since the solution of A is $[u_1, \cdots, u_k]$. where u_i is eigen-
-vectors.

$$\sum_{i=1}^{n} \text{trace}\left(AA^T \tilde{x}_i \tilde{x}_i^T\right) = \sum_{i=1}^{n} \text{trace}\left(A^T \tilde{x}_i \tilde{x}_i^T A\right)$$

$$= \sum_{i=1}^{n} \text{trace}\left(\begin{bmatrix} u_1 \\ \vdots \\ u_k \end{bmatrix} \tilde{x}_i \tilde{x}_i^T [u_1, \cdots, u_k]\right)$$

$$= \cancel{n\,\text{trace}}\; n \cdot \frac{1}{n} \sum_{i=1}^{n} \text{trace}\left(\begin{bmatrix} u_1 \\ \vdots \\ u_k \end{bmatrix} \tilde{x}_i \tilde{x}_i^T [u_1 \cdots u_k]\right) = n \cdot \text{trace}\left(\begin{bmatrix} u_1 \\ \vdots \\ u_k \end{bmatrix} U \Lambda U^T [u_1 \cdots u_k]\right)$$

$$= n \cdot \sum_{j=1}^{k} \lambda_j$$

Thus. $J^* = n \sum_{j=1}^{d} \lambda_j - n \sum_{j=1}^{k} \lambda_j = n \sum_{j=k+1}^{d} \lambda_j = \min_{\vec{\mu}, A, \{\vec{\theta}_i\}} \text{Obj.fun}$

3) EM Algorithm for Mixed Linear Regression

$$f(y|\vec{x};\vec{\theta}) \sim \sum_{k=1}^{K} \varepsilon_k \phi(y; \vec{w}_k^T\vec{x}+b_k, \sigma_k^2)$$

where $\vec{\theta} = (\varepsilon_1, \cdots, \varepsilon_K, \vec{w}_1, \cdots, \vec{w}_K, b_1, \cdots, b_K, \sigma_1^2, \cdots, \sigma_K^2)$

a. $\underline{\vec{x}} = (\vec{x}_1, \cdots, \vec{x}_n)$ $\quad y = (y_1 \cdots, y_n)$

$$\begin{aligned} l(\vec{\theta}; y|\underline{\vec{x}}) &= \log f(y|\underline{\vec{x}};\vec{\theta}) \\ &= \log \prod_{i=1}^{n} f(y_i|\vec{x}_i;\vec{\theta}) \\ &= \sum_{i=1}^{n} \log f(y_i|\vec{x}_i;\vec{\theta}) \\ &= \sum_{i=1}^{n} \log \sum_{k=1}^{K} \varepsilon_k \phi(y_i; \vec{w}_k^T\vec{x}_i + b_k, \sigma_k^2) \end{aligned}$$

Introduce a hidden variable $S=(S_1, \cdots, S_n)$ for each data $\underline{\vec{x}} = (\vec{x}_1, \cdots, \vec{x}_n)$

Then complete data is $\underline{\vec{z}} = (\underline{\vec{x}}, \underline{S})$

S_i: describe the component responsible for generating $\vec{x}_i$

complete log-likelihood:

$$\begin{aligned} l(\vec{\theta}; y, \underline{S}|\vec{x}) &= \log L(\theta; y, \underline{S}|\vec{x}) \\ &= \log \prod_{i=1}^{n} \mathbb{P}(y_i, s_i|\vec{x}_i;\vec{\theta}) \\ &= \log \prod_{i=1}^{n} \mathbb{P}(S_i=s_i) \cdot f(y_i|\vec{x}_i, S_i=s_i;\vec{\theta}) \\ &= \sum_{i=1}^{n} \log \mathbb{P}(S_i=s_i;\vec{\theta}) \cdot f(y_i|\vec{x}_i, S_i=s_i;\vec{\theta}) \\ &= \sum_{i=1}^{n} \log \varepsilon_{s_i} \phi(y_i; \vec{w}_{s_i}^T\vec{x}_i + b_{s_i}, \sigma_{s_i}^2) \end{aligned}$$

Define $\Delta_{ik} = \begin{cases} 1 & S_i = k \\ 0 & \text{otherwise}. \end{cases}$ Then

$$\begin{aligned} l(\vec{\theta}; y|\underline{\vec{z}}) &= \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \Delta_{ik} \cdot \varepsilon_k \phi(y_i; \vec{w}_k^T\vec{x}_i + b_k, \sigma_k^2)\right) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \Delta_{ik} \log(\varepsilon_k \phi(y_i; \vec{w}_k^T\vec{x}_i + b_k, \sigma_k^2)) \end{aligned}$$

E-step:

$$Q(\theta,\theta^{(j)}) = \mathbb{E}_{S|y,\vec{x}}[l(\vec{\theta}; y|\vec{x}, S) \mid y, \vec{x}; \theta^{(j)}]$$

$$= \mathbb{E}_S[\sum_{i=1}^n \sum_{k=1}^K \Delta_{ik} (\log \varepsilon_k + \log \phi(y_i; \vec{w}_k^T \vec{x}_i + b_k, \sigma_k^2))]$$

$$= \sum_{i=1}^n \sum_{k=1}^K [\mathbb{E}_S[\Delta_{ik}] \log \varepsilon_k \phi(y_i; \vec{w}_k^T \vec{x}_i + b_k, \sigma_k^2)]$$

Suppose $\gamma_{ik}^{(j)} = \mathbb{E}_S[\Delta_{ik}] = \mathbb{E}[\Delta_{ik} \mid y_i, \vec{x}_i; \theta^{(j)}]$

$$= \mathbb{P}(S_i = k \mid x_i, y_i; \theta^{(j)}]$$

Bayes Rule $\rightarrow$ $$= \frac{\mathbb{P}(S_i=k;\theta^{(j)}) \cdot f(y_i, \vec{x}_i \mid S_i = k; \theta^{(j)})}{f(y_i, \vec{x}_i; \theta^{(j)})}$$

$$= \frac{\varepsilon_k^{(j)} \phi(y_i; \vec{w}_k^{(j)T} \vec{x}_i + b_k^{(j)}, \sigma_k^{(j)2})}{\sum_{l=1}^K \varepsilon_l^{(j)} \phi(y_i; \vec{w}_l^{(j)T} \vec{x}_i + b_l^{(j)}, \sigma_l^{(j)2})}$$

b. M-step:

$$\theta^{(j+1)} = \arg\max_\theta Q(\theta, \theta^{(j)})$$

where $$Q(\theta,\theta^{(j)}) = \sum_{i=1}^n \sum_{k=1}^K \gamma_{ik}^{(j)} [\log \varepsilon_k + \log((2\pi^{-\frac{d}{2}})/\sigma_k \cdot e^{-\frac{1}{2\sigma_k^2}\|y_i - \vec{w}_k^T \vec{x}_i - b_k\|^2}]$$

$$= \sum_{i=1}^n \sum_{k=1}^K \gamma_{ik}^{(j)} [\log \varepsilon_k - \frac{d}{2}\log(2\pi) - \frac{1}{2\sigma_k^2}\|y_i - \vec{w}_k^T \vec{x}_i - b_k\|^2 - \frac{1}{2}\log \sigma_k^2]$$

First optimize ε_k: it has constraints $\sum_{k=1}^K \varepsilon_k = 1$. by Lagrange multiplier theory: $L(\varepsilon_1, \cdots, \varepsilon_K) = \sum_{i=1}^n \sum_{k=1}^K \gamma_{ik}^{(j)} \log \varepsilon_k + \lambda(\sum_{k=1}^K \varepsilon_k - 1)$

$$\frac{\partial L}{\partial \varepsilon_k} = \sum_{i=1}^n \gamma_{ik}^{(j)}/\varepsilon_k + \lambda \overset{!}{=} 0.$$

$$\Rightarrow \varepsilon_k^* = -\frac{\sum_{i=1}^n \gamma_{ik}^{(j)}}{\lambda}, \text{ since } \sum_{k=1}^K \varepsilon_k^* = 1.$$

$$\Rightarrow -\frac{\sum_{k=1}^K \sum_{i=1}^n \gamma_{ik}^{(j)}}{\lambda} = 1 \Rightarrow -\lambda = \sum_{i=1}^n 1 = n$$

Thus, $\varepsilon_k^* = \frac{1}{n}\sum_{i=1}^n \gamma_{ik}^{(j)}$ i.e. $\varepsilon_k^{(j+1)} = \frac{1}{n}\sum_{i=1}^n \gamma_{ik}^{(j)}$

3.

Second optimize $(\vec{w}_k, b_k)$, Suppose σ_k^2 is fixed.

Then $\max\limits_{(\vec{w}_k, b_k)} Q(\theta, \theta^{(j)}) \Leftrightarrow \min\limits_{(\vec{w}_k, b_k)} \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik}^{(j)} \| y_i - \vec{w}_k^T \vec{x}_i - b_k \|^2$

It has the same form as weighted least squared regression.

Thus, the solution:

$$\begin{bmatrix} b_k^{(j+1)} \\ \vec{w}_k^{(j+1)} \end{bmatrix} = (X^T C_k^{(j)} X)^{-1} X^T C_k^{(j)} \vec{y}$$

where $X = \begin{pmatrix} 1 & \vec{x}_1^T \\ \vdots & \vdots \\ 1 & \vec{x}_n^T \end{pmatrix}$, $C_k^{(j)} = \begin{bmatrix} \gamma_{1k}^{(j)} & & 0 \\ & \ddots & \\ 0 & & \gamma_{nk}^{(j)} \end{bmatrix}$

Last optimize σ_k^2, plug in results above. use Lagarange multiplyer theory

$$L(\sigma_1^2, \cdots, \sigma_K^2) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik}^{(j)} \left(-\frac{1}{2} \log \sigma_k^2 - \frac{1}{2} \| y_i - \vec{w}_k^T \vec{x}_i - b_k \|^2 / \sigma_k^2\right)$$

$$\frac{\partial L}{\partial \sigma_k^2} = \sum_{i=1}^{n} \gamma_{ik}^{(j)} \left(-\frac{1}{2\sigma_k^2} + \frac{\| y_i - \vec{w}_k^T - \vec{x}_i - b_k \|^2}{2(\sigma_k^2)^2}\right) \stackrel{!}{=} 0$$

(here we treat σ_k^2 as a variable)

$$\Rightarrow \sigma_k^{(j+1)^2} = \frac{\sum_{i=1}^{n} \gamma_{ik}^{(j)} \| y_i - w_k^{(j+1)} x_i - b_k^{(j+1)} \|^2}{\sum_{i=1}^{n} \gamma_{ik}^{(j)}}$$

4) Ncut and Normalized Spectral Clustering

$K=2$. $\quad Ncut(A, \bar{A}) = \frac{1}{2}\left(\frac{C(A, \bar{A})}{Vol(A)} + \frac{C(A, \bar{A})}{Vol(\bar{A})}\right) = \frac{1}{2} C(A, \bar{A})\left(\frac{1}{Vol(A)} + \frac{1}{Vol(\bar{A})}\right)$

where $Vol(A) = \sum_{i \in A} \sum_{j \in V} w_{ij}$

Given $A \in \{1, 2, \cdots, n\}$ define $\vec{f}_A = (f_{A1}, \cdots, f_{An})^T \in \mathbb{R}^n$ by

$$f_{Ai} = \begin{cases} +\sqrt{\frac{Vol(\bar{A})}{Vol(A)}} & \text{if } i \in A \\ -\sqrt{\frac{Vol(A)}{Vol(\bar{A})}} & \text{if } i \notin A. \end{cases}$$

Compute : $\vec{f}_A^T L \vec{f}_A = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_{A_i} - f_{A_j})^2$

$$= \frac{1}{2}\sum_{i\in A, j\in \bar{A}} w_{ij}\left(\sqrt{\frac{vol(\bar{A})}{vol(A)}} + \sqrt{\frac{vol(A)}{vol(\bar{A})}}\right)^2$$

$$+ \frac{1}{2}\sum_{i\in \bar{A}, j\in A} w_{ij}\left(-\sqrt{\frac{vol(\bar{A})}{vol(A)}} - \sqrt{\frac{vol(A)}{vol(\bar{A})}}\right)^2$$

$$= \sum_{i\in A, j\in \bar{A}} w_{ij}\left(\sqrt{\frac{vol(\bar{A})}{vol(A)}} + \sqrt{\frac{vol(A)}{vol(\bar{A})}}\right)^2$$

$$= \sum_{i\in A, j\in \bar{A}} w_{ij}\left(\frac{vol(\bar{A})}{vol(A)} + \frac{vol(A)}{vol(\bar{A})} + 2\right)$$

$$= (vol(\bar{A}) + vol(A))\left(\frac{\sum_{i\in A, j\in \bar{A}} w_{ij}}{vol(A)} + \frac{\sum_{i\in A, j\in \bar{A}} w_{ij}}{vol(\bar{A})}\right)$$

$$= (vol(\bar{A}) + vol(A))\left(\frac{C(A,\bar{A})}{vol(A)} + \frac{C(A,\bar{A})}{vol(\bar{A})}\right)$$

$$= 2(vol(\bar{A}) + vol(A))\, Ncut(A,\bar{A})$$

$$\vec{1} D \vec{f}_A = \sum_{i=1}^{n} d_i f_{A_i} = \sum_{i\in A} d_i \cdot \sqrt{\frac{vol(\bar{A})}{vol(A)}} - \sum_{j\in \bar{A}} d_j \sqrt{\frac{vol(A)}{vol(\bar{A})}}$$

since $d_i = \sum_{j=1}^{n} w_{ij}$

$$= vol(A)\sqrt{\frac{vol(\bar{A})}{vol(A)}} - vol(\bar{A})\cdot\sqrt{\frac{vol(A)}{vol(\bar{A})}} = 0.$$

$$\vec{f}_A^T D \vec{f}_A = \sum_{i=1}^{n} d_i f_{A_i}^2 = \sum_{i\in A} \frac{d_i\, vol(\bar{A})}{vol(A)} + \sum_{j\in \bar{A}} \frac{d_j\, vol(A)}{vol(\bar{A})}$$

$$= vol(\bar{A}) + vol(A)$$

Then we claim. $\vec{f}_A^T L \vec{f}_A = 2\vec{f}_A^T D \vec{f}_A \cdot Ncut(A,\bar{A})$

There, Ncut can be written as the following optimation problem:

$$\min_{A\in\{1,\cdots,n\}} \vec{f}_A^T L \vec{f}_A \quad s.t. \quad \vec{1} D \vec{f}_A = 0, \quad \vec{f}_A^T D \vec{f}_A = vol(A) + vol(\bar{A}) = vol(V)$$

The relaxation:

$$\min_{\vec{f}\in\mathbb{R}^n} \vec{f}^T L \vec{f} \quad s.t. \quad \vec{1} D \vec{f} = 0, \quad \vec{f}^T D \vec{f} = vol(V)$$

4.

Suppose $g = D^{\frac{1}{2}} f$. Then

$$\vec{f}_A^T D \vec{f}_A = (D^{\frac{1}{2}} f)^T (D^{\frac{1}{2}} f) = g^T g.$$

The optimazation problem is:

$$\min_{g \in \mathbb{R}^n} g^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} g \quad s.t. \quad D^{\frac{1}{2}} g = 0, \quad g^T g = vol(V)$$

Define: $L_g = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$

also, $\hat{L} = D^{-1} L = I - D^{-1} W$

The optimazation problem is:

$$\min_{g \in \mathbb{R}^n} g^T L_g g \quad s.t. \quad D^{\frac{1}{2}} g = 0. \quad g^T g = vol(V)$$

Notice L_g is symmetric $n \times n$ matrix, $D^{\frac{1}{2}} \mathbb{1}$ is the first eigenvector

$vol(V)$ is constant, Apply Rayleigh-Ritz Theorem.

The solution of opt problem is the second eigenvector of L_g.

re-substitute $f = D^{-\frac{1}{2}} g$.

suppose u_2 is eigenvector for L_g. Then $L_g u_2 = \lambda_2 u_2$

$u_2' := D^{-\frac{1}{2}} u_2$. Then $\hat{L} u_2' = D^{-1} L u_2' = D^{-1} L D^{-\frac{1}{2}} u_2 = D^{-\frac{1}{2}} L_g u_2$

$$= D^{-\frac{1}{2}} \lambda_2 u_2 = \lambda_2 u_2'$$

Thus, u_2' is eigenvector of $\hat{L}$ u_2' corresponds to u_2

Since $f = D^{-\frac{1}{2}} g$, it shows that finding second eigenvector for L_g is equivalent to finding second eigenvector for $\hat{L}$

And the second eigenvector of $\hat{L}$ is solution for f

b. The condition: $\vec{x}_i \in \mathbb{R}^d$ $\vec{\theta}_i \in \mathbb{R}^k$. Then $\lambda^k \neq \lambda^{k+1}$ is the condition

i.e. $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda^k > \lambda^{k+1} \geq \lambda^{k+2} \cdots \geq \lambda^d$

Here λ_i is eigenvalues for sample covariance matrix: $S = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)(x_i-\mu)^T$. And spectral decomposition of S is $S = U\Lambda U^T$, where $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_d)$. $U = [\vec{u}_1, \cdots, \vec{u}_d]$, λ_i is eigenvalue, $\vec{u}_i$ is eigenvector.

Suppose $\lambda_k = \lambda_{k+1}$. The corresponding eigenvectors don't have to be the same $\vec{u}_k \neq \vec{u}_{k+1}$. Then, if we choose A to be a $d \times k$ matrix. we can see that $A = [u_1, \cdots, u_k]$ and $A = [u_1, \cdots, u_{k-1}, u_{k+1}]$ are both solutions for $\min \sum_{i=1}^{n} \|\vec{x}_i - \mu - A\vec{\theta}_i\|^2$, however, the subspace $\langle A \rangle$ are different (not unique). Vice versa. if A is not unique, only $\vec{u}_k$ is possible to be replaced. it corresponds that $\lambda_k = \lambda_{k+1}$. Thus, our condition is necessary and sufficient

1) PCA

a. obj. fun $= \sum_{i=1}^{n} \|\vec{x}_i - \vec{\mu} - A\vec{\theta}_i\|^2 . =: J$

since it's a convex function of $\vec{\theta}_i \in \mathbb{R}^k$.

we compute $\frac{\partial J}{\partial \vec{\theta}_i} = -2A^T(\vec{x}_i - \vec{\mu} - A\vec{\theta}_i) \stackrel{!}{=} 0$.

$\Rightarrow \vec{\theta}_i = A^T(\vec{x}_i - \vec{\mu})$. $(\because A^TA = I_{k\times k})$ is optimal $\vec{\theta}_i$

for $\underline{\min J}$

$\Rightarrow J = \sum_{i=1}^{n} \|\vec{x}_i - \vec{\mu} - AA^T(\vec{x}_i - \vec{\mu})\|^2$, also, J is a convex function of $\vec{\mu} \in \mathbb{R}^d$

$\frac{\partial J}{\partial \vec{\mu}} = -\sum_{i=1}^{n} 2(\vec{x}_i - \vec{\mu} - A\vec{\theta}_i) \stackrel{!}{=} 0. \Rightarrow \sum_{i=1}^{n}(I - AA^T)(\vec{x}_i - \vec{\mu}) \stackrel{!}{=} 0$.

$\Rightarrow \vec{\mu} = \frac{1}{n}\sum_{i=1}^{n}\vec{x}_i = \bar{x}$ is optimal $\vec{\mu}$.

Suppose $\tilde{x}_i = \vec{x}_i - \bar{x}_i$, then

$J^* = \sum_{i=1}^{n} \|\tilde{x}_i - AA^T\tilde{x}_i\|^2 = \sum_{i=1}^{n} \text{trace}[(\tilde{x}_i - AA^T\tilde{x}_i)(\tilde{x}_i - AA^T\tilde{x}_i)^T]$

$= \sum_{i=1}^{n} \text{trace}[\tilde{x}_i\tilde{x}_i^T - \tilde{x}_i\tilde{x}_i^TAA^T - AA^T\tilde{x}_i\tilde{x}_i^T + AA^T\tilde{x}_i\tilde{x}_i^TAA^T]$

$= \sum_{i=1}^{n} \text{trace}(\tilde{x}_i\tilde{x}_i^T) - 2\sum_{i=1}^{n}\text{trace}(AA^T\tilde{x}_i\tilde{x}_i^T) + \sum_{i=1}^{n}\text{trace}(AA^T\tilde{x}_i\tilde{x}_i^T)$.

(because of the property of trace: $tr(A) = tr(A^T)$. $tr(ABC) = tr(CAB)$.)

$\Rightarrow J^* = \sum_{i=1}^{n}\text{trace}(\tilde{x}_i\tilde{x}_i^T) - \sum_{i=1}^{n}\text{trace}(AA^T\tilde{x}_i\tilde{x}_i^T)$

since $S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T = \frac{1}{n}\sum_{i=1}^{n}\tilde{x}_i\tilde{x}_i^T$.

$\Rightarrow tr(S) = \frac{1}{n}\sum_{i=1}^{n}\text{trace}(\tilde{x}_i\tilde{x}_i^T) = \sum_{j=1}^{d}\lambda_j$

$\Rightarrow J^* = n\sum_{j=1}^{d}\lambda_j - \sum_{i=1}^{n}\text{trace}(AA^T\tilde{x}_i\tilde{x}_i^T)$

EECS 545 Homework 5

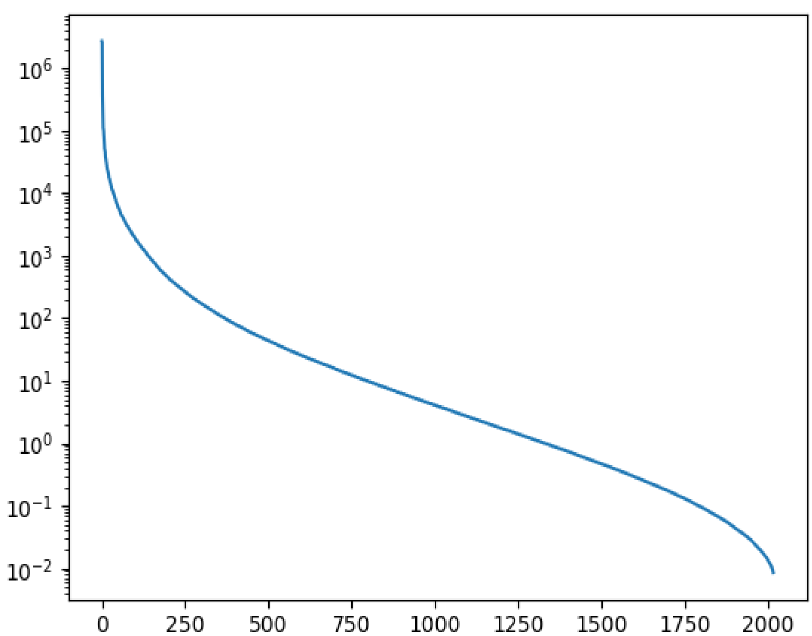
Yuan Yin

November 21, 2018

Problem 2) Eigenfaces

a.

The plot of sorted eigenvalues is as follows:

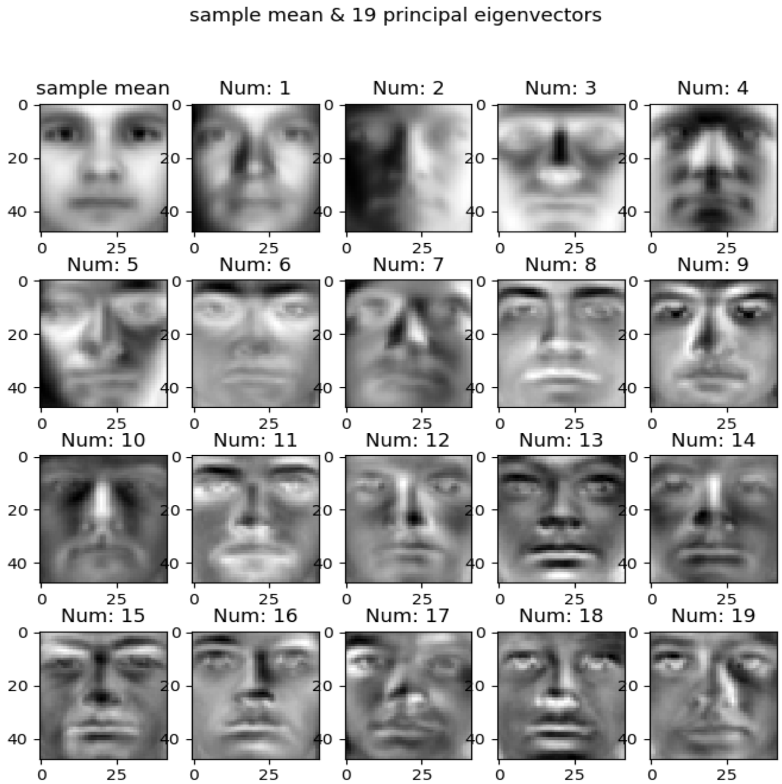


Also, the result of code is as follows:

```
number of principal components needed to represent 95% of total variation is: 43
percentage reduction in dimension is: 97.87%
number of principal components needed to represent 99% of total variation is: 167
percentage reduction in dimension is: 91.72%
```

b.

the plot of principal eigenvectors is as below:



From the figure above we can find that the first 19 eigenfaces capture both facial and lighting variations. And sometimes the eigenface can capture both of them. Like the first eigenfaces, it looks like capturing the whole shape of the face. And the second captures faces with different lightness on left and right. Some of them capture the emotional expression of faces like number: 9, 13. Some of them capture both lightness and fiacial characteristics like number: 4, 8, 14, 16 and others capture only lightness of faces like number: 3, 6, 10, 11

c.

The code is as follows:

```
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt

# Proceed with the data
yale = sio.loadmat('yalefaces.mat')
yalefaces = yale['yalefaces']
```

```
n = yalefaces.shape[2]
d = yalefaces.shape[0] * yalefaces.shape[1]
x = np.zeros([d, n])
for i in range(0, yalefaces.shape[2]):
    x[:, i] = np.reshape(yalefaces[:, :, i], d)

# Compute sample covariance matrix
mu = x.mean(axis=1)
x_bar = np.tile(mu, (n, 1)).T
s = (x - x_bar).dot((x - x_bar).T) / n

# Singular value decomposition
lamda, u = np.linalg.eig(s)
idx = lamda.argsort()[::-1]
lamda = lamda[idx]
u = u[:,idx]
plt.semilogy(lamda)
for i in range(d):
    percent = sum(lamda[:i]) / sum(lamda)
    if percent > 0.95:
        print('number of principal components needed to represent 95% of total variation is:
            ', i)
        print('percentage reduction in dimension is: ', '%.2f%%' % ((d - i) / d * 100))
        break
for i in range(d):
    percent = sum(lamda[:i]) / sum(lamda)
    if percent > 0.99:
        print("number of principal components needed to represent 99% of total variation is:
            ", i)
        print("percentage reduction in dimension is: ", '%.2f%%' % ((d - i) / d * 100))
        break

# Plot eigenvectors
fig = plt.figure(num='eigenfaces',figsize=(8,8.5))
fig.suptitle('sample mean & 19 principal eigenvectors')
plt.subplot(4,5,1)
plt.title('sample mean')
plt.imshow(np.reshape(mu, (yalefaces.shape[0], yalefaces.shape[1])), cmap =
    plt.get_cmap('gray'))

for i in range(19):
    plt.subplot(4,5,2+i)
    plt.title('Num: %d' % (1 + i))
    plt.imshow(np.reshape(u[:,i], (yalefaces.shape[0], yalefaces.shape[1])), cmap =
```

```
        plt.get_cmap('gray'))
plt.show()
plt.close()
```

Problem 3) EM Algorithm for Mixed Linear Regression

c.

The code is as follows:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import pylab as pl

# Generating data
np.random.seed(0)
n = 200 # sample size
K = 2 # number of lines
e = np.array([0.7, 0.3]) # mixing weights
w = np.array([-2, 1]) # slopes of lines
b = np.array([0.5, -0.5]) # offsets of lines
v = np.array([0.2, 0.1]) # variances
x = np.zeros([n])
y = np.zeros([n])
for i in range(0,n):
    x[i] = np.random.rand(1)
    if np.random.rand(1) < e[0]:
        y[i] =w[0] * x[i] + b[0] + np.random.randn(1) * np.sqrt(v[0])
    else:
        y[i] = w[1] * x[i] + b[1] + np.random.randn(1) * np.sqrt(v[1])

X = np.c_[np.ones(n).T, x]
plt.plot(x, y, 'bo')
t = np.linspace(0, 1, num = 100)
plt.plot(t, w[0] * t + b[0], 'k')
plt.plot(t, w[1] * t + b[1], 'k')

# Implement EM algorithm
## Initialization
e_weight = np.array([.5, .5])
w_slope = np.array([1., -1.])
b_offset = np.array([0., 0.])
```

```
sig_var = np.array([np.var(y), np.var(y)])
iteration = 500
gamma = np.zeros([n, K])
Loglike = np.zeros(iteration)
loglike = 0
for j in range(iteration):
    new_loglike = 0
    for i in range(n):
        s = 0
        for k in range(K):
            s += e_weight[k] * norm.pdf(y[i], loc=w_slope[k] * x[i] + b_offset[k],
                scale=np.sqrt(sig_var[k]))
        new_loglike += np.log(s)
    ## E step
    sum = np.zeros(n)
    for i in range(n):
        sum[i] = e_weight[0] * norm.pdf(y[i], loc=w_slope[0] * x[i] + b_offset[0],
            scale=np.sqrt(sig_var[0])) \
                + e_weight[1] * norm.pdf(y[i], loc=w_slope[1] * x[i] + b_offset[1],
                    scale=np.sqrt(sig_var[1]))
        for k in range(K):
            gamma[i, k] = e_weight[k] * norm.pdf(y[i], loc=w_slope[k] * x[i] + b_offset[k],
                scale=np.sqrt(sig_var[k])) / sum[i]

    ## M step
    e_weight = np.sum(gamma, axis=0) / n
    for k in range(K):
        [b_offset[k], w_slope[k]] = np.asarray((np.asmatrix(X).T *
            np.asmatrix(np.diag(gamma[:, k])) * np.asmatrix(X)).I
                                            * np.asmatrix(X).T * np.asmatrix(np.diag(gamma[:,
                                                k])) * np.asmatrix(y).T)
        numerator = 0; denominator = 0
        for i in range(n):
            numerator += (y[i] - w_slope[k] * x[i] - b_offset[k])**2 * gamma[i, k]
            denominator += gamma[i, k]
        sig_var[k] = numerator / denominator
    if abs(new_loglike - loglike) < 10**-4:
        times = j
        break
    else:
        loglike = new_loglike
        Loglike[j] = loglike
print("The number of iterations to reach convergence is: ", times + 1)
print("The estimated model parameters are as follows: ")
```

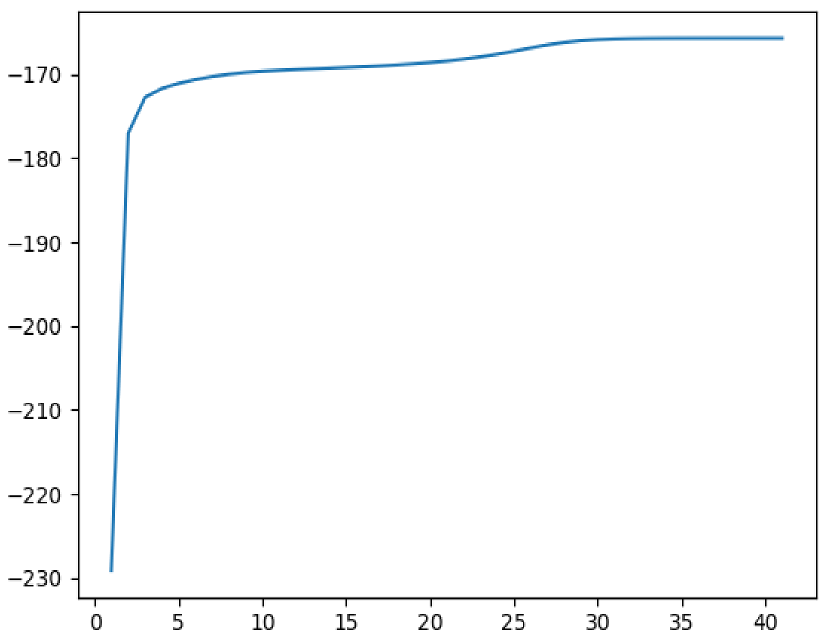
```
print("mixing weights: ", e_weight)
print("slopes of lines: ", w_slope)
print("offsets of lines: ", b_offset)
print("variances: ", sig_var)
plt.plot(t, w_slope[0] * t + b_offset[0], '--')
plt.plot(t, w_slope[1] * t + b_offset[1], '--')

plt.figure()
iter = pl.frange(1,times)
plt.plot(iter, Loglike[:times], label = 'log-likelihood')
plt.show()
```

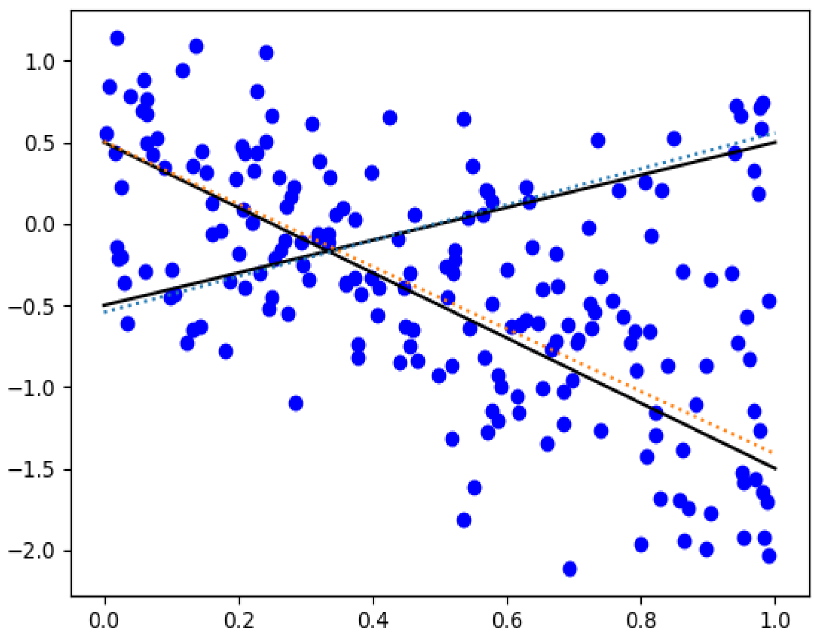
And the result is as follows:

```
The number of iterations to reach convergence is: 42
The estimated model parameters are as follows:
mixing weights: [0.18000251 0.81999749]
slopes of lines: [1.1010309 -1.91659525]
offsets of lines: [-0.54270376 0.50551544]
variances: [0.04026842 0.25301968]
```

The plot of log-likelihood as a function of iteration number is as follows:



The plot of showing the data, true lines (solid) and estimated lines (dotted) is as follows:


1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0