Variational Autoencoder

예제 및 실습

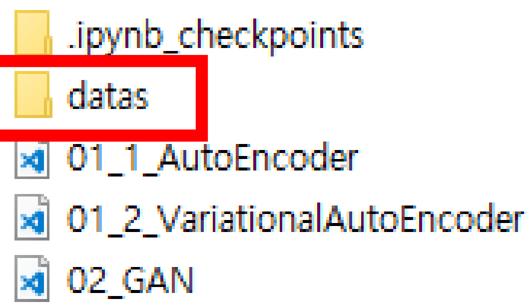
수업목표

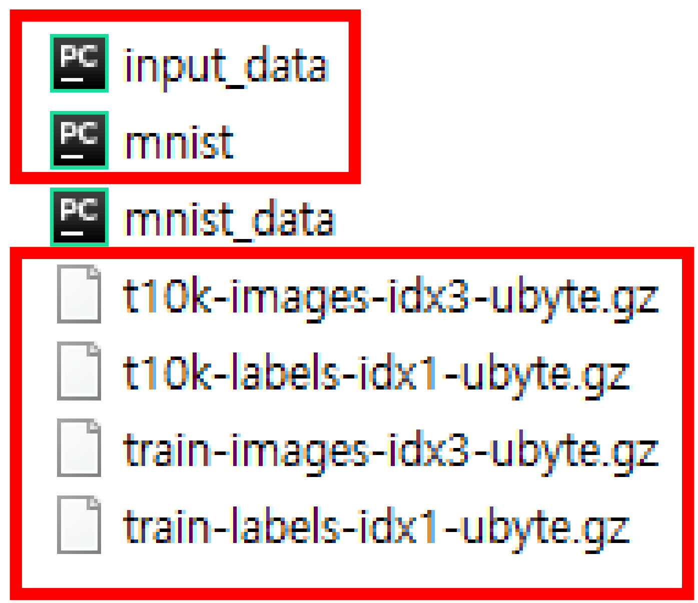
- MNIST dataset
- Variational Autoencoder

MNIST dataset

MNIST dataset

1. Import MNIST data set(local)





MNIST dataset

2. Extract

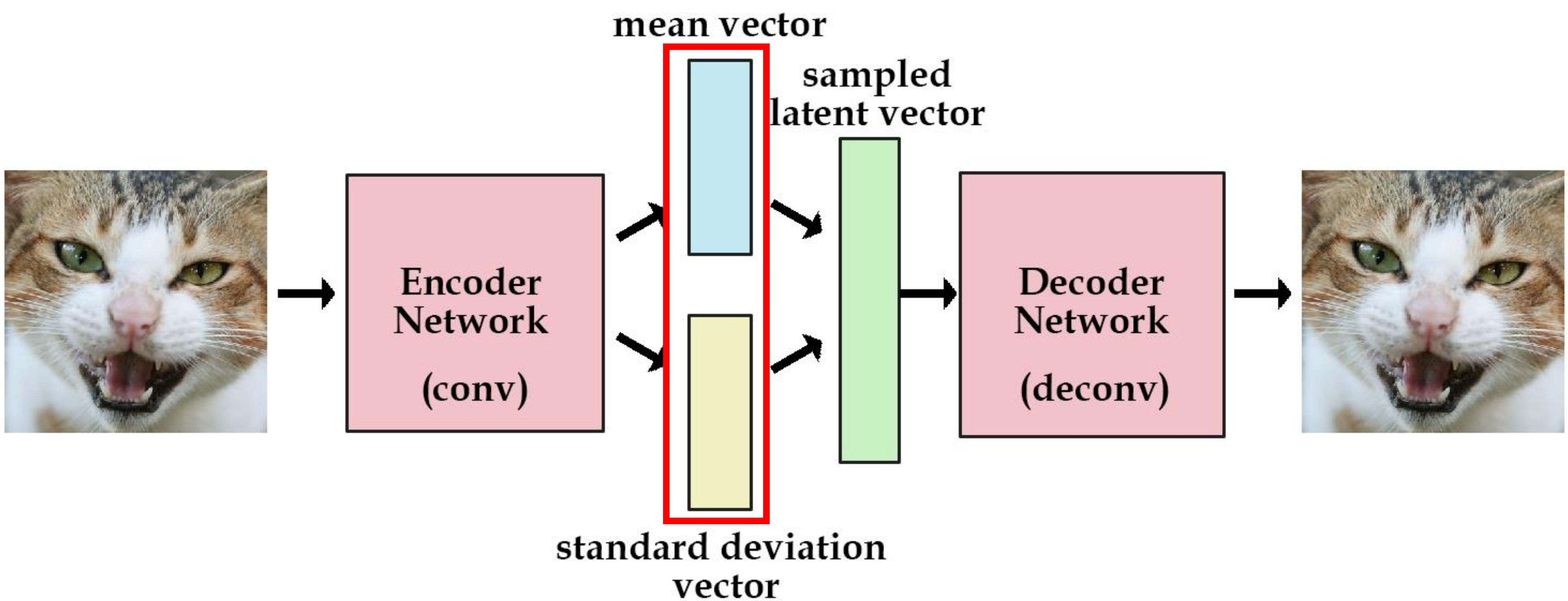
- from data import input_data
- mnist = input_data.read_data_sets("./data/", one_hot=True)

```
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
```

Variational Autoencoder

Variational Autoencoder

1. Overview



Variational Autoencoder

2. Import library

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

Variational Autoencoder

3. Set parameters(training)

- learning_rate = 0.01 # learning late
- num_steps = 30000 # epoch
- batch_size = 256 # batch size

- display_step = 1000 # display step(unit)

Variational Autoencoder

4. Set parameters(network)

- num_hidden_1 = 256 # 첫번째 hidden layer
- num_hidden_2 = 128 # 두번째 hidden layer
- num_input = 784 # MNIST 28*28

Variational Autoencoder

5. 변수 선언

- weight, bias, z_mean, z_std 초기화

```
# 모델의 wright와 bias의 배열값
# Variables
weights = {
    'encoder_h1': tf.Variable(glorot_init([num_input, num_hidden_1])),
    'z_mean': tf.Variable(glorot_init([num_hidden_1, num_hidden_2])),
    'z_std': tf.Variable(glorot_init([num_hidden_1, num_hidden_2])),
    'decoder_h1': tf.Variable(glorot_init([num_hidden_2, num_hidden_1])),
    'decoder_out': tf.Variable(glorot_init([num_hidden_1, num_input]))
}
biases = {
    'encoder_b1': tf.Variable(glorot_init([num_hidden_1])),
    'z_mean': tf.Variable(glorot_init([num_hidden_2])),
    'z_std': tf.Variable(glorot_init([num_hidden_2])),
    'decoder_b1': tf.Variable(glorot_init([num_hidden_1])),
    'decoder_out': tf.Variable(glorot_init([num_input]))
}
```

Variational Autoencoder

6. Encoder, decoder 정의

```python
# 디코더 설정
def decode_func(z) :
    de1 = tf.matmul(z, weights['decoder_h1']) + biases['decoder_b1']
    de1 = tf.nn.tanh(de1)
    de2 = tf.matmul(de1, weights['decoder_out']) + biases['decoder_out']
    recon = tf.nn.sigmoid(de2)
    return recon
```

```python
# VAE Loss function 정의
def vae_loss(x_reconstructed, x_true):
    # Reconstruction loss
    encode_decode_loss = x_true * tf.log(1e-10 + x_reconstructed) \
                         + (1 - x_true) * tf.log(1e-10 + 1 - x_reconstructed)
    encode_decode_loss = -tf.reduce_sum(encode_decode_loss, 1)
    # KL Divergence loss
    kl_div_loss = 1 + z_std - tf.square(z_mean) - tf.exp(z_std)
    kl_div_loss = -0.5 * tf.reduce_sum(kl_div_loss, 1)
    return tf.reduce_mean(encode_decode_loss + kl_div_loss)
```

Variational Autoencoder

7. Build model

```
# 모델 생성
encode_op = encode_func(X)
decode_op = decode_func(encode_op)
```

Variational GAN

8. Loss function / Optimizer

```
# Loss Function 및 optimizer 설정
loss_op = vae_loss(decode_op, X)
optimizer = tf.train.RMSPropOptimizer(learning_rate=learning_rate)
train_op = optimizer.minimize(loss_op)
```

Variational Autoencoder

9. 전체 변수 초기화

- Global variables initializer 사용

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

Variational Autoencoder

9. Training

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

# initializer 실행
sess.run(init)

# 학습 시작
# 학습횟수(epoch = num_steps = 30000)
for epoch in range(1, num_steps+1):
    # batch_size 만큼 다음 mini batch를 가져옴
    X_images, _ = mnist.train.next_batch(batch_size)

    # 로그
    _, l = sess.run([train_op, loss_op], feed_dict={X: X_images})

    # Display logs per step
    if epoch % display_step == 0 or epoch == 1:
        print('epoch %i: Minibatch Loss: %f' % (epoch, l))

print("학습완료! (loss : " + str(l) + ")")
```

Variational Autoencoder

9. Training - Result

```
epoch 1: Minibatch Loss: 626.888855
epoch 1000: Minibatch Loss: 128.563766
epoch 2000: Minibatch Loss: 123.270302
epoch 3000: Minibatch Loss: 119.959656
epoch 4000: Minibatch Loss: 127.980499
epoch 5000: Minibatch Loss: 117.810875
epoch 6000: Minibatch Loss: 117.190178
epoch 7000: Minibatch Loss: 119.162460
epoch 8000: Minibatch Loss: 112.696289
epoch 9000: Minibatch Loss: 116.761063
epoch 10000: Minibatch Loss: 111.186523
epoch 11000: Minibatch Loss: 111.680763
epoch 12000: Minibatch Loss: 111.398865
epoch 13000: Minibatch Loss: 113.417824
epoch 14000: Minibatch Loss: 111.933083
epoch 15000: Minibatch Loss: 111.075142
epoch 16000: Minibatch Loss: 107.956711
epoch 17000: Minibatch Loss: 106.295944
epoch 18000: Minibatch Loss: 107.166389
epoch 19000: Minibatch Loss: 109.261871
epoch 20000: Minibatch Loss: 105.386642
epoch 21000: Minibatch Loss: 104.854485
epoch 22000: Minibatch Loss: 111.801895
epoch 23000: Minibatch Loss: 112.239304
epoch 24000: Minibatch Loss: 106.498993
epoch 25000: Minibatch Loss: 102.629196
epoch 26000: Minibatch Loss: 110.207878
epoch 27000: Minibatch Loss: 112.944283
epoch 28000: Minibatch Loss: 108.290062
epoch 29000: Minibatch Loss: 108.509995
epoch 30000: Minibatch Loss: 104.559189
학습완료! (loss : 104.55919)
```

Variational Autoencoder

10. Test

```python
# 테스트 시작

# Generator takes noise as input
noise_input = tf.placeholder(tf.float32, shape=[None, num_hidden_2])
# Rebuild the decoder to create image from noise
decoder = tf.matmul(noise_input, weights['decoder_h1']) + biases['decoder_b1']
decoder = tf.nn.tanh(decoder)
decoder = tf.matmul(decoder, weights['decoder_out']) + biases['decoder_out']
decoder = tf.nn.sigmoid(decoder)

n = 4
canvas_orig = np.empty((28 * n, 28 * n))
canvas_recon = np.empty((28 * n, 28 * n))

for i in range(n):
    # MNIST test set
    test_X, _ = mnist.train.next_batch(batch_size)

    g = sess.run(decode_op, feed_dict={X: test_X})

    # 원본 이미지를 가져와서 출력
    for j in range(n):
        canvas_orig[i * 28:(i + 1) * 28, j * 28:(j + 1) * 28] = test_X[j].reshape([28, 28])
    # 재생성된 이미지를 가져와서 출력
    for j in range(n):
        # Draw the generated digits
        canvas_recon[i * 28:(i + 1) * 28, j * 28:(j + 1) * 28] = g[j].reshape([28, 28])

# 테스트 결과 출력
print("Original Images")
plt.figure(figsize=(n, n))
plt.imshow(canvas_orig, origin="upper", cmap="gray")
plt.show()

print("Reconstructed Images")
plt.figure(figsize=(n, n))
plt.imshow(canvas_recon, origin="upper", cmap="gray")
plt.show()
```

Variational Autoencoder

10. Test - Result