Unsupervised Learning Algorithm

```
import warnings
warnings.filterwarnings('ignore')
```

Import Packages

```
import numpy as np
import pandas as pd
```

I. Load Datasets

1) MNIST Dataset

```
from keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

```
X_train = X_train.reshape(-1, 28 * 28)
X_test = X_test.reshape(-1, 28 * 28)
```

```
X_train.shape, X_test.shape
```

```
((60000, 784), (10000, 784))
```

2) 'mnist_32.csv' Data

- AutoEncoder 'Latent Space' Data

```
url = 'https://raw.githubusercontent.com/rusita-ai/pyData/master/mnist_32.csv'
mnist_32 = pd.read_csv(url)

mnist_32.shape
```

```
(10000, 32)
```

```
mnist_32.head()
```

	0	1	2	3	4	5	6	7	
0	3.688405	3.930005	9.707697	11.878195	8.254636	5.676317	17.008430	5.514244	-(
1	8.770631	13.023509	8.275418	10.167507	4.191549	8.799183	7.692517	14.583757	-(
2	3.612655	4.127336	6.994210	11.199030	4.834584	2.894823	7.723920	3.293408	(
3	4.778088	11.318460	19.590998	9.894455	8.185436	5.662807	9.577927	10.103026	-(
4	4.581480	6.314673	9.630608	5.819597	4.213900	6.164255	1.878163	4.087563	-(

II. K-means Clustering

1) Modeling - sklearn.cluster

```
from sklearn.cluster import KMeans

Clusters = KMeans(n_clusters = 10,
                  n_init = 10,
                  random_state = 2045)

Clusters.fit(mnist_32)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=10, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=2045, tol=0.0001, verbose=0)
```

2) Visualization

- Clusters.labels_ : 0부터 9까지 10개의 군집 정보
  - 각 군집에 속한 이미지 시각화
  - 'X_test' Data 사용

```
import matplotlib.pyplot as plt

plt.figure(figsize = (12, 12))

for i in range(10):
    images = X_test[Clusters.labels_ == i]
    for n in range(10):
        plt.subplot(10, 10, i*10 + n + 1)
        plt.imshow(images[n].reshape(28, 28), cmap = 'gray')
        plt.axis('off')

plt.show()
```

```
plt.show()
```

III. t-SNE(Stochastic Neighbor Embedding)

- 각 데이터의 유사도를 확률적(Stochastic)으로 표현
- 하나의 데이터로부터 다른 데이터에 대한 거리를 't-분포'의 확률로 치환
  - 가까운 거리의 데이터는 높은 확률값
  - 먼 거리의 데이터는 확률값이 낮음
  - https://bcho.tistory.com/1210
- 고차원과 저차원에서 확률값을 계산 후, 저차원 확률값이 고차원에 가까워지도록 학습
- 연산에 많은 시간이 걸리기 때문에 50차원 이하의 데이터 사용을 권장
  - 32차원의 'mnist_32.csv' Data

1) Modeling - sklearn.manifold

- 약 2분

- n_components : 축소 차원 공간
- learning_rate : 학습률(10 ~ 1000)
- perplexity : 고려할 최근접 이웃의 숫자(5 ~ 50)

```
%%time

from sklearn.manifold import TSNE

tsne = TSNE(n_components = 2,
            learning_rate = 100,
            perplexity = 15,
            random_state = 2045)

tsne_vector = tsne.fit_transform(mnist_32)
```

```
CPU times: user 2min 39s, sys: 445 ms, total: 2min 39s
Wall time: 1min 24s
```

- 2차원 정보로 축소

```
tsne_vector.shape
```

```
(10000, 2)
```

2) Visualization

- tsne_vector : 2차원 축소 정보
  - Label 별로 잘 뭉쳐있는 것을 확인
  - 'y_test' Data 사용

```
plt.figure(figsize = (9, 7))

fig = plt.scatter(tsne_vector[:, 0],
                  tsne_vector[:, 1],
                  marker = '.',
                  c = y_test,
                  cmap = 'rainbow')
cb = plt.colorbar(fig, ticks = range(10))

plt.show()
```

- 'y_test' 정보와 비교

```
from matplotlib.offsetbox import TextArea, DrawingArea, OffsetImage, AnnotationBbox

plt.figure(figsize = (15, 15))

ax = plt.subplot(1, 1, 1)
ax.scatter(tsne_vector[:, 0],
           tsne_vector[:, 1],
           marker = '.',
           c = y_test,
           cmap = 'rainbow')

for i in range(200):
    imagebox = OffsetImage(X_test[i].reshape(28, 28))
    ab = AnnotationBbox(imagebox,
                        (tsne_vector[i, 0], tsne_vector[i, 1]),
                        frameon = False,
                        pad = 0.0)
    ax.add_artist(ab)

ax.set_xticks([])
ax.set_yticks([])

plt.show()
```

3) 'perplexity'별 결과 비교

- 약 11분

- perplexity : 고려할 최근접 이웃의 숫자(5 ~ 50)
  - 'perplexity' 변화에 따른 군집형태 확인
  - 'y_test' Data 사용

```
%%time

perplexities = [5, 10, 25, 50, 75, 100]

plt.figure(figsize = (10, 15))

for i in range(6):
    tsne = TSNE(n_components = 2,
                learning_rate = 100,
                perplexity = perplexities[i],
                random_state = 2045)

    tsne_vector = tsne.fit_transform(mnist_32)

    plt.subplot(3, 2, i + 1)
    plt.scatter(tsne_vector[:, 0],
                tsne_vector[:, 1],
                marker = '.',
                c = y_test,
                cmap = 'rainbow')
    plt.title('perplexity: {0}'.format(perplexities[i]))

plt.show()
```

#

#

#

The End

#

#

#

✓ 10분 18초 오전 8:55에 완료됨