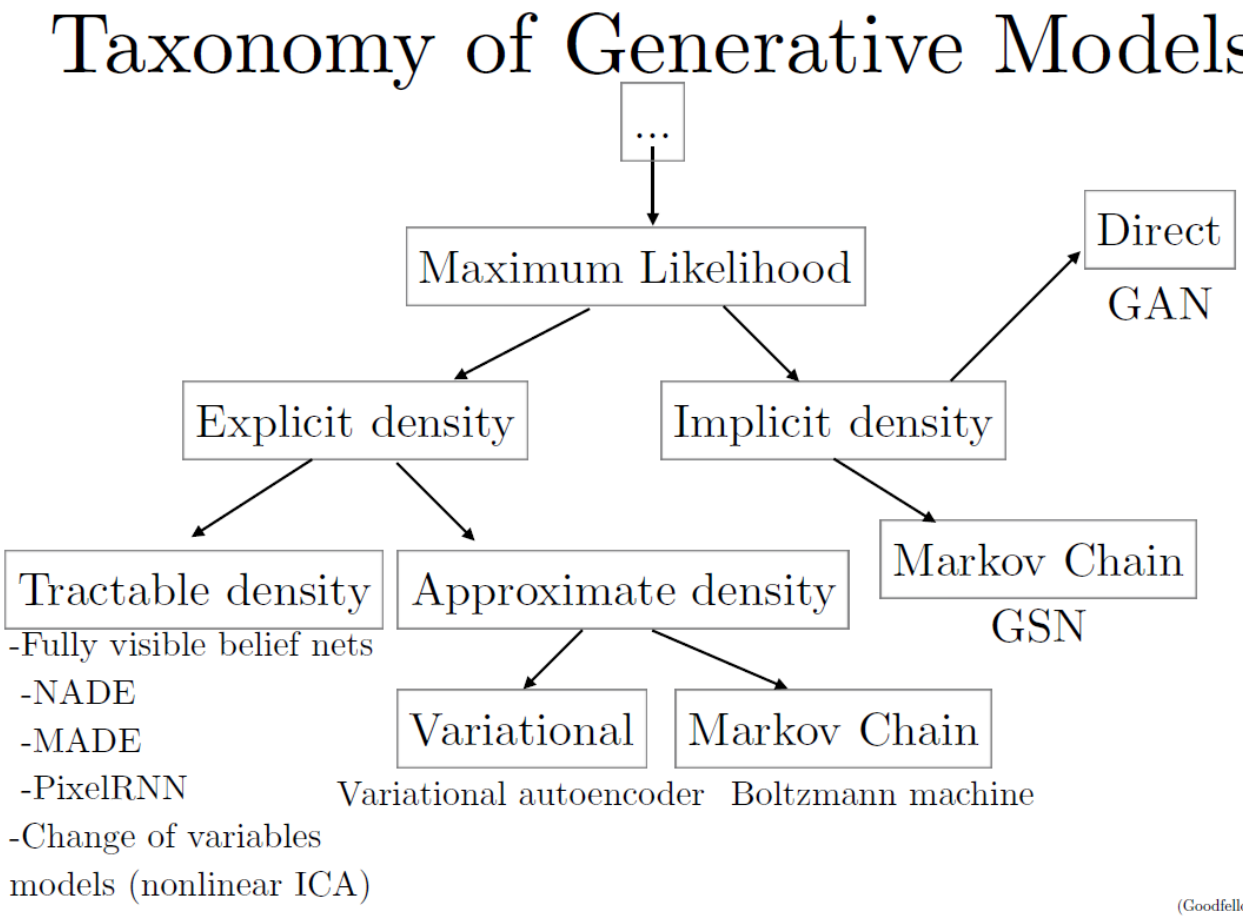
2.1 GAN是生成模型的一种



“生成模型”表示任何一种可以接受训练集（从一个分布p_{data}采样的样本）学会表示该分布的估计的模型。其结果是一个概率分布p_{model}。有些时候，模型会显式地估计p_{model}，比如说图上半部分所示。还有些时候，模型只能从p_{model}中生成样本，比如说图下半部分。有些模型能够同时这两件事情。虽说 GANs 经过设计可以做到这两点，我们这里把精力放在 GANs 样本生成上。

- *作者虽然说GANs可以都做，但是现在的应用一般都是在样本生成上*
- *关于判别模型和生成模型的区别，请参考《统计学习方法》第17-18页*

2.2 GAN与其他生成模型的对比



上图是作者将要与GAN比较的集中生成模型

GANs是针对其他生成式模型的缺点进行设计的:

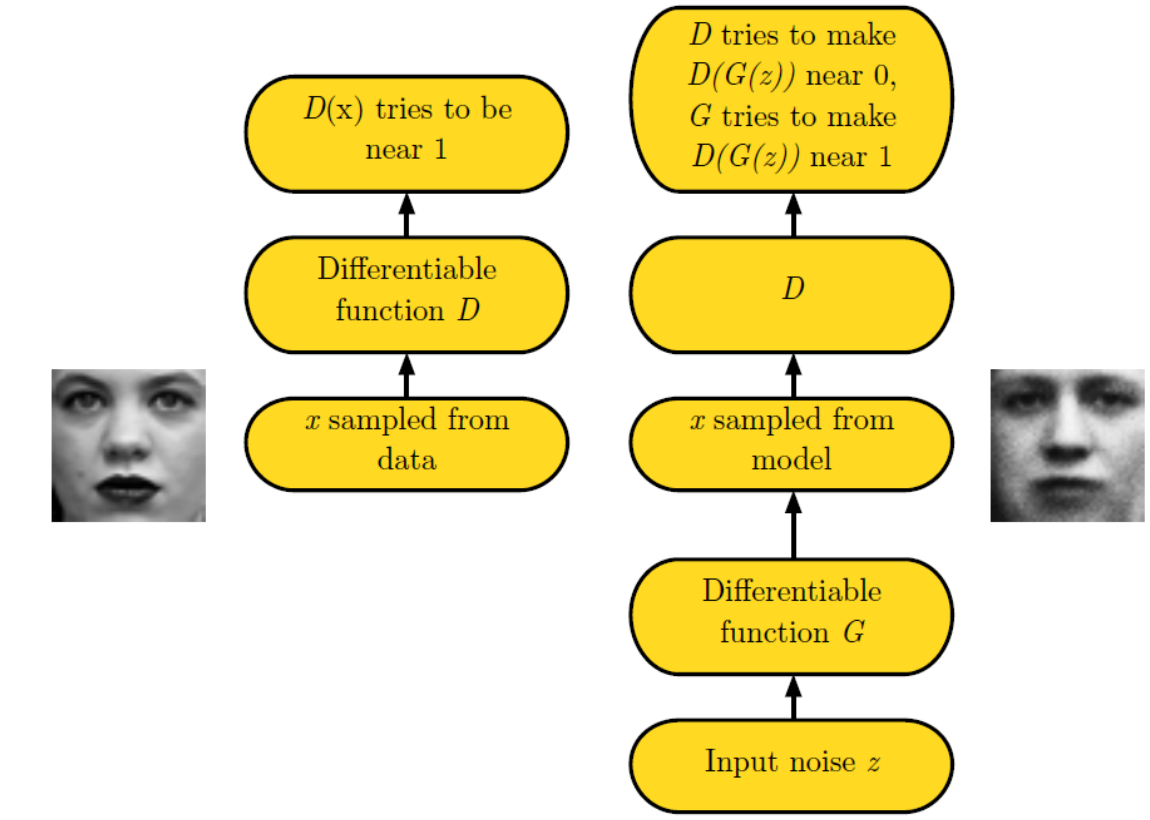
作者原话：GANs were designed to avoid many disadvantages associated with other generative models,相当直白，毫不客气

- 相对于FVBN，可以并行生成样本，而不是随 x 的维度变化的运行时间。
- 相对于玻尔茨曼机，生成器函数的设计只有很少的限制。在玻尔茨曼机中，只有很少概率分布能够给出易解的 Markov chain 采样。而非线性ICA中生成器必须是可逆的而且隐含编码 z 必须要和样本 x 有着同样的维度。
- 相对于玻尔茨曼机和GSNs，不需要 Markov chain。
- 不需要有变分界，在 GANs 框架中可用的特定模型族已经证实是通用近似子，所以 GANs 其实是渐进一致的。某些 VAEs 被猜想是渐进一致的，但还没有被证实。
- 主观上判断 GANs 能够产生比其他方法更好的样本。（仅仅是主观，ian 本人也说，这事儿没法衡量什么是「好」或「不好」）
- 同时，GANs 也有新的缺点，训练 GANs 需要找到博弈的 Nash 均衡，这个其实是一个比优化目标函数更加困难的问题。

---

3. 生成对抗网络

GANs 的基本思想是设置两个参与人的博弈。其中一个是生成器(generator)。生成器的目的产生来自和训练样本一样的分布的样本（制作跟真币一样的假币）。另外一个判别器(discriminator)。判别器检查这些样本来确定他们是真实的还是伪造的（识别真币和假币）。判别器使用传统的监督学习技术进行训练，将输入分成两类（真实的或者伪造的）。生成器训练的目标就是欺骗判别器。



博弈中的两个参与人由两个函数表示，**每个都是关于输入和参数可微分的。**判别器是一个以 x（真实数据）作为输入和使用 $\theta^{(D)}$ 为参数的函数 D 定义。生成器由一个以 z（噪音数据即假数据）为输入使用 $\theta^{(G)}$ 为参数的函数 G 定义。

双方的 *costfunction* 都有双方定义的参数。判别器希望仅控制住 $\theta^{(D)}$ 情形下最小化 $J^{(D)}(\theta^{(D)},\theta^{(G)})$。生成器希望在仅控制 $\theta^{(D)}$ 情形下最小化 $J^{(G)}(\theta^{(D)},\theta^{(G)})$。因为每个人的 *costfunction* 都依赖于另一个人的参数，但是每个人都不能控制别人的参数，这个场景其实更为接近一个博弈而非优化问题。优化问题的解是一个局部最小，而一个博弈的解是一个纳什均衡。在这样的设定下，Nash 均衡是一个元组，$(\theta^{(D)},\theta^{(G)})$ 既是关于 $\theta^{(D)}$ 的 $J^{(D)}$ 的局部最小值和也是关于 $\theta^{(G)}$ 的 $J^{(G)}$ 局部最小值。

3.1 生成模型

生成器是一个可微分函数 G。当 z 从某个简单的先验分布中采样出来时，$G(z)$ 产生一个从 $pmodel$ 中的样本 x。一般来说，深度神经网络可以用来表示 G。注意函数 G 的输入不需要和深度神经网络的第一层的输入相同；输入可能放在网络的任何地方。例如，我们可以将 z 划分成两个向量 $z^{(1)}$ 和 $z^{(2)}$，然后让 $z^{(1)}$ 作为神经网络的第一层的输入，将 $z^{(2)}$ 作为神经网络的最后一层的输入。如果 $z^{(2)}$ 是 Gaussian，这就使得 x 成为 $z^{(1)}$ 条件高斯。另外一个流行的策略是将噪声加到或者乘到隐含层或者将噪声拼接到神经网络的隐含层上。总之，我们看到其实对于生成式网络只有很少的限制。如果我们希望 p{model} 是 x 空间的支集(support)，我们需要 z 的维度需要至少和 x 的维度一样大，而且 G 必须是可微分的，但是这些其实就是仅有的要求了。特别地，注意到使用非线性 ICA 方法的任何模型都可以成为一个 GAN 生成器网络。GANs 和变分自编码器的关系更加复杂一点；一方面 GAN 框架可以训练一些 VAE 不能的训练模型，反之亦然，但是两个框架也有很大的重合部分。 最为显著的差异是，如果采用标准的反向传播，VAEs 不能在生成器输入有离散变量，而 **GANs 不能够在生成器的输出层有离散变量。**

3.2 训练过程

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.

训练过程包含同时随机梯度下降 simultaneous SGD。在每一步，会采样两个 minibatch：一个来自数据集的x 的 minibatch 和一个从隐含变量的模型先验采样的z 的 minibatch。然后两个梯度步骤同时进行：一个更新 来降低 J(D)，另一个更新 θ^(G) 来降低 J(G)。这两个步骤都可以使用你选择的基于梯度的优化算法。 Adam (Kingmaand Ba, 2014) 通常是一个好的选择。

很多作者推荐其中某个参与人运行更多步骤(包括14年的文章上，算法过程在上图)，但是在 2016 年的年末，观点是最好的机制就是同时梯度下降，每个参与人都是一步。

下面这张图可能更加容易理解训练过程

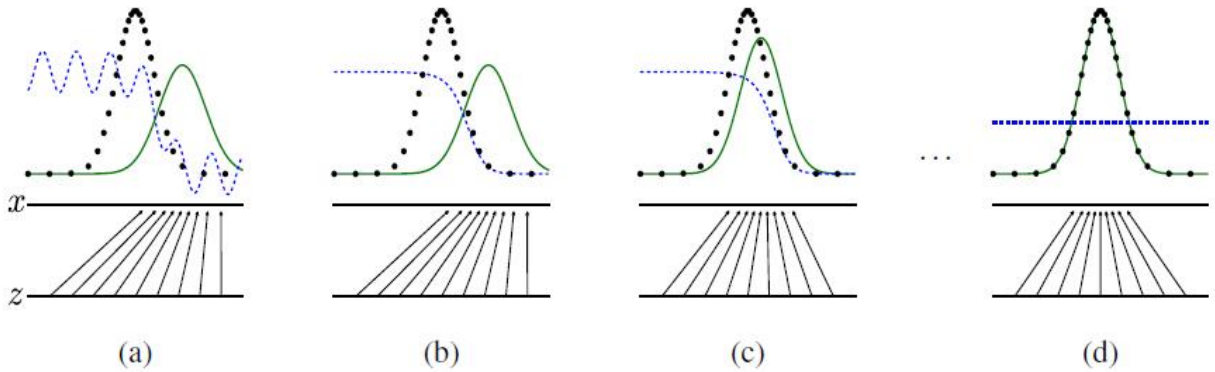


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

训练对抗的生成网络时，同时更新判别分布（D，蓝色虚线）使D能区分数据生成分布 p*x（黑色虚线）中的样本和生成分布p_g（G，绿色实线）中的样本。下面的水平线为均匀采样z的区域，上面的水平线为x的部分区域。朝上的箭头显示映射x=G(z)如何将非均匀分布p_g作用在转换后的样本上。G在pg高密度区域收缩，且在pg的低密度区域扩散。(a)考虑一个接近收敛的对抗的模型对：p_g与p{data}相似，且D是个部分准确的分类器。(b)算法的内循环中，训练D来判别数据中的样本，收敛到：D^*(x)={p{data}(x)} \over {p{data}(x)+pg(x)}。(c)在G的1次更新后，D的梯度引导G(z)流向更可能分类为数据的区域。(d)训练若干步后，如果G和D性能足够，它们接近某个稳定点并都无法继续提高性能，因为此时p_g=p{data}。判别器将无法区分训练数据分布和生成数据分布，即D(x)=1 \over 2。

3.3 cost function

目前为 GANs 设计的所有不同的博弈针对判别器 J(D) 使用了同样的代价函数。他们仅仅是生成器 J(G) 的代价函数不同。

3.3.1 判别模型 cost function

$$J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right). \quad (8)$$

其实就是标准的训练一个 sigmoid 输出的标准的二分类器交叉熵代价。唯一的不同就是分类器在两个 minibatch 的数据上进行训练；一个来自数据集（其中的标签均是 1），另一个来自生成器（其标签均是 0）。

GAN 博弈的所有版本都期望判别器能够最小化（8）式。所有情况下，判别器有同样最优策略。

3.3.2生成模型 cost function

- Minimax

最简单的博弈版本是零和博弈，其中所有参与人的代价总是 0。在这个版本的博弈中

$$J^{(G)} = -J^{(D)}. \quad (10)$$

- 启发式，非饱和博弈

在 minimax 博弈中用在生成器上的代价对理论分析很有用但是在实践中表现很糟糕。

最小化目标类和分类器预测的分布的交叉熵是很高效的，因为代价不会在分类器有错误的输出的时候饱和。最终代价会饱和到\theta，但是仅仅是在分类器选择了正确的类标的情况下。

在 minimax 博弈中，判别器最小化交叉熵，但是生成器是最大化同一个交叉熵。这对于生成器是不利的，因为判别器成功地以高置信度反对生成器产生的样本时，生成器的梯度会消失。

为了解决这个问题，一种方式是继续使用交叉熵来最小化生成器。不过我们不是去改变判别器代价函数的正负号来获得生成器的代价。我们是将用来构造交叉熵代价的目标的正负号。所以，生成器的代价函数就是：

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z})) \quad (13)$$

-最大似然博弈

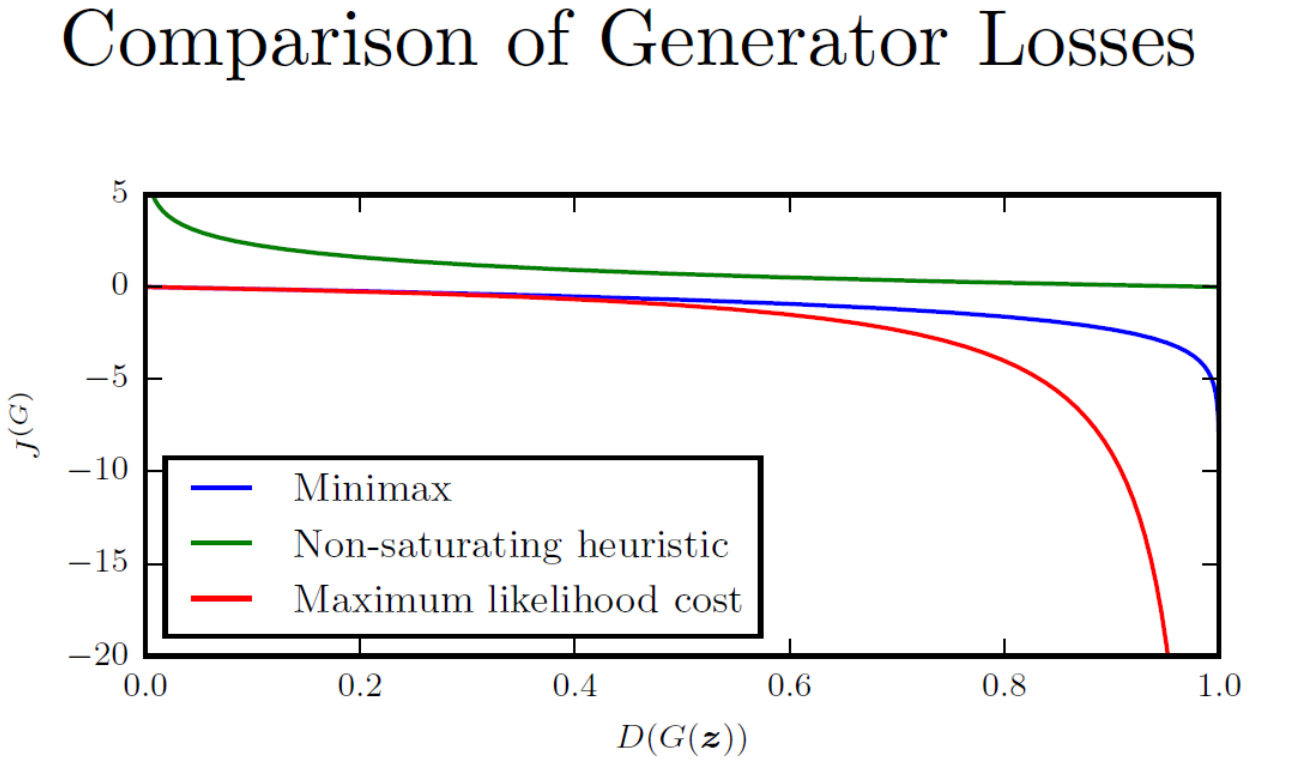
我们可能能够使用 GANs 进行最大似然学习，这就意味着可以最小化数据和模型之间的 KL 散度

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D_{\text{KL}}\left(p_{\text{data}}(\boldsymbol{x}) \| p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})\right). \quad (4)$$

有很多中方式能够使用 GAN 框架来近似（4）式：

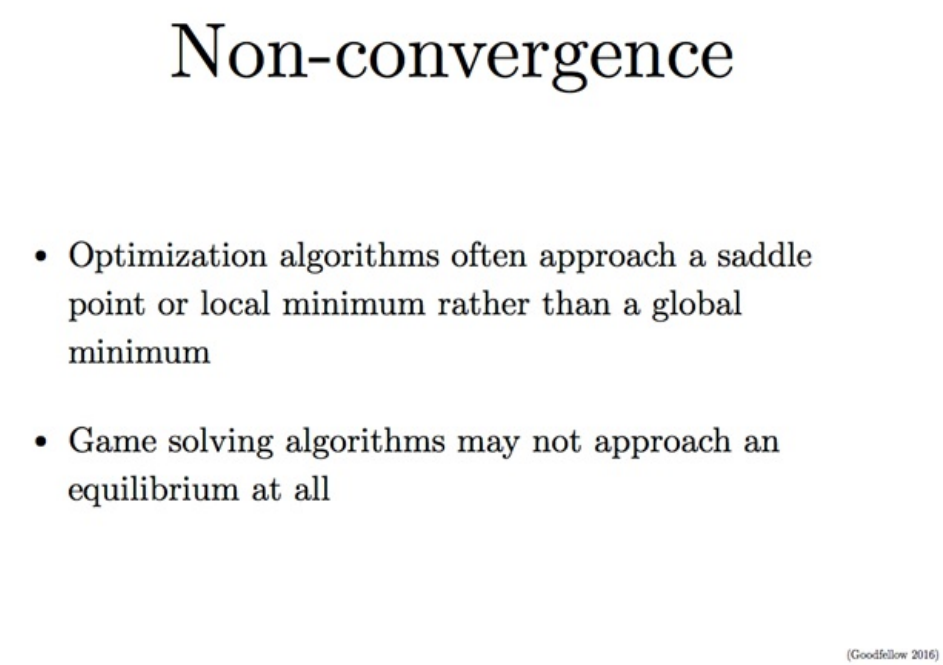
$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{z} \exp\left(\sigma^{-1}\left(D(G(\boldsymbol{z}))\right)\right), \quad (14)$$

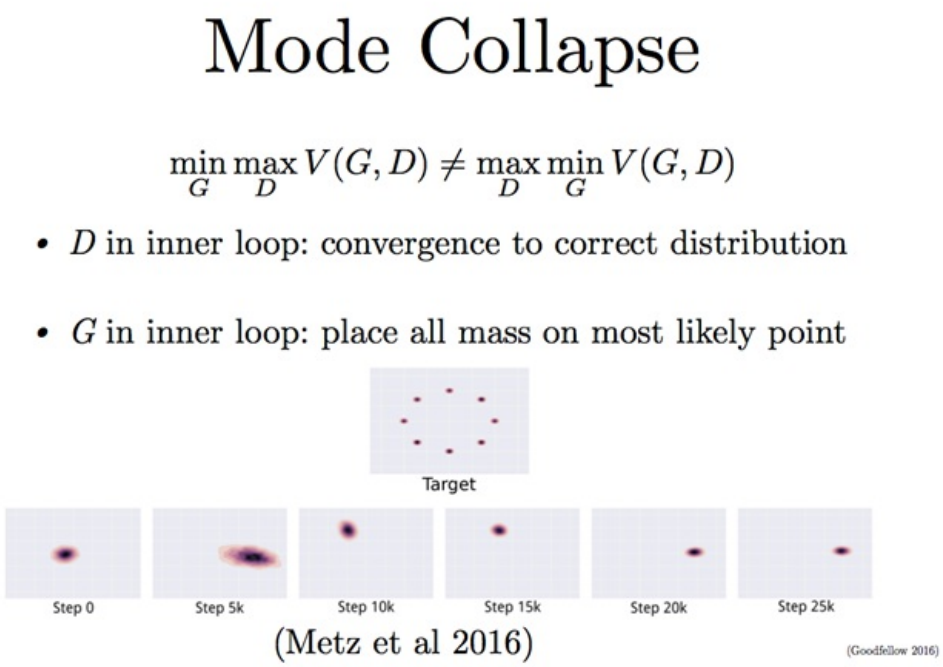
上述不同损失函数的比较：



(Goodfellow 2014)

3.4 缺点
最大的缺点就是训练困难，导致无法收敛和模式崩溃





现在 GAN 面临的最大问题就是不稳定，很多情况下都无法收敛（non-convergence）。原因是我们使用的优化方法很容易只找到一个局部最优点，而不是全局最优点。或者，有些算法根本就没法收敛。
模式崩溃（mode collapse）就是一种无法收敛的情况，这在 Ian 2014 年的首篇论文中就被提及了。比如，对于一个最小最大博弈的问题，我们把最小（min）还是最大（max）放在内循环？minmax V(G,D) 不等于 maxmin V(G,D)。如果 maxD 放在内圈，算法可以收敛到应该有的位置，如果 minG 放在内圈，算法就会一股脑地扑向其中一个聚集区，而不会看到全局分布

不过值得庆幸的是，现在已经有算法解决这些问题，这也是2016年GAN才火的一个原因吧

4. 后续发展
github上有人总结的GAN的文章：AdversarialNetsPapers

5. 参考
Generative Adversarial Nets
NIPS 2016 Tutorial: Generative Adversarial Networks
GAN之父NIPS 2016演讲现场直击：全方位解读生成对抗网络的原理及未来