Principal Component Analysis

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING
인공지능학과
Department of Artificial Intelligence

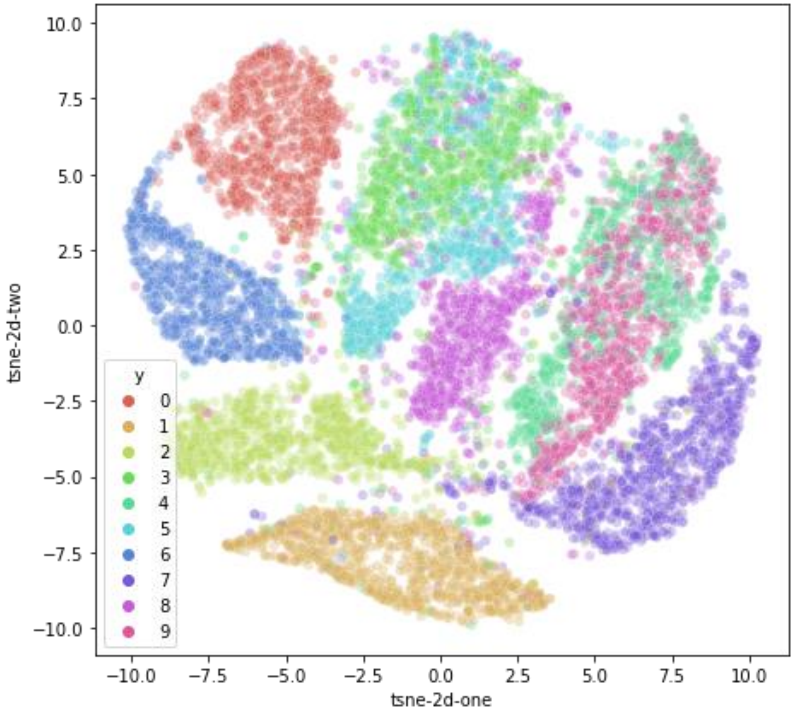
정 우 환 (whjung@hanyang.ac.kr)

Fall 2021

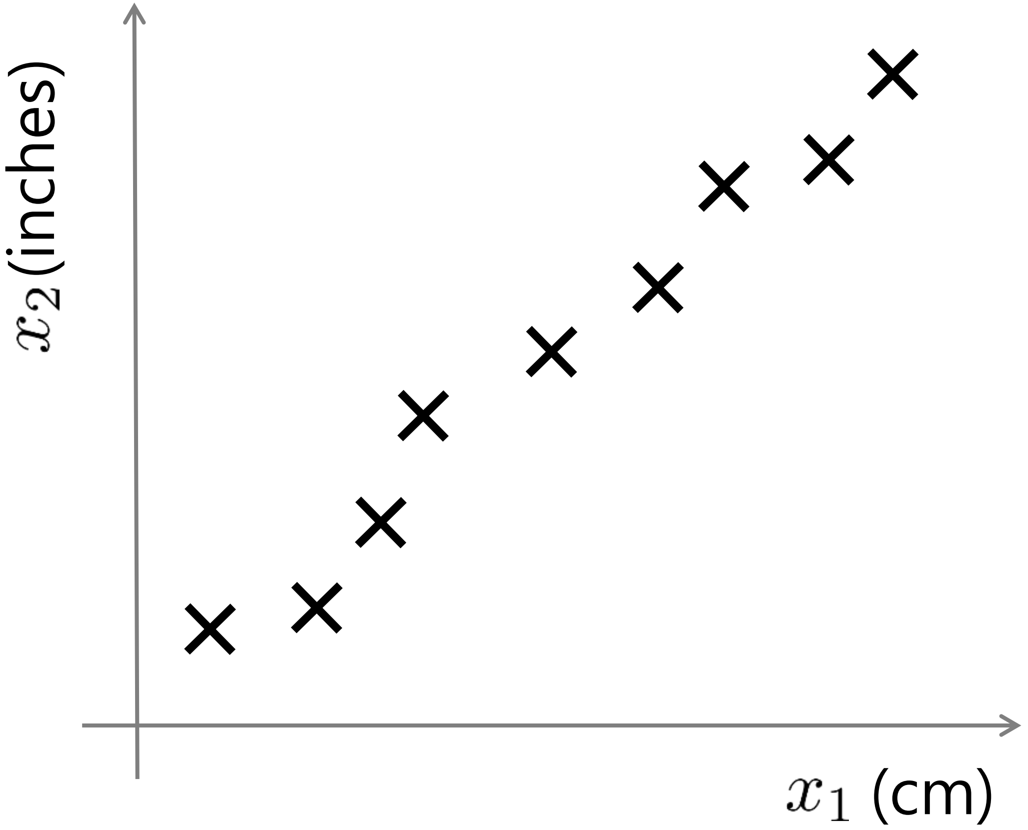
Dimensionality reduction

- Summarize data with a lower dimensional real valued vector

- Given data points in d dimensions
- Convert them to data points in $r<d$ dimensions
- With minimal loss of information

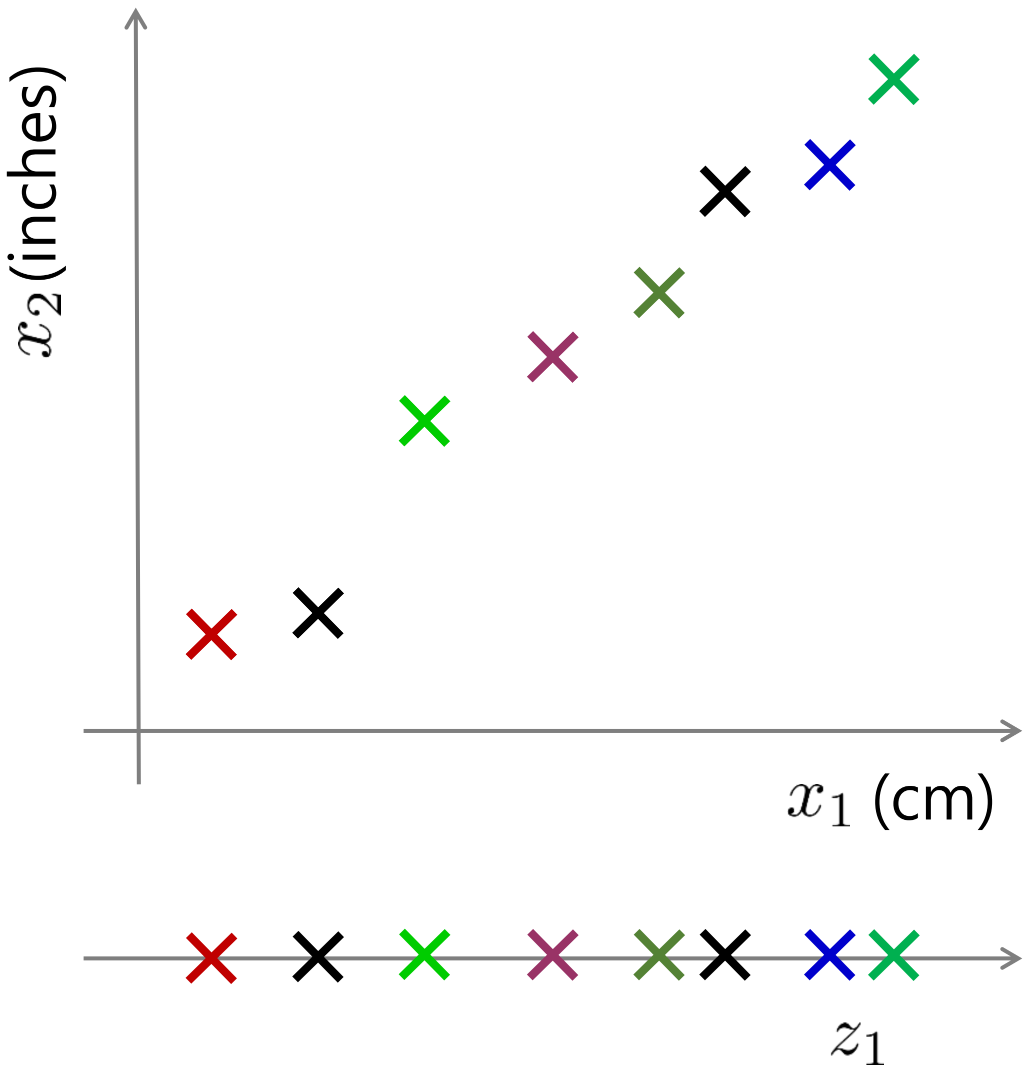


Data Compression



Reduce data from 2D to 1D

Data Compression



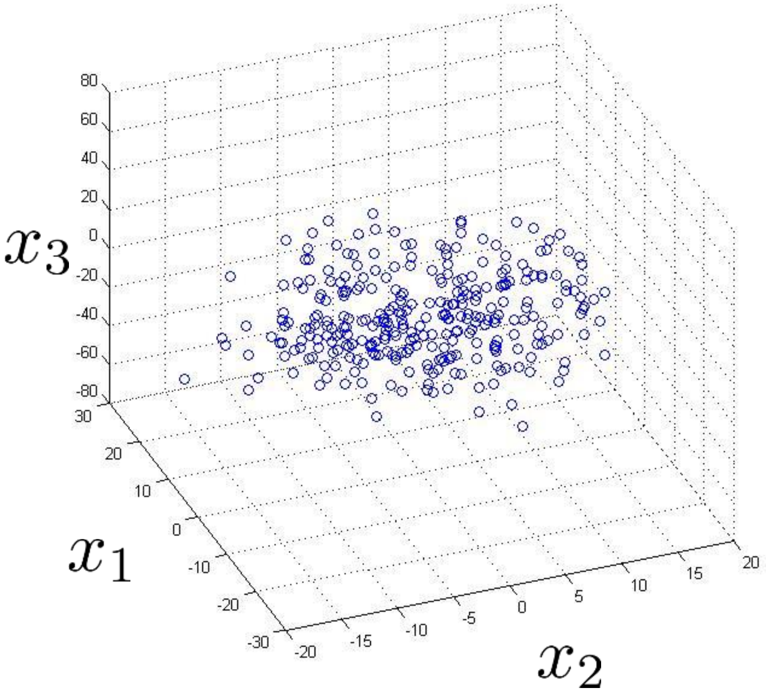
Reduce data from 2D to 1D

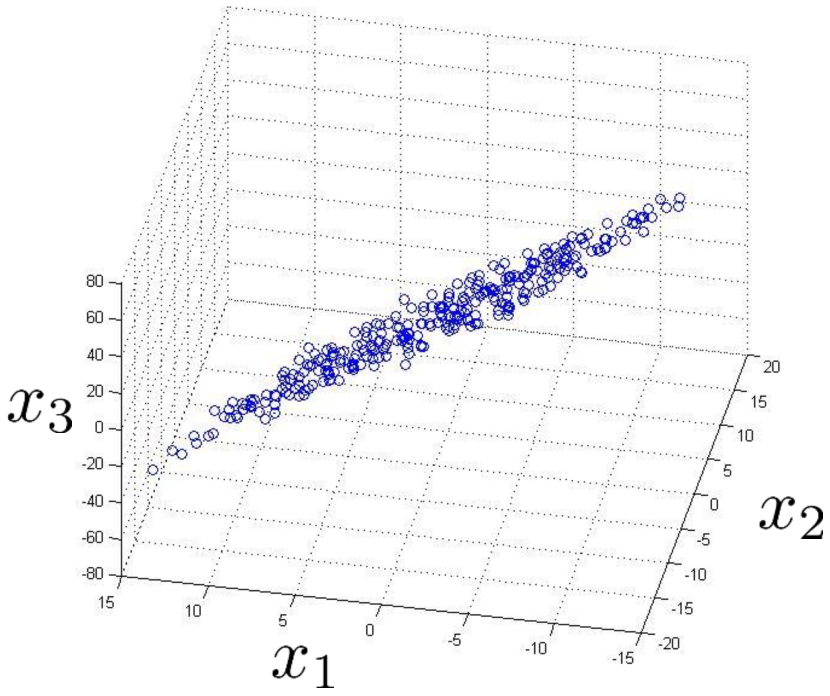
$$x^{(1)} \rightarrow z^{(1)}$$
$$x^{(2)} \rightarrow z^{(2)}$$
$$\vdots$$
$$x^{(m)} \rightarrow z^{(m)}$$

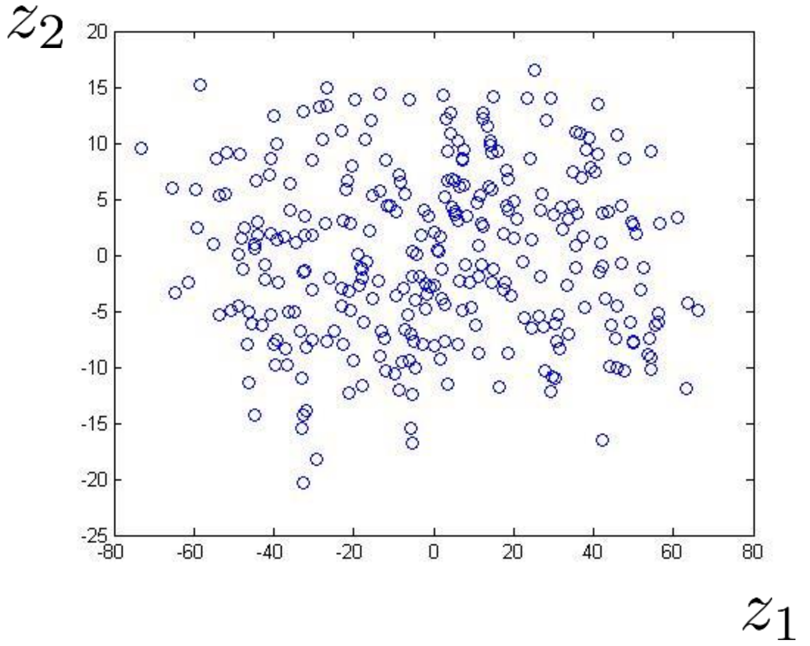
Andrew Ng

Data Compression

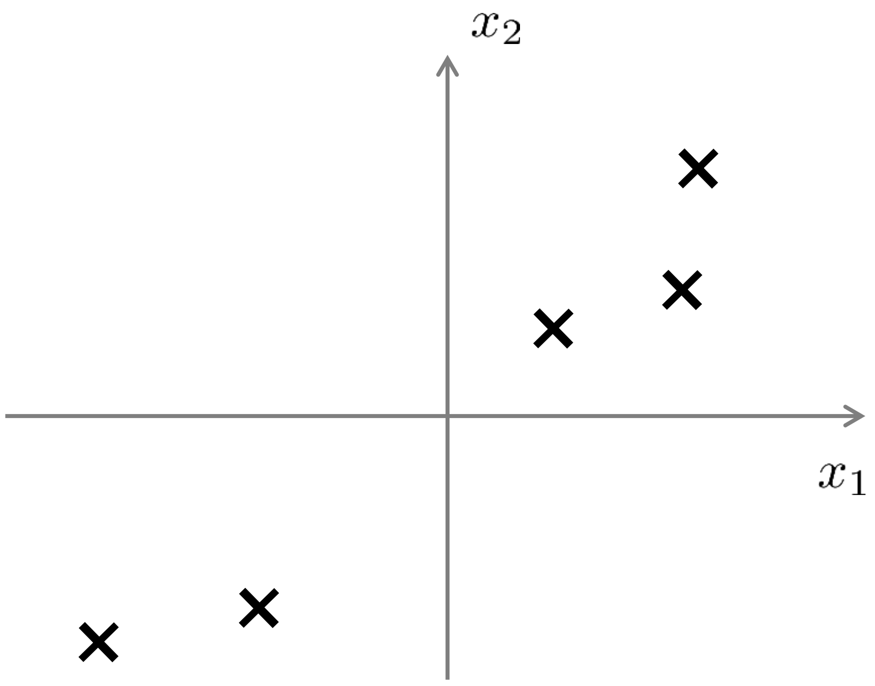
Reduce data from 3D to 2D



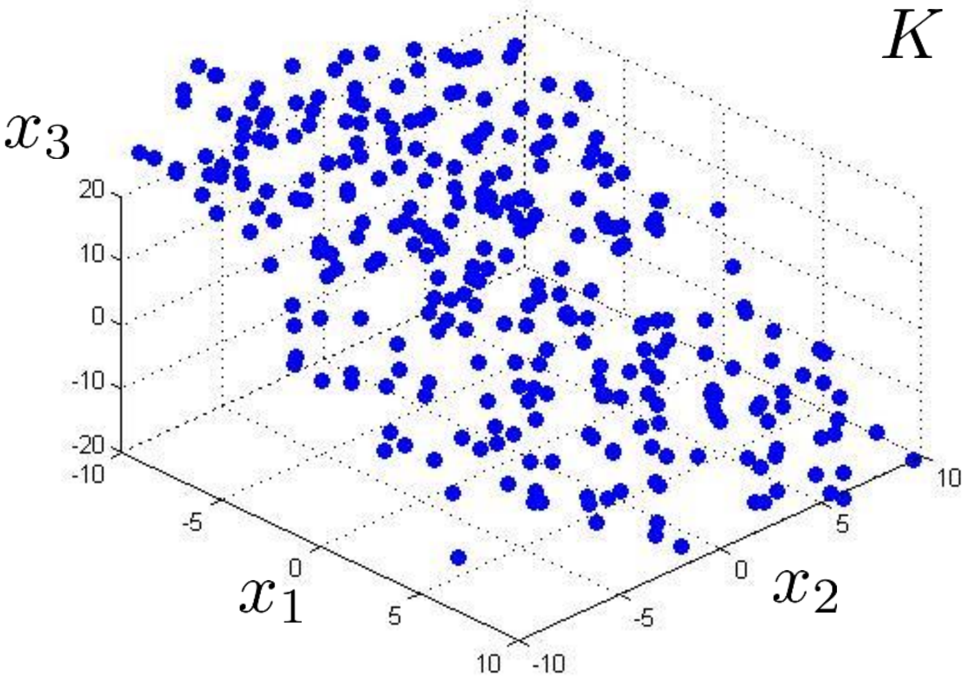




Principal Component Analysis (PCA) problem formulation



$3D \rightarrow 2D$
$K = 2$



Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find $k$vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

Principal Component Analysis

Goal: Find r-dim projection that best preserves variance

1. Compute mean vector μ and covariance matrix Σ of original points

2. Compute eigenvectors and eigenvalues of Σ

3. Select top r eigenvectors

4. Project points onto subspace spanned by them:

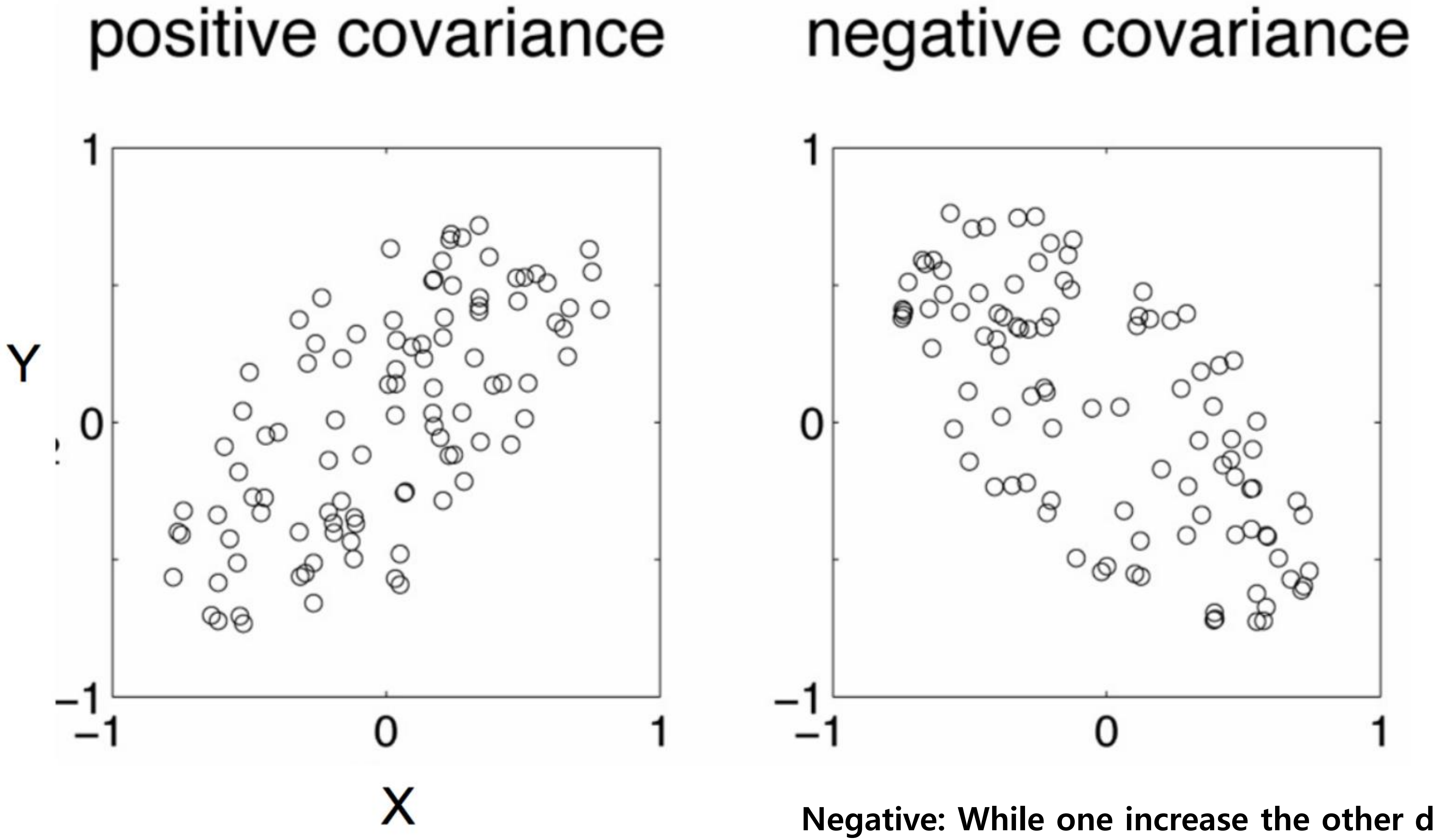
$$y = A(x - \mu)$$

   where y is the new point, x is the old one, and the rows of A are the eigenvectors

Covariance

- Variance and Covariance:
  - Measure of the “spread” of a set of points around their center of mass(mean)
- Variance:
  - Measure of the deviation from the mean for points in one dimension
- Covariance:
  - Measure of how much each of the dimensions vary from the mean with **respect to each other**

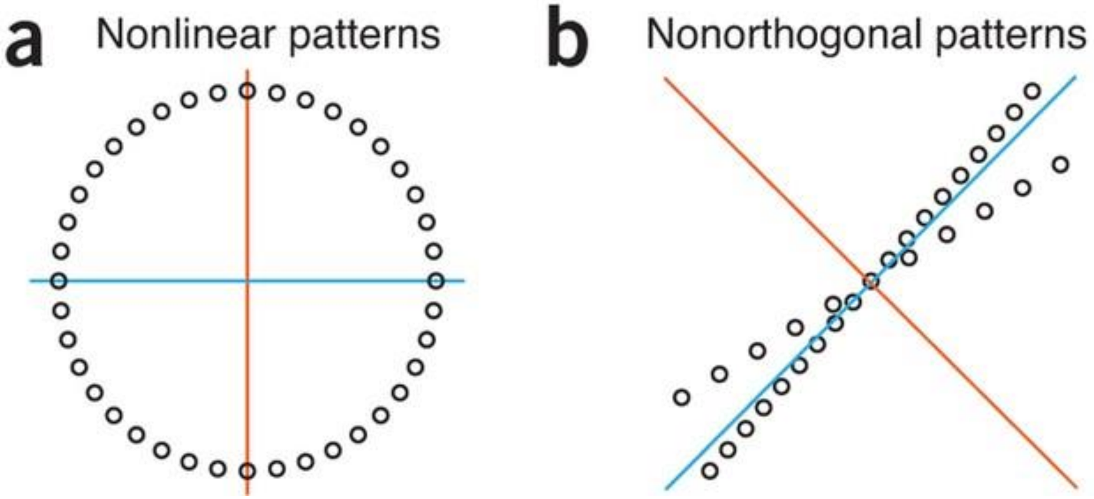
- **Covariance is measured between two dimensions**
- **Covariance sees if there is a relation between two dimensions**
- **Covariance between one dimension is the variance**

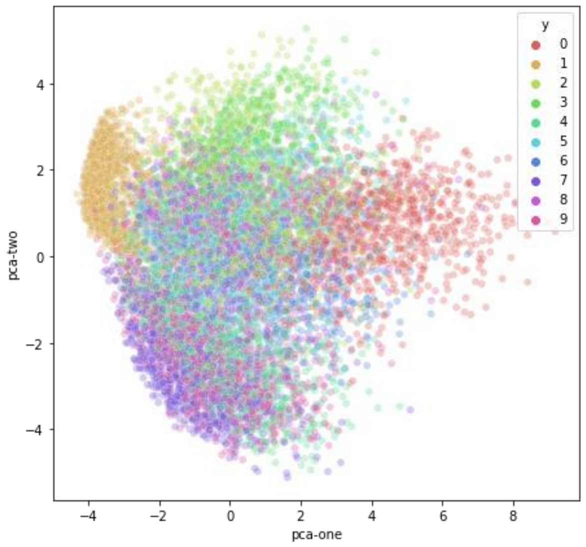


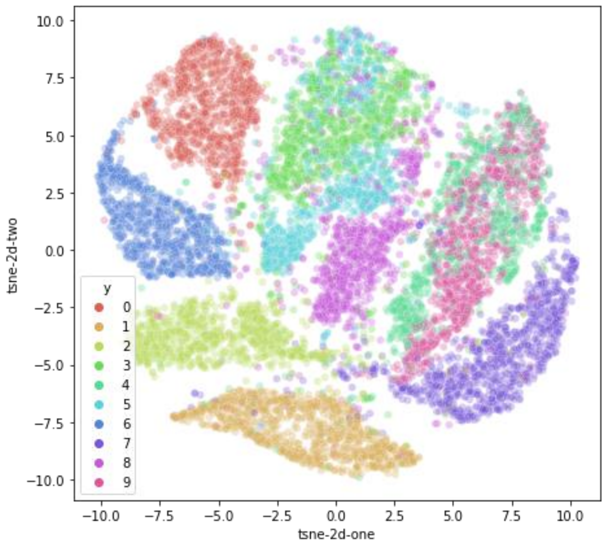
Negative: While one increase the other decrease

Positive: Both dimensions increase or decrease together

Limitations of PCA







- T-Distributed Stochastic Neighbor Embedding (tSNE) can be used!

Term project

MNIST visualization

Term project

- MNIST visualization
- Objective:
  - Deep neural network 구현 및 Visualization
  - Visualization을 통한 Neural network에 대한 이해 향상

실습

MNIST 불러오기

In [1]:
```python
from __future__ import print_function
import time
import numpy as np
import pandas as pd

from sklearn.datasets import fetch_openml
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns

%matplotlib inline
```

In [2]:
```python
# Load MNIST data
mnist = fetch_openml('mnist_784', version=1, cache=True)
X = mnist.data / 255.0
y = mnist.target
```

In [3]:
```python
print(f'X.shape : {X.shape}')
print(f'y.shape : {y.shape}')
```

```
X.shape : (70000, 784)
y.shape : (70000,)
```

In [4]:
```python
feat_cols = [f'pixel{i}' for i in range(X.shape[1])]
df = pd.DataFrame(X.values, columns=feat_cols)
df['y'] = y
df
```

Out[4]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

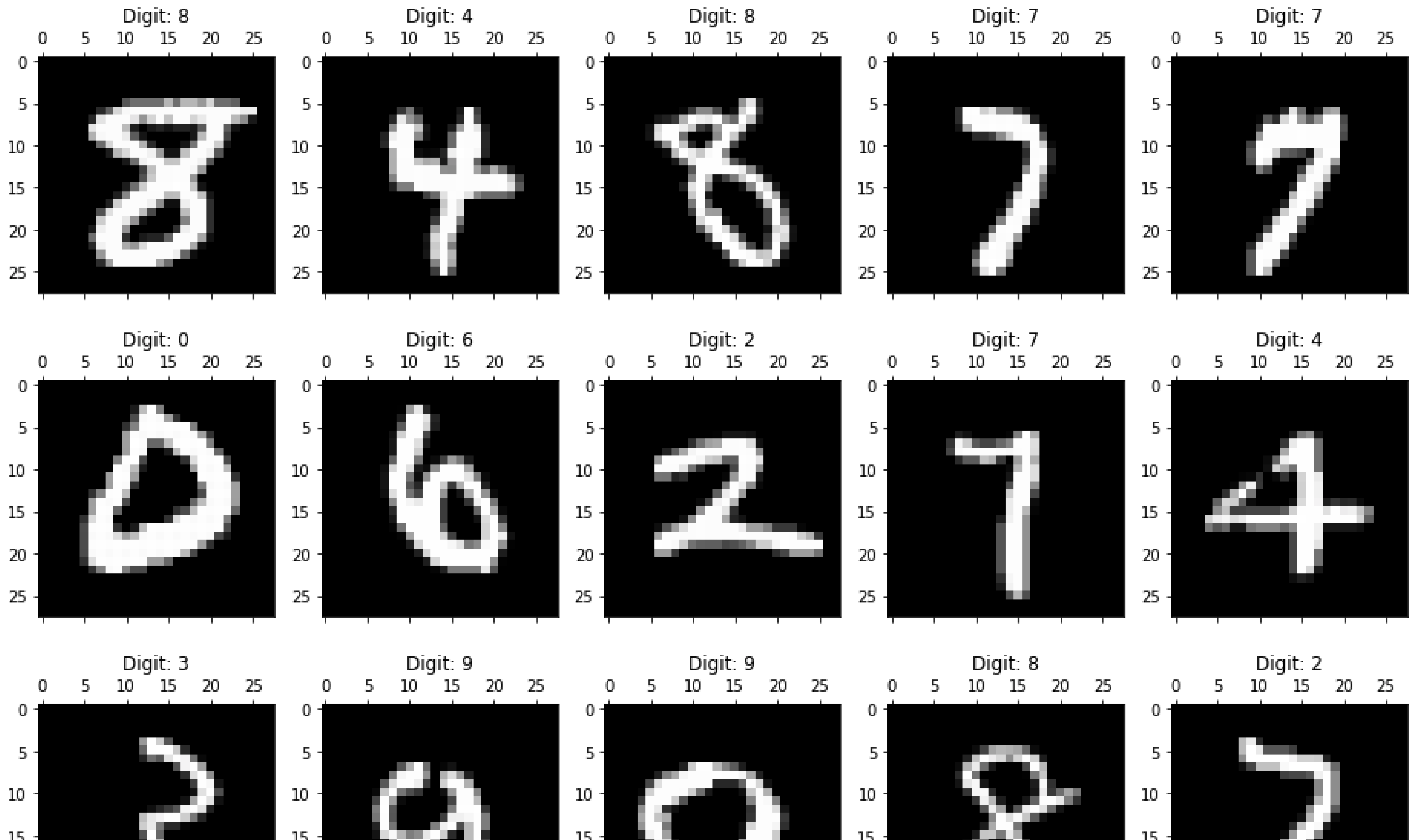
실습

MNIST 확인

```
In [5]: # For reproducability of the results
        np.random.seed(42)
        rndperm = np.random.permutation(df.shape[0])
```

```
In [6]: plt.gray()
        fig = plt.figure( figsize=(16,11) )
        for i in range(0,15):
            ax = fig.add_subplot(3,5,i+1, title="Digit: {}".format(str(df.loc[rndperm[i],'y'])) )
            ax.matshow(df.loc[rndperm[i],feat_cols].values.reshape((28,28)).astype(float))
        plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



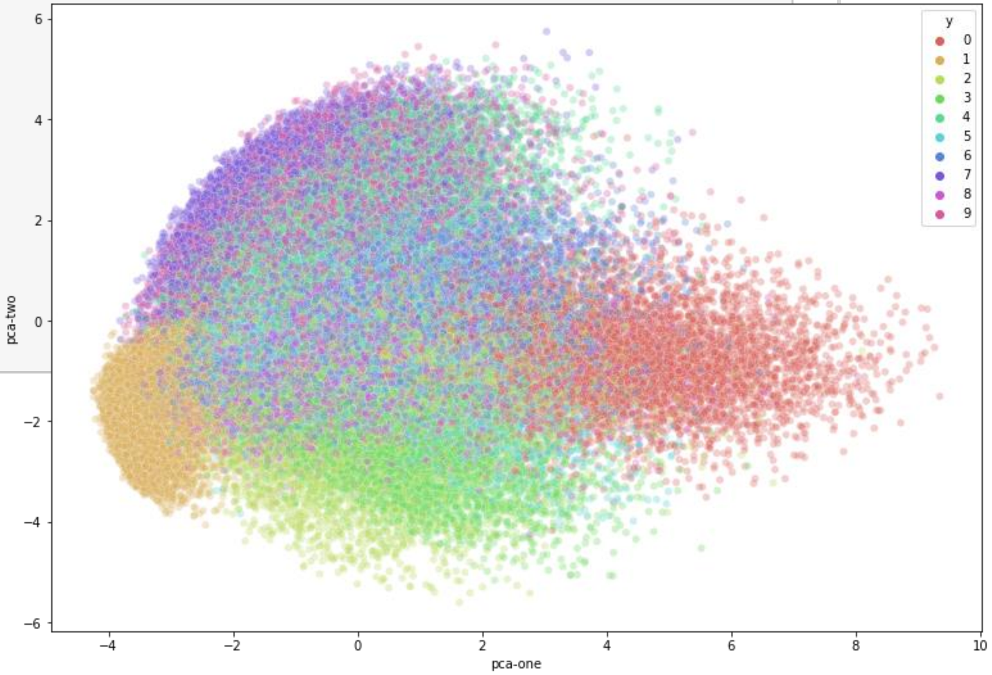
실습

PCA 시각화

```
In [7]: pca = PCA(n_components=2)
        pca_result = pca.fit_transform(df[feat_cols].values)
        df['pca-one'] = pca_result[:,0]
        df['pca-two'] = pca_result[:,1]
        print('Explained variation per principal component: {}'.format(pca.explained_variance_ratio_))

        Explained variation per principal component: [0.09746116 0.07155445]
```

```
In [16]: plt.figure(figsize=(13,9))
         sns.scatterplot(
             x="pca-one", y="pca-two",
             hue="y",
             palette=sns.color_palette("hls", 10),
             data=df.loc[rndperm,:],
             legend="full",
             alpha=0.3
         )
         plt.show()
```



실습

t-SNE 시각화

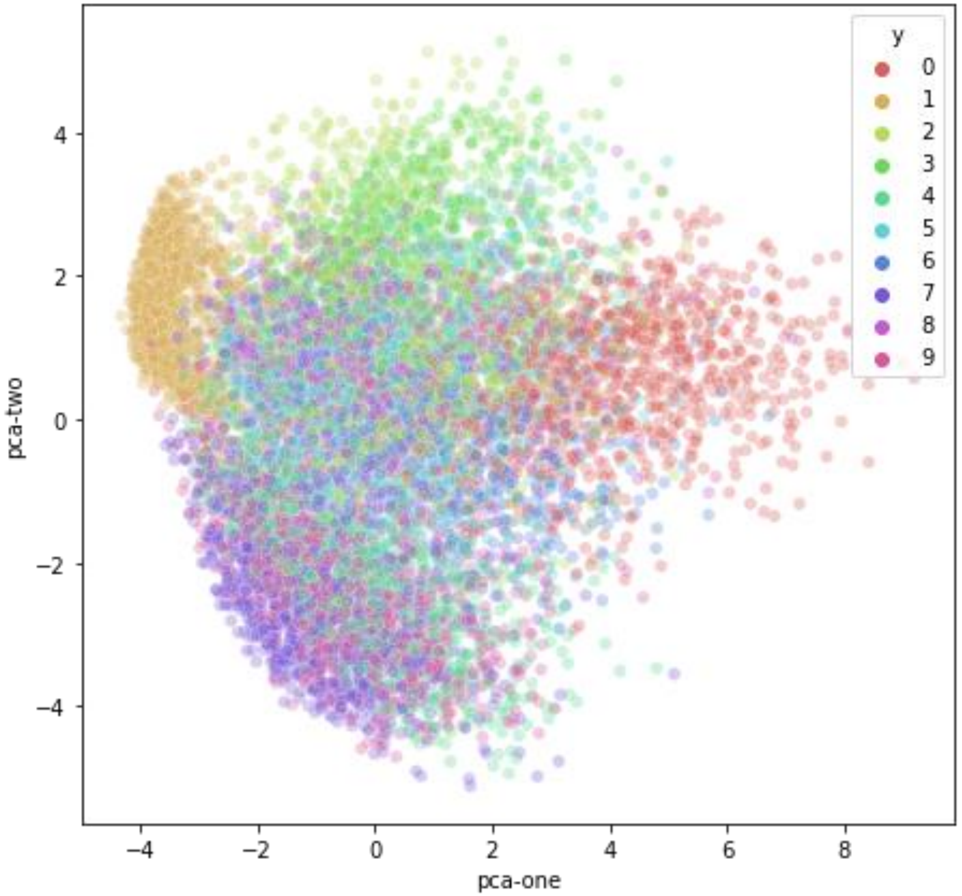
In [8]:
```
N = 10000
df_subset = df.loc[rndperm[:N],:].copy()
data_subset = df_subset[feat_cols].values
pca = PCA(n_components=2)
pca_result = pca.fit_transform(data_subset)
df_subset['pca-one'] = pca_result[:,0]
df_subset['pca-two'] = pca_result[:,1]
print('Explained variation per principal component: {}'.format(pca.explained_variance_ratio_))
```

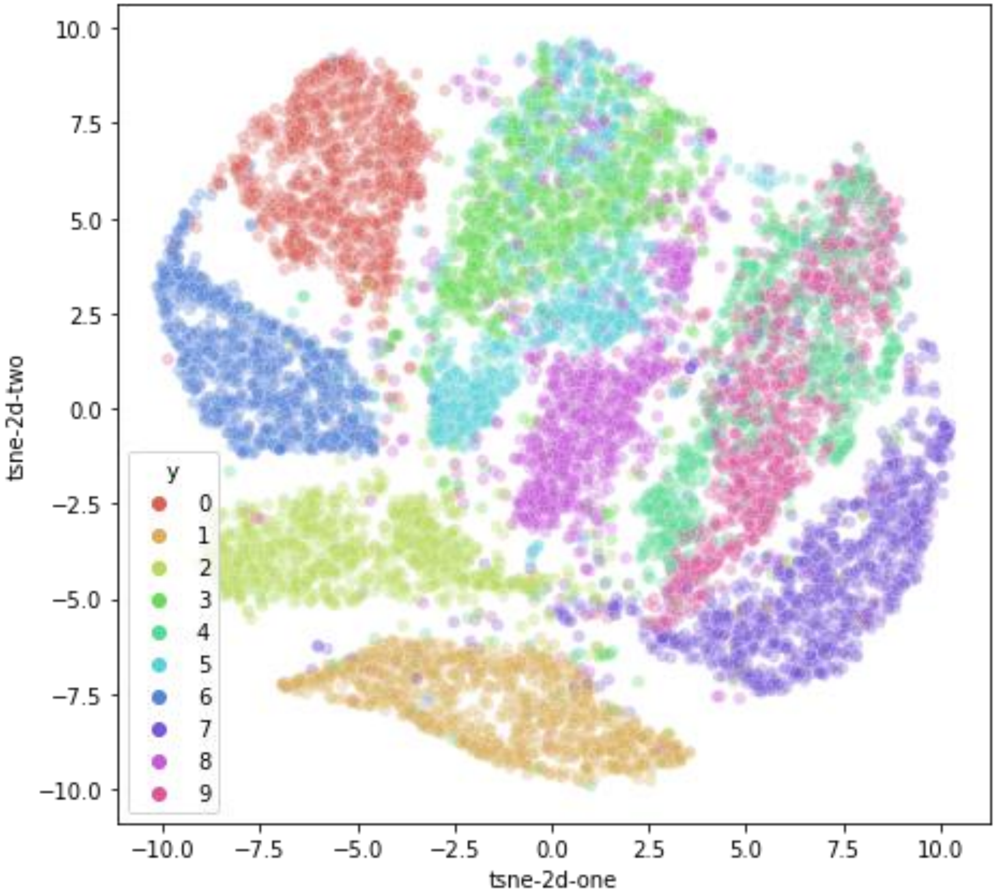
```
Explained variation per principal component: [0.09819946 0.07123677]
```

In [9]:
```
time_start = time.time()
tsne = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=300)
tsne_results = tsne.fit_transform(data_subset)
df_subset['tsne-2d-one'] = tsne_results[:,0]
df_subset['tsne-2d-two'] = tsne_results[:,1]
print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))
```

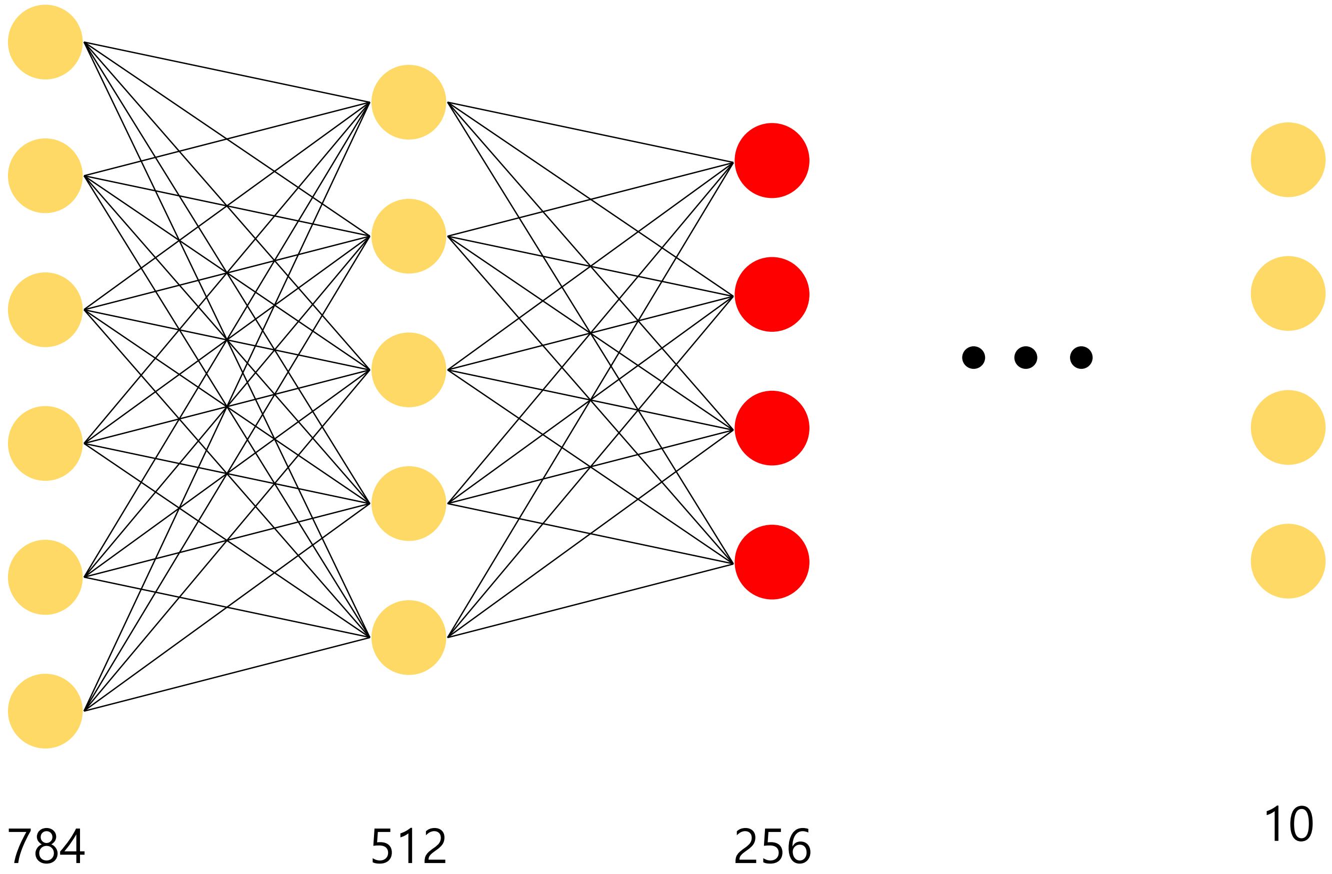
```
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 10000 samples in 0.006s...
[t-SNE] Computed neighbors for 10000 samples in 1.705s...
[t-SNE] Computed conditional probabilities for sample 1000 / 10000
[t-SNE] Computed conditional probabilities for sample 2000 / 10000
[t-SNE] Computed conditional probabilities for sample 3000 / 10000
[t-SNE] Computed conditional probabilities for sample 4000 / 10000
[t-SNE] Computed conditional probabilities for sample 5000 / 10000
[t-SNE] Computed conditional probabilities for sample 6000 / 10000
[t-SNE] Computed conditional probabilities for sample 7000 / 10000
[t-SNE] Computed conditional probabilities for sample 8000 / 10000
[t-SNE] Computed conditional probabilities for sample 9000 / 10000
[t-SNE] Computed conditional probabilities for sample 10000 / 10000
[t-SNE] Mean sigma: 2.117975
[t-SNE] KL divergence after 250 iterations with early exaggeration: 85.791290
[t-SNE] KL divergence after 300 iterations: 2.802236
t-SNE done! Time elapsed: 7.769311904907227 seconds
```

PCA vs t-SNE

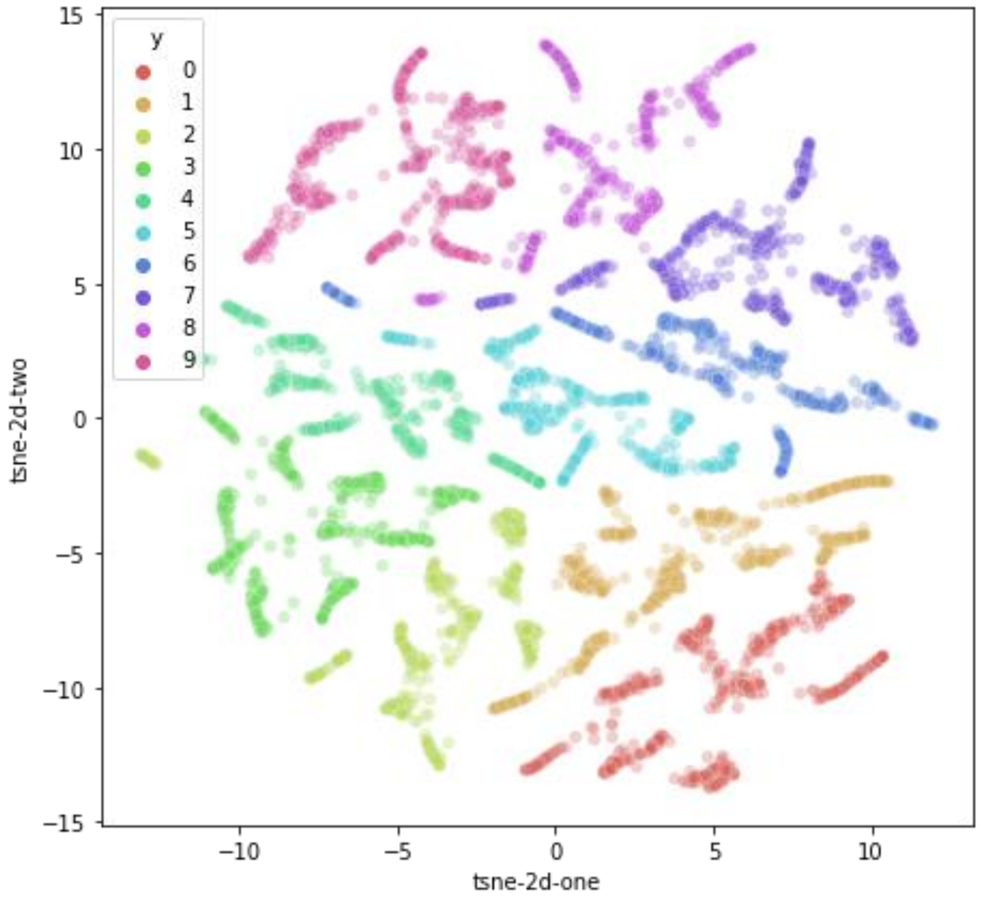

y
0
1
2
3
4
5
6
7
8
9
pca-two
pca-one
4
2
0
−2
−4
−4
−2
0
2
4
6
8


y
0
1
2
3
4
5
6
7
8
9
tsne-2d-two
tsne-2d-one
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Hidden layer's PCA & t-SNE visualization



PCA vs t-SNE (Hidden layer 256)



Term project

- Due: 11월 10일 23:59
- 제출물
  - 코드
  - 리포트 (pdf, 최대 3페이지)
    - 중요 코드 설명
    - Input image에 대한 PCA, t-SNE 결과
    - hidden vector (z[1], a[1], z[2], a[2])에 대한 PCA, ,t-SNE결과
    - 결과 분석 (자유롭게)

Reference

- https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b
- Dimensionality Reduction, Fereshteh Sadeghi