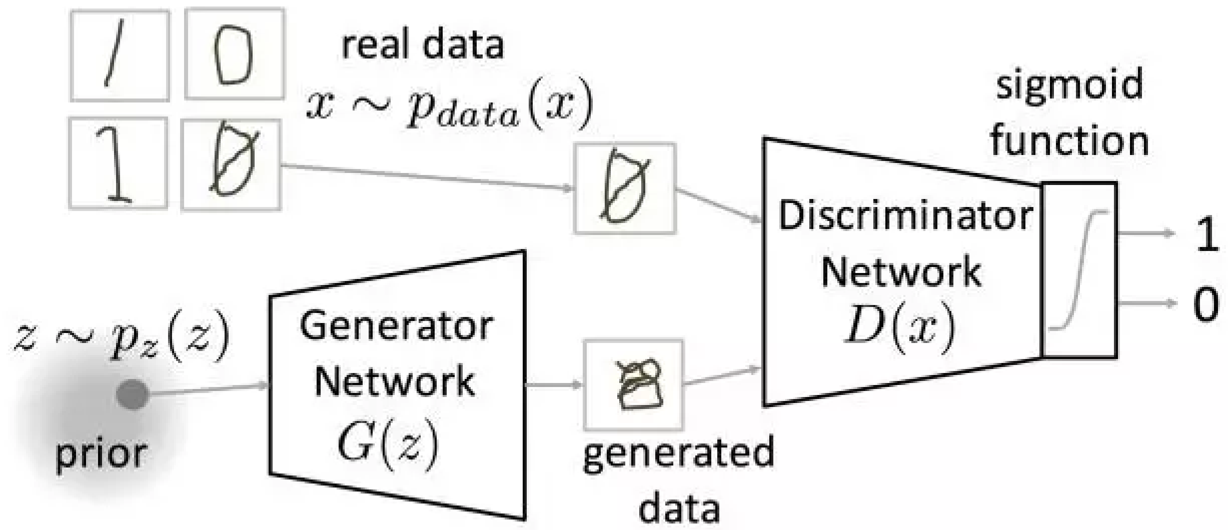
GAN入门理解及公式推导

2017-03-23 ALI 神经网络与强化学习

首先，当我们拿到一个崭新的网络的时候，先不管它到底是什么，作为一个黑盒来研究研究它的外观。
GAN:生成对抗网络
输入：原始数据x和随机噪声信号z （比如高斯分布或者均匀分布）
输出：一个概率值或一个标量值。
首先举个简单例子好个初始的印象　警察与造假币者
　造假者按照真实钱币的样子来造假，警察来分辨遇到的钱币是真还是假。最初的时候，造假者的造假能力不高，所以警察可以很容易的分辨出来。当被识别出来的时候，造假者就会继续修炼自己的技艺 与此同时 警察的分辨能力也要相应提高。这样的过程就是一种生成对抗的过程。对抗到最后，造假者已经能够创造出可以以假乱真的钱币，警察难以区分真假 所以猜对的概率变成0.5。最后得到的造假者 已经是一个可以很好的刻画钱币的人，他掌握了钱币的各种特征，他也就是我们希望得到的生成器。

好，接下来开始研究一下黑盒内部，GAN的组成和工作原理。
生成对抗网络GAN，就是在这种对抗与生成的交替进行中对原数据进行刻画的。　GANGAN启发自博弈论中的二人零和博弈[1]（此为注解，不理解的读者可以跳到最后查看解答）是由Ian Goodfellow 于2014年开创性的提出。GAN是由两部分主成：生成模型G（generative model）和判别模型D（discriminative model）。生成模型捕捉样本数据的分布，判别模型是一个二分类器，判别输入是真实数据还是生成的样本。



X是真实数据，真实数据符合Pdata(x)分布。z是噪声数据，噪声数据符合Pz(z)分布，比如高斯分布或者均匀分布。然后从噪声z进行抽样，通过G之后生成数据x=G(z)。然后真实数据与生成数据一起送入分类器D，后面接一个sigmoid函数 输出判定类别。

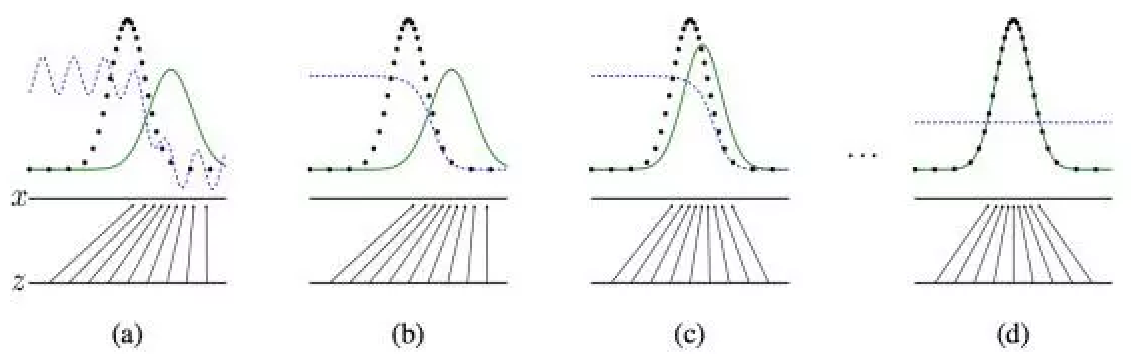
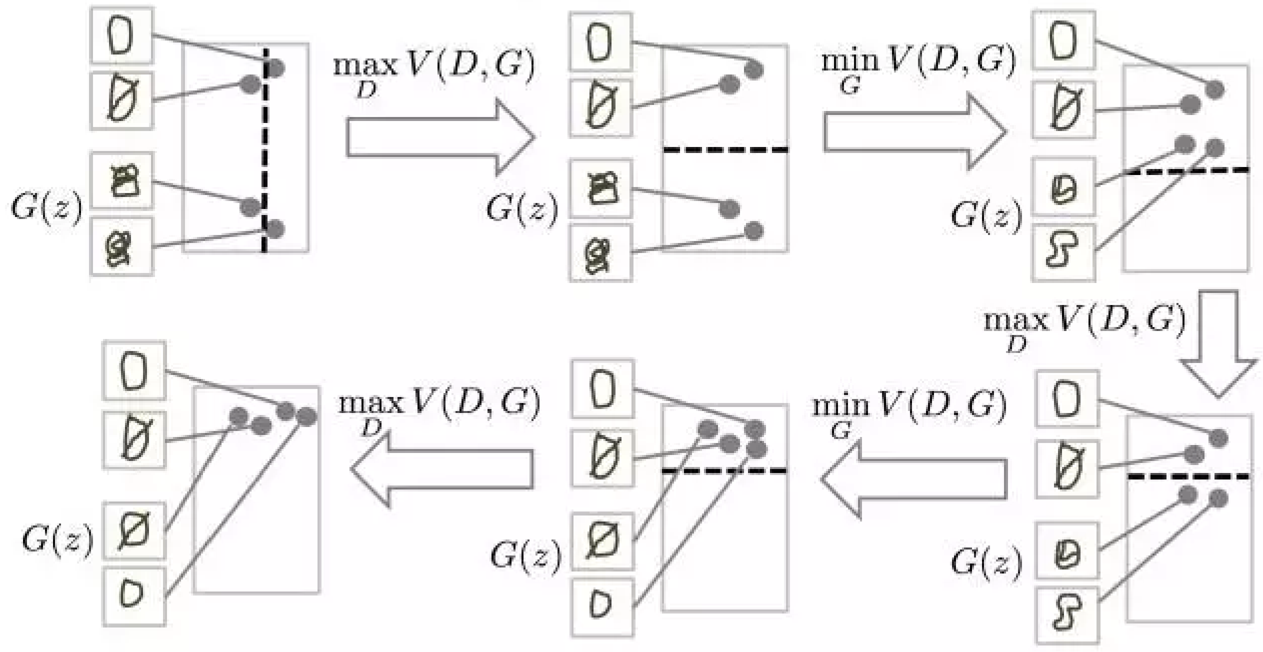


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution 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. $D(\boldsymbol{x}) = \frac{1}{2}$.

这里，z是噪声数据，从z中采样作为x，形成绿色的高耸的部分，原始数据Xdata是不变的。蓝色虚线是分类器(sigmoid),黑色虚线是原始数据，绿色实线是生成数据。最初的时候D可以很好的区分开真实数据和生成数据，看图(b)，对于蓝色虚线而言，单纯的绿色部分可以很明确的划分为0，黑色虚线的部分可以很明确的划分成1，二者相交的部分划分不是很明确。看图(c)绿色实现生成数据更新，与原始数据相似度增加。最后一张图，经过多次更新与对抗生成之后，生成数据已经和原始数据非常相像的了，这时分类器已经无法判断到底是输出1还是0，于是就变成了0.5 一条水平线。

$$\min_G \max_D V(D,G)$$



这张图形象的说明了一下G和D的优化方向。优化D的时候需要很好的画出黑色虚线，使它能够区分开真实数据和生成数据。优化G的时候，生成数据更加接近原始数据的样子，使得D难以区分数据真假。如此反复直到最后再也画不出区分的黑色虚线。

主要特点和应用场景

(1)GAN可以用来产生data。

(2)根据GAN的组成结构我们可以知道，GAN 整个的过程就是，G产生一个自创的假数据和真数据放在一起 让D来区分，在这种不停的较量中G就模拟出来了 跟真实数据十分相近的数据。所以GAN主要的应用场景就是能够学习出这样模拟分布的数据，并且可以用模拟分布代替原始数据的地方！

GAN优化目标函数：

$$\min_G \max_D V(D,G)$$

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

D想办法增加V的值，G想办法减小V的值，两人在相互的对抗。

下面讲这个式子的数学意义。

首先固定G训练D ：

$$\max_D(\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))])$$

1）训练D的目的是希望这个式子的值越大越好。真实数据希望被D分成1，生成数据希望被分成0。

第一项，如果有一个真实数据被分错，那么log(D(x))<<0,期望会变成负无穷大。

第二项，如果被分错成1的话，第二项也会是负无穷大。

很多被分错的话，就会出现很多负无穷，那样可以优化的空间还有很多。可以修正参数 使V的数值增大。

2）训练G ，它是希望V的值越小越好，让D分不开真假数据。

因为目标函数的第一项不包含G，是常数，所以可以直接忽略 不受影响。

$$\min_G(\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))])$$

There is no G in this term.

对于G来说 它希望D在划分他的时候能够越大越好，他希望被D划分1(真实数据)。

$$\Rightarrow \min_G(\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))])$$

$D(G(z))$ should not be 0 => $D(G(z))$ should be 1

$$\Rightarrow \max_G(\mathbb{E}_{z\sim p_z(z)}[\log(D(G(z))])$$

第二个式子和第一个式子等价。在训练的时候，第二个式子训练效果比较好 常用第二个式子的形式。

证明V是可以收敛导最佳解的。
（1）global optimum 存在
（2）global optimum训练过程收敛

全局优化首先固定G优化D，D的最佳情况为：

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

1。证明D*G(x)是最优解
由于V是连续的所以可以写成积分的形式来表示期望：

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z))]$$
$$= \int_x p_{data}(x)log(D(x))dx + \int_z p_z(z)log(1-D(G(z)))dz$$
$$x = G(z) \Rightarrow z = G^{-1}(x) \Rightarrow dz = (G^{-1})'(x)dx$$
$$\Rightarrow p_g(x) = p_z(G^{-1}(x))(G^{-1})'(x)$$
$$= \int_x p_{data}(x)log(D(x))dx + \int_x p_z(G^{-1}(x))log(1-D(x))(G^{-1})'(x)dx$$
$$= \int_x p_{data}(x)log(D(x))dx + \int_x p_g(x)log(1-D(x))dx$$
$$= \int_x p_{data}(x)log(D(x)) + p_g(x)log(1-D(x))dx$$

$$1 = \int p_g(x)dx = \int p_z(z)dz$$
$$= \int p_z(z)(G^{-1})'(x)dx$$
$$= \int p_z(G^{-1}(x))(G^{-1})'(x)dx$$
从而推出
$$p_g(x) = p_z(G^{-1}(x))(G^{-1})'(x)$$

通过假设x=G(z)可逆进行了变量替换，整理式子后得到：

$$V(G,D) = \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x})\log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z})\log(1-D(g(\boldsymbol{z})))dz$$
$$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x})\log(D(\boldsymbol{x})) + p_g(\boldsymbol{x})\log(1-D(\boldsymbol{x}))dx$$

然后对V(G,D)进行最大化：对D进行优化令V取最大

$$\max_D V(D,G) = \max_D \int_x p_{data}(x)log(D(x)) + p_g(x)log(1-D(x))dx$$

$$\frac{\partial}{\partial D(x)}(p_{data}(x)log(D(x)) + p_g(x)log(1-D(x))) = 0$$

$$\Rightarrow \frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1-D(x)} = 0$$

$$\Rightarrow D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

取极值，对V进行求导并令导数等于0.求解出来可得D的最佳解D*G(x)结果一样。

2。假设我们已经知道D*G(x)是最佳解了，这种情况下G想要得到最佳解的情况是：G产生出来的分布要和真实分布一致 即：

$$p_{data}(x) = p_g(x)$$

在这个条件下，D*G(x)=1/2.

接下来看G的最优解是什么，因为D的这时已经找到最优解了，所以只需要调整G ，令

$$C(G) = \max_D V(G, D)$$
$$= \max_D \int_x p_{data}(x)log(D(x)) + p_g(x)log(1 - D(x))dx$$

对于D的最优解我们已经知道了，D*G(x)，可以直接把它带进来 并去掉前面的Max

$$= \int_x p_{data}(x)log(D_G^*(x)) + p_g(x)log(1 - D_G^*(x))dx$$
$$= \int_x p_{data}(x)log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}) + p_g(x)log(\frac{p_g(x)}{p_{data}(x) + p_g(x)})dx$$

然后对 log里面的式子分子分母都同除以2，分母不动，两个分子在log里面除以2 相当于在log外面 -log(4) 可以直接提出来：

$$= \int_x p_{data}(x)log(\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}) + p_g(x)log(\frac{p_g(x)}{\frac{p_{data}(x)+p_g(x)}{2}})dx - log(4)$$
$$= KL[p_{data}(x)||\frac{p_{data}(x) + p_g(x)}{2}] + KL[p_g(x)||\frac{p_{data}(x) + p_g(x)}{2}] - log(4)$$

结果可以整理成两个KL散度-log(4)

$$C(G) = \underbrace{KL[p_{data}(x)||\frac{p_{data}(x) + p_g(x)}{2}]}_{\geq 0} + \underbrace{KL[p_g(x)||\frac{p_{data}(x) + p_g(x)}{2}]}_{\geq 0} - log(4)$$

$$\min_G C(G) = 0 + 0 - log(4) = -log(4)$$

KL散度是大于等于零的，所以C的最小值是 -log（4）
当且仅当

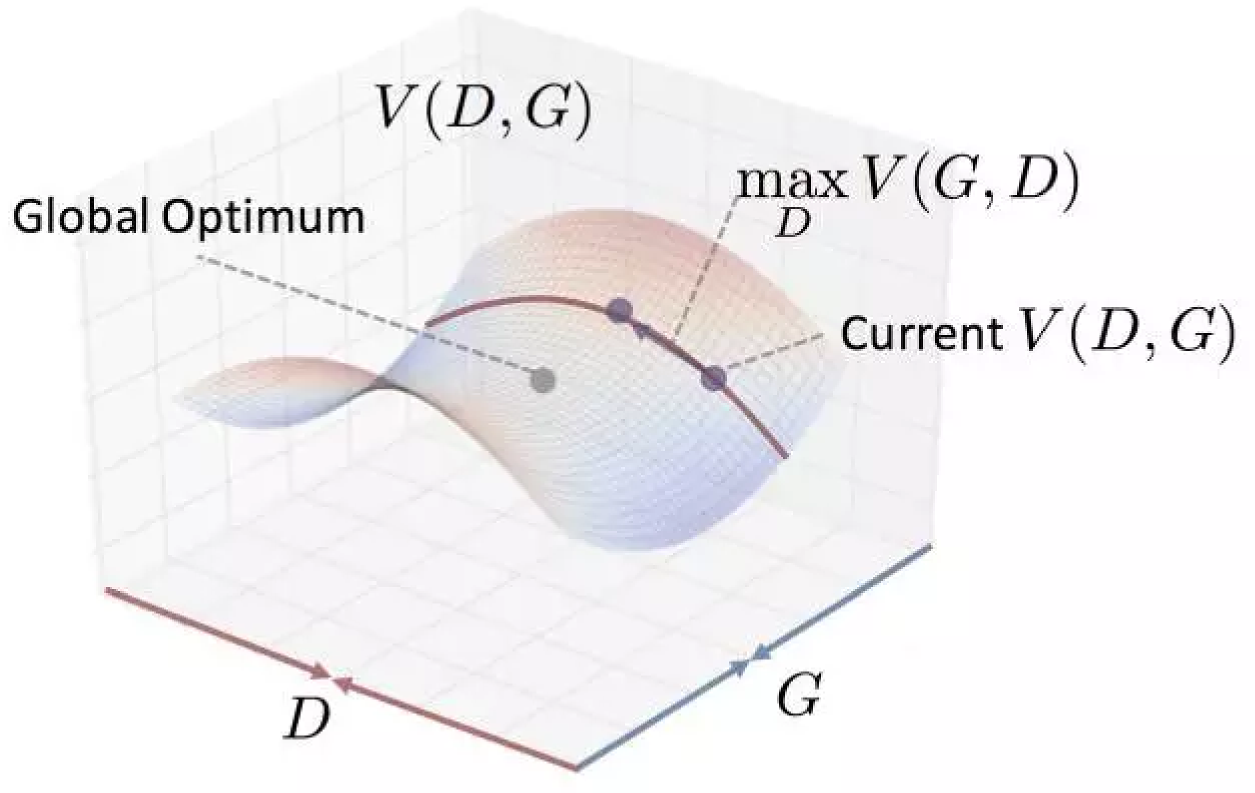
$$p_{data}(x) = \frac{p_{data}(x) + p_g(x)}{2}$$

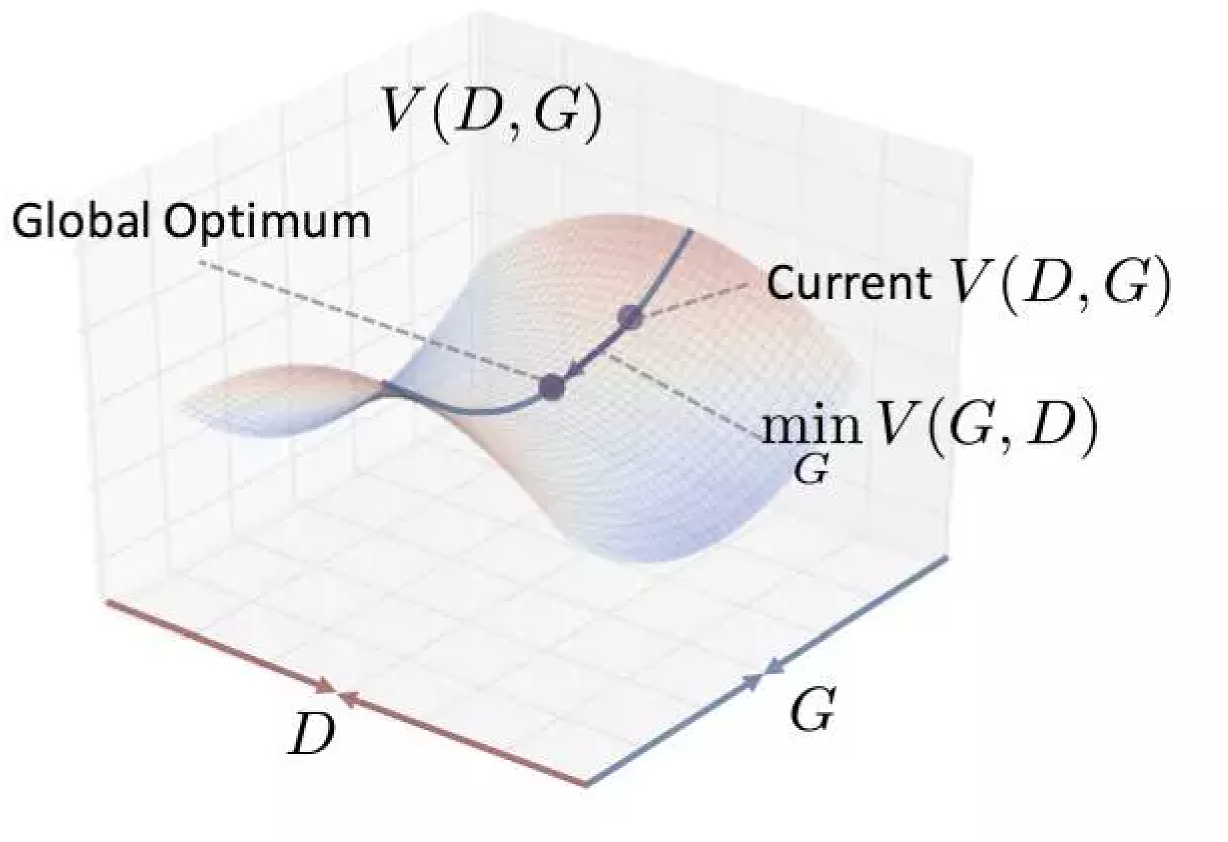
即

$$\Rightarrow p_{data}(x) = p_g(x)$$

所以证明了 当G产生的数据和真实数据是一样的时候，C取得最小值也就是最佳解。

证明收敛：





核心部分伪代码描述：

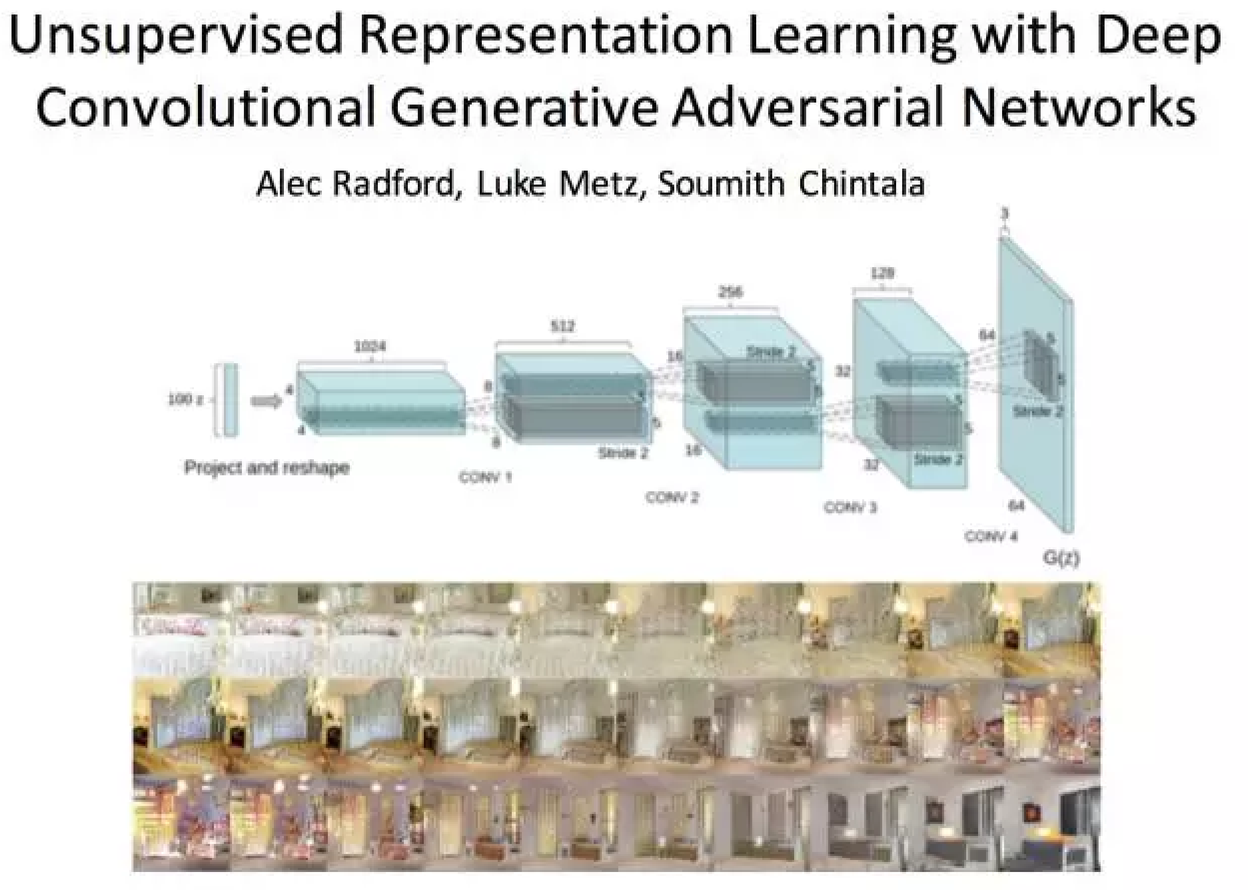
For number of training steps
        Sample n training images, X
        Compute n generated images, G

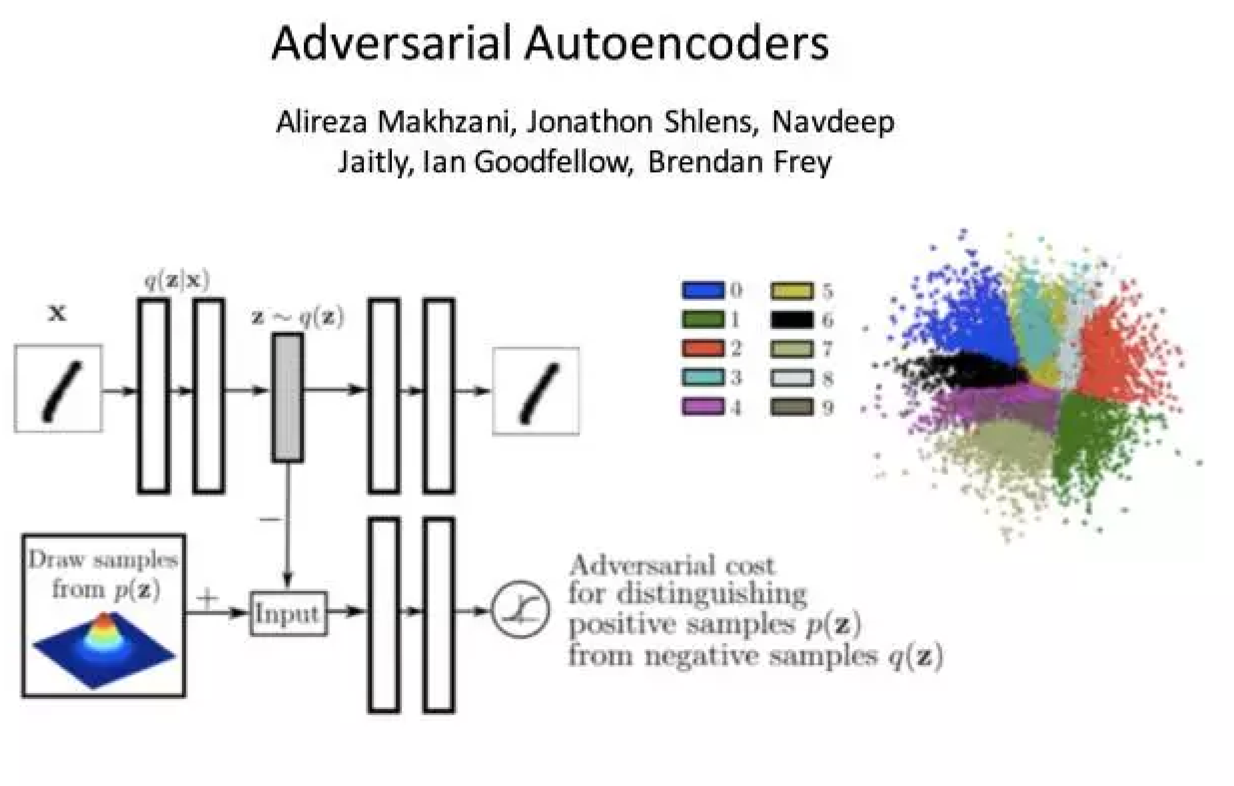
        Compute discriminator probabilities for X and G
        Label training images 1 and generated images 0
                Cost = (1/n)sum[log(D(X)) + log(1 - D(G))]
        Update discriminator weights, hold generator weights constant

        Label generated images 1
                Cost = (1/n)sum[log(D(G))]
        Update generator weights, hold discriminator weights constant

---

Further Research：DCGAN和Adversrial Autoencoders





---

[1]零和博弈（zero-sum game），又称零和游戏，与非零和博弈相对，是博弈论的一个概念，属非合作博弈。指参与博弈的各方，在严格竞争下，一方的收益必然意味着另一方的损失，博弈各方的收益和损失相加总和永远为“零”，双方不存在合作的可能。就像下棋的游戏一样，你走的每一步和对方走的每一步都是向着对自己有利的方向走，然后你和对手轮流走步 每一步都向着自己最大可能能赢的地方走。这就是零和博弈。

$$v_s = \max_{a_s} \min_{a_{s-}} v_s(a_s, a_{s-})$$

- s : the current player
- s^- : the opponent
- a_s : the action taken by current player
- a_{s-} : the action taken by opponent
- v_s : the value function of current player

Vs代表S能赢的可能性，S是当前棋手，S-是对方，aS是当前棋手可以做的选择，aS-是对方可做的选择。对于S来讲V越大越容易赢。

==

参考资料：

文章：2014 GAN 《Generative Adversarial Networks》-Ian Goodfellow, arXiv:1406.2661v1
视频：https://www.youtube.com/watch?v=7zXDGuW0Eq4
slides：https://www.slideshare.net/ckmarkohchang/generative-adversarial-networks
blog：http://blog.csdn.net/solomon1558/article/details/52537114
http://www.cnblogs.com/Charles-Wan/p/6238033.html
http://www.cnblogs.com/wangxiaocvpr/p/6069026.html
DCGAN：
文章：2015 DCGAN《Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks》 - Alec Radford & Luke Metz, arxiv:1511.06434
代码：https://github.com/kkihara/GAN
post：http://kenkihara.com/projects/GAN.html
slides：https://github.com/kkihara/GAN/blob/master/presentation.pdf