eigenfaces-mypca

May 12, 2023

M = 5, N^2 = 9

[ ]:
```python
import numpy as np
X = np.random.randint(10, size=(5, 9))
print(X)
```

```
[[7 6 7 5 9 3 7 9 4]
 [6 6 3 3 8 1 9 7 3]
 [8 5 2 8 1 1 8 5 5]
 [3 0 6 9 5 1 8 9 2]
 [6 5 9 9 8 8 1 2 8]]
```

Matrix form of X

[ ]:
```python
print(X.T)
```

```
[[7 6 8 3 6]
 [6 6 5 0 5]
 [7 3 2 6 9]
 [5 3 8 9 9]
 [9 8 1 5 8]
 [3 1 1 1 8]
 [7 9 8 8 1]
 [9 7 5 9 2]
 [4 3 5 2 8]]
```

The Mean array: Mean of Xi in X

[ ]:
```python
phi = np.mean(X, axis = 0)
print(phi)
```

```
[6.  4.4 5.4 6.8 6.2 2.8 6.6 6.4 4.4]
```

New created X matrix with Ai = Xi - phi

[ ]:
```python
A = X - phi
A = A.T
print(A)
```

```
[[ 1.   0.   2.  -3.   0. ]
 [ 1.6  1.6  0.6 -4.4  0.6]
```

```
 [ 1.6 -2.4 -3.4  0.6  3.6]
 [-1.8 -3.8  1.2  2.2  2.2]
 [ 2.8  1.8 -5.2 -1.2  1.8]
 [ 0.2 -1.8 -1.8 -1.8  5.2]
 [ 0.4  2.4  1.4  1.4 -5.6]
 [ 2.6  0.6 -1.4  2.6 -4.4]
 [-0.4 -1.4  0.6 -2.4  3.6]]
```

Covaraince Matrix: M x M size since At = M x N^2 and A = N^2 x M and Cov = At x A

[ ]:
```
cov = A.T @ A
print(cov)
```

```
[[ 24.32  13.32 -22.88  -8.48  -6.28]
 [ 13.32  37.32   0.12  -7.48 -43.28]
 [-22.88   0.12  51.92  -1.68 -27.48]
 [ -8.48  -7.48  -1.68  52.72 -35.08]
 [ -6.28 -43.28 -27.48 -35.08 112.12]]
```

[ ]:
```
values1, vectors1 = np.linalg.eig(cov)
print(values1)
print()
print(vectors1)
```

```
[1.47660727e+02 4.53996194e-15 7.69412295e+00 6.68712060e+01
 5.61739441e+01]

[[-0.01443128 -0.4472136   0.70226399 -0.55295725 -0.0292457 ]
 [-0.3228464  -0.4472136  -0.63093265 -0.44767294  0.31190246]
 [-0.2410243  -0.4472136   0.23940211  0.64458993  0.51874631]
 [-0.28974997 -0.4472136  -0.11562837  0.26447449 -0.79544219]
 [ 0.86805195 -0.4472136  -0.19510509  0.09156576 -0.00596088]]
```

[ ]:
```
cov_d = A @ A.T
print(cov_d)
```

```
[[ 14.   16.   -7.   -6.   -4.    2.   -1.   -8.    8. ]
 [ 16.   25.2  -3.8 -16.6  10.6   7.4  -4.2  -9.8  10.2]
 [ -7.   -3.8  33.2  11.4  23.6  28.4 -29.2  -6.8  12.2]
 [ -6.  -16.6  11.4  28.8 -16.8  11.8 -17.4 -12.6   9.4]
 [ -4.   10.6  23.6 -16.8  42.8  18.2 -13.6   4.6   2.6]
 [  2.    7.4  28.4  11.8  18.2  36.8 -38.4 -25.6  24.4]
 [ -1.   -4.2 -29.2 -17.4 -13.6 -38.4  41.2  28.8 -26.2]
 [ -8.   -9.8  -6.8 -12.6   4.6 -25.6  28.8  35.2 -24.8]
 [  8.   10.2  12.2   9.4   2.6  24.4 -26.2 -24.8  21.2]]
```

5 eigenvectors of Cov_d which are At x Ei where Ei are the eigenvectors of Cov

```
[ ]: e1 = A @ vectors1[0]
     # print(e1)
     e2 = A @ vectors1[1]
     # print(e2)
     e3 = A @ vectors1[2]
     # print(e3)
     e4 = A @ vectors1[3]
     # print(e4)
     e5 = A @ vectors1[4]
     # print(e5)
     cov_d_eig = np.array([e1, e2, e3, e4, e5])
     print(cov_d_eig)
```

```
[[ 3.04896844  2.09819107 -1.77453384  1.28725827 -3.88625835  0.38126846
  -0.7062798  -2.59802684  2.27504282]
 [-0.24169288  0.54624682  3.55617448  1.22472095  2.67052733  4.30379802
  -4.45915276 -2.76074352  2.57394189]
 [-1.69598986 -3.48248726  2.12794725  4.97987767 -2.56450805  1.86307477
  -2.83711285 -1.83670425  1.18662092]
 [-1.31443018 -2.88947181 -1.70205807  0.91407865 -2.76418223 -3.65718794
   3.47364822  3.32778093 -2.82570866]
 [ 0.20314448  0.14981243  3.15903332  0.09112278  2.51949894  1.13396906
  -0.83766596  2.5260529  -0.07940181]]
```

EigenValues of Cov and Cov_d

```
[ ]: print(values1)
```

```
[1.47660727e+02 4.53996194e-15 7.69412295e+00 6.68712060e+01
 5.61739441e+01]
```

Let's select the Top k highest EigValues and corresponding EigVectors of Cov_d

```
[ ]: k = 3
     indices = np.argsort(values1)[-3 : ][::-1]
     print(f"The indices of the top {k} Eigenvalues are:")
     print(indices)
     print(f"\nThe corresponding top {k} Eigenvectors of Cov_d are:")
     print(cov_d_eig[indices])
```

```
The indices of the top 3 Eigenvalues are:
[0 3 4]

The corresponding top 3 Eigenvectors of Cov_d are:
[[ 3.04896844  2.09819107 -1.77453384  1.28725827 -3.88625835  0.38126846
  -0.7062798  -2.59802684  2.27504282]
 [-1.31443018 -2.88947181 -1.70205807  0.91407865 -2.76418223 -3.65718794
   3.47364822  3.32778093 -2.82570866]
 [ 0.20314448  0.14981243  3.15903332  0.09112278  2.51949894  1.13396906
```

```
  -0.83766596  2.5260529  -0.07940181]]
```

These top K Eigenvectors are called EIGENFACES

Now we want to represent each Ai which is Xi - phi as the linear combination of each of the Top k eigenvectors as Ai = Xi - phi = (wi,1 x E1) + (wi,2 x E2) + …. + (wi,k x Ek)

0.0.1 Let's see PCA

[ ]:
```python
from sklearn.decomposition import PCA
from sklearn.datasets import load_wine
from sklearn.preprocessing import StandardScaler
wine = load_wine()
X, target = wine.data, wine.target
print(X.shape)
print(X)
```

```
(178, 13)
[[1.423e+01 1.710e+00 2.430e+00 … 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 … 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 … 1.030e+00 3.170e+00 1.185e+03]
 …
 [1.327e+01 4.280e+00 2.260e+00 … 5.900e-01 1.560e+00 8.350e+02]
 [1.317e+01 2.590e+00 2.370e+00 … 6.000e-01 1.620e+00 8.400e+02]
 [1.413e+01 4.100e+00 2.740e+00 … 6.100e-01 1.600e+00 5.600e+02]]
```

0.0.2 Scaling using StandardScaler

[ ]:
```python
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
print(X_scaled)
```

```
[[ 1.51861254 -0.5622498   0.23205254 …  0.36217728  1.84791957
   1.01300893]
 [ 0.24628963 -0.49941338 -0.82799632 …  0.40605066  1.1134493
   0.96524152]
 [ 0.19687903  0.02123125  1.10933436 …  0.31830389  0.78858745
   1.39514818]
 …
 [ 0.33275817  1.74474449 -0.38935541 … -1.61212515 -1.48544548
   0.28057537]
 [ 0.20923168  0.22769377  0.01273209 … -1.56825176 -1.40069891
   0.29649784]
 [ 1.39508604  1.58316512  1.36520822 … -1.52437837 -1.42894777
  -0.59516041]]
```

0.0.3 Standard Scaler working: replace each Xi in X with (Xi - mean)/(std), where mean = sum(Xi)/N (it is itself an array like Xi) and std = root(sum(Xi - mean)/N)

[ ]:

```
means = np.mean(X, axis = 0)
stds = np.std(X, axis = 0)
My_X_scaled = (X - means)/stds
print(My_X_scaled)
```

```
[[ 1.51861254 -0.5622498   0.23205254 …  0.36217728  1.84791957
   1.01300893]
 [ 0.24628963 -0.49941338 -0.82799632 …  0.40605066  1.1134493
   0.96524152]
 [ 0.19687903  0.02123125  1.10933436 …  0.31830389  0.78858745
   1.39514818]
 …
 [ 0.33275817  1.74474449 -0.38935541 … -1.61212515 -1.48544548
   0.28057537]
 [ 0.20923168  0.22769377  0.01273209 … -1.56825176 -1.40069891
   0.29649784]
 [ 1.39508604  1.58316512  1.36520822 … -1.52437837 -1.42894777
  -0.59516041]]
```

0.0.4 Covariance Matrix Cov = (A x At)/(M - 1)

[ ]:

```
cov = np.cov(My_X_scaled.T)
print(cov)
print()
Mycov = (My_X_scaled.T @ My_X_scaled)/(My_X_scaled.shape[0] - 1)
print(Mycov)
```

```
[[ 1.00564972  0.09493026  0.21273976 -0.31198788  0.27232816  0.29073446
   0.23815287 -0.15681042  0.13747022  0.549451   -0.07215255  0.07275191
   0.64735687]
 [ 0.09493026  1.00564972  0.16497228  0.29013035 -0.05488343 -0.3370606
  -0.41332866  0.29463237 -0.22199334  0.25039204 -0.56446685 -0.37079354
  -0.19309537]
 [ 0.21273976  0.16497228  1.00564972  0.44587209  0.28820583  0.12970824
   0.11572743  0.1872826   0.00970647  0.2603499  -0.07508874  0.00393333
   0.22488969]
 [-0.31198788  0.29013035  0.44587209  1.00564972 -0.0838039  -0.32292752
  -0.353355    0.36396647 -0.19844168  0.01883781 -0.27550299 -0.27833221
  -0.44308618]
 [ 0.27232816 -0.05488343  0.28820583 -0.0838039   1.00564972  0.21561254
   0.19688989 -0.25774204  0.23777643  0.20107967  0.05571118  0.06637684
   0.39557317]
 [ 0.29073446 -0.3370606   0.12970824 -0.32292752  0.21561254  1.00564972
   0.86944804 -0.45247731  0.61587304 -0.05544792  0.43613151  0.70390388
```

```
  0.50092909]
 [ 0.23815287 -0.41332866  0.11572743 -0.353355    0.19688989  0.86944804
   1.00564972 -0.54093859  0.65637929 -0.17335329  0.54654907  0.79164133
   0.49698518]
 [-0.15681042  0.29463237  0.1872826   0.36396647 -0.25774204 -0.45247731
  -0.54093859  1.00564972 -0.36791202  0.13984265 -0.26412347 -0.50611293
  -0.31314443]
 [ 0.13747022 -0.22199334  0.00970647 -0.19844168  0.23777643  0.61587304
   0.65637929 -0.36791202  1.00564972 -0.02539259  0.29721399  0.52199968
   0.33228346]
 [ 0.549451    0.25039204  0.2603499   0.01883781  0.20107967 -0.05544792
  -0.17335329  0.13984265 -0.02539259  1.00564972 -0.52476129 -0.43123763
   0.31788599]
 [-0.07215255 -0.56446685 -0.07508874 -0.27550299  0.05571118  0.43613151
   0.54654907 -0.26412347  0.29721399 -0.52476129  1.00564972  0.56866303
   0.23751782]
 [ 0.07275191 -0.37079354  0.00393333 -0.27833221  0.06637684  0.70390388
   0.79164133 -0.50611293  0.52199968 -0.43123763  0.56866303  1.00564972
   0.31452809]
 [ 0.64735687 -0.19309537  0.22488969 -0.44308618  0.39557317  0.50092909
   0.49698518 -0.31314443  0.33228346  0.31788599  0.23751782  0.31452809
   1.00564972]]

[[ 1.00564972  0.09493026  0.21273976 -0.31198788  0.27232816  0.29073446
   0.23815287 -0.15681042  0.13747022  0.549451   -0.07215255  0.07275191
   0.64735687]
 [ 0.09493026  1.00564972  0.16497228  0.29013035 -0.05488343 -0.3370606
  -0.41332866  0.29463237 -0.22199334  0.25039204 -0.56446685 -0.37079354
  -0.19309537]
 [ 0.21273976  0.16497228  1.00564972  0.44587209  0.28820583  0.12970824
   0.11572743  0.1872826   0.00970647  0.2603499  -0.07508874  0.00393333
   0.22488969]
 [-0.31198788  0.29013035  0.44587209  1.00564972 -0.0838039  -0.32292752
  -0.353355    0.36396647 -0.19844168  0.01883781 -0.27550299 -0.27833221
  -0.44308618]
 [ 0.27232816 -0.05488343  0.28820583 -0.0838039   1.00564972  0.21561254
   0.19688989 -0.25774204  0.23777643  0.20107967  0.05571118  0.06637684
   0.39557317]
 [ 0.29073446 -0.3370606   0.12970824 -0.32292752  0.21561254  1.00564972
   0.86944804 -0.45247731  0.61587304 -0.05544792  0.43613151  0.70390388
   0.50092909]
 [ 0.23815287 -0.41332866  0.11572743 -0.353355    0.19688989  0.86944804
   1.00564972 -0.54093859  0.65637929 -0.17335329  0.54654907  0.79164133
   0.49698518]
 [-0.15681042  0.29463237  0.1872826   0.36396647 -0.25774204 -0.45247731
  -0.54093859  1.00564972 -0.36791202  0.13984265 -0.26412347 -0.50611293
  -0.31314443]
 [ 0.13747022 -0.22199334  0.00970647 -0.19844168  0.23777643  0.61587304
```

```
  0.65637929 -0.36791202  1.00564972 -0.02539259  0.29721399  0.52199968
  0.33228346]
 [ 0.549451    0.25039204  0.2603499   0.01883781  0.20107967 -0.05544792
 -0.17335329  0.13984265 -0.02539259  1.00564972 -0.52476129 -0.43123763
  0.31788599]
 [-0.07215255 -0.56446685 -0.07508874 -0.27550299  0.05571118  0.43613151
  0.54654907 -0.26412347  0.29721399 -0.52476129  1.00564972  0.56866303
  0.23751782]
 [ 0.07275191 -0.37079354  0.00393333 -0.27833221  0.06637684  0.70390388
  0.79164133 -0.50611293  0.52199968 -0.43123763  0.56866303  1.00564972
  0.31452809]
 [ 0.64735687 -0.19309537  0.22488969 -0.44308618  0.39557317  0.50092909
  0.49698518 -0.31314443  0.33228346  0.31788599  0.23751782  0.31452809
  1.00564972]]
```

0.0.5 EigenValues and EigenVectors

[ ]:
```
values, vectors = np.linalg.eig(cov)
print(values)
print()
max_abs_idx = np.argmax(np.abs(vectors), axis=0)
signs = np.sign(vectors[max_abs_idx, range(vectors.shape[0])])
vectors = vectors*signs[np.newaxis,:]
vectors = vectors.T
print(vectors)
```

```
[4.73243698 2.51108093 1.45424187 0.92416587 0.85804868 0.64528221
 0.55414147 0.10396199 0.35046627 0.16972374 0.29051203 0.22706428
 0.25232001]

[[ 0.1443294  -0.24518758 -0.00205106 -0.23932041  0.14199204  0.39466085
   0.4229343  -0.2985331   0.31342949 -0.0886167   0.29671456  0.37616741
   0.28675223]
 [ 0.48365155  0.22493093  0.31606881 -0.0105905   0.299634    0.06503951
  -0.00335981  0.02877949  0.03930172  0.52999567 -0.27923515 -0.16449619
   0.36490283]
 [-0.20738262  0.08901289  0.6262239   0.61208035  0.13075693  0.14617896
   0.1506819   0.17036816  0.14945431 -0.13730621  0.08522192  0.16600459
  -0.12674592]
 [-0.0178563   0.53689028 -0.21417556  0.06085941 -0.35179658  0.19806835
   0.15229479 -0.20330102  0.39905653  0.06592568 -0.42777141  0.18412074
  -0.23207086]
 [-0.26566365  0.03521363 -0.14302547  0.06610294  0.72704851 -0.14931841
  -0.10902584 -0.50070298  0.13685982 -0.07643678 -0.17361452 -0.10116099
  -0.1578688 ]
 [ 0.21353865  0.53681385  0.15447466 -0.10082451  0.03814394 -0.0841223
  -0.01892002 -0.25859401 -0.53379539 -0.41864414  0.10598274  0.26585107
   0.11972557]
```

```
[-0.05639636  0.42052391 -0.14917061 -0.28696914  0.3228833  -0.02792498
 -0.06068521  0.59544729  0.37213935 -0.22771214  0.23207564 -0.0447637
  0.0768045 ]
[ 0.01496997  0.02596375 -0.14121803  0.09168285  0.05677422 -0.46390791
  0.83225706  0.11403985 -0.11691707 -0.0119928  -0.08988884 -0.15671813
  0.01444734]
[ 0.39613926  0.06582674 -0.17026002  0.42797018 -0.15636143 -0.40593409
 -0.18724536 -0.23328465  0.36822675 -0.03379692  0.43662362 -0.07810789
  0.12002267]
[-0.26628645  0.12169604 -0.04962237 -0.05574287  0.06222011 -0.30388245
 -0.04289883  0.04235219 -0.09555303  0.60422163  0.259214    0.60095872
 -0.07940162]
[-0.50861912  0.07528304  0.30769445 -0.20044931 -0.27140257 -0.28603452
 -0.04957849 -0.19550132  0.20914487 -0.05621752 -0.08582839 -0.1372269
  0.57578611]
[-0.22591696  0.07648554 -0.49869142  0.47931378  0.07128891  0.30434119
 -0.02569409  0.11689586 -0.23736257  0.0318388  -0.04821201  0.0464233
  0.53926983]
[ 0.21160473 -0.30907994 -0.02712539  0.05279942  0.06787022 -0.32013135
 -0.16315051  0.21553507  0.1341839  -0.29077518 -0.52239889  0.52370587
  0.162116  ]]
```

[ ]:
```
indices = np.argsort(values)[::-1]
eig = vectors[indices]
print(eig)
```

```
[[ 0.1443294  -0.24518758 -0.00205106 -0.23932041  0.14199204  0.39466085
   0.4229343  -0.2985331   0.31342949 -0.0886167   0.29671456  0.37616741
   0.28675223]
 [ 0.48365155  0.22493093  0.31606881 -0.0105905   0.299634    0.06503951
  -0.00335981  0.02877949  0.03930172  0.52999567 -0.27923515 -0.16449619
   0.36490283]
 [-0.20738262  0.08901289  0.6262239   0.61208035  0.13075693  0.14617896
   0.1506819   0.17036816  0.14945431 -0.13730621  0.08522192  0.16600459
  -0.12674592]
 [-0.0178563   0.53689028 -0.21417556  0.06085941 -0.35179658  0.19806835
   0.15229479 -0.20330102  0.39905653  0.06592568 -0.42777141  0.18412074
  -0.23207086]
 [-0.26566365  0.03521363 -0.14302547  0.06610294  0.72704851 -0.14931841
  -0.10902584 -0.50070298  0.13685982 -0.07643678 -0.17361452 -0.10116099
  -0.1578688 ]
 [ 0.21353865  0.53681385  0.15447466 -0.10082451  0.03814394 -0.0841223
  -0.01892002 -0.25859401 -0.53379539 -0.41864414  0.10598274  0.26585107
   0.11972557]
 [-0.05639636  0.42052391 -0.14917061 -0.28696914  0.3228833  -0.02792498
  -0.06068521  0.59544729  0.37213935 -0.22771214  0.23207564 -0.0447637
   0.0768045 ]
 [ 0.39613926  0.06582674 -0.17026002  0.42797018 -0.15636143 -0.40593409
```

```
 -0.18724536 -0.23328465  0.36822675 -0.03379692  0.43662362 -0.07810789
  0.12002267]
 [-0.50861912  0.07528304  0.30769445 -0.20044931 -0.27140257 -0.28603452
  -0.04957849 -0.19550132  0.20914487 -0.05621752 -0.08582839 -0.1372269
   0.57578611]
 [ 0.21160473 -0.30907994 -0.02712539  0.05279942  0.06787022 -0.32013135
  -0.16315051  0.21553507  0.1341839  -0.29077518 -0.52239889  0.52370587
   0.162116  ]
 [-0.22591696  0.07648554 -0.49869142  0.47931378  0.07128891  0.30434119
  -0.02569409  0.11689586 -0.23736257  0.0318388  -0.04821201  0.0464233
   0.53926983]
 [-0.26628645  0.12169604 -0.04962237 -0.05574287  0.06222011 -0.30388245
  -0.04289883  0.04235219 -0.09555303  0.60422163  0.259214    0.60095872
  -0.07940162]
 [ 0.01496997  0.02596375 -0.14121803  0.09168285  0.05677422 -0.46390791
   0.83225706  0.11403985 -0.11691707 -0.0119928  -0.08988884 -0.15671813
   0.01444734]]
```

0.0.6 Select the top K eigenvectors and Final Projection

[ ]:
```python
k = 2
W = eig[:k]
X_proj = My_X_scaled @ W.T
print(X_proj.shape)
print(X_proj[:10])
```

```
(178, 2)
[[ 3.31675081  1.44346263]
 [ 2.20946492 -0.33339289]
 [ 2.51674015  1.0311513 ]
 [ 3.75706561  2.75637191]
 [ 1.00890849  0.86983082]
 [ 3.05025392  2.12240111]
 [ 2.44908967  1.17485013]
 [ 2.05943687  1.60896307]
 [ 2.5108743   0.91807096]
 [ 2.75362819  0.78943767]]
```

0.0.7 Built-in PCA

[ ]:
```python
model = PCA(n_components = k)
X_2D = model.fit_transform(X_scaled)
print(X_2D[:10])
```

```
[[ 3.31675081 -1.44346263]
 [ 2.20946492  0.33339289]
 [ 2.51674015 -1.0311513 ]
 [ 3.75706561 -2.75637191]
```

```
 [ 1.00890849 -0.86983082]
 [ 3.05025392 -2.12240111]
 [ 2.44908967 -1.17485013]
 [ 2.05943687 -1.60896307]
 [ 2.5108743  -0.91807096]
 [ 2.75362819 -0.78943767]]
```

1 YES !!!! SAME RESULT FROM BUILT-IN AND MY OWN PCA!!