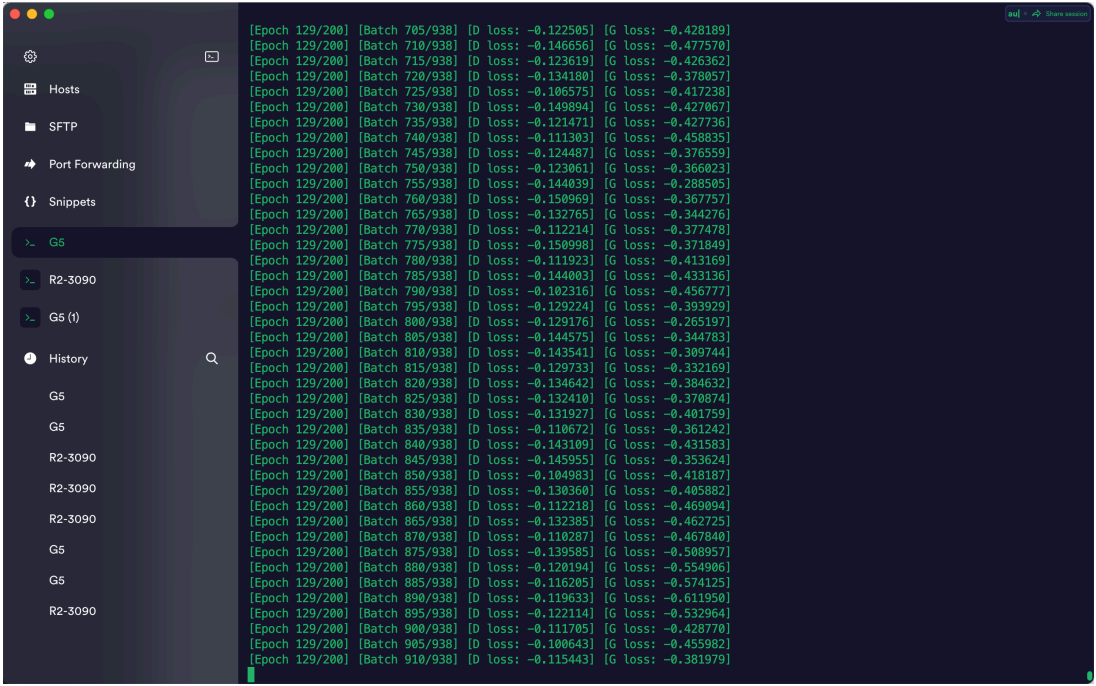


文柯力 CV 作业 #Week9

- 文柯力 CV 作业 #Week9
- WGAN 实验结果展示
- WGAN 阅读笔记
- WGAN 代码对照
- Reference

WGAN 实验结果展示

训练过程展示



训练结果展示



从左到右分别是训练第 1, 96401, 187201 Batch 得到的结果。

WGAN 阅读笔记

在知道WGAN的优势前，我们想知道原始的GAN有什么问题？原始的GAN有“the - log D alternative”或“the - log D trick”两种形式。

形式一，当我们找个一个足够优秀的判别器 D 的时候，标准GAN的优化目标等价于 $JS(P_r||P_f)$ ，然而，当 P_r 和 P_f 没有交集的时候， $JS(P_r||P_f)$ 始终为 \log_2 。也就是说，这种情况下， G 的梯度消失了。

形式二，简单来说最小化第二种生成器loss函数，会等价于最小化一个不合理的距离衡量，导致两个问题，一是梯度不稳定，二是collapse mode即多样性不足。

总结就是：在原始GAN的（近似）最优判别器下，第一种生成器loss面临梯度消失问题，第二种生成器loss面临优化目标荒谬、梯度不稳定、对多样性与准确性惩罚不平衡导致mode collapse这几个问题。

而 EMD 距离也就是 W 距离相较于 JS 和 KL 散度更为优秀。它其实计算的是一种匹配。

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|] \tag{1}$$

直观上可以把 $E(x, y) \sim \gamma[\|x - y\|]$ 理解为在 γ 这个“路径规划”下把 P_r 这堆“沙土”挪到 P_g “位置”所需的“消耗”，而 $W(P_r, P_g)$ 就是“最优路径规划”下的“最小消耗”，所以才叫Earth-Mover（推土机）距离。与KL散度和JS散度比较，KL散度和JS散度是突变的，要么最大要么最小，**Wasserstein距离却是平滑的**，如果我们要用梯度下降法优化 θ 这个参数，前两者根本提供不了梯度，Wasserstein距离却可以。类似地，在高维空间中如果两个分布不重叠或者重叠部分可忽略，则KL和JS既反映不了远近，也提供不了梯度，**但是Wasserstein却可以提供有意义的梯度**。

但是原始的 Wasserstein 距离定义中的 $\inf_{\gamma \in \Pi(P_r, P_g)}$ 没法直接求解，作者用了一个已有的定理把它变换为如下形式

$$W(P_r, P_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim P_r} [f(x)] - \mathbb{E}_{x \sim P_g} [f(x)] \tag{2}$$

然后构造一个含参数 ω 、最后一层不是非线性激活层的判别器网络 f_ω ，在限制 ω 不超过某个范围的条件下，使得 L 尽可能取到最大，此时L就会近似真实分布与生成分布之间的Wasserstein距离（忽略常数倍数K）。注意原始GAN的判别器做的是真假二分类任务，所以最后一层是sigmoid，但是现在WGAN中的判别器fw做的是近似拟合Wasserstein距离，属于回归任务，所以要把最后一层的sigmoid拿掉。

$$L = \mathbb{E}_{x \sim P_r} [f_\omega(x)] - \mathbb{E}_{x \sim P_g} [f_\omega(x)] \tag{3}$$

WGAN与原始GAN第一种形式相比，进行了四点改进：

- 判别器最后一层去掉sigmoid 【为了拟合W距离】

- 生成器和判别器的loss不取log
- 每次更新判别器的参数之后把它们的绝对值截断到不超过一个固定常数c
- 不要用基于动量的优化算法（包括momentum和Adam），推荐RMSProp，SGD也行

WGAN 代码对照

下面是生成器的代码

```
1 class Generator(nn.Module):
2     def __init__(self):
3         super(Generator, self).__init__()
4         ...
```

下面的判别网络，利用整个判别器去拟合 W 距离：

```
1 class Discriminator(nn.Module):
2     def __init__(self):
3         super(Discriminator, self).__init__()
4
5         self.model = nn.Sequential(
6             nn.Linear(int(np.prod(img_shape)), 512),
7             nn.LeakyReLU(0.2, inplace=True),
8             nn.Linear(512, 256),
9             nn.LeakyReLU(0.2, inplace=True),
10            nn.Linear(256, 1),
11            # 最后不能带 sigmoid
12        )
13
14    def forward(self, img):
15        img_flat = img.view(img.shape[0], -1)
16        validity = self.model(img_flat)
17        return validity
```

优化算法部分：Finally, as a negative result, we report that WGAN training becomes unstable at times when one uses a momentum based optimizer such as **Adam** [8] (with $\beta_1 > 0$) on the critic, or when one uses high learning rates. We therefore switched to **RMSProp** [21] which is known to perform well even on very nonstationary（不稳定） problems [13].





此时求解公式(2)可以近似变成求解如下形式

$$K \cdot W(P_r, P_g) \approx \max_{w: \|f_w\|_L} \leq K \mathbb{E}_{x \sim P_r} [f_w(x)] - \mathbb{E}_{x \sim P_g} [f_w(x)] \quad (1)$$

为了满足公式 $\|f_w\|_L \leq K$ 这个限制。我们其实不关心具体的 K 是多少，只要它不是正无穷就行，因为它只是会使得梯度变大 K 倍，并不会影响梯度的方向。所以作者采取了一个非常简单的做法，就是限制神经网络 f_θ 的所有参数 w_i 的不超过某个范围 $[-c, c]$ ，比如 $w_i \in [-0.01, 0.01]$ ，此时关于输入样本 x 导数 $\frac{\partial f_w}{\partial f_x}$ 也不会超过某个范围。

```
1  # Optimizers
2  optimizer_G = torch.optim.RMSprop(generator.parameters(), lr=opt.lr)
3  optimizer_D = torch.optim.RMSprop(discriminator.parameters(), lr=opt.lr)
4
5  # Clip weights of discriminator
6  for p in discriminator.parameters():
7      p.data.clamp_(-opt.clip_value, opt.clip_value)
```

Reference

-  令人拍案叫绝的Wasserstein GAN
-  WGAN的成功，可能跟Wasserstein距离没啥关系
-  GAN论文阅读笔记2：不懂W距离也能理解WGAN
- Wasserstein GAN PDF
-  PyTorch-GAN: wgan