04. 대표적 비지도 학습법 - Autoencoder ¶

학습 내용

01. Autoencoder란?

02. 간단한 예제를 보자.

03. 왜 사용되는가?

01. Autoencoder란?

- 머신러닝 학습 방법은 크게 지도학습과 비지도 학습으로 나눌 수 있다.
- 비지도 학습 중 가장 널리 쓰이는 신경망으로 오토인코더(Autoencoder)가 있다.
- 입력값과 출력값을 같게 하는 신경망이다.
- 가운데 계층의 노드 수가 입력값보다 적은 것이 특징이다 - 노이즈 제거에 매우 효과적

가. 비지도 학습 중 가장 널리 쓰이는 신경망으로 오토 인코더(AutoEncoder)가 있다.

나. 오토 인코더는 입력값과 출력값을 같게 하는 신경망이다.

- 입력층으로 들어온 데이터를 인코더를 통해 은닉층으로 내보낸다.
- 은닉층의 데이터를 디코더를 통해 출력층으로 내보낸다.
- 만들어진 출력값을 입력값과 비슷해지도록 만드는 가중치를 찾아내는 것이다.

다. 가운데 계층의 노드 수가 입력값보다 적은 것이 특징이다.

- 결과적으로 입력 데이터를 압축하는 효과를 얻는다. 이 과정이 노이즈 제거에 많이 효과적이다.

라. 오토 인코더는 변이형 오토 인코더(Variational Autoencoder), 잡음제거 오토 인코더(Denoising Autoencoder)등 다양한 방식이 있다.

메모 :

- (01) 입력층으로 들어온 데이터를 인코더를 통해 은닉층으로 내보낸다.
- (02) 은닉층의 데이터를 디코더를 통해 출력층으로 내보낸다.
- (03) 만들어진 출력값과 입력값이 같아지도록 만드는 가중치를 찾아낸다.

02. 간단한 예제를 보자.

In [2]:
```
1 import tensorflow as tf
2 import numpy as np
3 import matplotlib.pyplot as plt
```

```
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
```

In [3]:
```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("./mnist/data/", one_hot=True)
```

```
WARNING:tensorflow:From <ipython-input-3-4dcbd946c02b>:2: 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₩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
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.
Extracting ./mnist/data/t10k-images-idx3-ubyte.gz
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: D
ataSet.__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.
```

In [5]:
```
learning_rate = 0.01    # 학습율 0.01
training_epoch = 20     # 훈련 횟수 20회
batch_size = 100        # 배치 사이즈 100
n_hidden = 256          # 은닉층의 개수 256
n_input = 28 * 28       # 784개 (입력층)
```

인코더 만들기

- (1) 맨처음은 n_hidden개의 뉴런을 만든다.
  - 가중치와 편향 변수를 원하는 뉴런의 개수만큼 설정한다.
- (2) 변수들을 입력값과 곱하고 더한 후, 활성화 함수 sigmoid 함수를 적용한다.
  - n_input 값보다 n_hidden 값이 더 작다.

In [7]:
```
X = tf.placeholder(tf.float32, [None, n_input]) # X 플레이스 홀더, 비지도학습 Y가 없음.
W_encode = tf.Variable(tf.random_normal([n_input, n_hidden]))
b_encode = tf.Variable(tf.random_normal([n_hidden]))

encoder = tf.nn.sigmoid(tf.add(tf.matmul(X, W_encode), b_encode))
```

디코더 만들기

- 입력층을 은닉층의 크기로 출력값을 입력층의 크기로 만듬.

In [8]:
```
W_decode = tf.Variable(tf.random_normal([n_hidden, n_input]))
b_decode = tf.Variable(tf.random_normal([n_input]))
decoder = tf.nn.sigmoid(tf.add(tf.matmul(encoder, W_decode), b_decode))
```

In [9]:
```
cost = tf.reduce_mean(tf.pow(X- decoder, 2))
optimizer = tf.train.RMSPropOptimizer(learning_rate).minimize(cost)
```

학습을 진행

In [10]:
```
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

total_batch = int(mnist.train.num_examples/batch_size)

for epoch in range(training_epoch):
    total_cost = 0

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        _, cost_val = sess.run([optimizer, cost],
                               feed_dict={X:batch_xs})
        total_cost += cost_val

    print('Epoch:', '%04d' % (epoch + 1),
          'Avg. cost=', '{:.4f}'.format(total_cost / total_batch))

print('최적화 완료!')
```

```
Epoch: 0001 Avg. cost= 0.1994
Epoch: 0002 Avg. cost= 0.0594
Epoch: 0003 Avg. cost= 0.0485
Epoch: 0004 Avg. cost= 0.0425
Epoch: 0005 Avg. cost= 0.0384
Epoch: 0006 Avg. cost= 0.0361
Epoch: 0007 Avg. cost= 0.0352
Epoch: 0008 Avg. cost= 0.0346
Epoch: 0009 Avg. cost= 0.0326
Epoch: 0010 Avg. cost= 0.0315
Epoch: 0011 Avg. cost= 0.0312
Epoch: 0012 Avg. cost= 0.0309
Epoch: 0013 Avg. cost= 0.0307
Epoch: 0014 Avg. cost= 0.0304
Epoch: 0015 Avg. cost= 0.0302
Epoch: 0016 Avg. cost= 0.0301
Epoch: 0017 Avg. cost= 0.0295
Epoch: 0018 Avg. cost= 0.0282
Epoch: 0019 Avg. cost= 0.0276
Epoch: 0020 Avg. cost= 0.0274
최적화 완료!
```

총 10개의 테스트 데이터를 가져와 디코더를 이용해 출력값을 만든다.

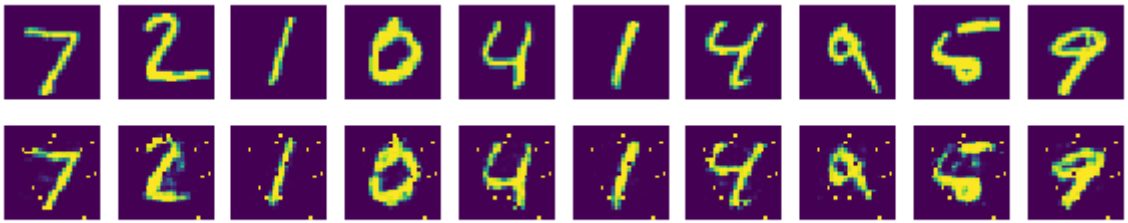
In [11]:
```
sample_size = 10
samples = sess.run(decoder,
                   feed_dict = {X:mnist.test.images[:sample_size]})
```

위쪽이 원본 데이터, 아래쪽이 신경망이 생성한 이미지

In [14]:
```
fig, ax = plt.subplots(2, sample_size, figsize=(sample_size, 2))

for i in range(sample_size):
    ax[0][i].set_axis_off()
    ax[1][i].set_axis_off()
    ax[0][i].imshow(np.reshape(mnist.test.images[i], (28,28)))
    ax[1][i].imshow(np.reshape(samples[i], (28,28)))

plt.show()
```



In [ ]:
```

```