GAN의 구현

In [2]:
```
### 2016년 관심 분야 비지도학습(Unsupervised) 방법.
```

In [1]:
```python
# https://arxiv.org/abs/1406.2661
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("./mnist/data/", one_hot=True)

```

```
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩h5py₩__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype fr
om `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters

WARNING:tensorflow:From <ipython-input-1-4a88a2be3f8a>:7: read_data_sets (from tensorflow.contrib.learn.python.learn.datasets.mnis
t) is deprecated and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflow/models.
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:260: m
aybe_download (from tensorflow.contrib.learn.python.learn.datasets.base) is deprecated and will be removed in a future version.
Instructions for updating:
Please write your own downloading logic.
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩base.py:252: _i
nternal_retry.<locals>.wrap.<locals>.wrapped_fn (from tensorflow.contrib.learn.python.learn.datasets.base) is deprecated and will
be removed in a future version.
Instructions for updating:
Please use urllib or similar directly.
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:262: e
xtract_images (from tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting ./mnist/data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:267: e
xtract_labels (from tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting ./mnist/data/train-labels-idx1-ubyte.gz
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:110: d
ense_to_one_hot (from tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a future version.
Instructions for updating:
```

```
Please use tf.one_hot on tensors.
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting ./mnist/data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting ./mnist/data/t10k-labels-idx1-ubyte.gz
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:290: DataSet.__init__ (from tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflow/models.
```

01. 기본 옵션 설정

In [3]:

```python
total_epoch = 100   # epoch 수 설정
batch_size = 100    # 배치 사이즈
learning_rate = 0.0002  # 학습률
# 신경망 레이어 구성 옵션
n_hidden = 256          # 은닉층 노드
n_input = 28 * 28       # 입력
n_noise = 128  # 생성기의 입력값으로 사용할 노이즈의 크기

```

02. 신경망 모델 구성

- 노이즈를 이용하여 데이터 생성
- 비지도학습이므로 Y가 없음

In [4]:

```python
# GAN 도 Unsupervised 학습이므로 Autoencoder 처럼 Y 를 사용하지 않습니다.
X = tf.placeholder(tf.float32, [None, n_input])

# 노이즈 Z를 입력값으로 사용합니다.
Z = tf.placeholder(tf.float32, [None, n_noise])
```

생성자 신경망, 판별자 신경망 변수 선언

In [5]:
```
G_W1 = tf.Variable(tf.random_normal([n_noise, n_hidden], stddev=0.01))
G_b1 = tf.Variable(tf.zeros([n_hidden]))
G_W2 = tf.Variable(tf.random_normal([n_hidden, n_input], stddev=0.01))
G_b2 = tf.Variable(tf.zeros([n_input]))
```

In [6]:
```
# 판별기 신경망에 사용하는 변수들입니다.
D_W1 = tf.Variable(tf.random_normal([n_input, n_hidden], stddev=0.01))
D_b1 = tf.Variable(tf.zeros([n_hidden]))
# 판별기의 최종 결과값은 얼마나 진짜와 가깝냐를 판단하는 한 개의 스칼라값입니다.
D_W2 = tf.Variable(tf.random_normal([n_hidden, 1], stddev=0.01))
D_b2 = tf.Variable(tf.zeros([1]))
```

2-1 생성자(D) 신경망 구성

- 무작위 생성한 노이즈를 받아, 가중치와 편향을 반영하여 은닉층 구성.
- sigmoid 함수를 이용하여 최종 결과값 0~1 사이의 값 반환

In [7]:
```
def generator(noise_z):
    hidden = tf.nn.relu(
                    tf.matmul(noise_z, G_W1) + G_b1)
    output = tf.nn.sigmoid(
                    tf.matmul(hidden, G_W2) + G_b2)

    return output
```

2-2 구분자(D) 신경망 구성

- 구분자 신경망 구성, 가중치와 편향을 반영한 데이터 출력
- sigmoid 함수를 이용하여 최종 결과값 0~1 사이의 값 반환

In [9]:
```
def discriminator(inputs):
    hidden = tf.nn.relu(
                    tf.matmul(inputs, D_W1) + D_b1)
    output = tf.nn.sigmoid(
                    tf.matmul(hidden, D_W2) + D_b2)

    return output
```

2-3 위의 노이즈 발생을 위한 노이즈 생성 함수

In [10]:
```
# 랜덤한 노이즈(Z)를 만듭니다.
def get_noise(batch_size, n_noise):
    return np.random.normal(size=(batch_size, n_noise))
```

In [13]:
```
# 노이즈를 이용해 랜덤한 이미지를 생성합니다.
# Z에는 실행 시, noise가 입력됨.
G = generator(Z)
# 노이즈를 이용해 생성한 이미지가 진짜 이미지인지 판별한 값을 구합니다.
D_fake = discriminator(G)
# 진짜 이미지를 이용해 판별한 값을 구합니다.
D_real = discriminator(X)

```

- GAN은 생성자(Generator) : 구분자가 1로 예측하도록 하는 것을 목표로 학습시킴.
- GAN은 구분자(Discriminator) : 진짜 데이터를 받으면 1로 가짜 데이터를 받으면 0으로 예측하도록 학습시킴.

GAN의 모델의 최적화

- loss_G와 loss_D를 최대화 하는 것. 단, 서로의 손실이 연관되어 있어, 두 손실값이 같이 증가가 어려움.
- loss_D를 최대화하기 위해서는 D_gene값을 최소화시킴.
- 판별기에 진짜 이미지를 넣었을 때에도 최대값을 : tf.log(D_real)
- 가짜 이미지를 넣었을 때에도 최대값을 : tf.log(1 - D_gene)

In [12]:
```
loss_D = tf.reduce_mean(tf.log(D_real) + tf.log(1 - D_fake))
```

- loss_G(생성자 손실)를 최대화하기 위해서는 D_gene값을 최대화 한다.
- 가짜 이미지를 넣었을 때, 판별기가 최대한 실제 이미지라고 판단하도록 생성기 신경망을 학습

In [14]:
```
# 결국 D_gene 값을 최대화하는 것이므로 다음과 같이 사용할 수 있습니다.
loss_G = tf.reduce_mean(tf.log(D_gene))
```

In [15]:
```
# loss_D 를 구할 때는 판별기 신경망에 사용되는 변수만 사용하고,
# loss_G 를 구할 때는 생성기 신경망에 사용되는 변수만 사용하여 최적화를 합니다.
D_var_list = [D_W1, D_b1, D_W2, D_b2]
G_var_list = [G_W1, G_b1, G_W2, G_b2]
```

In [16]:
```
# GAN 논문의 수식에 따르면 loss 를 극대화 해야하지만, minimize 하는 최적화 함수를 사용하기 때문에
# 최적화 하려는 loss_D 와 loss_G 에 음수 부호를 붙여줍니다.
train_D = tf.train.AdamOptimizer(learning_rate).minimize(-loss_D,
                                                         var_list=D_var_list)
train_G = tf.train.AdamOptimizer(learning_rate).minimize(-loss_G,
                                                         var_list=G_var_list)
```

03. 모델 학습

In [17]:
```
sess = tf.Session()
sess.run(tf.global_variables_initializer())

total_batch = int(mnist.train.num_examples/batch_size)
loss_val_D, loss_val_G = 0, 0
```

In [19]:

```
%%time

for epoch in range(total_epoch):
    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        noise = get_noise(batch_size, n_noise)

        # 판별기와 생성기 신경망을 각각 학습시킵니다.
        _, loss_val_D = sess.run([train_D, loss_D],
                                 feed_dict={X: batch_xs, Z: noise})
        _, loss_val_G = sess.run([train_G, loss_G],
                                 feed_dict={Z: noise})

    print('Epoch:', '%04d' % epoch,
          'D loss: {:.4}'.format(loss_val_D),
          'G loss: {:.4}'.format(loss_val_G))

    #########
    # 학습이 되어가는 모습을 보기 위해 주기적으로 이미지를 생성하여 저장
    ######
    if epoch == 0 or (epoch + 1) % 10 == 0:
        sample_size = 10
        noise = get_noise(sample_size, n_noise)
        samples = sess.run(G, feed_dict={Z: noise})

        fig, ax = plt.subplots(1, sample_size, figsize=(sample_size, 1))

        for i in range(sample_size):
            ax[i].set_axis_off()
            ax[i].imshow(np.reshape(samples[i], (28, 28)))

        plt.savefig('samples/{}.png'.format(str(epoch).zfill(3)), bbox_inches='tight')
        plt.close(fig)

print('최적화 완료!')
```

```
Epoch: 0000 D loss: -0.6861 G loss: -2.217
Epoch: 0001 D loss: -0.7462 G loss: -1.865
Epoch: 0002 D loss: -0.7107 G loss: -2.114
Epoch: 0003 D loss: -0.7527 G loss: -2.307
```

```
Epoch: 0004 D loss: -0.6416 G loss: -2.131
Epoch: 0005 D loss: -0.7226 G loss: -1.945
Epoch: 0006 D loss: -0.746 G loss: -2.024
Epoch: 0007 D loss: -0.6154 G loss: -1.935
Epoch: 0008 D loss: -0.6017 G loss: -2.067
Epoch: 0009 D loss: -0.6282 G loss: -1.834
Epoch: 0010 D loss: -0.5948 G loss: -2.301
Epoch: 0011 D loss: -0.5937 G loss: -2.055
Epoch: 0012 D loss: -0.6531 G loss: -2.498
Epoch: 0013 D loss: -0.7168 G loss: -2.18
Epoch: 0014 D loss: -0.6469 G loss: -1.945
Epoch: 0015 D loss: -0.6318 G loss: -2.035
Epoch: 0016 D loss: -0.6361 G loss: -2.106
Epoch: 0017 D loss: -0.5714 G loss: -2.345
Epoch: 0018 D loss: -0.5852 G loss: -2.074
Epoch: 0019 D loss: -0.5421 G loss: -2.169
Epoch: 0020 D loss: -0.6326 G loss: -2.139
Epoch: 0021 D loss: -0.6079 G loss: -2.274
Epoch: 0022 D loss: -0.5379 G loss: -2.266
Epoch: 0023 D loss: -0.6803 G loss: -2.079
Epoch: 0024 D loss: -0.455 G loss: -2.504
Epoch: 0025 D loss: -0.4151 G loss: -2.34
Epoch: 0026 D loss: -0.5889 G loss: -2.282
Epoch: 0027 D loss: -0.599 G loss: -2.117
Epoch: 0028 D loss: -0.5311 G loss: -2.229
Epoch: 0029 D loss: -0.5944 G loss: -2.399
Epoch: 0030 D loss: -0.5939 G loss: -2.431
Epoch: 0031 D loss: -0.5153 G loss: -2.42
Epoch: 0032 D loss: -0.5849 G loss: -2.569
Epoch: 0033 D loss: -0.5698 G loss: -2.468
Epoch: 0034 D loss: -0.6851 G loss: -2.465
Epoch: 0035 D loss: -0.5331 G loss: -2.357
Epoch: 0036 D loss: -0.6993 G loss: -2.183
Epoch: 0037 D loss: -0.4732 G loss: -2.478
Epoch: 0038 D loss: -0.6145 G loss: -2.386
Epoch: 0039 D loss: -0.6042 G loss: -2.155
Epoch: 0040 D loss: -0.6145 G loss: -2.209
Epoch: 0041 D loss: -0.5462 G loss: -2.418
Epoch: 0042 D loss: -0.5685 G loss: -2.724
Epoch: 0043 D loss: -0.5667 G loss: -2.322
Epoch: 0044 D loss: -0.4701 G loss: -2.653
Epoch: 0045 D loss: -0.5048 G loss: -2.377
```

```
Epoch: 0046 D loss: -0.552 G loss: -2.668
Epoch: 0047 D loss: -0.4565 G loss: -2.328
Epoch: 0048 D loss: -0.5723 G loss: -2.367
Epoch: 0049 D loss: -0.6002 G loss: -1.859
Epoch: 0050 D loss: -0.4801 G loss: -2.711
Epoch: 0051 D loss: -0.4849 G loss: -2.619
Epoch: 0052 D loss: -0.5439 G loss: -2.394
Epoch: 0053 D loss: -0.4535 G loss: -2.766
Epoch: 0054 D loss: -0.5805 G loss: -2.332
Epoch: 0055 D loss: -0.5657 G loss: -2.638
Epoch: 0056 D loss: -0.6457 G loss: -2.311
Epoch: 0057 D loss: -0.5186 G loss: -2.578
Epoch: 0058 D loss: -0.4823 G loss: -2.942
Epoch: 0059 D loss: -0.4393 G loss: -2.88
Epoch: 0060 D loss: -0.4891 G loss: -2.727
Epoch: 0061 D loss: -0.5531 G loss: -2.468
Epoch: 0062 D loss: -0.4788 G loss: -2.869
Epoch: 0063 D loss: -0.5832 G loss: -2.327
Epoch: 0064 D loss: -0.5149 G loss: -2.613
Epoch: 0065 D loss: -0.4985 G loss: -2.632
Epoch: 0066 D loss: -0.4478 G loss: -2.738
Epoch: 0067 D loss: -0.4604 G loss: -2.457
Epoch: 0068 D loss: -0.4922 G loss: -2.554
Epoch: 0069 D loss: -0.5522 G loss: -2.669
Epoch: 0070 D loss: -0.4112 G loss: -2.918
Epoch: 0071 D loss: -0.4938 G loss: -2.609
Epoch: 0072 D loss: -0.6117 G loss: -2.678
Epoch: 0073 D loss: -0.5208 G loss: -2.435
Epoch: 0074 D loss: -0.4941 G loss: -2.496
Epoch: 0075 D loss: -0.5028 G loss: -2.812
Epoch: 0076 D loss: -0.3788 G loss: -2.59
Epoch: 0077 D loss: -0.5201 G loss: -3.027
Epoch: 0078 D loss: -0.5821 G loss: -2.644
Epoch: 0079 D loss: -0.6687 G loss: -2.473
Epoch: 0080 D loss: -0.5094 G loss: -2.772
Epoch: 0081 D loss: -0.5823 G loss: -2.538
Epoch: 0082 D loss: -0.4968 G loss: -2.59
Epoch: 0083 D loss: -0.459 G loss: -2.481
Epoch: 0084 D loss: -0.6422 G loss: -2.365
Epoch: 0085 D loss: -0.5191 G loss: -2.543
Epoch: 0086 D loss: -0.5272 G loss: -2.613
Epoch: 0087 D loss: -0.4131 G loss: -2.734
```

```
Epoch: 0088 D loss: -0.5643 G loss: -2.563
Epoch: 0089 D loss: -0.4333 G loss: -2.544
Epoch: 0090 D loss: -0.4739 G loss: -2.46
Epoch: 0091 D loss: -0.4366 G loss: -2.952
Epoch: 0092 D loss: -0.5228 G loss: -3.015
Epoch: 0093 D loss: -0.5097 G loss: -2.731
Epoch: 0094 D loss: -0.5056 G loss: -2.693
Epoch: 0095 D loss: -0.431 G loss: -3.032
Epoch: 0096 D loss: -0.4503 G loss: -2.769
Epoch: 0097 D loss: -0.3913 G loss: -2.829
Epoch: 0098 D loss: -0.5614 G loss: -3.046
Epoch: 0099 D loss: -0.548 G loss: -2.632
최적화 완료!
Wall time: 12min 28s
```

In [ ]:

```

```