



SeetaTech  
中 科 视 拓

# GAN的诞生

主讲人：黄焯恒

# 目录

- 一、GAN的介绍
- 二、代码实现
- 三、模块实现细节
- 四、实验结果分析
- 五、模型训练技巧

# 一、GAN的介绍

- 1. Generative Adversarial Networks ( GAN ) : 生成式对抗网络，在2014年提出的一种无监督深度学习模型。
- 2. GAN模型组成：生成模型G ( Generative Model ) 和判别器D ( Discriminative Model )
- 3. 网络结构：



# 一、GAN的介绍

- 4. 训练优化过程：“二元极小极大值博弈”

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

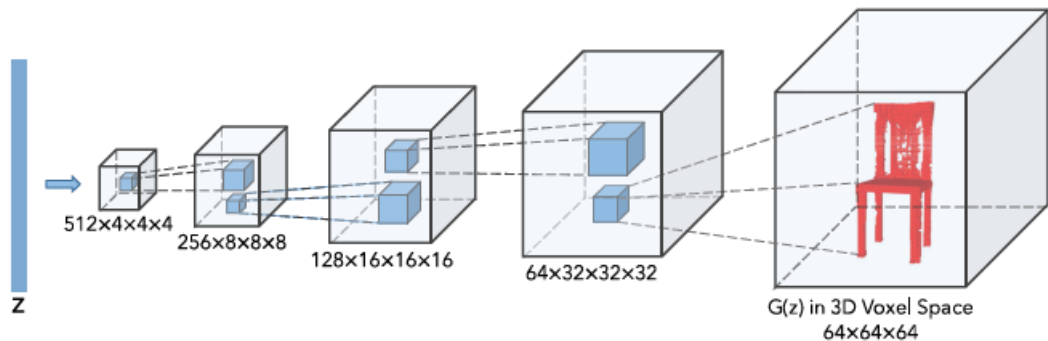
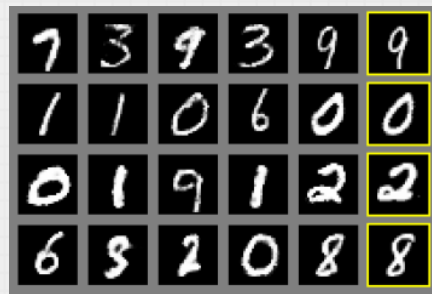
这是关于判别网络D和生成网络G的价值函数，其中：

- (1) 最大化  $\log D(\mathbf{x})$ ：训练网络D使得最大概率地分对训练样本的标签；
- (2) 最小化  $\log(1 - D(G(\mathbf{z})))$ ：训练网络G生成能“欺骗”网络D的数据，最大化D的损失。

# 一、GAN的介绍

## 5. 应用案例介绍：

- 计算机视觉：生成图像、风格迁移、生成模型等。
- 自然语言处理：对话系统，纯文本生成（如诗歌生成），机器翻译，IR，中文分词，文本分类（半监督学习）
- 常用的GAN
  - DCGAN
  - CGAN
  - InfoGAN
  - WGAN
  - VAEGAN



## 二、代码实现

- 1. 环境配置 ( GPU优先 )
  - Python 2.7.13
  - TensorFlow 1.3.0
  - pandas 0.19.2
  - dask 0.15.0
- 2. 数据集准备：MNIST手写数字图片
- 3. 实现目标：利用GAN生成手写数字

## 二、代码实现

- 5. 程序实现流程

(1) 导入相关的包：TensorFlow，numpy，matplotlab等；

(2) 读数据集数据：mnist = input\_data.read\_data\_sets( './mnist' , one\_hot=True)

(3) 定义输入变量和参数：

输入 - X：数据集数据（真）

参数 - w1，w2：权值

- Z：随机噪声（假）

- b1，b2：偏置

(4) 构建G和D模型；

(5) 往模型喂入数据和参数；

(6) 迭代计算，更新G和D的损失（Loss）；

(7) 生成手写数字图片，观察Loss收敛。

# 三、模块实现细节

## 1. 生成模型

```
#生成模型
def generator(z):
    G_h1 = tf.nn.relu(tf.matmul(z, G_W1) + G_b1)
    G_log_prob = tf.matmul(G_h1, G_W2) + G_b2
    G_prob = tf.nn.sigmoid(G_log_prob)

    return G_prob
```

## 2. 判别模型

```
#判别模型
def discriminator(x):
    D_h1 = tf.nn.relu(tf.matmul(x, D_W1) + D_b1)
    D_logit = tf.matmul(D_h1, D_W2) + D_b2
    D_prob = tf.nn.sigmoid(D_logit)

    return D_prob, D_logit
```



# 三、模块实现细节

## 3. 随机噪声采样函数

```
#随机噪声采样函数
def sample_z(m, n):
    return np.random.uniform(-1., 1., size=[m, n])
```

## 4. 参数设置

```
#生成模型的输入和参数初始化
Z = tf.placeholder(tf.float32, shape=[None, 100])

G_W1 = tf.Variable(xavier_init([100, 128]))
G_b1 = tf.Variable(tf.zeros(shape=[128]))

G_W2 = tf.Variable(xavier_init([128, 784]))
G_b2 = tf.Variable(tf.zeros(shape=[784]))

theta_G = [G_W1, G_W2, G_b1, G_b2]
```

```
#判别模型的输入和参数初始化
X = tf.placeholder(tf.float32, shape=[None, 784])

D_W1 = tf.Variable(xavier_init([784, 128]))
D_b1 = tf.Variable(tf.zeros(shape=[128]))

D_W2 = tf.Variable(xavier_init([128, 1]))
D_b2 = tf.Variable(tf.zeros(shape=[1]))

theta_D = [D_W1, D_W2, D_b1, D_b2]
```

# 三、模块实现细节

## 5. 训练网络过程

(1) 喂入数据

**#喂入数据**

```
G_sample = generator(Z)
D_real, D_logit_real = discriminator(X)
D_fake, D_logit_fake = discriminator(G_sample)
```

(2) 计算G 和 D 的损失 ( loss ) —— 交叉熵 ( 度量两个概率分布间的差异性信息 )

```
def sigmoid_cross_entropy_with_logits(_sentinel=None, labels=None, logits=None, name=None):
```

代码为例：

```
D_loss_real = tf.reduce_mean( tf.nn.sigmoid_cross_entropy_with_logits( logits = D_logit_real,
                                                                           targets = tf.ones_like(D_logit_real) ) )
```

## 三、模块实现细节

### (3) 实现了Adam算法的优化器

动态调整每个参数的学习率。Adam的优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳。

代码为例：

```
D_solver = tf.train.AdamOptimizer().minimize(D_loss, var_list=theta_D)
```

```
G_solver = tf.train.AdamOptimizer().minimize(G_loss, var_list=theta_G)
```

## 三、模块实现细节

### (4) 开始训练

```
for it in range(1000000):
    if it % 1000 == 0:
        samples = sess.run(G_sample, feed_dict={Z: sample_Z(16, Z_dim)})

        fig = plot(samples)
        plt.savefig('out/{}.png'.format(str(i).zfill(3)), bbox_inches='tight')
        i += 1
        plt.close(fig)

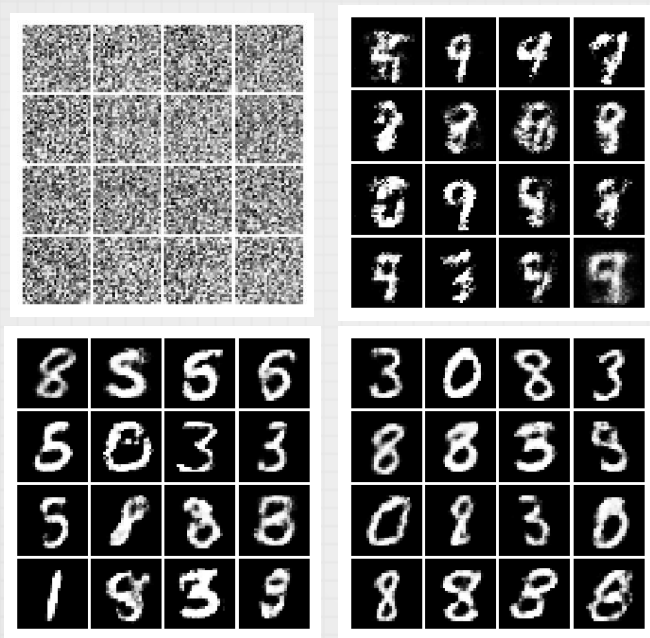
    X_mb, _ = mnist.train.next_batch(mb_size)

    _, D_loss_curr = sess.run([D_solver, D_loss], feed_dict={X: X_mb, Z: sample_Z(mb_size, Z_dim)})
    _, G_loss_curr = sess.run([G_solver, G_loss], feed_dict={Z: sample_Z(mb_size, Z_dim)})

    if it % 1000 == 0:
        print('Iter: {}'.format(it))
        print('D loss: {:.4}'.format(D_loss_curr))
        print('G_loss: {:.4}'.format(G_loss_curr))
```

# 四、实验结果分析

## 1. 生成图像和输出的Loss



```
Iter: 996000  
D_loss: 0.1958  
G_loss: 3.884  
( )  
Iter: 997000  
D_loss: 0.1026  
G_loss: 4.046  
( )  
Iter: 998000  
D_loss: 0.2147  
G_loss: 3.404  
( )  
Iter: 999000  
D_loss: 0.1724  
G_loss: 3.704  
( )
```

# 五、模型训练技巧

- 1. 训练判别模型D时，可以将真实数据和生成数据一起喂入吗？

不能。训练判别模型D时，真实数据和生成数据需要分开喂入，因为判别网络需要分别学习真实数据和生成数据的特征，而当判别模型训练好之后，在测试集中用学习到的特征来判断数据真假。

- 2. 输入标准化

在-1和1之间对数据（图像）进行归一化处理

- 3. 损失函数的修改

在GAN论文（2014）中，优化G的损失函数是 $\min(\log 1-D)$ ，但实际上人们实际上使用 $\max \log D$ 。

# 实验材料

1. 论文 : Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.[NIPS2014]
2. 数据集mnist
3. 源码 : gan.py
4. 输出结果 : out文件夹下的图片

Thank you.