ECE 271-B HW#1

Problem 1

Hyperplane $w^T x + b = 0$, w, b unknown

a) Point closest to origin x_0: as func of w and b

The distance of point x_0 to the origin is $\|x_0\|$. So in our function we should minimize $\|x\|$, w.r.t x. We can use the hyperplane equation as a constraint.

$$\min_x \|x\|, \text{ constraint} \Rightarrow w^T x + b = 0$$

We can use Lagrangian optimization.

$$L(x, \lambda) = \|x\| - \lambda(w^T x + b)$$

However this way the second derivative will become 0. Also this way I can't find $w^T x$ in terms of λ, so I can't reach a sensible solution. I will use $\|x\|^2$ as the distance to be minimized. This is usually done for distance functions, as $\|x\|^2$ is more well behaved.

$$\min_x \|x\|^2, \text{ constraint} \Rightarrow w^T x + b = 0$$

For the Lagrangian we have 3 conditions:

i) $\nabla_x L(x^*, \lambda^*) = 0$

ii) $\nabla_\lambda L(x^*, \lambda^*) = 0$

iii) $y^T \underbrace{\nabla^2_{xx} L(x^*, \lambda^*)}_{\text{Hessian positive definite}} y \geq 0, \forall y \text{ s.t. } \nabla h(x^*)^T y = 0$

The Lagrangian is $L(x,\lambda) = \|x\|^2 - \lambda(w^T x + b)$

$\nabla_x L = 2x - \lambda w$, $\nabla_\lambda L = w^T x + b$

Setting derivatives to 0, $2x - \lambda w = 0$, $w^T x + b = 0$

$2x = \lambda w$

$2w^T x = \lambda w^T w$, $w^T w = \|w\|^2$

$\frac{2w^T x}{\|w\|^2} = \lambda$

$w^T x = \lambda \frac{\|w\|^2}{2}$

$w^T x + b = 0$, $\frac{\lambda\|w\|^2}{2} + b = 0$

$\lambda = \frac{-2b}{\|w\|^2}$

$2x - \lambda w = 0$, $2x + \frac{2bw}{\|w\|^2} = 0$, $x = \frac{-bw}{\|w\|^2}$

The final condition of the Lagrangian is the Hessian

$\nabla_{xx}^2 L(x^*, \lambda^*) = 0$, where $x^* = \frac{-bw}{\|w\|^2}$, $\lambda^* = \frac{-2b}{\|w\|^2}$

$\nabla_{xx}^2 L(x,\lambda) = \nabla_x (2x - \lambda w) = 2$ or in $(2I)$

where I is the identity matrix. (I) is always P.D.

So the Hessian is always P.D, for every x, λ.

We have proved that $x^* = \frac{-bw}{\|w\|^2}$, $\lambda^* = \frac{-2b}{\|w\|^2}$ is the optimal solution.

b) $x^* = \frac{-bw}{\|w\|^2}$, and $w^T x + b = 0$.

$w^T x^* = \frac{-b w^T w}{\|w\|^2}$, $w^T x^* = -b$, $b = -w^T x^*$

$w^T x - w^T x^* = 0$, $w^T(x - x^*) = 0$, we can call $x^* = x_0$

So the eqn is $w^T(x - x_0) = 0$. This is a well known eqn.

It represents a plane that contains x_0, and which has the normal vector w^T. Which is what we were given in the problem definition.

We are also given that $\|w\| = 1$, previously we had: $x' = x_0 = \frac{-bw}{\|w\|^2}$, $x_0 = -bw$. Here we know that w is now a unit vector. Therefore $\|x_0\| = |b|\|w\|$, $\|x_0\| = |b|$

So, x_0 is a point on the plane $w^T(x - x_0) = 0$, with magnitude of $|b|$. Which is as expected. Such a plane is always at least $|b|$ distant from the origin. If $b = 0$, $w^T x = 0$, and the plane would be passing from the origin.

Problem 2

We are given the entropy of the pdf of $p(x)$

$h(x) = -\int p(x) \log[p(x)]\, dx$. As this is a pdf function we will have several constraints for the Lagrangian. The most obvious one is that:

$\int p(x)\, dx = 1$, the prob. dist. sums to 1.

We are also given that mean $= \mu$ and variance $= \sigma^2$. These should also be constraints because they define the prob. dist. $p(x)$ itself. We know:

$\mu = E[X] = \int x f(x)\, dx = \int x p(x)\, dx$ in our case.

$$\sigma^2 = Var(x) = E[(x - E[x]^2)] = \int (x - E[x]^2) f(x)\, dx$$

$$= \int (x-\mu)^2 p(x)\, dx$$

Therefore the Lagrangian Optimization will have 3 seperate constraints. I will call their Lagrange multipliers, λ_p, λ_μ and λ_σ respectively. We have:

$$L(p(x), \lambda_p, \lambda_\mu, \lambda_\sigma) = \int p(x) \log[p(x)]\, dx - \lambda_p \left(\int p(x)\, dx - 1\right) - \lambda_\mu \left(\int x p(x)\, dx - \mu\right) - \lambda_\sigma \left(\int (x-\mu)^2 p(x)\, dx - \sigma^2\right)$$

(1) (2) (3) (4) — 4 parts

The next step is to set the derivatives to 0, and then check the Hessian. We will derivate w.r.t. $\partial p(x)$, $\partial \lambda_p$, $\partial \lambda_\mu$, $\partial \lambda_\sigma$. We derivate w.r.t. $p(x)$ instead of x, because $p(x)$ is what we are concerned with.

Finally we are minimizing $-\int p(x) \log(x)\, dx$, to maximize

$$h(x) = -\int p(x) \log(x)\, dx.$$

The derivatives w.r.t. λ_p, λ_μ and λ_σ are trivial:

$$\frac{\partial L}{\partial \lambda_p} = \int p(x)\, dx - 1 = 0 \Rightarrow \int p(x)\, dx = 1$$

$$\frac{\partial L}{\partial \lambda_\mu} = \int x p(x)\, dx - \mu = 0 \Rightarrow \int x p(x)\, dx = \mu$$

$$\frac{\partial L}{\partial \lambda_p} = \int (x-\mu)^2 p(x)\, dx - \sigma^2 = 0 \Rightarrow \int (x-\mu)^2 p(x)\, dx = \sigma^2$$

Same equations as the constraints.

We just get the same eqns. defining the prob. dist.

The derivative w.r.t. $\partial p(x)$ is more complex. I will derivate each of the 4 parts of the Lagrangian seperately:

① $\frac{\partial}{\partial p(x)} \left(\int p(x) \log[p(x)] dx \right)$, from the lecture notes on functional derivatives, this is equivalent to $\frac{\partial}{\partial p(v)} \left[p(v) \log[p(v)] \right]$

$= \log[p(v)] + 1$

The other integrals will follow the same pattern.

② $\frac{\partial}{\partial p(x)} \left(-\lambda_p \int p(x) dx - 1 \right) \Rightarrow \frac{\partial}{\partial p(v)} \left(-\lambda_p p(v) - 1 \right) = -\lambda_p$

③ $\frac{\partial}{\partial p(x)} \left(-\lambda_\mu \int x p(x) dx - \mu \right) \Rightarrow \frac{\partial}{\partial p(v)} \left(-\lambda_\mu v p(v) - \mu \right) = -\lambda_\mu v$

④ $\frac{\partial}{\partial p(x)} \left(-\lambda_\sigma \int (x-\mu)^2 p(x) dx - \sigma^2 \right) \Rightarrow \frac{\partial}{\partial p(v)} \left(-\lambda_\sigma (v-\mu)^2 p(v) - \sigma^2 \right)$

$= -\lambda_\sigma (v-\mu)^2$

Combining the 4 parts we get:

$$\frac{\partial L}{\partial p(v)} = \log[p(v)] + 1 - \lambda_p - \lambda_\mu v - \lambda_\sigma (v-\mu^2) = 0$$

$$\log[p(v)] = \lambda_p + \lambda_\mu v + \lambda_\sigma (v-\mu)^2 - 1$$

$$p(v) = \exp\{ \lambda_p + \lambda_\mu v + \lambda_\sigma (v-\mu)^2 - 1 \}$$

This is one exponential that has a quadratic expression of v. I know that every normal dist. is the exponential of a quadratic function, $p(v)$ is most probably a Gaussian.

Therefore I will try to 'Gaussify' it using methods learned from ECE 271A. For simplicity, I will only consider the function inside of the exponential:

$\lambda_\rho + \lambda_{\mu v} v + \lambda_\sigma v^2 - 2\lambda_\sigma v\mu + \lambda_\sigma \mu^2 - 1$, λ_σ has the highest power term attached to it, so:

$\lambda_\sigma (v^2 - 2v\mu + \mu^2) + \lambda_{\mu v} v + \lambda_\rho - 1$, $\lambda_{\mu v}$ should also be integrated to the equ

$= \lambda_\sigma \left(v^2 - 2v\mu - \frac{\lambda_{\mu v}}{\lambda_\sigma} + \mu^2\right) + \lambda_\rho - 1$

$= \lambda_\sigma \left(v^2 - 2v\left(\mu - \frac{\lambda_\mu}{2\lambda_\sigma}\right) + \mu^2\right) + \lambda_\rho - 1.$

Now I will use the 'completing the squares' trick:

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

$= a\left(x + \frac{b}{a}\right)^2 + c - \frac{b^2}{a}$ (from ECE 271A Lecture notes)

$\lambda_\sigma \left(v^2 - 2v\left(\mu - \frac{\lambda_\mu}{2\lambda_\sigma}\right) + \left(\mu - \frac{\lambda_\mu}{2\lambda_\sigma}\right)^2 - \left(\mu - \frac{\lambda_\mu}{2\lambda_\sigma}\right)^2 + \mu^2\right) + \lambda_\rho - 1$

Putting this into the original exponent:

$$p(v) = \underbrace{\exp\left\{\lambda_\sigma \left(v - \left(\mu - \frac{\lambda_\mu}{2\lambda_\sigma}\right)\right)^2\right\}}_{①} \underbrace{\exp\left\{-\lambda_\sigma\left[\left(\mu - \frac{\lambda_\mu}{2\lambda_\sigma}\right)^2 - \mu^2\right] + \lambda_\rho - 1\right\}}_{②}$$

② is constant in terms of v. I will call it C.

① is the equation of a Gaussian of form $p(x) = C \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Problem 2 part b)

b) For ①, the Gaussian,

$$\text{Mean} = \mu - \frac{\lambda_\mu}{2\lambda_\sigma} \quad \text{and} \quad \text{Variance} = -\frac{1}{\lambda_\sigma}\cdot\frac{1}{2}$$

We were given that Mean $= \mu$ and Variance $= \sigma^2$. Therefore

$$\mu - \frac{\lambda_\mu}{2\lambda_\sigma} = \mu, \ \lambda_\mu = 0 \quad \text{and} \quad \frac{-1}{2\lambda_\sigma} = \sigma^2, \ \lambda_\sigma = \frac{-1}{2\sigma^2}$$

Now the only unknown (constant) left in the $p(v)$ eqn. is λ_ρ.

As previously stated, $p(v) = C \cdot \exp\left\{\frac{-(x-\mu)^2}{2\sigma^2}\right\}$

For a scalar Gaussian: $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(x-\mu)^2}{2\sigma^2}\right\}$

Therefore we know that $C = 1/\sqrt{2\pi\sigma^2}$

$$C = \exp\left\{-\lambda_\sigma\left[\left(\mu - \frac{\lambda_\mu}{2\lambda_\sigma}\right)^2 - \mu^2\right] + \lambda_\rho - 1\right\} = \left(2\pi\sigma^2\right)^{-1/2}$$

$$\log[C] = -\lambda_\sigma\left[\left(\mu - \frac{\lambda_\mu}{2\lambda_\sigma}\right)^2 - \mu^2\right] + \lambda_\rho - 1 = -\frac{1}{2}\log\left[2\pi\sigma^2\right]$$

if we plug $\lambda_\mu = 0$, $\lambda_\rho - 1 = -\frac{1}{2}\log[2\pi\sigma^2]$, $\lambda_\rho = 1 - \frac{1}{2}\log[2\pi\sigma^2]$

We have found all the constant parameters defining $p(v)$

$$p(v) = \exp\left\{\lambda_\rho - 1\right\} \cdot \exp\left\{\lambda_\sigma (v - (\mu))^2\right\}$$

Where $\lambda_\rho = 1 - \frac{1}{2}\log[2\pi\sigma^2]$, $\lambda_\sigma = -\frac{1}{2\sigma^2}$

$$p(v) = \exp\left\{\log\left[(2\pi\sigma^2)^{-1/2}\right]\right\} \exp\left\{-\frac{(v-\mu)^2}{2\sigma^2}\right\}$$

$$p(x) = \frac{1}{2\pi\sigma^2}\, e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

The pdf with the highest entropy is the Normal Dist.

Problem 3

We have image patches $X = \{X_1, \ldots, X_{64}\}$ i.i.d. but not independent.

Correlation coefficient: $\rho_{ij} \frac{E[X_i X_j]}{\sqrt{E[X_i^2]\, E[X_j^2]}}$

a) The PCA of X should not be effected by the change of variable $Z = X - \mu_x$, where $\mu_x = E[X]$.

For PCA, we compute the sample variance $\hat{\Sigma}$

$\hat{\Sigma} = \frac{1}{n} \sum_i (x_i - \hat{\mu})(x_i - \hat{\mu})^T$ and compute its eigenvalues / eigenvectors

for the change of $Z = X - \mu_x$ to have no effect, these eigenvalue/vector pairs should remain the same. These values specifically define matrix $\hat{\Sigma}$ itself. Therefore the matrix $\hat{\Sigma}$ should remain the same. Even without calculations, we know that a shift of μ applied to the distribution will not change the $(x_i - \mu)(x_i - \mu)^T$ values. Because the distances to the mean value μ, will remain the same for all points. Mathematically:

$$\Sigma_x = E\left[(x - \mu_x)(x - \mu_x)^T\right] \text{ and } \Sigma_z = E\left[(z - \mu_z)(z - \mu_z)^T\right]$$

$$\mu_z = E[Z] = E[X - \mu_x] = E[X] - E[\mu_x] = \mu_x - \mu_x = 0$$

$\Sigma_z = E[ZZ^T]$, where $Z = X - \mu_x$.

$\Sigma_z = E[(X - \mu_x)(x - \mu_x)^T] = \Sigma_x$, as expected

b) from a) we are assuming now $\mu_x = E[X] = 0$

In the extreme of highly correlated pixel values $\rho_{i,j} \to 1 \quad \forall i,j$

For pixel patches (p.p.) we can assume they are highly correlated. Vector $\underline{1}$ represents creates the DC coefficient in the DCT. If $\underline{1}$ is the largest principal component, the DC coefficient should be the most unique feature of this patch. Which is what we expect.

When $\rho_{i,j} \to 1$, we will have $\dfrac{E[X_i X_j]}{\sqrt{E[X_i^2]E[X_j^2]}} \longrightarrow 1$

In the most extreme case, $\rho_{i,j} = 1 \Rightarrow E[X_i X_j] = \sqrt{E[X_i^2]E[X_j^2]}$

As we have $E[X] = 0$, $E[X_i X_j] = E\left[(X_i - E[X])(X_j - E[X])\right]$

Therefore $E[X_i X_j] = \Sigma_{pp}$ (the cov. matrix of a pixel patch with highly correlated pixels)

$$\Sigma_{pp} = E[X_i X_j] = \sqrt{E[X_i^2]E[X_j^2]}$$

again because $E[X] = 0$, $E[X_i^2] = E\left[(X_i - E[X])^2\right] = \sigma_i^2$

Therefore $\Sigma_{pp} = \sqrt{\sigma_i^2 \sigma_j^2} = \sigma_i \sigma_j$. We are not assuming the pixels to be independent. However we assume them to be identically distributed. Which means they should have the same mean and variance. Therefore:

$\sigma = \sigma_i = \sigma_j \quad \forall i,j$ in the pixel patch.

$\Sigma_{pp} = \sigma_i \sigma_j = \sigma^2 A$ where A is a matrix

As we assume $\sigma_i = \sigma_j$ for all i, j, all the elements of Σ_{pp} should have the same value. Therefore matrix A should be:

$$A = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \end{bmatrix}_{d \times d}$$ where $d = 64$ in our case.

So A is a matrix of ones. It can be represented as an outer product, $A = \underline{1}\,\underline{1}^T$.

For matrices of the form $A = \underline{u}\,\underline{v}^T$ the eigen values are all zero except $\underline{v}^T\underline{u}$. The eigen vector of this eigen value is $\underline{u}$. (From general Linear Algebra knowledge)

Therefore $\Sigma_{pp} = \sigma A = \sigma^2 \underline{1}\,\underline{1}^T$ has the largest eigenvector of $\underline{1}$. Therefore $\underline{1}$ is the largest principal component.

c) $\bar{\Phi}$ is the matrix whose colums are ϕ_i, the principal comp.

Coefficients (features) z resulting from $z = \bar{\Phi}^T x$

DCT coefficients $z_i = \phi_i^T x$, $z_1 = \underline{1}^T x$ the DC coefficient.

We should show that $E[z_i] = 0$, $\forall i > 1$, AC coeffs have 0 mean.

From $z_i = \phi_i^T x$, $E[\phi_i^T x] = 0$

$$\phi_i = \begin{bmatrix} \phi_{i,1} \\ \vdots \\ \phi_{i,n} \end{bmatrix} \quad \text{and} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \text{where } n = 64$$

$$\phi_i^T x = \sum_{n=1}^{64} \phi_{i,n} \cdot x_n, \quad E[\phi_i^T x] = E\left[\sum_{n=1}^{64} \phi_{i,n} \cdot x_n\right]$$

$E[\Phi_{i,n}] = \Phi_{i,n}$

$E[\phi_i^T x] = \sum_{n=1}^{64} E[\Phi_{i,n} x_n]$, here $\Phi_{i,n}$ values are constants

Therefore $\rightarrow = \sum_{n=1}^{64} \Phi_{i,n} E[x_n] = 0$ where $X = \{x_1, \dots, x_{64}\}$

is identically distributed. Therefore $E[x_1] = E[x_n] = \mu$

$E[\phi_i^T x] = \sum_{n=1}^{64} \Phi_{i,n} \mu = \mu \sum_{n=1}^{64} \Phi_{i,n} = 0$, where $\mu = E[x]$

Here, we need to know $\sum_{n=1}^{64} \Phi_{i,n}$ to go on.

We know one of the components, $\phi_1 = \frac{1}{\sqrt{64}}$ from $z_1 = \phi_1^T x$.
We also know that Φ is orthonormal. We should combine this information to get a result.

Orthonormality means $\phi_i^T \phi_j = 0$ for every i, j except $i = j$

We can use $\phi_1 = \frac{1}{\sqrt{64}}$ as ϕ_i, $\frac{1}{\sqrt{64}} \phi_j = 0$ for every $j \neq 1$.

$\frac{1}{\sqrt{64}} \phi_j = 0$ yields $\sum_{n=1}^{64} \Phi_{j,n} = 0$ for every $j \neq 1$ or $j > 1$

$\rightarrow E[\phi_j^T x] = \mu \sum_{n=1}^{64} \Phi_{j,n} = 0$, as $\sum_{n=1}^{64} \Phi_{j,n} = 0$ for $j > 1$

Therefore we have proven that for all $j > 1$, (all the AC components), the mean $E[z_i] = 0$

Problem 4

We have 2 Gaussian classes $P_{x|y}(x|i) = \mathcal{G}(x, \mu_i, \Sigma_i)$ $i \in \{1, 2\}$

$P_y(1) = P_y(2) = 1/2$, We should show:

$\mu_x = E[x] = \frac{1}{2}[\mu_1 + \mu_2]$ and

$\Sigma_x = E[(x-\mu_x)(x-\mu_x)^T] = \frac{1}{2}[\Sigma_1+\Sigma_2] + \frac{1}{4}(\mu_1-\mu_2)(\mu_1-\mu_2)^T$

For $E[X]$ we have: $\rightarrow P_Y(1)\, E[X \mid Y=1]$

$\rightarrow P_Y(2)\, E[X \mid Y=2]$

Therefore $E[X] = P_Y(1) E[X|Y=1] + P_Y(2) E[X|Y=2]$

$E[X] = \frac{1}{2}\mu_1 + \frac{1}{2}\mu_2 = \frac{1}{2}[\mu_1+\mu_2]$ ($Y=2$)

For covariance $\Sigma = E[(x-\mu_x)(x-\mu_x)^T]$ we have

$\Sigma = P_Y(1) E[(x-\mu_1)(x-\mu_1)^T \mid Y=1] + P_Y(2) E[(x-\mu_2)(x-\mu_2)^T | Y$

$\Sigma_1 = E[(x-\mu_1)^2]$, $\Sigma_2 = E[(x-\mu_2)^2]$

) I should somehow yield Σ_1, Σ_2 from this equation but I couldn't find a way.

f) For the derivation of PCA we have from part a)

$\mu_x = \frac{1}{2}[\mu_1+\mu_2]$, $\Sigma_x = \frac{1}{2}[\Sigma_1+\Sigma_2] + \frac{1}{4}(\mu_1-\mu_2)(\mu_1-\mu_2)^T$

$\mu_1 = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$, $\mu_2 = -\begin{bmatrix} \alpha \\ 0 \end{bmatrix}$, $\mu_1-\mu_2 = \begin{bmatrix} 2\alpha \\ 0 \end{bmatrix}$

$\frac{1}{4}(\mu_1-\mu_2)(\mu_1-\mu_2)^T = \begin{bmatrix} \alpha^2 & 0 \\ 0 & 0 \end{bmatrix}$, $\Sigma_1 = \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & \sigma^2 \end{bmatrix}$

Therefore $\Sigma_x = \begin{bmatrix} 1 & 0 \\ 0 & \sigma^2 \end{bmatrix} + \begin{bmatrix} \alpha^2 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \alpha^2+1 & 0 \\ 0 & \sigma^2 \end{bmatrix}$

Σ_x has two eigenvalues, α^2+1 and σ^2 and corresponding eigenvectors $v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $v_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

The PCA can choose either of these pairs as the first component. This will be decided by $\max(1+\alpha^2, \sigma^2)$

If it chooses v_1, the projection will be $z = v_1^T x$

$$z = [1 \ 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1$$

If it chooses v_2, $z = v_2^T x$, $z = [0 \ 1] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_2$

If the projection axis is x_1, the μ_1 is on α while μ_2 is on $-\alpha$. This is a very good projection direction to distinguish the classes.

If the projection axis is x_2, $\mu_1 = 0 = \mu_2$ in this axis this will be a very bad projection to distinguish the classes. Just because variance is high along this axis, we would be making a very bad choice. Therefore PCA is not always reliable for classification, it can give the worst choices possible

LDA, on the other hand, will take into account the classes, their separation and also variances/covariances. As the covariance between the classes are 0, LDA will choose x_1 for projection, which maximizes between class separation.

LDA is generally better for classification. Especially for our case of Gaussian functions. PCA can give good results too, but it can also give very bad ones. The main problem of PCA is that it may ignore very discriminant dimensions, if they have low variance. LDA takes class separation into account.

Arda Cankat Bati

ECE 271B Homework #1, Computer Assignment

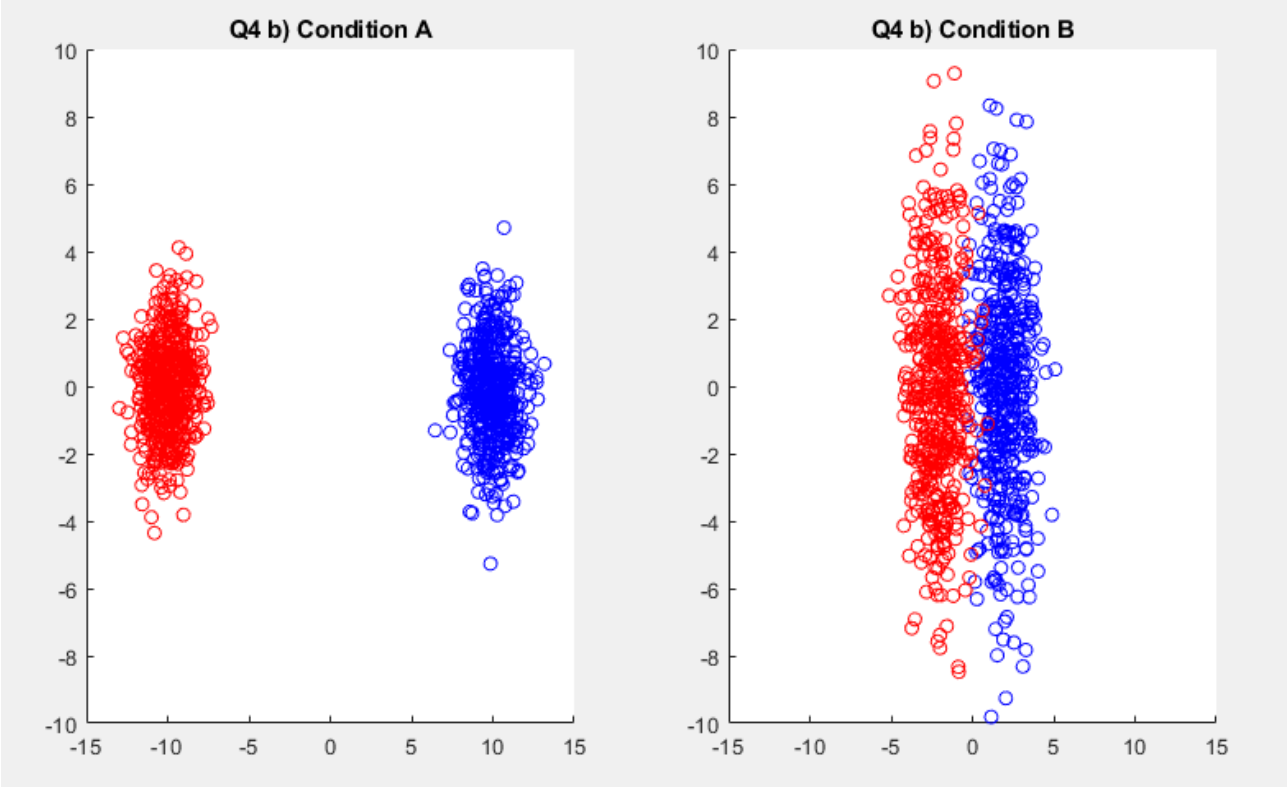
Std Id = A53284500

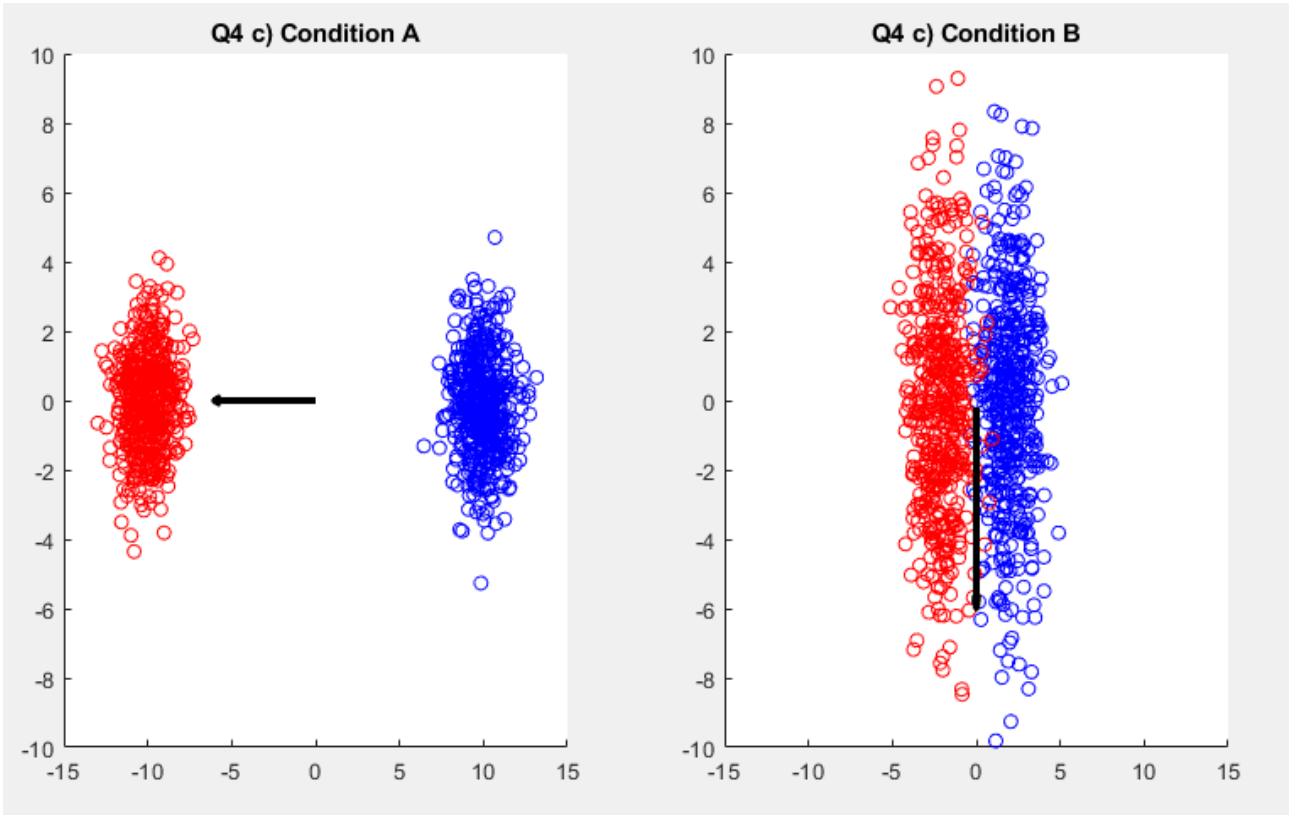
The MATLAB Code used for Questions 4 & 5 will be given at the end of the report.

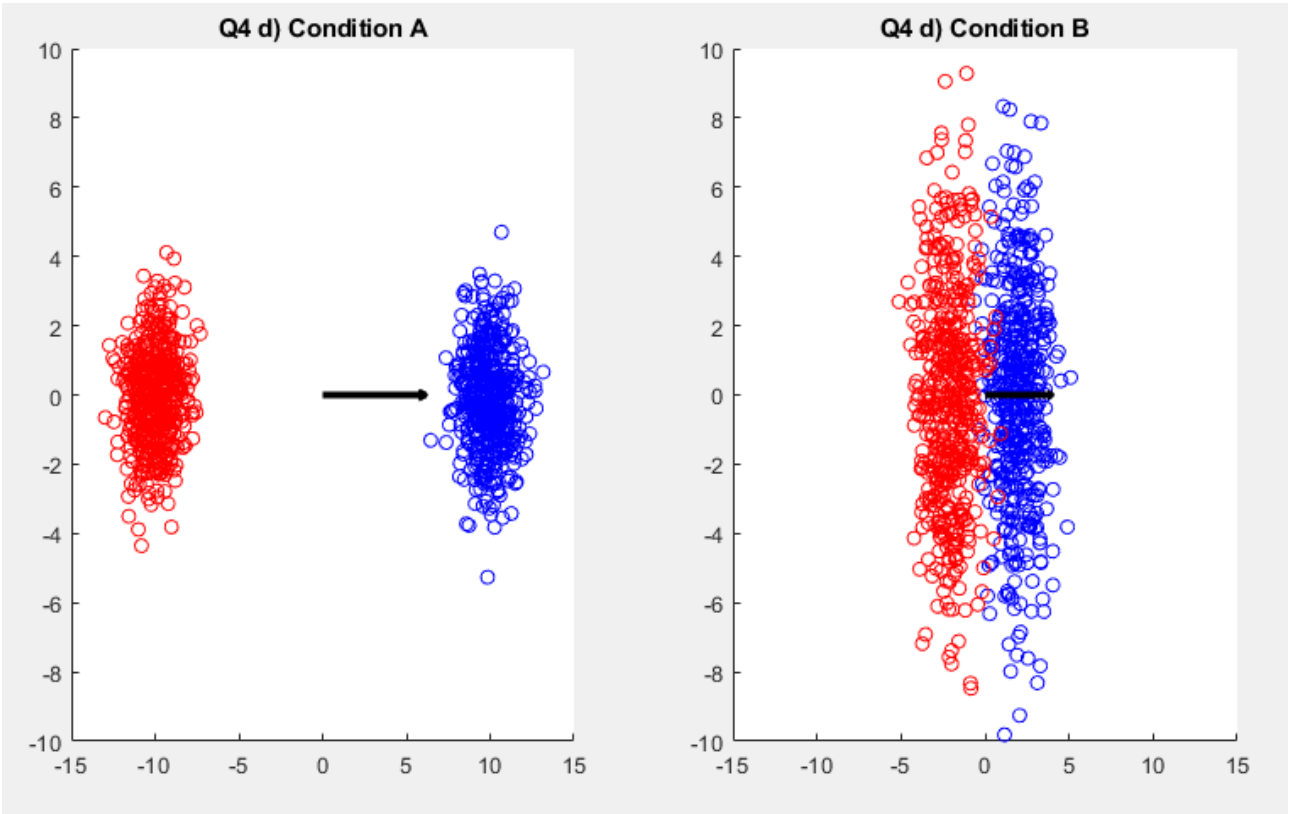
ANSWERS

Q4 b & c & d)

The plots required in these parts are given below. The blue points are of the first class, while the red ones are of the second. I represented the principal components as an arrow starting from the origin. (The arrow points to the direction of the principal component. However the size does not reflect the principal component).




Q4 c) Condition A
Q4 c) Condition B


Q4 d) Condition A
Q4 d) Condition B

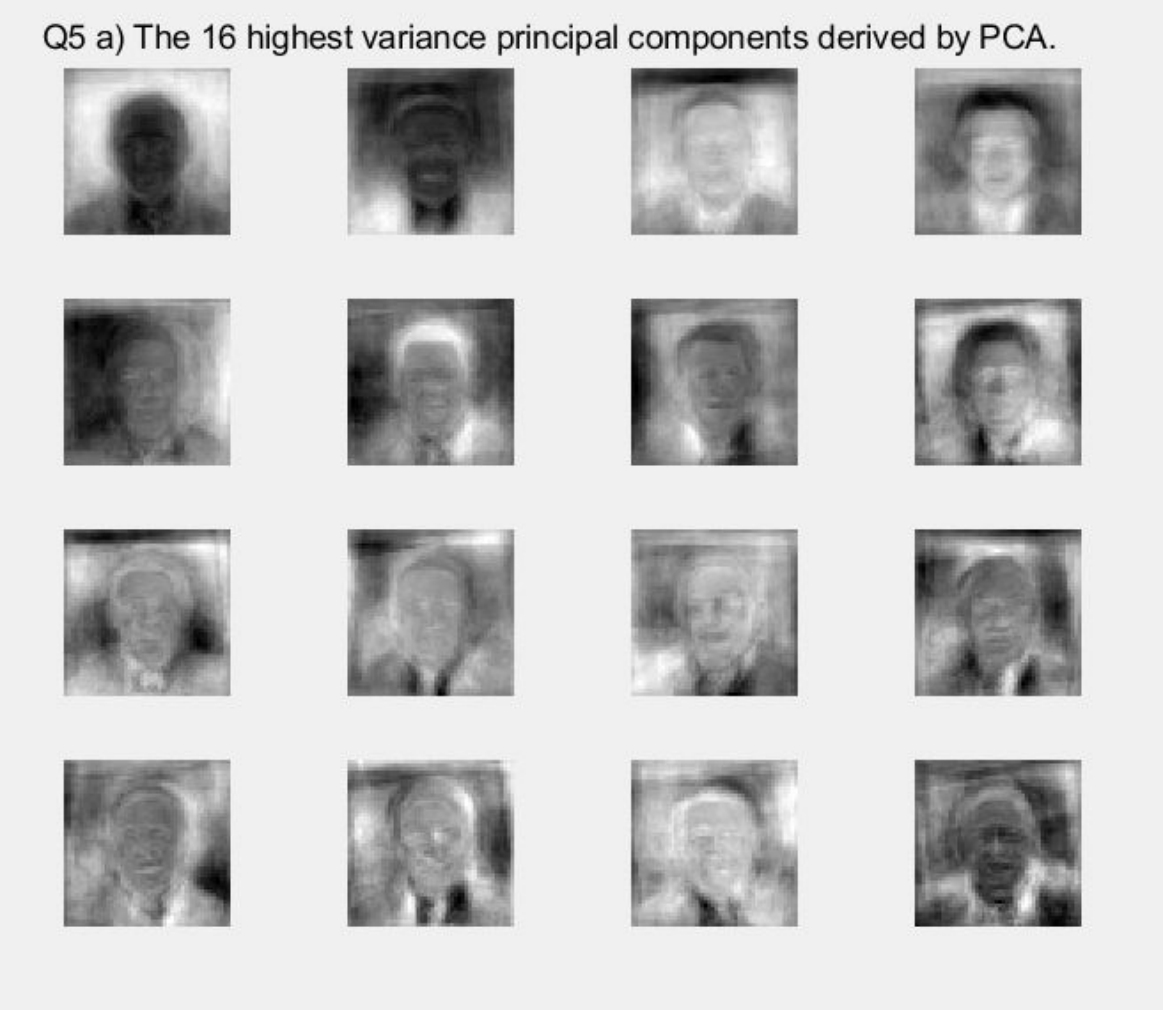
Q4 e)

From the results seen above, it can be seen that PCA is not always a good method when it comes to classification. While it yields a good principal component in Condition A, the choice for Condition B is a very poor one. When projected into the principal component from condition b, both classes have the same mean, and the same variance. This is the hardest possible case to do classification. As PCA searches for the highest variance dimension for component selection, it may choose dimension that are not very discriminant. In contrast, LDA also takes into account the class labels (by the separation between the class means) along with variance. Therefore it chooses more discriminant directions, especially in the case of underlying Gaussian distributions. In general, LDA is more successful for classification problems.

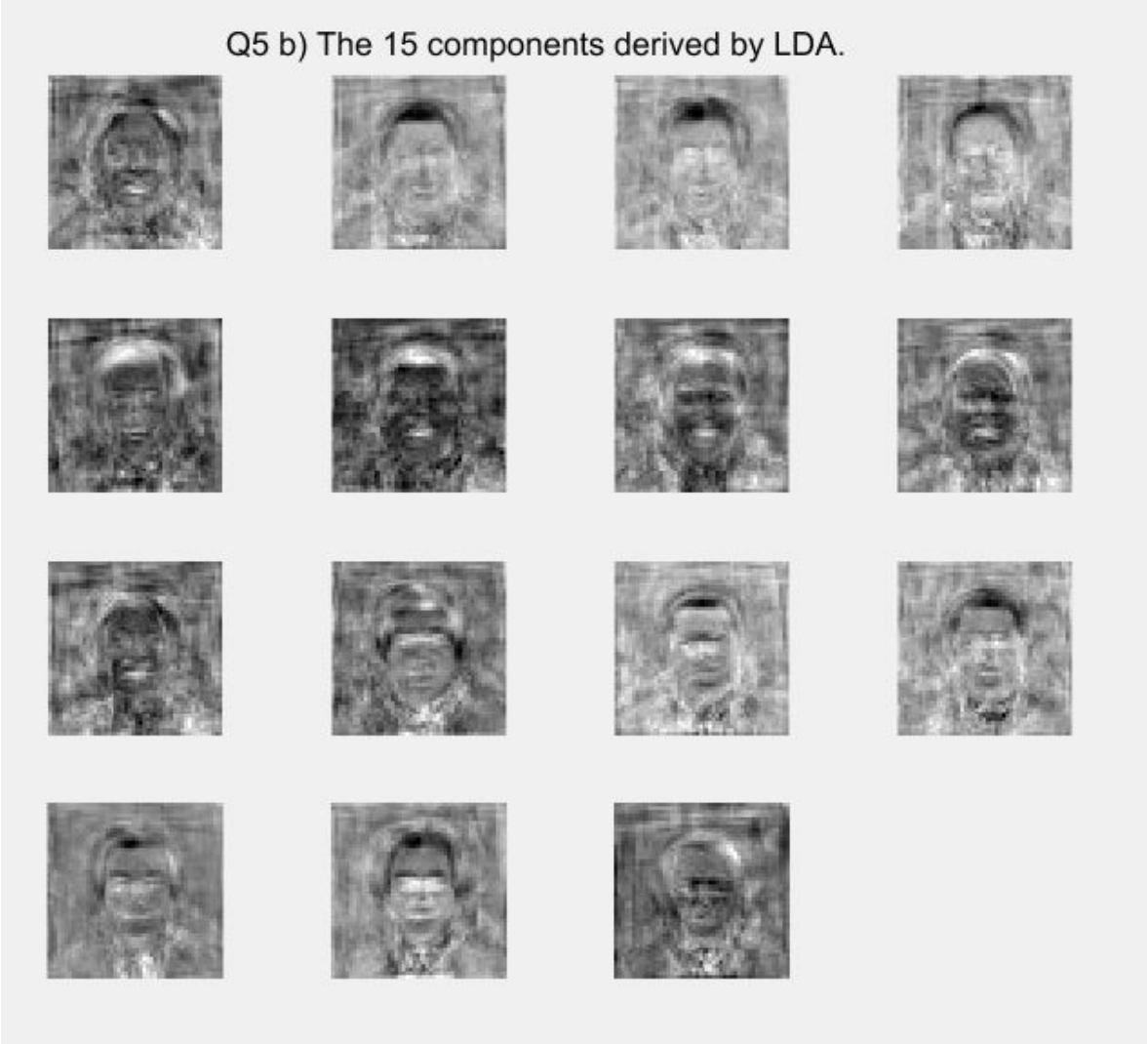
Q4 f)

I have answered this question in the hand written section of my report.

Q5 a)



Q5 b)



Q5 c)

PCA error rates, from person 1 to 6, in percent are as follows:

Person 1: 20%
Person 2: 20%
Person 3: 40%
Person 4: 30%
Person 5: 20%
Person 6: 70%

Average error in percent: 33.3333%

Q5 d)

LDA error rates, from person 1 to 6, in percent are as follows:

Person 1: 20%
Person 2: 30%
Person 3: 20%
Person 4: 20%
Person 5: 10%
Person 6: 50%

Average error in percent: 25%

Q5 e)

PCA + LDA error rates, from person 1 to 6, in percent are as follows:

Person 1: 30%
Person 2: 30%
Person 3: 30%
Person 4: 40%
Person 5: 30%
Person 6: 30%

Average error in percent: 31.6667%

```
clc
clear all

muA = [10; 0]; sigA = [1 0;0 2];
muB = [2; 0]; sigB = [1 0;0 10];

mu1A = muA; mu2A = -muA;
mu1B = muB; mu2B = -muB;

%For each class, 500 points
cases = 500;

%r = mvnrnd(MU,SIGMA,cases)
res1A = mvnrnd(mu1A,sigA,cases);
res2A = mvnrnd(mu2A,sigA,cases);

res1B = mvnrnd(mu1B,sigB,cases);
res2B = mvnrnd(mu2B,sigB,cases);

resB = mvnrnd(muB,sigB,cases);

% Drawing the distributions
figure();
subplot(1,2,1);
scatter(res1A(:,1),res1A(:,2),'b');
hold on
scatter(res2A(:,1),res2A(:,2),'r');
title('Q4 b) Condition A')
xlim([-15 15]);
ylim([-10 10]);
hold off

subplot(1,2,2);
scatter(res1B(:,1),res1B(:,2),'b');
hold on
scatter(res2B(:,1),res2B(:,2),'r');
xlim([-15 15]);
ylim([-10 10]);
title('Q4 b) Condition B')
hold off

% Creating two different datasets from the 2 conditions
CondA = [res1A; res2A];
CondB = [res1B; res2B];

% Sample Means & Covariances
MuA = mean(CondA); CovA = cov(CondA);
MuB = mean(CondB); CovB = cov(CondB);

% PCA Section
[egVec_A, egVal_A] = eig(CovA);
[value, index] = max(max(egVal_A));
```

```
compA = egVec_A(:,index);
[egVec_B, egVal_B] = eig(CovB);
[value, index] = max(max(egVal_B));
compB = egVec_B(:,index);

drawArrow = @(x,y) quiver( x(1),y(1),x(2)-x(1),y(2)-
y(1),0, 'k', 'LineWidth',3) ;

% Drawing PCA Components

figure();
subplot(1,2,1);
scatter(res1A(:,1),res1A(:,2),'b');
hold on
scatter(res2A(:,1),res2A(:,2),'r');
%Draw function a bit funky, I will print compA for clarification
compA'
drawArrow(6*([0;0] - compA) - [6 0], [0;0]);
title('Q4 c) Condition A')
xlim([-15 15]);
ylim([-10 10]);

subplot(1,2,2);
scatter(res1B(:,1),res1B(:,2),'b');
hold on
scatter(res2B(:,1),res2B(:,2),'r');
%Draw function a bit funky, I will print compB for clarification
compB'
drawArrow([0;0], 6*([0;0] - compB));
title('Q4 c) Condition B')
xlim([-15 15]);
ylim([-10 10]);


% Sample Means & Covariances
Mu1A = mean(res1A); Cov1A = cov(res1A);
Mu2A = mean(res2A); Cov2A = cov(res2A);
Mu1B = mean(res1B); Cov1B = cov(res1B);
Mu2B = mean(res2B); Cov2B = cov(res2B);

% LDA with regularization

SWA = Cov1A + Cov2A + eye(size(Cov1A,1));
SWB = Cov1B + Cov2B + eye(size(Cov1A,1));
SBA = (Mu2A' - Mu1A') * (Mu2A' - Mu1A')';
SBB = (Mu2B' - Mu1B') * (Mu2B' - Mu1B')';
LD_A = inv(SWA) * SBA;
LD_B = inv(SWB) * SBB;

[egVec_A_LD, egVal_A_LD] = eig(LD_A);
[value, index] = max(max(egVal_A_LD));
compA_LD = egVec_A_LD(:,index);

[egVec_B_LD, egVal_B_LD] = eig(LD_B);
```

```
[value, index] = max(max(egVal_B_LD));
compB_LD = egVec_B_LD(:,index);

% Drawing LDA Components

figure();
subplot(1,2,1);
scatter(res1A(:,1),res1A(:,2),'b');
hold on
scatter(res2A(:,1),res2A(:,2),'r');
%Draw function a bit funky, I will print the component for
 clarification
compA_LD'
drawArrow(6*([0;0] - compA_LD) + [6 0], [0;0]);
title('Q4 d) Condition A')
xlim([-15 15]);
ylim([-10 10]);

subplot(1,2,2);
scatter(res1B(:,1),res1B(:,2),'b');
hold on
scatter(res2B(:,1),res2B(:,2),'r');
%Draw function a bit funky, I will print the component for
 clarification
compB_LD'
drawArrow(4*([0;0] - compB_LD) + [4 0], [0;0]);
title('Q4 d) Condition B')
xlim([-15 15]);
ylim([-10 10]);
```

Published with MATLAB® R2018b

```
clc
clear all
trainset = 'trainset\subset';
testset = 'testset\subset';

TrainSet = [];
TestSet = [];

for i = 0:5
    current = strcat(trainset,int2str(i));
    jpgfiles = dir(fullfile(current,'\*.jpg*'));
    n = numel(jpgfiles);

    for j = 1:40
        im = jpgfiles(j).name;
        im1 = imread(fullfile(current,im));
        [irow, icol] = size(im1);
        temp = reshape(im1, irow*icol, 1);
        TrainSet = [TrainSet temp];
        %figure()
        %imshow(im1);
    end
end

for i = 6:11
    current = strcat(testset,int2str(i));
    jpgfiles = dir(fullfile(current,'\*.jpg*'));
    n = numel(jpgfiles);

    for j = 1:10
        im = jpgfiles(j).name;
        im1 = imread(fullfile(current,im));
        [irow, icol] = size(im1);
        temp = reshape(im1, irow*icol, 1);
        TestSet = [TestSet temp];
        %figure()
        %imshow(im1);
    end
end

TrainSet = im2double(TrainSet);
TestSet = im2double(TestSet);
m = mean(TrainSet,2);
Train_Number = size(TrainSet,2);

A = [ ];

for i = 1 : Train_Number
    temp = double(TrainSet(:,i)) - m;
    A = [A temp];
end
```

```
L = A'*A;

[V, D] = eig(L);
D_vector = diag(D);
[values, indices] = sort(D_vector,'descend');

Eigenfaces_PCA = zeros(2500,30);
for i = 1 : 30
    Eigenfaces_PCA(:,i) = A * V(:,indices(i));
end

figure()
for i = 1:16
    eigenface = Eigenfaces_PCA(:, i);
    eigenface = reshape(eigenface,50,50);
    subplot(4,4,i);
    min1 = min(min(eigenface));
    max1 = max(max(eigenface));
    eigenface=((eigenface-min1).*1)./(max1-min1);
    imshow(eigenface)
end
sgtitle('Q5 a) The 16 highest variance principal components derived by
 PCA.');

count = 0;
Eigenfaces_LDA = zeros(2500,15);
combs = combnk([0 1 2 3 4 5],2);

for k = 1: size(combs,1)
    combin = combs(k,:);
    i = combin(1); j = combin(2);
    count = count + 1;
    class0 = TrainSet(:,i*40 + 1:(i+1)*40);
    class1 = TrainSet(:,j*40 + 1:(j+1)*40);

    mu0 = mean(class0,2);
    mu1 = mean(class1,2);
    SB = (mu1 - mu0) * (mu1 - mu0)';

    E0 = cov(class0');
    E1 = cov(class1');
    SW = E0 + E1 + eye(size(E0,1));

    %LDA = inv(SW)*SB;
    LDA = SW\SB;
    [vectors, values] = eig(LDA);
    vectors = real(vectors);
    values = real(values);
    values_vector = diag(values);
    [values, indices] = sort(values_vector, 'descend');

    Eigenfaces_LDA(:,count) = vectors(:,indices(1));

end
```

```
figure()
for i = 1:15
    eigenface = Eigenfaces_LDA(:, i);
    eigenface = reshape(eigenface,50,50);
    subplot(4,4,i);
    min1 = min(min(eigenface));
    max1 = max(max(eigenface));
    eigenface=((eigenface-min1).*1)./(max1-min1);
    imshow(eigenface)
end
sgtitle('Q5 b) The 15 components derived by LDA.');


% Test Phase, Part C

Eigenfaces_PCA_Test = Eigenfaces_PCA(:,1:15)';
Z_Values_PCA = Eigenfaces_PCA_Test * TrainSet;

mu_values = zeros(15,6);
cov_values = cell(6);
for i = 1:6
    mu_values(:,i) = mean(Z_Values_PCA(:,(i-1)*40 + 1:i*40),2);
    cov_values{i} = cov(Z_Values_PCA(:,(i-1)*40 + 1:i*40)') +
 eye(15)*0.85;
end

TestP_PCA = Eigenfaces_PCA_Test * TestSet;

id_errors = zeros(6,1);
for i = 1:size(TestP_PCA,2)
    point = TestP_PCA(:,i);
    y = zeros(6,1);

    for j = 1:6
        y(j) = mvnpdf(point',mu_values(:,j)',cov_values{j});
    end

    [M,indices] = sort(y,'descend');
    class_id = indices(1);
    class_id = class_id - 1;
    real_id = floor(i / 10);
    if real_id == 6
        real_id = 5;
    end
    error = (class_id ~= real_id);
    id_errors(real_id + 1) = id_errors(real_id + 1) + error;
end
disp('Q5 Part C PCA error rates, from person 1 to 6, in percent')
id_errors = ((id_errors / 10)')*100;
disp(id_errors);
disp('Average error in percent:')
disp(mean(id_errors));
disp(' ');
```

```
% Test Phase, Part D

Eigenfaces_LDA_Test = Eigenfaces_LDA';
Z_Values_LDA = Eigenfaces_LDA_Test * TrainSet;

mu_values = zeros(15,6);
cov_values = cell(6);
for i = 1:6
    mu_values(:,i) = mean(Z_Values_LDA(:,(i-1)*40 + 1:i*40),2);
    cov_values{i} = cov(Z_Values_LDA(:,(i-1)*40 + 1:i*40)');
end

TestP_LDA = Eigenfaces_LDA_Test * TestSet;

id_errors = zeros(6,1);
for i = 1:size(TestP_LDA,2)
    point = TestP_LDA(:,i);
    y = zeros(6,1);

    for j = 1:6
        y(j) = mvnpdf(point',mu_values(:,j)',cov_values{j});
    end

    [M,indices] = sort(y,'descend');
    class_id = indices(1);
    class_id = class_id - 1;
    real_id = floor(i / 10);
    if real_id == 6
        real_id = 5;
    end
    error = (class_id ~= real_id);
    id_errors(real_id + 1) = id_errors(real_id + 1) + error;
end

disp('Q5 Part D LDA error rates, from person 1 to 6, in percent')
id_errors = ((id_errors / 10)')*100;
disp(id_errors);
disp('Average error in percent:')
disp(mean(id_errors));
disp(' ');

% Test Phase, Part E

TrainSet_PartE = Eigenfaces_PCA' * TrainSet;

count = 0;
combs = combnk([0 1 2 3 4 5],2);
Eigenfaces_PCA_LDA = zeros(30, 15);
for k = 1: size(combs,1)
    combin = combs(k,:);
    i = combin(1); j = combin(2);
    count = count + 1;
    class0 = TrainSet_PartE(:,i*40 + 1:(i+1)*40);
```

```
    class1 = TrainSet_PartE(:,j*40 + 1:(j+1)*40);

    mu0 = mean(class0,2);
    mu1 = mean(class1,2);
    SB = (mu1 - mu0) * (mu1 - mu0)';

    E0 = cov(class0');
    E1 = cov(class1');
    SW = E0 + E1;

    %LDA = inv(SW)*SB;
    LDA = SW\SB;
    [vectors, values] = eig(LDA);
    vectors = real(vectors);
    values = real(values);
    values_vector = diag(values);
    [values, indices] = sort(values_vector, 'descend');

    Eigenfaces_PCA_LDA(:,count) = vectors(:,indices(1));

end

TrainSet_PartE_LDA = Eigenfaces_PCA_LDA' * TrainSet_PartE;

mu_values = zeros(15,6);
cov_values = cell(6);
for i = 1:6
    mu_values(:,i) = mean(TrainSet_PartE_LDA(:,(i-1)*40 + 1:i*40),2);
    cov_values{i} = cov(TrainSet_PartE_LDA(:,(i-1)*40 + 1:i*40)');
end

TestP_PCA_LDA = Eigenfaces_PCA_LDA' * (Eigenfaces_PCA' * TestSet);

id_errors = zeros(6,1);
for i = 1:size(TestP_PCA_LDA,2)
    point = TestP_PCA_LDA(:,i);
    y = zeros(6,1);

    for j = 1:6
        y(j) = mvnpdf(point',mu_values(:,j)',cov_values{j});
    end

    [M,indices] = sort(y,'descend');
    class_id = indices(1);
    class_id = class_id - 1;
    real_id = floor(i / 10);
    if real_id == 6
        real_id = 5;
    end
    error = (class_id ~= real_id);
    id_errors(real_id + 1) = id_errors(real_id + 1) + error;
end
```

```
disp('Q5 Part E PCA + LDA error rates, from person 1 to 6, in
 percent:')
id_errors = ((id_errors / 10)')*100;
disp(id_errors);
disp('Average error in percent:')
disp(mean(id_errors));
disp(' ');
```

Published with MATLAB® R2018b