Lab6 : InfoGAN

0516069 翁英傑

1. Introduction

Train a generative adversarial network to learn to generate mnist digits.

2. Experiment setups

A. Adversarial loss :

$$\mathcal{L}_{\mathrm{D}} = -E_{x\sim p_r}[\log D(x)] - E_{x\sim p_g}[\log(1 - D(x))]$$

$$\mathcal{L}_{\mathrm{G}} = E_{x\sim p_g}[-\log D(x)]$$

Simultaneously minimize both loss function for generator and discriminator, we can generate well image.

B. Maximizing mutual information:

$$\begin{aligned} I(c; G(z,c)) &= H(c) - H(c|G(z,c)) \\ &= \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log P(c'|x)]] + H(c) \\ &= \mathbb{E}_{x\sim G(z,c)}[\underbrace{D_{\mathrm{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\geq 0} + \mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\ &\geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \end{aligned}$$

Mutual information is the entropy between the class predicted by a Q network from an image generated from the generator with a class condition and that condition class used in generator.

C. How you generate fixed noise and images:

```
idx = np.arange(noiseSize).repeat(noiseSize).reshape(noiseSize, noiseSize).transpose().reshape(noiseSize**2)
one_hot = np.zeros((noiseSize**2, 10))
one_hot[range(noiseSize**2), idx] = 1
fix_noise = torch.Tensor(np.random.normal(0, 1, (noiseSize, nz-10)).repeat(noiseSize, axis=0))
```

D. Which loss function of generator you used:

$$\mathcal{L}_{\mathrm{G}} = E_{x\sim p_g}[-\log D(x)] + \mathrm{L}_{\mathrm{I}}(G, Q)$$

3. Discussion

In my experiment, I've encounter a situation so called mode collapse, which is usually referred to a problem that generator outputs are identical. It may be caused by the distribution of true data to be multi-peaks, and that the generator

is only able to produce sub-group of the true data. And this may be triggered in a seemingly random fashion.

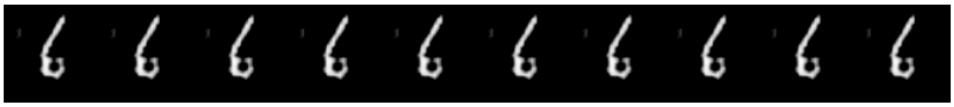
4. Results of testing

Fixed Condition (same digit different style)

```
# fixed random condition
single_idx = np.random.randint(10, size=1).repeat(noiseSize)
single_one_hot = np.zeros((noiseSize, 10))
single_one_hot[range(noiseSize), single_idx] = 1
single_fix_noise = torch.Tensor(np.random.normal(0, 1, (1, nz-10)).repeat(noiseSize, axis=0))

single_noise.data.copy_(single_fix_noise)
single_dis_c.data.copy_(torch.Tensor(single_one_hot))
const_condition = torch.cat([single_noise, single_dis_c], 1).view(-1, 64, 1, 1)

x_save = net_G(const_condition)
save_image(x_save.data, './tmp/const_condition.png', nrow=10)
Image('./tmp/const_condition.png')
```



Fixed meaningless noise (different digit same style)

```
# fixed random noise
single_idx = np.arange(noiseSize)
single_one_hot = np.zeros((noiseSize, 10))
single_one_hot[range(noiseSize), single_idx] = 1
single_fix_noise = torch.Tensor(np.random.normal(0, 1, (noiseSize, nz-10)))

single_noise.data.copy_(single_fix_noise)
single_dis_c.data.copy_(torch.Tensor(single_one_hot))
const_noise = torch.cat([single_noise, single_dis_c], 1).view(-1, 64, 1, 1)

x_save = net_G(const_noise)
save_image(x_save.data, './tmp/const_noise.png', nrow=10)
Image('./tmp/const_noise.png')
```