哈爾濱工業大學

HARBIN INSTITUTE OF TECHNOLOGY

模式识别与深度学习

实验报告

对抗式生成网络

黄海

1160300329

计算机科学与技术

指导教师 左旺孟

2019 年 6 月 13 日

目录

Chapter 1

最基本的 GAN

1.1 原理

对抗式生成网络，顾名思义，需要最基本的两个部件来进行生成，即生成器和判别器。在给定 TRUE DATA 的条件下，生成 FAKE DATA 提供给判别器，进行对抗。GAN 的难点在于调参，使两者中的任何一个都不会训练过头。

1.2 代码

以下是训练的部分代码

```
def GAN_TRAIN(SELF):
    PLT.ION()
    F, AX = PLT.SUBPLOTS(FIGSIZE=(8, 8))
    B_DATA = SELF.GET_DATA()
    X, Y = NP.MGRID[-1:2:COMPLEX(0, 100), -1:2:COMPLEX(0, 100)]
    print('START GENERATOR TRAINNING:')
    for STEP in RANGE(STEPS):
        print('THIS IS STEP:', STR(STEP + 1))
        GENERATOR_IDEAS = VARIABLE(TORCH.RANDN(8192, G_INPUT_SIZE))
        GENERATOR = SELF.G(GENERATOR_IDEAS)
```

```
GENERATOR_POINTS = GENERATOR.DETACH().NUMPY()
PROB_ART_FIRST = SELF.D(SELF.ART)
PROB_ART_SECOND = SELF.D(GENERATOR)
DISCRIMINATOR_LOSS = - TORCH.MEAN(TORCH.LOG(PROB_ART_FIRST)
                     + TORCH.LOG(1 - PROB_ART_SECOND))
GENERATOR_LOSS = TORCH.MEAN(TORCH.LOG(1 - PROB_ART_SECOND))
SELF.D_OPTIMIZER.ZERO_GRAD()
DISCRIMINATOR_LOSS.BACKWARD(RETAIN_GRAPH=TRUE)
SELF.D_OPTIMIZER.STEP()
SELF.G_OPTIMIZER.ZERO_GRAD()
GENERATOR_LOSS.BACKWARD(RETAIN_GRAPH=TRUE)
SELF.G_OPTIMIZER.STEP()
if STEP % 100 == 99:
    B_OUTPUT = SELF.D(B_DATA).DATA.NUMPY()
    B_OUTPUT = NP.RESHAPE(B_OUTPUT, (100, 100))
    ORIGINAL_DATA = PD.DATAFRAME(DATA().MAT_DATA,
                                 COLUMNS=['X', 'Y'])
    OUTPUT_DATA = PD.DATAFRAME(GENERATOR_POINTS, COLUMNS=['X', 'Y'])
    AX.SET_XLIM([-1, 2])
    AX.SET_YLIM([-1, 2])
    AX.SET_TITLE('GAN:THIS IS STEP:' +
                 STR(STEP + 1),
                 COLOR='RED', FONTWEIGHT=800)
    PCM = AX.PCOLOR(X, Y, B_OUTPUT, CMAP=S)
    BAR = F.COLORBAR(PCM, AX=AX)
    AX = SNS.SCATTERPLOT(X='X', Y='Y',
              DATA=ORIGINAL_DATA, MARKER='X', COLOR='B', S=30)
    AX = SNS.SCATTERPLOT(X='X', Y='Y',
              DATA=OUTPUT_DATA, MARKER='P', COLOR='R', ALPHA=0.5, S=30)
    F.SAVEFIG('./RESULTS/ADAM/GAN/STEP'
              + STR(STEP+1) + '.PNG')
    PLT.PAUSE(0.05)
    BAR.REMOVE()
```

```
PLT.IOFF()
F.SHOW()
```

1.3 实验结果

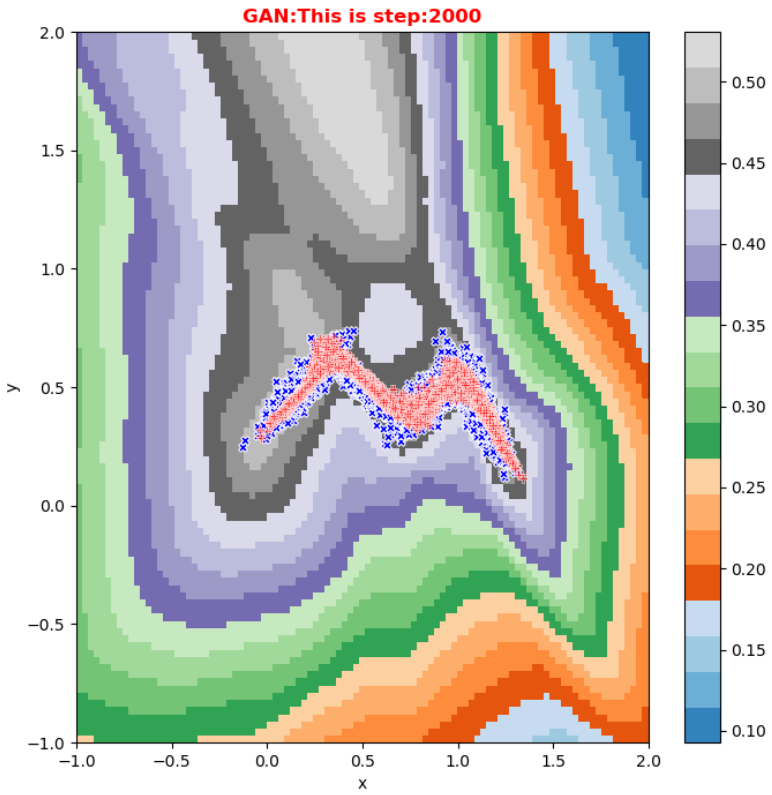

图 1.1: GAN 训练结果

Chapter 2

WGAN

2.1 解释

GAN 的难点也是自己的不足，两个部件很难训练到一个恰当的程度。这个问题产生的根本在于生成的 NOISE 和真正的 TRUE DATA 之间的联系权重过高，使得判别器有一丁点的改动生成器就会过度反应。解决的方法在于为参数进行减权，即在判别器更新之后，对参数赋予一个权重系数。

WGAN 个 GAN 的区别主要存在于以下四点

1. LOSS 函数不需要进行对数操作
2. 网络中判别器最后不需要 SIGMOD
3. 对参数赋予权重
4. 不使用基于动量的优化算法

2.2 代码

以下展示了于 GAN 不同的地方

```
# 没有 sigmod
class WGANDiscriminator(nn.Module):
    def __init__(self):
```

```
        SUPER(WGANDiscriminator, self).__init__()
        self.n1 = nn.Linear(D_INPUT_SIZE, D_HIDDEN_SIZE)
        self.n2 = nn.Linear(D_HIDDEN_SIZE, D_HIDDEN_SIZE)
        self.n3 = nn.Linear(D_HIDDEN_SIZE, D_RETURN_SIZE)
        self.relu = nn.ReLU()
        # self.sig = nn.Sigmoid()

    def forward(self, x):
        x = self.relu(self.n1(x))
        x = self.relu(self.n2(x))
        x = self.n3(x)
        return x

# loss 函数
discriminator_loss = - torch.mean(prob_art_first + 1 - prob_art_second)
generator_loss = torch.mean(1 - prob_art_second)

# 优化器
self.d_optimizer = optim.RMSprop(self.D.parameters(), lr=d_lr)
self.g_optimizer = optim.RMSprop(self.G.parameters(), lr=g_lr)

# 参数截断
for i in self.D.parameters():
    i.data.clamp_(-c, c)
```

2.3 结果

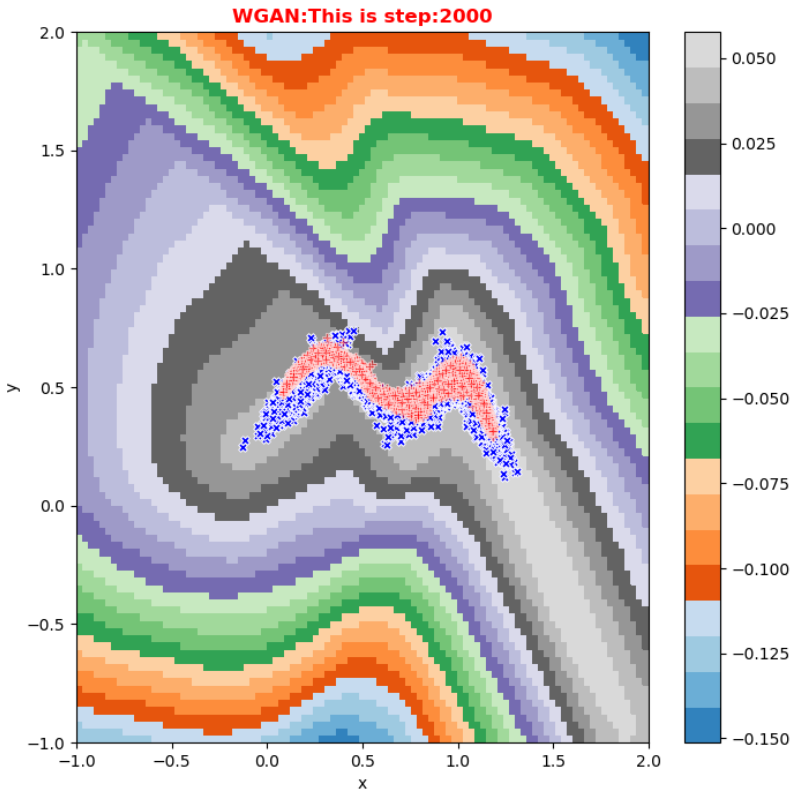

图 2.1: WGAN 训练结果

Chapter 3

WGAN-GP

3.1 解释

WGAN 解决了部分训练过度的问题，但是也暴露出其他的缺点，例如收敛过慢，在精细调参后的效果不如 GAN 等等。WGAN-GP 有解决了这类问题，对于 WGAN，只需要对梯度进行更改。这里需要有梯度惩罚措施，加快收敛速度但是不至于过快的下降。

3.2 代码

以下展示了于 WGAN 不同的地方

```
# 梯度惩罚
def CALC_GRADIENT_PENALTY(SELF, NETD, REAL_DATA, FAKE_DATA):
    # print real_data.size()
    ALPHA = TORCH.RAND(8192, 1)
    ALPHA = ALPHA.EXPAND(REAL_DATA.SIZE())
    INTERPOLATES = ALPHA * REAL_DATA + ((1 - ALPHA) * FAKE_DATA)
    INTERPOLATES = AUTOGRAD.VARIABLE(INTERPOLATES, REQUIRES_GRAD=TRUE)
    DISC_INTERPOLATES = NETD(INTERPOLATES)
    GRADIENTS = AUTOGRAD.GRAD(OUTPUTS=DISC_INTERPOLATES, INPUTS=INTERPOLATES,
              GRAD_OUTPUTS=TORCH.ONES(
```

```
                DISC_INTERPOLATES.SIZE()),
                CREATE_GRAPH=TRUE, RETAIN_GRAPH=TRUE, ONLY_INPUTS=TRUE)[0]
    GRADIENT_PENALTY = ((GRADIENTS.NORM(2, DIM=1) - 1) ** 2).MEAN() * LAMBDA
    return GRADIENT_PENALTY
```

3.3 结果

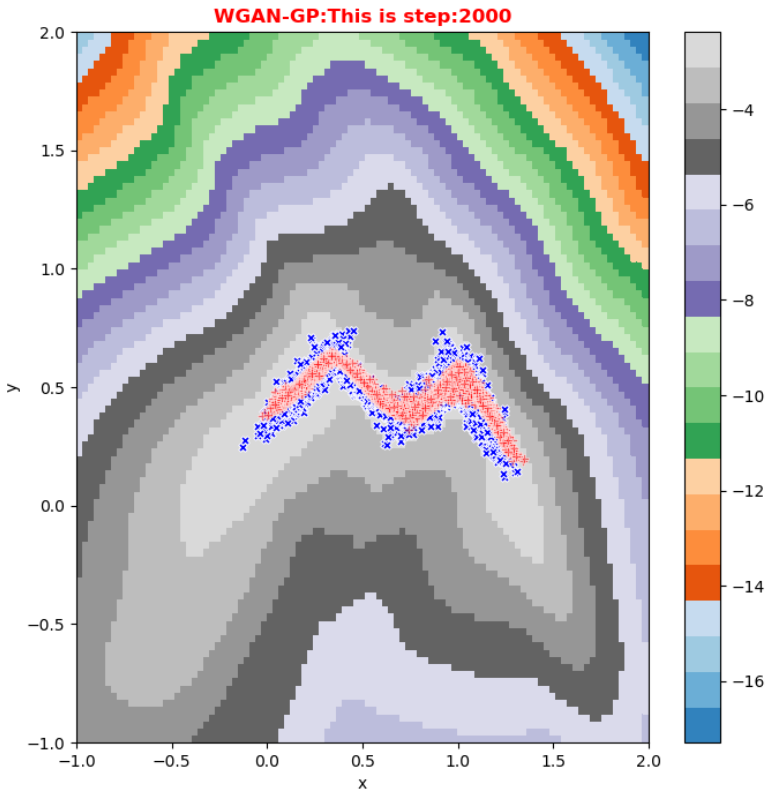

图 3.1: WGAN-GP 训练结果