Lab 6: InfoGAN

陽明大學 不分系二年級 張凱博

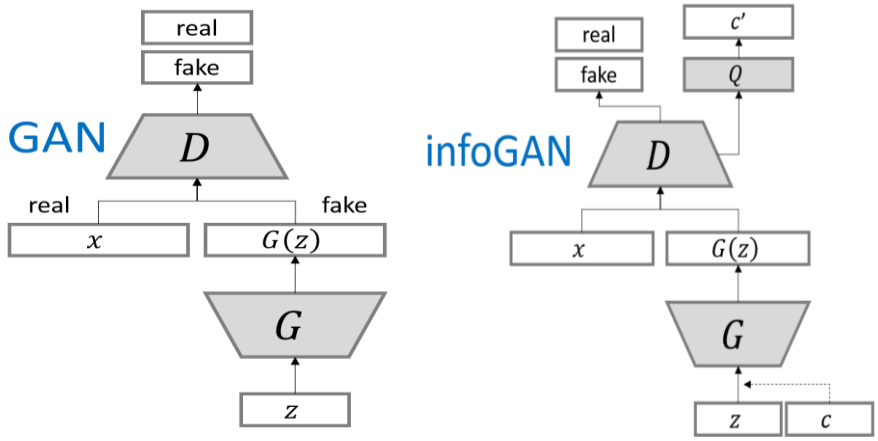
(一) Introduction

InfoGAN 之於 GAN 就像 Conditional VAE 之於 VAE，因為他們都有共通點，就是多加了限制條件。InfoGAN 在 Generator Network 加入一個新的潛變量 c，使得 c 與生成的樣本有較高的 Mutual Information。c 用於表示 label，z 用於表示真實樣本 x 中與 c 無關的雜訊，在最後時現網絡的時候，infoGAN 可以看成由 3 個網路組成:

(1) Generator Network: x = G(c, z)

(2) Discriminator Network: y1 = D1(x)

(3) Classifer Network: y2 = D2(x)，當 c 用於代表類別訊息的時候，網絡最後一層是 softmax 層；且除了網絡最後一層以外，D1 與 D2 共享網絡參數。



和傳統的 GAN 是一樣，InfoGAN 也是由生成器和判別器兩部分組成，但是，在對生成器進行輸入的部分做了一定的改變，由原來的只輸入噪聲 Z，改變為一種無監督的方式，輸入噪聲 Z 和 label c，即生成數據可以表示成：G(z,c)。在判別器輸出的部分，也做出了一定的調整，會輸出類別 C'，與此同時，會有一個簡單的 Q 網絡輸出（如上圖所示），對於 Q 網絡來說，其實，Q 網絡和 D 網絡共享同一個網絡，只是在最後一層作出了改變，其他部分都是一樣的。

(二) Experiment setups

A. How you implement InfoGAN

I. Adversarial loss

Adversarial loss 就是藉由輸入 fake data 給 discriminator，讓 discriminator 判斷它和 1(real data label)的差距，如果 binary cross entropy loss 的計算結果越接近 1 的話，那就代表 generator 的所產出的 data 越會接近 real data 的分布，之後 discriminator 再將 binary cross entropy loss 的計算結果和 mutual information loss 一起回饋給 generator 健行 optimization。

程式碼如下:

```
## D and Q part ##
optimG.zero_grad()

# adversarial loss
fe_out = FE(fake_x) # fe_out.size = ([100, 512, 4, 4])
probs_fake = D(fe_out) # probs_fake.size = ([100, 1])
label.data.fill_(1.0)

reconstruction_loss = criterionD(probs_fake, label)
```

II. Maximizing mutual information

在標準 GAN 中，如果直接輸入 G(z,c)作為網絡的輸入進行訓練，那麼生成器因為有很高的自由度，所以很容易可以一組解使得: PG(z|c) = PG(z)，即 z 與 c 相互獨立、不相關，如此一來，會忽略隱藏編碼 c 的作用。為了解決這個問題，文章提出正則化約束項：使得隱藏編碼 c 與生成樣本 G(z,c)的互信息量應該較大，即 I(c;G(z,c))應該較大。

使得 infoGAN 的損失函數為:

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c;G(z,c))$$

D 想辦法增加 V 的值 (真實數據希望被 D 分成 1，生成數據希望被分成 0)，G 想辦法減小 V 的值 (讓 D 分不開真假數據)，兩人在相互的對抗，最後加入正則向: mutual infomation。

```
# mutual info
q_output = Q(fe_out) # q_output.size = ([100, 10])
class_ = torch.LongTensor(idx).cuda() # class_.size = ([100])
target = Variable(class_)
dis_loss = criterionQ_dis(q_output, target)

G_loss = reconstruction_loss + dis_loss
G_LOSS.append(G_loss)

G_loss.backward()
optimG.step()
```

III. How you generate fixed noise and images

Batch size 的設定為 100，所以輸入到 generator 的矩陣為 100*64*1*1，其中第 2 個維度的組成是 54 + 10，54 是用 normal distribution sampling 的 noise (fixed noise)，另外 10 是將 minst 的數字組成標籤 0~9 轉為 one hot vector 的形式，最後將 2 者 concatenate 後的 tensor 輸入 generator 後產生圖片儲存即可。

```
# fixed random variables
c = np.linspace(-1, 1, 10).reshape(1, -1)
c = np.repeat(c, 10, 0).reshape(-1, 1)

idx = np.arange(10).repeat(10)
one_hot = np.zeros((100, 10))
one_hot[range(100), idx] = 1
fix_noise = torch.Tensor(100, 54).uniform_(-1, 1)
```

```
noise.data.copy_(fix_noise) # fix_noise.size = ([64, 54])
dis_c.data.copy_(torch.Tensor(one_hot))

z = torch.cat([noise, dis_c], 1).view(100, 64, 1, 1)
x_save = G(z)
save_image(x_save.data, '%s/real_samples.png' % '.', normalize=True, nrow=10)
```

B. Which loss function of generator you used

我使用的是:

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

程式碼如下:

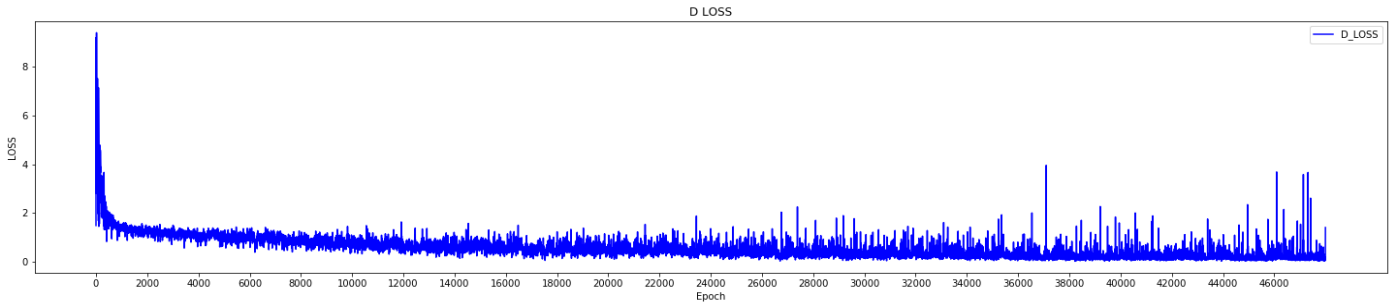
```
G_loss = reconstruction_loss + dis_loss
```

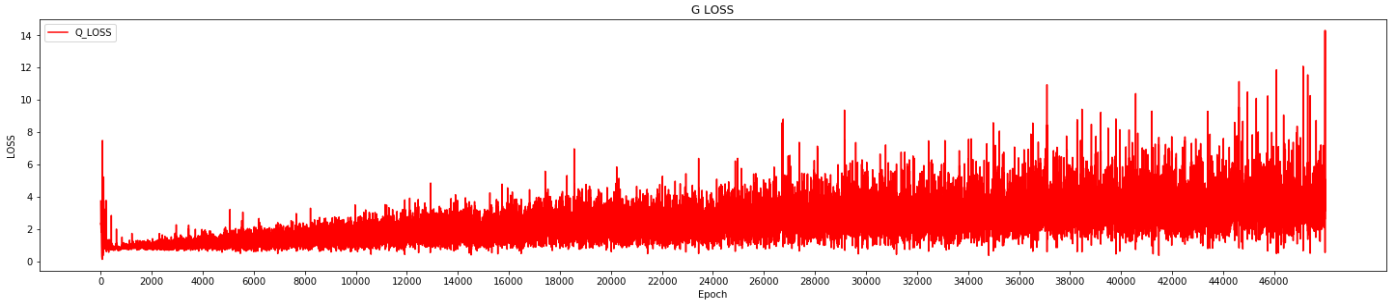
是 adversial loss + mututal information loss 的加總。

(三) Results

A. Results of your samples (shown as in the expected results section)

B. Training loss curves





(四) Discussion

在這次的 lab 中，是要利用 infoGAN 的 generator 產生出 0~9 的 image ，但是我產生出來的 image 卻有幾個長得不太像是 0~9 的數字，原因並不是訓練次數不夠多的問題，因為在最後的 discriminato loss 已經趨近於零，若當 discriminato loss 等於 0，那 generator loss 就會逼近 50、60，那麼所產生出來的 model 就會很會是很模糊的圖片，我在想可能是要把 noise 設定得再更多樣一點，就是有 noise1 和 noise2，使得一開始 discriminator 的 loss 很大，使他的學習時間拉長，這樣可能會有比較好的學習效果。