# Title:

Generative Adversarial Nets

# Abstract:

我们提出了一种新的框架，通过对抗过程来估计生成模型。在这个框架中，我们同时训练两个模型：一个生成模型G，用于捕捉数据分布；一个判别模型D，用于估计样本来自训练数据而不是G的概率。对于G的训练过程是最大化D犯错的概率。这个框架对应于一个最小最大二人博弈（博弈论内著名的游戏）。在任意函数G和D的空间中，存在唯一的解，其中G恢复训练数据分布，D等于二分之一。当G和D由多层感知器定义时，整个系统可以使用反向传播进行训练。在训练或生成样本的过程中，不需要任何马尔可夫链或近似的推理网络展开。实验通过定性和定量评估生成的样本展示了该框架的潜力。

# Intro：

深度学习在生成模型上进展不多，因为要在最大化似然函数的时候对概率分布做很多近似，这个近似带来了很大的计算困难。本文可以不再去近似似然函数，用别的方法得到一个计算上更好的模型。

这是一个对抗网络框架，框架内是MLP。框架内有个生成模型和判别模型，下面举一个例子来说明这两个模型：生成模型就像造假的人，目标是生成假币，判别模型就像警察，目的是区分真币和假币，造假者和警察会不断学习，造假者会不断提升造假的性能，警察会不断学习区分真币假币的技能，最后希望造假者能赢，生成出和真币无二，警察没有能力分辨出是否真币。这个时候，就能用这个模型来生成和真实数据无二的数据了。

因为两个模型都是基于MLP，可以通过基于误差的反向传递来计算，从而不用使用像马尔科夫链一样的算法来对一个分布进行复杂的采样，从而在计算上有优势。

# Relate work：

之前的方法总是想构造出一个分布函数出来，然后提供一些参数让它可学习，这些参数通过最大化它的对数似然函数来做，这样的坏处是采样一个分布时计算困难，特别是对于高维的分布。

最近有个工作叫“generative mechines”，它是学一个模型来近似结果，坏处是不知道最后的分布是什么样子，好处是计算比较容易

作者观察到，对f的期望求导等价于对f自己求导，这是使用误差的方向传递求解的原因



VAE是一个和GAN类似的工作

NCE也是类似的工作，但是使用的损失函数比较复杂，求解性没有GAN这么好

讲述了和predictabilty minimization的区别

最后讲了和adversarial example的区别：对抗性示例是构造出一些假的样本，但是和真的样本长得很像，能使分类器出错，从而测试算法的稳定性

# Adversarial nets

GAN这个框架最简单的应用是生成器和辨别器都是MLP的时候：

生成器需要学习一个在数据上的分布，接下来，生成器如何生成呢？在一个输入噪音变量上面定义一个先验，生成模型就是把映射成，映射空间为：生成器是个MLP，可学习参数为。

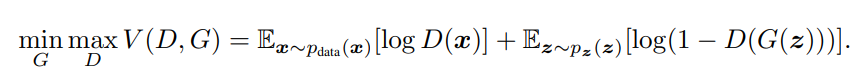
辨别器也是一个MLP，可学习参数为，判别器生成一个标量（0~1），判别图片是生成出来的还是来自于数据的，如果来自于的就输出1，来自生成器就输出0。

在训练D的时候，也会去训练G，训练G来最小化这一项：

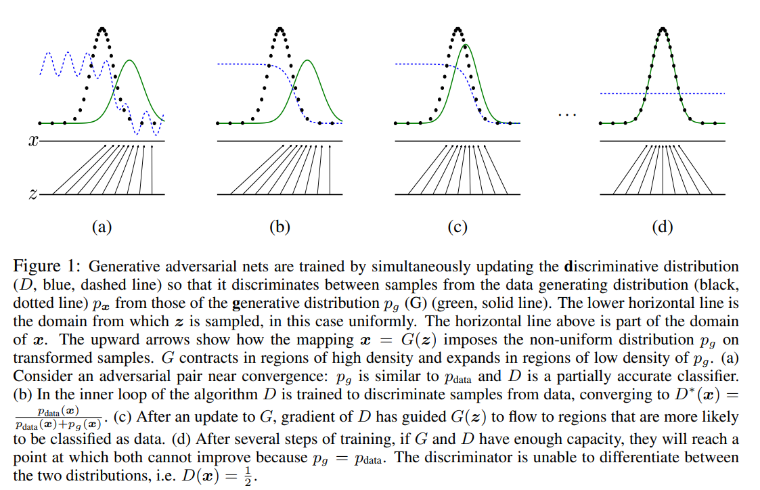


解释：为随机噪音，放入生成一个图片，再放入D中判别，输出0~1，log的输出为负无穷至0，若是训练一个G最小化这一项就是让D尽可能的犯错，让D无法分别是生成的还是真实的

我们要训练两个模型，一个叫G，一个叫D，目标函数为以下两人的min-max游戏：



解释：V是一个价值函数，第一项是期望，x采样真实分布，把x放入辨别器内加上log，假设辨别器能分辨出，这一项就等于0；第二项采样于噪音分布，在D是完美的情况下，这一项应该等于0；但D不是完美时，这两项会变成一个负数值，所以要训练D，就要最大化这两个期望，而要训练G，就要最小化第二项；最终D和G都不能再进步了，就认为达到了一个均衡（纳什均衡Nash equilibrium）。



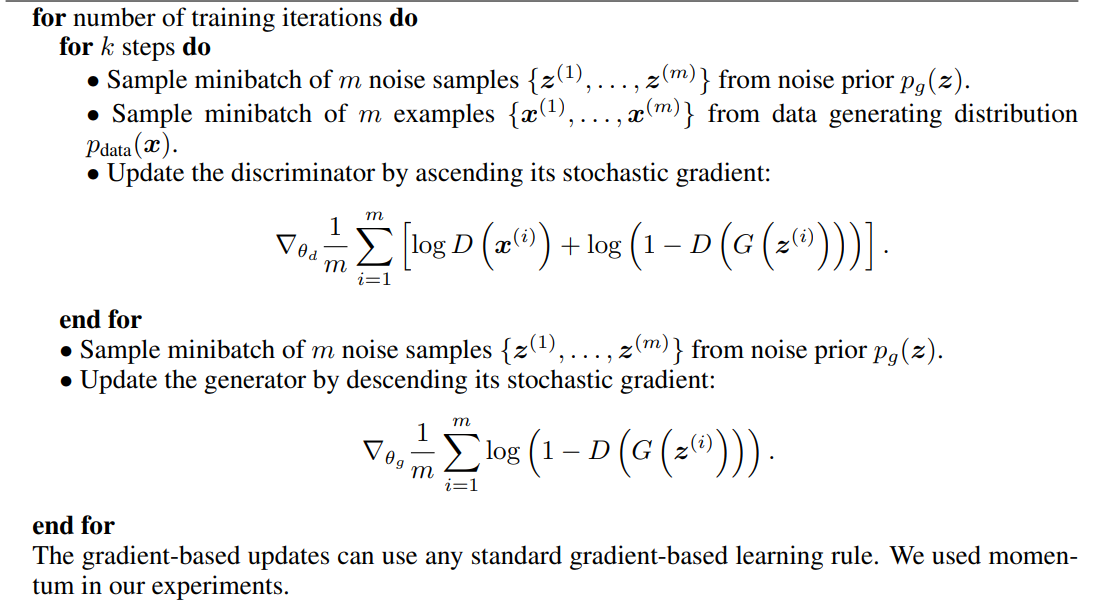
z和x都是一维标量，z为均匀分布采样，拟合的x为高斯分布

绿色线是生成器，辨别器是蓝色的线，黑色点是x采样

a，b，c，d效果越来越好，最后拟合

# Theoretical Results

## Algorithm 1

迭代N次直到完成

做k步（超参数）

2m大小的批量

放入价值函数对辨别器求梯度

再在噪音采样m个样本

放入生成器求梯度

如果辨别器更新太完美，这项会变成0，导致训练不动

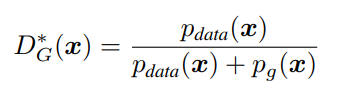
GAN的收敛非常不稳定，之后有非常多的工作对其进行改进

目标函数有全局最优解，当且仅当生成器学到的分布和真实的分布相等的情况

## Global Optimality of pg = pdata

**结论1：**

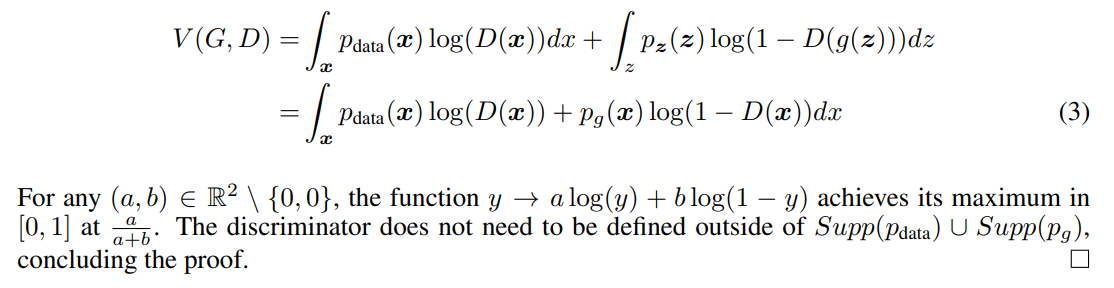
生成器固定住，最优的辨别器D是



生成器拟合的分布和真实分布完全相等时，D=1/2，表示这两个分布是完全分不开的

在统计领域非常有用：训练一个二分类器，如果分类器能完全分开这两个分布就说明他们不相等，如果不能分开，就说明他们来自同一个分布

例：在训练集训练的模型部署到另一个环境，要看它真实的数据是否和训练数据分布一样的时候，就可以用这个方法，避免模型和环境不匹配的问题

**证明：**

x=g(z)，z和x一一对应，可以对第二项进行换元操作

将pdata(x)替代成a，D(x)替代成y，pg(x)替代成b，成为一个一元凸函数，最大值就是a/(a+b)

***相对熵（KL散度）：***

相对熵又称KL散度,如果我们对于同一个随机变量 x 有两个单独的概率分布 P(x) 和 Q(x)，我们可以使用 KL 散度（Kullback-Leibler (KL) divergence）来**衡量这两个分布的差异**。

在机器学习中，P往往用来表示样本的真实分布，Q用来表示模型所预测的分布，那么KL散度就可以计算两个分布的差异，也就是Loss损失值。

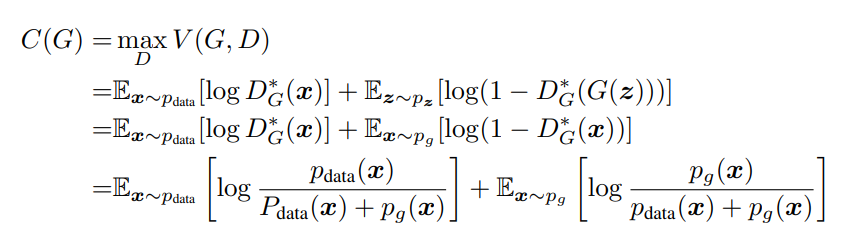
表示知道p的情况下，至少需要多少bit，能够把q描述出来

从KL散度公式中可以看到Q的分布越接近P（Q分布越拟合P），那么散度值越小，即损失值越小。

因为对数函数是凸函数，所以KL散度的值为非负数。有时会将KL散度称为KL距离，但它并不满足距离的性质：

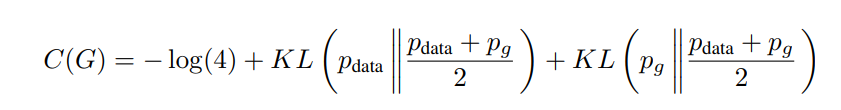
1.KL散度不是对称的；

2.KL散度不满足三角不等式。

接下来把最优解D带入价值函数，展开：

成为一个关于G的函数，当pdata=pg时，G取得全局最优解

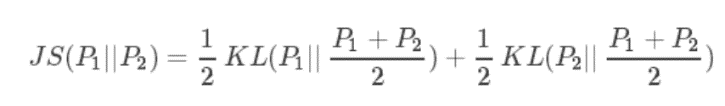
相当于两个KL散度相加



KL散度是大于等于0的，当取最优解时，两个KL散度等于0

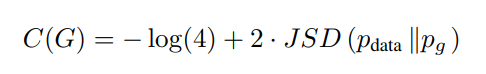
***JS散度（Jensen-Shannon divergence琴森香农散度）：***

JS散度度量了两个概率分布的相似度，基于KL散度的变体，解决了KL散度非对称的问题。一般地，JS散度是对称的，其取值是0到1之间。定义如下：



跟KL散度的区别：JS散度是对称的，而KL散度不是对称的（p、q不能互换）

上一个式子可以变成：



## Convergence of Algorithm 1

**命题2：**

当D和G有足够的容量时，在算法1中每一步D可以达到它的最优解，对G的迭代换成下面这一个步骤：



最后pg会收敛到pdata

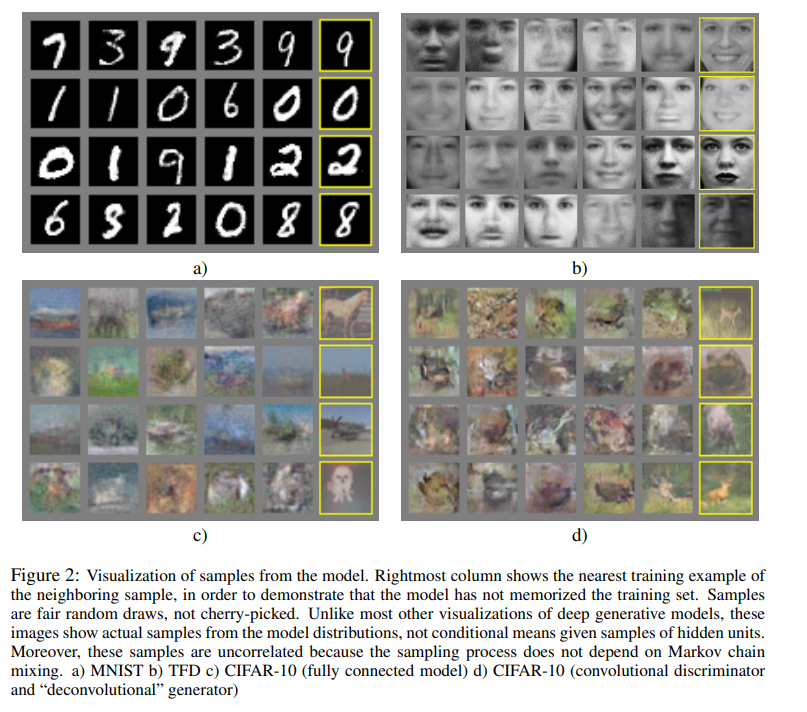
**证明：**

把目标函数看成pg的函数：



这是一个关于函数的函数（输入是一个函数：在python内函数接收一个closure：一个值和一个计算），之前在一个高维的值空间内做迭代，现在要在函数空间内做梯度下降。

pg是一个很简单的凸函数。一个凸函数的上限函数还是一个凸函数，最后做梯度下降时会得到一个最优解



这是在当时用GAN生成的一些图片

# Advantages and disadvantages

坏处是训练起来比较难，G和D要均衡好，没有平衡好的话生成器会训练不佳