CSE M146 HW-8

Gawun Kim 305-186-572.

① dimensionality for space : M

dimensional subspace : M+1 (from eigen vector with additional u_{M+1})

Constrain u_{M+1} to make it not relate eigenvector ($u_1 \sim u_M$)

For this, proving the orthogonal between u_{M+1} and ($u_1 \sim u_M$), by Lagrange multiplier η_i $i \in 1 \sim M$

The variance in the direction u_{M+1} can be $u_{M+1}^T S u_{M+1}$.

When maximizing $u_{M+1}^T S u_{M+1}$ by using a Lagrange Multiplier λ_{M+1} for the normalization constraint $u_{M+1}^T u_{M+1} = 1$

$u_{M+1}^T S u_{M+1} + \lambda_{M+1}(1 - u_{M+1}^T u_{M+1}) + \sum_{i=1}^{M} \eta_i u_{M+1}^T u_i$ and maximize this function with respect to u_{M+1}

$0 = 2S u_{M+1} - 2\lambda_{M+1} u_{M+1} + \sum_{i=1}^{M} \eta_i u_i$ for stationary points.

By left multiplying with u_i^T with orthogonal constraint, $\eta_i = 0$ from 1 to M. So that $S u_{M+1} = \lambda_{M+1} u_{M+1}$.

u_{M+1} is an eigenvector of S with eigen value u_{M+1}.

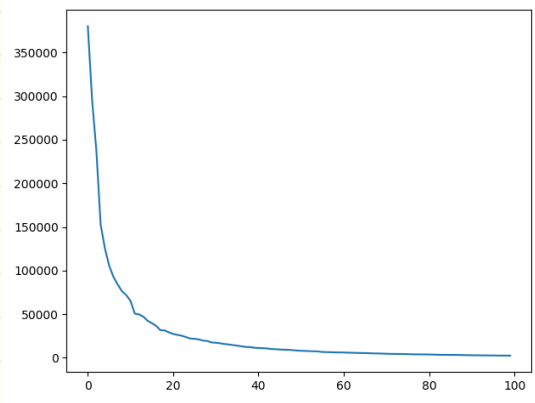
The variance in the u_{M+1} is an eigenvector to have the largest eigenvalue among the unselected previously.

When the result is for $M \leq D$, the result explicitly is M+1. and Also, based on the explained steps, the result holds for projection spaces of M+1 dimensionality. ← induction proved.

②

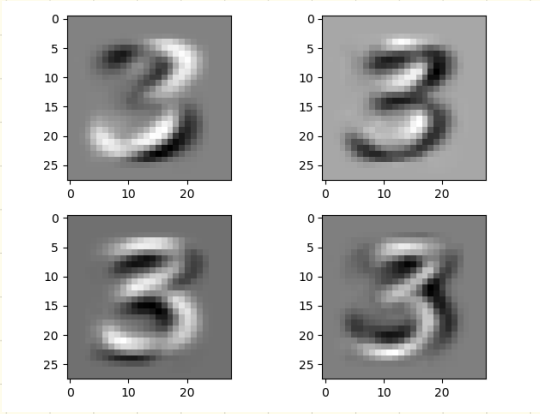
ⓐ

```
import numpy as np
from numpy import linalg as eg
import matplotlib.pyplot as plt
import matplotlib.cm as cm

fp =  np.genfromtxt('MNIST3.csv', delimiter=',') # fp, the shape
is (400, 784)
T_fp = np.transpose(fp)
cv = np.cov(T_fp)
E_value, E_vector = eg.eig(cv)
temp = sorted(E_value[:], reverse=True)
temp = temp[:100]  # save the 100 largest eigenvalues from
data
plt.plot(temp)
plt.show()
```



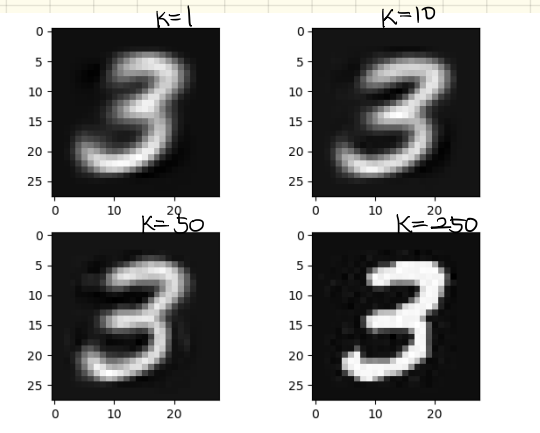
ⓑ

```
import numpy as np
from numpy import linalg as eg
import matplotlib.pyplot as plt

fp =  np.genfromtxt('MNIST3.csv', delimiter=',') # fp, the shape is (400, 784)
T_fp = np.transpose(fp)
cv = np.cov(T_fp)
E_value, E_vector = eg.eig(cv)

temp_vector = []
for i in range(0,4):
    temp_vector.append(E_vector[:, i])

fig, ax = plt.subplots(2, 2)
temp_vector[0] = np.transpose(np.array(temp_vector[0].reshape(28, 28),
dtype=float))
ax[0, 0].imshow(temp_vector[0], cmap='gray')
temp_vector[1] = np.transpose(np.array(temp_vector[1].reshape(28, 28),
dtype=float))
ax[1, 0].imshow(temp_vector[1], cmap='gray')
temp_vector[2] = np.transpose(np.array(temp_vector[2].reshape(28, 28),
dtype=float))
ax[0, 1].imshow(temp_vector[2], cmap='gray')
temp_vector[3] = np.transpose(np.array(temp_vector[3].reshape(28, 28),
dtype=float))
ax[1, 1].imshow(temp_vector[3], cmap='gray')
plt.show()
```



There are four figures and they look like four "3" is given in the front page. as the image

(c)



When M value is getting bigger, the image (plot) quality is increasing. When M=0, there is no difference with M=1.

```
import numpy as np
from numpy import linalg as eg
import matplotlib.pyplot as plt
import statistics

fp =  np.genfromtxt('MNIST3.csv', delimiter=',') # fp, the shape is (400, 784)
T_fp = np.transpose(fp)
cv = np.cov(T_fp)
E_value, E_vector = eg.eig(cv)

temp_vector = []
for i in range(0,400):
    temp_vector.append(E_vector[:, i])
temp_vector = np.array(temp_vector)


MEAN_Vec_X = []
for i in range(784):
    TEMP = 0
    for j in range(400):
        TEMP = TEMP + T_fp[i][j]
    MEAN_Vec_X.append(float(TEMP/400))

X1 = fp[0]
MEAN_Vec_X = np.array(MEAN_Vec_X)
M = 1
QQ = np.array(np.zeros(784))
for m in range(0,M):
    XX = np.dot(X1, temp_vector[m])
    YY = np.dot(MEAN_Vec_X, temp_vector[m])
    ZZ = (XX-YY)
    QQ = QQ + ZZ*temp_vector[m]
QQ = MEAN_Vec_X + QQ
print(QQ.shape)
plotting = np.transpose(np.array(QQ.reshape(28, 28), dtype=float))
plt.subplot(2,2,1)
plt.imshow(plotting, cmap='gray')


M = 10
QQ = np.array(np.zeros(784))
for m in range(0,M):
    XX = np.dot(X1, temp_vector[m])
    YY = np.dot(MEAN_Vec_X, temp_vector[m])
    ZZ = (XX-YY)
    QQ = QQ + ZZ*temp_vector[m]
QQ = MEAN_Vec_X + QQ
print(QQ.shape)
plotting = np.transpose(np.array(QQ.reshape(28, 28), dtype=float))
plt.subplot(2,2,2)
plt.imshow(plotting, cmap='gray')


M = 50
QQ = np.array(np.zeros(784))
for m in range(0,M):
    XX = np.dot(X1, temp_vector[m])
    YY = np.dot(MEAN_Vec_X, temp_vector[m])
    ZZ = (XX-YY)
    QQ = QQ + ZZ*temp_vector[m]
QQ = MEAN_Vec_X + QQ
print(QQ.shape)
plotting = np.transpose(np.array(QQ.reshape(28, 28), dtype=float))
plt.subplot(2,2,3)
plt.imshow(plotting, cmap='gray')


M = 250
QQ = np.array(np.zeros(784))
for m in range(0,M):
    XX = np.dot(X1, temp_vector[m])
    YY = np.dot(MEAN_Vec_X, temp_vector[m])
    ZZ = (XX-YY)
    QQ = QQ + ZZ*temp_vector[m]
QQ = MEAN_Vec_X + QQ
print(QQ.shape)
plotting = np.transpose(np.array(QQ.reshape(28, 28), dtype=float))
plt.subplot(2,2,4)
plt.imshow(plotting, cmap='gray')
plt.show()
```

③ A symmetric real Matrix M → positive. def

When $z^T M z$ is positive with non-zero column vec z

ⓐ $A=\begin{bmatrix}9 & 6\\ 6 & a\end{bmatrix}$ Let z is $[x\ Y]$

$$z^T A z = [x\ Y]\begin{bmatrix}9 & 6\\ 6 & a\end{bmatrix}\begin{bmatrix}x\\ Y\end{bmatrix} = \begin{bmatrix}9x+6Y & 6x+aY\end{bmatrix}\begin{bmatrix}x\\ Y\end{bmatrix}$$

$$= x(9x+6Y)+Y(6x+aY)$$

$$= 9x^2+6xY+6xY+aY^2$$

To satisfy the above function to be positive, a should be bigger than 4

then → $9x^2+6xY+6xY+4Y^2+a_2Y^2$ since a is bigger than 4

$= (3x+2Y)^2+(a_2Y^2)$ ← it shows always positive.

∴ When $a>4$, matrix A is positive definite.

ⓑ B is positive definite, B is real ※ Matrix and symmetric.

Let $B=\begin{bmatrix}a & c\\ c & b\end{bmatrix}$ and u is $\begin{bmatrix}x\\ Y\end{bmatrix}$, then $u^TBu=(ax+bY)^2$

B is positive definite and, to be that, u^TBu is positive ; it can be expressed $(a_1x+b_1Y)^2+a_2x^2+b_2Y^2$ $a_1, a_2>0$

$a=a_1+a_2$ $b=b_1+b_2$ $b_1, b_2>0$

$B^{-1}=\frac{1}{a^2b^2-c^2}\begin{bmatrix}b & -c\\ -c & a\end{bmatrix}$ in that result, $\frac{1}{a^2b^2-c^2}$ is constant

$$u^TB^{-1}u = [x\ Y]\left(\frac{1}{a^2b^2-c^2}\right)\begin{bmatrix}b^2-c\\ -c\ a^2\end{bmatrix}\begin{bmatrix}x\\ Y\end{bmatrix}$$

$$= \frac{1}{a^2b^2-c^2}\begin{bmatrix}b^2x-cY & -cx+a^2Y\end{bmatrix}\begin{bmatrix}x\\ Y\end{bmatrix}$$

$$= \frac{1}{a^2b^2-c^2}\left(b^2x^2-cxY-cxY+a^2Y\right)$$

$$= \frac{1}{a^2b^2-c^2}\left[(bx-aY)^2+b_2x^2+a_2Y^2\right]$$ and it is always positive.

So that B^{-1} is also positive definite

Or to prove differently, $y=Bx$, $y^TB^{-1}y = x^TB^TB^{-1}Bx$

$= x^TBx$. Since x^TBx is positive, so $y^TB^{-1}y$ is positive.

It means that B^{-1} is also positive definite.

④ N balls in a jar, each ball is labeled from 1 to N each.

ⓐ The probability is zero. Since pick each ball will end up being done when there is no ball anymore. (no replacement) It means that the ball 1 will be picked up in one of N.

ⓑ $\left(\frac{N-1}{N}\right)^N$

ⓒ $\lim_{N\to\infty}\left(\frac{N-1}{N}\right)^N$ → $\lim_{N\to\infty} N(\ln(N-1)-\ln(N))$

$$= \lim_{N\to\infty}\frac{(\ln(N-1)-\ln(N))}{1/N}$$

$$= \lim_{N\to\infty}\frac{1/(N-1)-1/N}{-1/N^2} = \lim_{N\to\infty}\frac{1/(N^2-N)}{-1/N^2}$$

$$= \lim_{N\to\infty}\frac{-N^2}{N^2-N} = -1$$

So that $\ln(\alpha)=-1$ and $\alpha=\frac{1}{e}$

Which means that the probability that I got in ⓒ approaches $\frac{1}{e}$ when $N\to\infty$

⑤ Constraint $\sum_{k=1}^{K}\pi_k=1$

By Lagrange multiplier and maximizing the below quantity

$\ln P(X|\pi,\mu,\Sigma)+\lambda\left(\sum_{k=1}^{K}\pi_k-1\right)$

which gives $0=\sum_{n=1}^{N}\frac{\mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x_n|\mu_j,\Sigma_j)}+\lambda$

$$\gamma(z_{nk}) = \sum_{n=1}^{N}\frac{\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x_n|\mu_j,\Sigma_j)}$$

$$= \pi_k\left[\sum_{n=1}^{N}\frac{\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x_n|\mu_j,\Sigma_j)}\right]$$

$$0=\sum_{n=1}^{N}\frac{\mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x_n|\mu_j,\Sigma_j)}+\lambda$$

$$= \sum_{n=1}^{N}\frac{1}{\pi_k}\gamma(z_{nk})+\lambda$$

$$= \frac{1}{\pi_k}\cdot N_k+\lambda$$

By the given constraint $\lambda=-N$

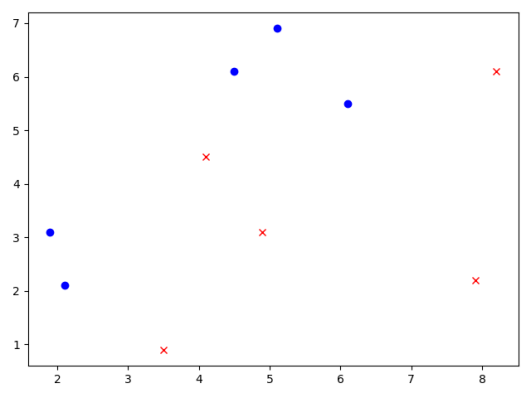
So that $\frac{1}{\pi_k}\cdot N_k+\lambda=0=\frac{1}{\pi_k}N_k-N$

$N=N_k\cdot\frac{1}{\pi_k}$ → $\pi_k=\frac{N_k}{N}$

6

@

```
import numpy as np
import matplotlib.pyplot as plt
ada = np.genfromtxt('AdaBoost_data.csv', delimiter=',')
for index in ada:
    if index[2] == 1:
        plt.plot(index[0],index[1],'o', color='blue')
    else:
        plt.plot(index[0],index[1],'x', color='red')
plt.show()
```



It is not linearly separable. This data set cannot be divided with single layer decision tree.