GANs生成对抗网总结

在自然界中**某个具体事务**的组成，是由特征随着一个特定的规律分布组合的，如果我们能够知道这个分布组合规律，那么我们就可以重现出这个**具体事务**。而生成对抗网络真实基于此过程，来学习这个分布函数的神经网络模型。

P_g 是x的分布函数，**这里的x是真实数据，比如一张图片的分布，一个数据的分布，是真正的分布。**

P_z 是噪声z的分布函数，z是无规律的随机生成的数据，然后计算出它的分布规律是什么；它是一个预先定义的量。

鉴别器**D(x)**，是一个神经网络模型，它的作用是**衡量 x** 与 **真实事务** 的 **特征分布组合规律** 的差距有多大。

在论文中x是真实的数据，由于刚开始**鉴别器**还是一个初始的状态，还不能根据 **特征分布组合规律** 来 有效的分清 给到的 **真实数据x参数** 与 **真实事务** 那么我们需要通过训练他，是它见到真实的数据 能够得出是真实数据，而虚假生成的数据 它能判别出是虚假的数据。

G(z) 是根据产生的随机噪声，然后生成的一个图片的**神经网络模型**。

小结：D(x)与G(z)都是神经网络模型，他们都是学习真实数据的分布，只是达到的效果不同。**D(x)** 学习到了 **真实数据分布** 后，鉴别虚假能力越来越强；**G(z)** 学习到了 **真实数据分布** 后，根据真实数据的分布，能够产生跟真实数据一样分布的数据。

D(x) 判别标准是 **真实数据分布**，拿到一个x，首先计算出他的分布，然后再进行判定；

G(z) 生成的规则是，学习到了 **真实数据分布** ，依据该 真实数据分布 产生数据。

论文中提到的训练逻辑

搞清楚他们是怎么操作的，我们要知道训练逻辑与论文中具体的公式推导意义。

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

在通常的神经网络模型的共性中，如果一个神经网络计算结果值越大，那么效果是越好的，所以我们知道D(x)，他是一个**增函数**，如果x越接近真实数据分布，那么D(x)值就越大，论文中也给出了**[0, 1]**区间。

鉴别器模型公式

首先查看鉴别器的模型公式

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

为了让鉴别器越来越强，我们需要这个公式产生的值越来越大。

使 $logD(x)$ 越来越大；

使 $log(1-D(G(z)))$ 越来越大，而要使得该公式越来越大，那么 $D(G(z))$ 越来越小，也就是希望G(z)的生成能力变差。

然后在看生成器模型公式：

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

生成器需要这个模型表现的很好，这里是一个减函数，让这个模型产生的值越来越小。那么也就是需要让G(z)的生成能力变强。

在这个论文的后续改进中，人们已经不这样做了，也把生成器改成增函数形式。即：$log(D(G(z)))$ 。

训练过程：

训练过程也是首先，先计算先计算鉴别器，然后使用优化函数，对鉴别器模型中的参数进行优化，让鉴别器表现更好。

让鉴别器表现更好同时，势必会拉低生成器模型的效果，因为鉴别器中第二项是生成器的模型。

紧接着下一步是优化生成器，生成器是**提高**生成器的能力，在生成器提高后，导致的问题鉴别器的能力不如以前，又拉低了鉴别器。

如此往复进行，直到他们互相平衡。

代码复现

```
# 鉴别真实数据
real_cpu = data[0].to(device)
b_size = real_cpu.size(0)
label = torch.full((b_size,), real_label, dtype=torch.float, device=device)
output = netD(real_cpu).view(-1)
errD_real = criterion(output, label)
# 反响传播鉴别器
errD_real.backward()
D_x = output.mean().item()

# 生成fake数据，然后鉴别
noise = torch.randn(b_size, nz, 1, 1, device=device)
fake = netG(noise)
label.fill_(fake_label)
output = netD(fake.detach()).view(-1)
errD_fake = criterion(output, label)
# 反向传播鉴别器
errD_fake.backward()
D_G_z1 = output.mean().item()
# 仅仅用于显示
errD = errD_real + errD_fake
# 鉴别器D 参数更新
optimizerD.step()
```

D识别真实图片数据

G生成Fake->D识别虚假的数据

然后优化更新鉴别器

我们可以发现到，DCGAN论文中的生成器模型公式符合：**D(x) + D[G(z)]**

```
##############################
# (2) Update G network: maximize log(D(G(z)))
##############################
netG.zero_grad()
label.fill_(real_label)
output = netD(fake).view(-1)
errG = criterion(output, label)

errG.backward()
D_G_z2 = output.mean().item()
# 生成器G 参数更新
optimizerG.step()
```

优化后的鉴别器 鉴别 Fake，然后优化

可以发现到生成器的公式**D(G(Z))**