GAN 예제 및 실습

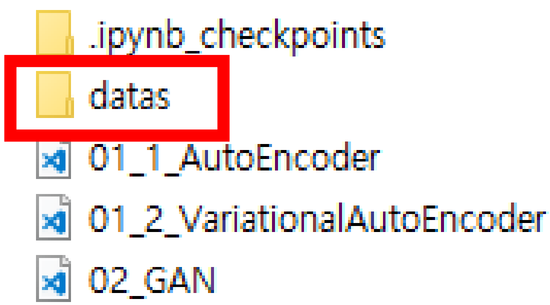
수업목표

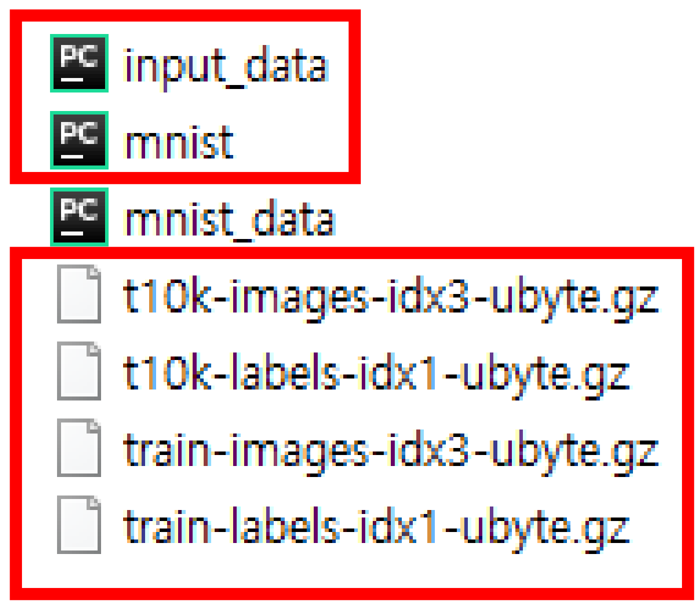
- MNIST dataset
- GAN

MNIST dataset

MNIST dataset

1. Import MNIST data set(local)





MNIST dataset

2. Extract

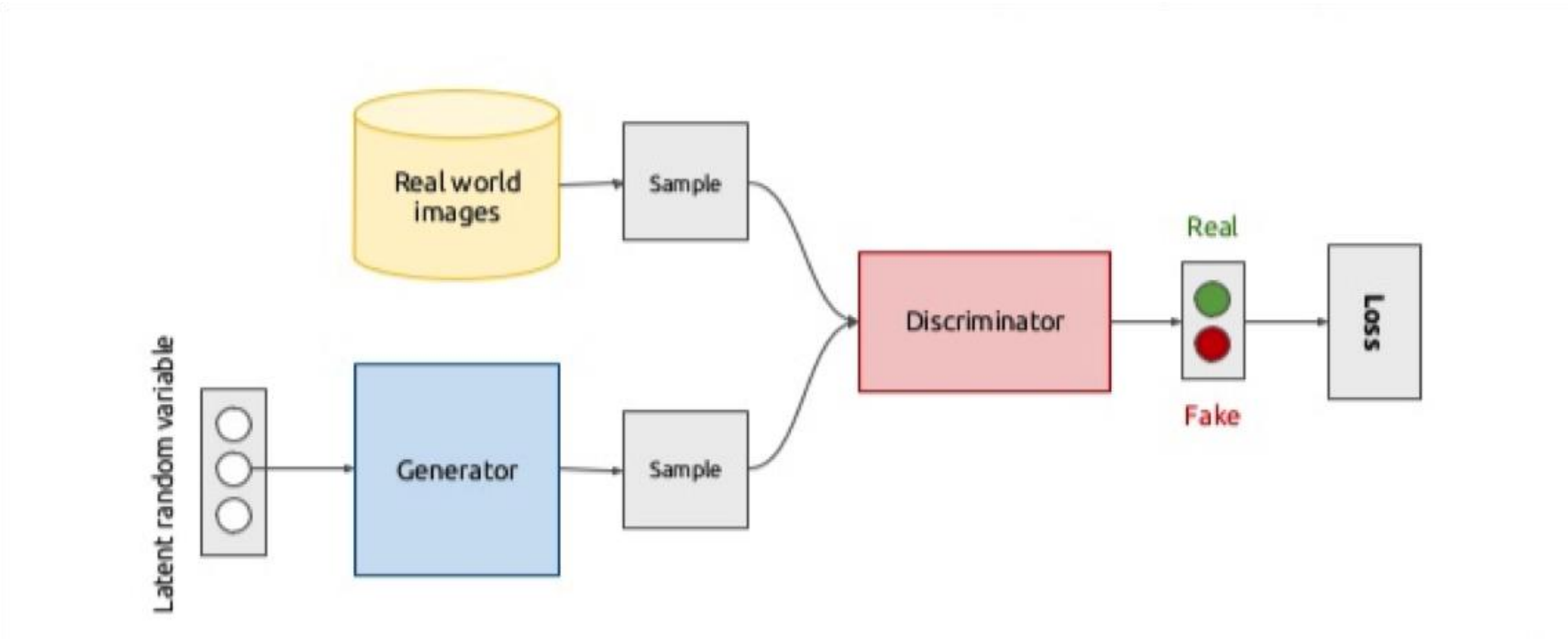
- from data import input_data
- mnist = input_data.read_data_sets("./data/", one_hot=True)

```
Extracting ./data/train-images-idx3-ubyte.gz
Extracting ./data/train-labels-idx1-ubyte.gz
Extracting ./data/t10k-images-idx3-ubyte.gz
Extracting ./data/t10k-labels-idx1-ubyte.gz
```

GAN

GAN

1. Overview



GAN

2. Import library

- import matplotlib.pyplot as plt
- import numpy as np
- import tensorflow as tf

GAN

3. Set parameters(training)

- learning_rate = 0.0002 # learning late
- num_steps = 70000 # epoch
- batch_size = 128 # batch size

- display_step = 2000 # display step(unit)

GAN

3. Set parameters(network)

- gen_hidden_dim = 256 # learning late
- disc_hidden_dim = 256 # epoch
- num_input = 784 # MNIST 28*28
- noise_dim = 100 # display step(unit)

GAN

4. 변수 초기화

- Xavier-Glorot initializer 사용
- weight, bias 초기화

```
# 변수 초기화 (see Xavier Glorot init)
def glorot_init(shape):
    return tf.random_normal(shape=shape, stddev=1. / tf.sqrt(shape[0] / 2.))
```

GAN

5. Generator, discriminator 정의

```
# Generator
def generator(x):
    hidden_layer = tf.matmul(x, weights['gen_hidden1'])
    hidden_layer = tf.add(hidden_layer, biases['gen_hidden1'])
    hidden_layer = tf.nn.relu(hidden_layer)
    out_layer = tf.matmul(hidden_layer, weights['gen_out'])
    out_layer = tf.add(out_layer, biases['gen_out'])
    out_layer = tf.nn.sigmoid(out_layer)
    return out_layer
```

```
# Discriminator
def discriminator(x):
    hidden_layer = tf.matmul(x, weights['disc_hidden1'])
    hidden_layer = tf.add(hidden_layer, biases['disc_hidden1'])
    hidden_layer = tf.nn.relu(hidden_layer)
    out_layer = tf.matmul(hidden_layer, weights['disc_out'])
    out_layer = tf.add(out_layer, biases['disc_out'])
    out_layer = tf.nn.sigmoid(out_layer)
    return out_layer
```

GAN

6. Build Networks - inputs

- gen_input, disc_input 설정(placeholder)

```
# Build Networks
# Network Inputs
gen_input = tf.placeholder(tf.float32, shape=[None, noise_dim], name='input_noise')
disc_input = tf.placeholder(tf.float32, shape=[None, num_input], name='disc_input')
```

GAN

6. Build Networks – Generator / Discriminator networks

```
# Generator Network
gen_sample = generator(gen_input)

# 두 개의 Discriminator Network 생성 (one from noise input, one from generated samples)
disc_real = discriminator(disc_input)
disc_fake = discriminator(gen_sample)
```

GAN

7. Loss function

```
# Loss Function
gen_loss = -tf.reduce_mean(tf.log(disc_fake))
disc_loss = -tf.reduce_mean(tf.log(disc_real) + tf.log(1. - disc_fake))

# Optimizer
optimizer_gen = tf.train.AdamOptimizer(learning_rate=learning_rate)
optimizer_disc = tf.train.AdamOptimizer(learning_rate=learning_rate)
```

GAN

8. Optimizer

```
# 각각의 optimizer 의 훈련 변수
# Generator Network 변수
gen_vars = [weights['gen_hidden1'], weights['gen_out'],
            biases['gen_hidden1'], biases['gen_out']]
# Discriminator Network Variables
disc_vars = [weights['disc_hidden1'], weights['disc_out'],
            biases['disc_hidden1'], biases['disc_out']]

# 최적화
train_gen = optimizer_gen.minimize(gen_loss, var_list=gen_vars)
train_disc = optimizer_disc.minimize(disc_loss, var_list=disc_vars)

# 변수 초기화
init = tf.global_variables_initializer()
```

GAN

9. Training

```
# TF session 시작
sess = tf.Session()

# initializer 실행
sess.run(init)

# 훈련 시작
for i in range(1, num_steps+1):
    batch_x, _ = mnist.train.next_batch(batch_size)

    # Generator 에 feed 하기 위한 노이즈 생성
    z = np.random.uniform(-1., 1., size=[batch_size, noise_dim])

    # 훈련
    feed_dict = {disc_input: batch_x, gen_input: z}
    _, _, gl, dl = sess.run([train_gen, train_disc, gen_loss, disc_loss],
                            feed_dict=feed_dict)
    if i % 2000 == 0 or i == 1:
        print('Step %i: Generator Loss: %f, Discriminator Loss: %f' % (i, gl, dl))
```

GAN

9. Training - Result

```
Step 1: Generator Loss: 0.449949, Discriminator Loss: 1.889206
Step 2000: Generator Loss: 4.648057, Discriminator Loss: 0.052475
Step 4000: Generator Loss: 3.667910, Discriminator Loss: 0.086267
Step 6000: Generator Loss: 4.074244, Discriminator Loss: 0.113183
Step 8000: Generator Loss: 3.833836, Discriminator Loss: 0.166927
Step 10000: Generator Loss: 3.207469, Discriminator Loss: 0.189841
Step 12000: Generator Loss: 4.008196, Discriminator Loss: 0.180366
Step 14000: Generator Loss: 3.992439, Discriminator Loss: 0.251904
Step 16000: Generator Loss: 3.327659, Discriminator Loss: 0.346477
Step 18000: Generator Loss: 4.003550, Discriminator Loss: 0.265334
Step 20000: Generator Loss: 3.574762, Discriminator Loss: 0.345076
Step 22000: Generator Loss: 4.155303, Discriminator Loss: 0.134706
Step 24000: Generator Loss: 3.558898, Discriminator Loss: 0.307450
Step 26000: Generator Loss: 3.405217, Discriminator Loss: 0.324923
Step 28000: Generator Loss: 3.964176, Discriminator Loss: 0.258023
Step 30000: Generator Loss: 3.134517, Discriminator Loss: 0.275423
Step 32000: Generator Loss: 2.855176, Discriminator Loss: 0.294660
Step 34000: Generator Loss: 3.167559, Discriminator Loss: 0.409081
Step 36000: Generator Loss: 3.188723, Discriminator Loss: 0.343197
Step 38000: Generator Loss: 3.468094, Discriminator Loss: 0.364154
Step 40000: Generator Loss: 2.867365, Discriminator Loss: 0.406178
Step 42000: Generator Loss: 2.837689, Discriminator Loss: 0.463374
Step 44000: Generator Loss: 3.051821, Discriminator Loss: 0.412852
Step 46000: Generator Loss: 3.280366, Discriminator Loss: 0.399309
Step 48000: Generator Loss: 3.142927, Discriminator Loss: 0.345826
Step 50000: Generator Loss: 2.790816, Discriminator Loss: 0.445501
Step 52000: Generator Loss: 2.788424, Discriminator Loss: 0.571878
Step 54000: Generator Loss: 2.504256, Discriminator Loss: 0.512286
Step 56000: Generator Loss: 2.881957, Discriminator Loss: 0.388795
Step 58000: Generator Loss: 2.686737, Discriminator Loss: 0.440981
Step 60000: Generator Loss: 2.650364, Discriminator Loss: 0.442958
Step 62000: Generator Loss: 3.026576, Discriminator Loss: 0.431689
Step 64000: Generator Loss: 2.889766, Discriminator Loss: 0.419922
Step 66000: Generator Loss: 3.166866, Discriminator Loss: 0.380643
Step 68000: Generator Loss: 3.080680, Discriminator Loss: 0.374044
Step 70000: Generator Loss: 2.872635, Discriminator Loss: 0.551213
```

GAN

10. Test

```
# 테스트 시작
# 노이즈로부터 이미지를 생성
n = 6
canvas = np.empty((28 * n, 28 * n))
for i in range(n):
    # 노이즈 입력
    z = np.random.uniform(-1., 1., size=[n, noise_dim])

    # 이미지 생성
    g = sess.run(gen_sample, feed_dict={gen_input: z})

    g = -1 * (g - 1)
    for j in range(n):
        canvas[i * 28:(i + 1) * 28, j * 28:(j + 1) * 28] = g[j].reshape([28, 28])

plt.figure(figsize=(n, n))
plt.imshow(canvas, origin="upper", cmap="gray")
plt.show()
```

GAN

10. Test - Result