0.1 On PCA

In [2]:

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
```

executed in 871ms, finished 10:20:04 2022-11-13

1 scale

In [5]:

```
def scale(X):
    '''Input should be a matrix.'''
    return (X-np.mean(X, axis=1))/np.std(X, axis=1, ddof=1)
```

executed in 7ms, finished 10:20:43 2022-11-13

In [6]:

```
X = np.matrix([[.009, .229, .525, .943, .477, .484, .577, .43, .871, .82],
          [-.388, -.098, .293, .845, .23, .239, .362, .168, .75, .682]])
X_norm = scale(X)
X_norm
```

executed in 21ms, finished 10:20:44 2022-11-13

Out[6]:

```
matrix([[-1.82764444, -1.0654041 , -0.03984438,  1.40841225, -0.206151
36,
         -0.18189826,  0.14032152, -0.36899362,  1.15895178,  0.982250
61],
        [-1.82756108, -1.06640538, -0.04015752,  1.40866298, -0.205512
04,
         -0.18188997,  0.14094504, -0.36824188,  1.15931887,  0.980840
98]])
```

2 SVD

In [7]:

```
U, s, V_t = np.linalg.svd(X_norm)
sigma = np.diag(s).dot(np.eye(2, 10))
print(U.shape, sigma.shape, V_t.shape)
print('U:\n', U)
print('sigma:\n', sigma)
print('V_transpose:\n', V_t)
```

executed in 12ms, finished 10:20:52 2022-11-13

```
(2, 2) (2, 10) (10, 10)
U:
 [[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
sigma:
 [[4.24264041e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.52560395e-03 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]]
V_transpose:
 [[-6.09200959e-01 -3.55301604e-01 -1.33336519e-02  4.69512569e-01
  -6.86105714e-02 -6.06313751e-02  4.68777621e-02 -1.22872590e-01
   3.86378467e-01  3.27181954e-01]
 [ 3.86331075e-02 -4.64086663e-01 -1.45139977e-01  1.16210920e-01
   2.96322098e-01  3.84500122e-03  2.88998287e-01  3.48425028e-01
   1.70144803e-01 -6.53352606e-01]
 [ 2.55904252e-02 -1.39814303e-01  9.85843547e-01  5.13635369e-03
   3.01968508e-02  1.19887396e-03  2.79132813e-02  3.60764829e-02
   1.15868782e-02 -6.89565278e-02]
 [ 4.35375667e-01  2.23668886e-01  1.61953998e-02  8.61930467e-01
  -6.93607556e-03  1.61984607e-02 -3.77902865e-02  3.34968174e-03
  -1.20070223e-01 -3.26673040e-02]
 [-1.47458579e-01  2.62114675e-01  2.78870500e-02  1.74918174e-02
   9.34748035e-01 -6.03136846e-03 -5.37636439e-02 -8.03875446e-02
  -5.15842588e-04  1.59138286e-01]
 [-6.12676452e-02 -1.11452294e-02 -2.70999257e-04  1.76683340e-02
  -3.09162203e-03  9.97687696e-01  1.31276651e-03 -5.24018644e-03
   1.44186724e-02  1.35069650e-02]
 [-3.07583020e-02  2.83343683e-01  2.84032391e-02 -1.61621520e-02
  -5.93631609e-02 -1.62697993e-03  9.43735848e-01 -7.04062370e-02
  -2.79799805e-02  1.33410734e-01]
 [-2.15311361e-01  2.97928030e-01  3.23434716e-02  3.28871340e-02
  -7.83110570e-02 -8.66989061e-03 -6.17969796e-02  9.02513130e-01
   9.58280146e-03  1.95201968e-01]
 [ 3.38347816e-01  2.54174989e-01  2.05227741e-02 -1.14260724e-01
  -2.14673687e-02  1.24588415e-02 -4.51984953e-02 -1.62449708e-02
   8.96894456e-01  9.82826579e-03]
 [ 4.99892765e-01 -5.35098619e-01 -5.96239995e-02 -8.99223655e-02
   1.50482366e-01  1.98764156e-02  1.12622527e-01  1.89602451e-01
  -4.13373209e-02  6.15429872e-01]]
```

In [8]:

```
# SVD is a stable algorithm.
U*sigma*V_t
```

executed in 8ms, finished 10:21:12 2022-11-13

Out[8]:

```
matrix([[-1.82764444, -1.0654041 , -0.03984438,  1.40841225, -0.206151
36,
         -0.18189826,  0.14032152, -0.36899362,  1.15895178,  0.982250
61],
        [-1.82756108, -1.06640538, -0.04015752,  1.40866298, -0.205512
04,
         -0.18188997,  0.14094504, -0.36824188,  1.15931887,  0.980840
98]])
```

3 To find the first PC

3.1 lecture notes method

只看row1

In [16]:

```
UtX = -U.T[0]*X_norm
print('U_transpose:\n', U.T[0])
print('X_norm:\n', X_norm)
print('UtX:\n', UtX)
```

executed in 10ms, finished 10:23:10 2022-11-13

```
U_transpose:
 [[0.70710678 0.70710678]]
X_norm:
 [[-1.82764444 -1.0654041  -0.03984438  1.40841225 -0.20615136 -0.1818
9826
   0.14032152 -0.36899362  1.15895178  0.98225061]
 [-1.82756108 -1.06640538 -0.04015752  1.40866298 -0.20551204 -0.18188
997
   0.14094504 -0.36824188  1.15931887  0.98084098]]
UtX:
 [[ 2.58462061  1.50741694  0.05656989 -1.991973    0.29108998  0.2572
3712
  -0.19888549  0.52130422 -1.6392649  -1.38811538]]
```

3.2 sklearn method

In [17]:

```
1 np.asarray(X_norm.T)
```

executed in 7ms, finished 10:23:12 2022-11-13

Out[17]:

```
array([[-1.82764444, -1.82756108],
       [-1.0654041 , -1.06640538],
       [-0.03984438, -0.04015752],
       [ 1.40841225,  1.40866298],
       [-0.20615136, -0.20551204],
       [-0.18189826, -0.18188997],
       [ 0.14032152,  0.14094504],
       [-0.36899362, -0.36824188],
       [ 1.15895178,  1.15931887],
       [ 0.98225061,  0.98084098]])
```

In [18]:

```
1 pca = PCA(n_components=1)
2 new_X = pca.fit_transform(np.asarray(X_norm.T))
3 print(new_X.T)
```

executed in 8ms, finished 10:23:14 2022-11-13

```
[[ 2.58462061  1.50741694  0.05656989 -1.991973    0.29108998  0.25723
712
  -0.19888549  0.52130422 -1.6392649  -1.38811538]]
```

The two methods are equal:

In [22]:

```
1 np.round(new_X.T,3) == np.round(UtX,3)
```

executed in 8ms, finished 10:23:50 2022-11-13

Out[22]:

```
array([[ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True]])
```

3.3 what about 等式右侧?

3.3.1 也可以乘出pca

In [25]:

```
-sigma*V_t
```

executed in 8ms, finished 10:24:12 2022-11-13

Out[25]:

```
matrix([[ 2.58462061e+00,  1.50741694e+00,  5.65698902e-02,
         -1.99197300e+00,  2.91089983e-01,  2.57237122e-01,
         -1.98885488e-01,  5.21304217e-01, -1.63926490e+00,
         -1.38811538e+00],
        [-5.89388215e-05,  7.08012446e-04,  2.21426122e-04,
         -1.77291839e-04, -4.52070164e-04, -5.86594906e-06,
         -4.40896928e-04, -5.31558600e-04, -2.59573584e-04,
          9.96757317e-04]])
```

3.3.2 As for 1 compoent form

In [147]:

```
sigma_1 = np.diag([1,0]).dot(sigma)
print(sigma_1)
```

executed in 7ms, finished 09:29:19 2022-11-13

```
[[4.24264041 0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]]
```

In [148]:

```
-sigma_1*V_t
```

executed in 7ms, finished 09:29:32 2022-11-13

Out[148]:

```
matrix([[ 2.58462061,  1.50741694,  0.05656989, -1.991973  ,  0.291089
98,
          0.25723712, -0.19888549,  0.52130422, -1.6392649 , -1.388115
38],
        [ 0.        ,  0.        ,  0.        ,  0.        ,  0.
,
          0.        ,  0.        ,  0.        ,  0.        ,  0.
]])
```

Lecture notes上也是这么写的： 第一主成分就是第一右奇异向量用最大奇异值的平方加权：

In [157]:

```
s[0]*V_t[0]
```

executed in 9ms, finished 10:09:44 2022-11-13

Out[157]:

```
matrix([[-2.58462061, -1.50741694, -0.05656989,  1.991973  , -0.291089
98,
         -0.25723712,  0.19888549, -0.52130422,  1.6392649 ,  1.388115
38]])
```