Generative Adversarial Nets

Abstract

本文提出了一种新架构，称为GAN（干！）。具体如下，框架使用了两种模型，一种是生成模型G，一种是判别模型D，不使用马尔可夫和推理，效果非常好。

Introduction

深度学习在判别模型上很有效，但是生成模型由于需要近似分布估计，因而始终没有什么进展。

GAN的思想就是道高一尺（D），魔高一丈（G）。G和D类似造假者和警察关系，G造假钞（生成），D来验钞（辨别），最后互相内卷，卷到最后D发现自己辨别不了假钞和真钞了，于是G的假钞就可以拿去用了。

同时，G可以把随机噪声通过MLP映射到任何一个高斯分布上，D也是个MLP。这种情况下，就可以使用简单的反向传播计算梯度，无需使用任何近似或者马尔可夫。

Related work

先前的工作是构造一个分布，试图使用最大似然函数学习分布，由于维度很高导致计算非常困难；现在的方法是不去构造分布函数，直接使用模型去近似结果，极大减少计算量，但是坏处是不知道分布。同时作者还发现对f（x）的期望E[f(x)]求导等于对f（x）求导

$$\lim_{\sigma \to 0} \nabla_{\boldsymbol{x}} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} f(\boldsymbol{x} + \epsilon) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}).$$

Adversarial nets

为了学习数据x的分布pg，我们先定义输入噪音z的分布pz（z），然后利用生成模型$G(z;\theta_g)$先整个生成输出，其中øg是可学习的参数。

接着，我们还要请出判别模型D（x），这个模型是为了辨别输入x是不是真实数据（和生成的造假数据相对），如果判别器D越肯定x是假的，则D（x）输出越接近于0；反之，如果判别器D越肯定x是真的，则D（x）输出越接近于1。

在整个训练步骤中，我们先要训练出一个效果足够好（不是最好，原因后面再说）的判别器D，来鉴别G生成的例子。为了做到这点，我们需要最大化以下公式，$\log(1 - D(G(z)))$。G此时生成的例子还十分粗糙，而我们的目标就是先让D性能足够优越，尽可能将他们判断为不真实的信息，因此要使D（G（Z))尽可能小，也就是使$\log(1 - D(G(z)))$尽可能大；

同时为了避免仅输入不真实数据G（z）导致出现奇怪的一刀断现象（相当于D摆烂表示反正我估摸你全是假数据，那我就面对问题输出，全都给你0)的结果，我们还需要用真实数据x训练D。因此，在提供数据时，我们一般会提供数量为m的真数据x、和数量同样为m的生成数据G（z)，也就是数据总共2m，这也就要求我们同时要训练D使得$\log D(x)$足够接近1。这两点合并起来，就是最大化以下式子期望了

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

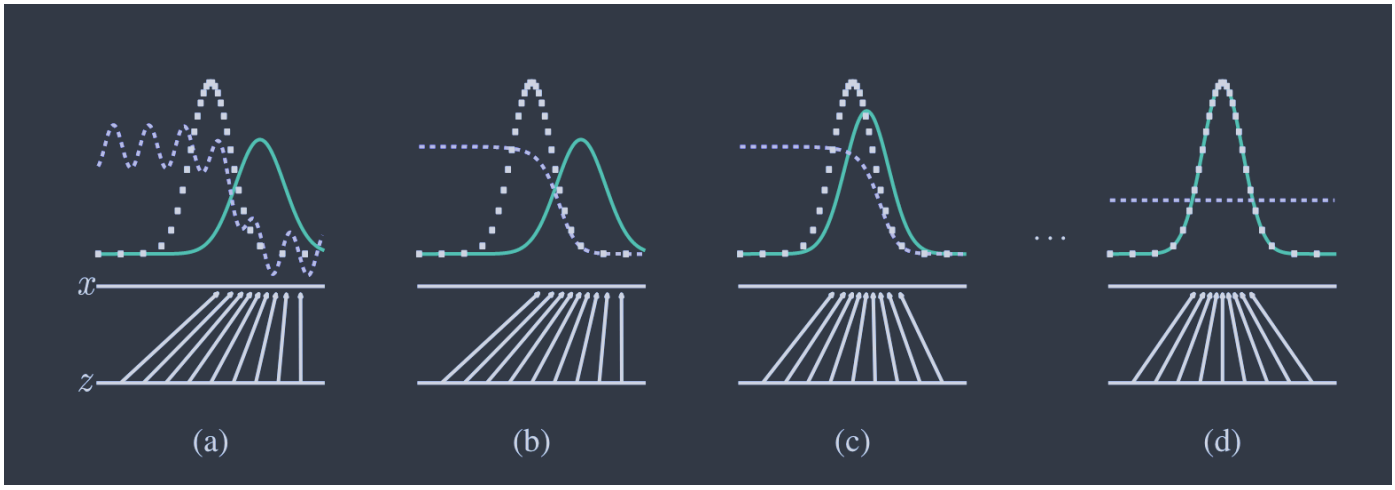
在将警察D训练地差不多后，我们就要训练造假犯人G了。这里可以认为D是已经固定了，因此我们要训练G使得其尽量瞒过D，让D输出0。因此也就是最小化上述的式子

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

在博弈论中也叫纳什均衡，这就是paper中公式的来源啦。

同时我们上面提到，不能让判断器D或者生成器G先训练地太完美，通俗来讲，为了促进造假生态繁荣发展，（假如我们的最终目的是得到完美的假钞）我们首先在初期不能有太厉害的警察，否则造假犯刚冒头就进局子了，无法发展造假技术；同时，我们也不能让造假者技术领先警察数个版本，导致警察叔叔根本认不出假钞，从而造假犯直接躺平，无法持续奋斗更新造假技术，这是如此的倦怠，不行！

从数学上的角度来说，如果一方性能过好，例如在训练G性能爆表时，会导致$\log(1-D(G(z)))$过小，最坏情况是趋近于负无穷，使得梯度过大，难以优化。所以作者也提出在训练时用最大化$\log D(G(z))$来替代最小化$\log(1-D(G(z)))$。



以图像为例子，白虚线是真实数据x；绿线是生成数据G（z）；蓝线是判别器D，数值越靠近1（曲线越高）表示越肯定这是真实数据x，越靠近0（曲线越低）表示越肯定这是生成数据G（z）。如图a，此时G和D都没有被充分训练；图b中D经过了一定的训练，因此可以很好的区分出x和G（z）；图c中G也经过了一定训练，所以D也开始遇到一定困难；图d中G已经训练地比较完美了，G（z）基本和x没有区别，做到了以假乱真的地步，因此D也无法辨别出G（z）和x，只能给一个1/2的中立结果表示“我也不知道啊”，这也就是为什么D的值到最后保持在1/2附近。数学上来解释即是，训练到最后，最优判别器D的结果是$\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}$，即真实数据概率pdata

（x)占总概率[pdata（x)+pg（x)]的比例，到了最后G趋近于完美时，pdata（x)和pg（x)基本相等，因而D最后输出为1/2。

具体证明如下，将

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

的期望展开，得到

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

要得到最优判别器D就是要最大化上式。

将后面那一块中的G（z)用x代替，则关于z的概率分布pz（z)可以被替代为pg（x)，得到

$$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx$$

对被积分式子分析，可以写作

$y \rightarrow a\log(y) + b\log(1-y)$，求导后得到y=a/（a+b)时该式最大。将a和b分别用pdata（x)和pg（x)替代，则得到最优判别器$D_G^*(\boldsymbol{x})$=

$$\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

因此以上优化D的式子可以变为

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

值得一提的是 $\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}$ 并不是一个概率，因为它的期望是1/2，但是如果将分母乘1/2，则期望变成1，成为完整的概率。而在计算时分母的1/2被log变换出去变成常数-log2，因而可以忽略。如果你非要看，也有完整式子：

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

具体算法流程如下

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

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

实验部分咱就不讲啦，还要上数电课呢～

过几天看懂了ViT和Gaitset再继续更新，就当是新人练手吧。