Homework week 12

Fishers Discriminant Analysis

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import eigh
np.random.seed(1)
```

Generate Data 1

In [ ]:
```
n = 100
x = np.random.randn(n, 2)
x[:n//2, 0] -= 4
x[n//2:, 0] += 4
x -= np.mean(x, axis=0)
x1 = x
y1 = np.concatenate([np.ones(n//2), 2 * np.ones(n//2)])
```

Compute FDA

In [ ]:
```
def FDA(x,y):
    m1 = np.mean(x[y == 1, :], axis=0).reshape(-1,2)
    m2 = np.mean(x[y == 2, :], axis=0).reshape(-1,2)

    # Center the data for each class
    x1 = x[y == 1, :] - m1
    x2 = x[y == 2, :] - m2

    # Calculate the between-class scatter matrix
    S_B = (n / 2) * (m1.T @ m1 + m2.T @ m2)
    # Calculate the within-class scatter matrix
    S_W = (x1.T @ x1) + (x2.T @ x2)

    # Compute the Fisher's discriminant vector and eigenvalue
    eigenvalues, eigenvectors = eigh(S_B, S_W, eigvals=(1, 1))
    v = eigenvalues[0]
    t = eigenvectors[:, 0]
    return t,v
```

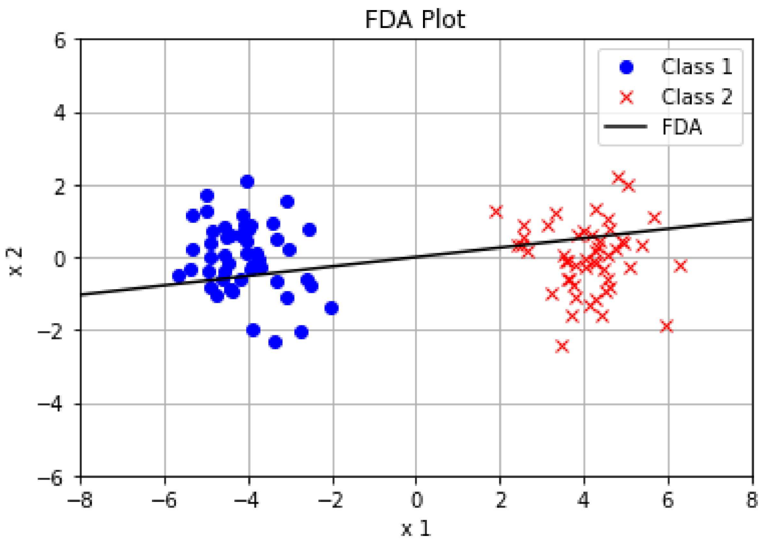
In [ ]:
```
t,v = FDA(x1,y1)
print(t,v)
```

```
[0.11823351 0.01532725] 23.377361848491706
```

Plotting

In [ ]:
```
plt.figure(1)
plt.clf()
plt.axis([-8, 8, -6, 6])

plt.plot(x1[y1 == 1, 0], x1[y1 == 1, 1], 'bo')
plt.plot(x1[y1 == 2, 0], x[y1 == 2, 1], 'rx')
```

```
plt.plot(np.array([-t[0], t[0]]) * 99, np.array([-t[1], t[1]]) * 99, color = "blac

plt.xlabel('x 1')
plt.ylabel('x 2')
plt.title('FDA Plot')
plt.legend(['Class 1', 'Class 2', 'FDA'])
plt.grid()
plt.show()
```



Generate Data 2

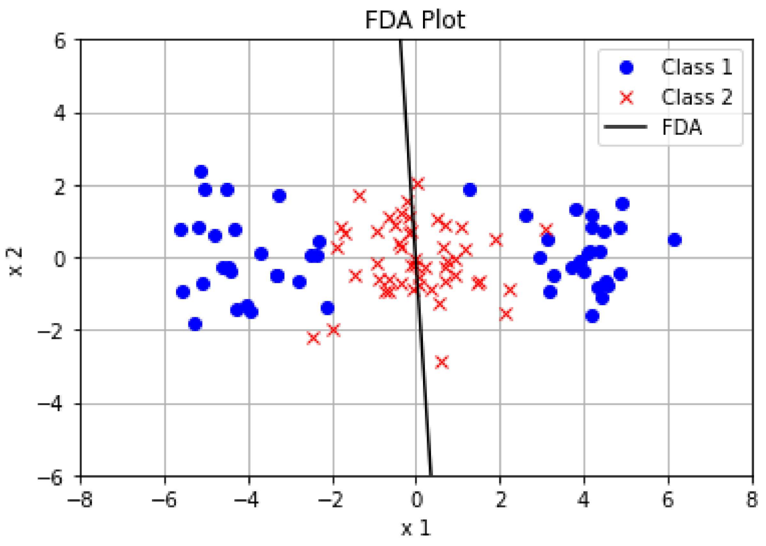
In [ ]:
```
n = 100
x = np.random.randn(n, 2)
x[:n // 4, 0] -= 4
x[n // 4:n // 2, 0] += 4
x = x - np.mean(x, axis=0)
x2 = x
y2 = np.concatenate((np.ones(n // 2), 2 * np.ones(n // 2)))
```

In [ ]:
```
t,v = FDA(x2,y2)
print(t,v)
```

```
[-0.00597065  0.09800593] 0.004195298716675413
```

Plotting

In [ ]:
```
plt.figure(2)
plt.clf()
plt.axis([-8, 8, -6, 6])

plt.plot(x2[y2 == 1, 0], x2[y2 == 1, 1], 'bo')
plt.plot(x2[y2 == 2, 0], x2[y2 == 2, 1], 'rx')
plt.plot(np.array([-t[0], t[0]]) * 99, np.array([-t[1], t[1]]) * 99, color = "blac

plt.xlabel('x 1')
plt.ylabel('x 2')
plt.title('FDA Plot')
plt.legend(['Class 1', 'Class 2', 'FDA'])
plt.grid()
plt.show()
```

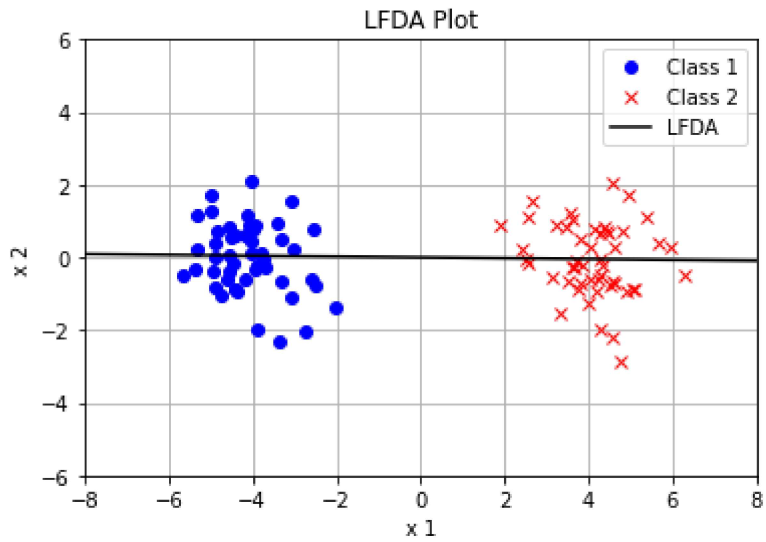


Compute LFDA

In [ ]:
```
def LFDA(x,y):
    # LFDA
    Sw = np.zeros((2, 2))
    Sb = np.zeros((2, 2))

    for j in range(1, 3):
        p = x[y == j, :]
        nj = np.sum(y == j)

        W = np.exp(-np.sum((p[:, None] - p[None]) ** 2, axis=2))
        G = p.T @ (p.T*np.sum(W, axis=1)).T - p.T @ W @ p

        Sb += G / n + p.T @ p * (1 - nj / n) + (p**2).T @ (p**2) / n
        Sw += G / nj

    # Compute the eigenvectors and eigenvalues
    eigenvalues, eigenvectors = eigh((Sb + Sb.T) / 2, (Sw + Sw.T) / 2, eigvals=(1,

    t = eigenvectors.flatten()
    v = eigenvalues[0]
    return t,v
```

Plotting Data - 1

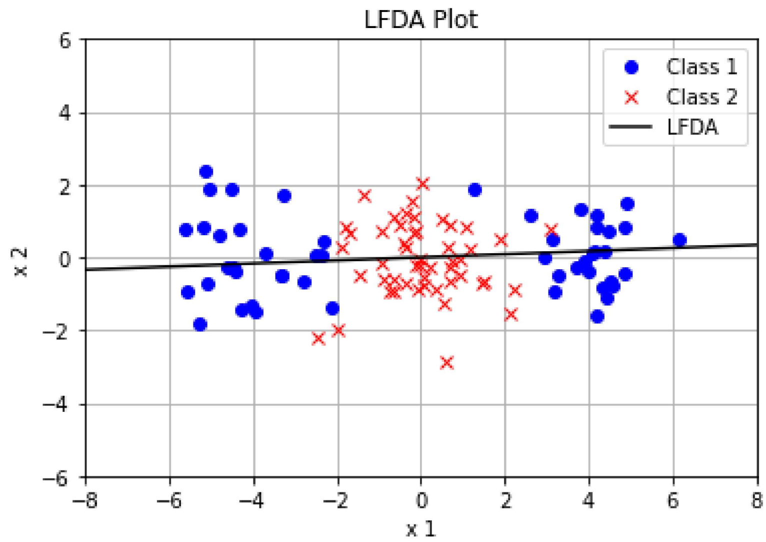
In [ ]:
```
t,v = LFDA(x1,y1)
plt.figure(3)
plt.clf()
plt.axis([-8, 8, -6, 6])

plt.plot(x1[y1 == 1, 0], x1[y1 == 1, 1], 'bo')
plt.plot(x1[y1 == 2, 0], x[y1 == 2, 1], 'rx')
plt.plot(np.array([-t[0], t[0]]) * 99, np.array([-t[1], t[1]]) * 99, color = "blac

plt.xlabel('x 1')
plt.ylabel('x 2')
plt.title('LFDA Plot')
plt.legend(['Class 1', 'Class 2', 'LFDA'])
plt.grid()
plt.show()
```



Plotting Data - 2

In [ ]:

```
t,v = LFDA(x2,y2)
plt.figure(4)
plt.clf()
plt.axis([-8, 8, -6, 6])

plt.plot(x2[y2 == 1, 0], x2[y2 == 1, 1], 'bo')
plt.plot(x2[y2 == 2, 0], x2[y2 == 2, 1], 'rx')
plt.plot(np.array([-t[0], t[0]]) * 99, np.array([-t[1], t[1]]) * 99, color = "blac

plt.xlabel('x 1')
plt.ylabel('x 2')
plt.title('LFDA Plot')
plt.legend(['Class 1', 'Class 2', 'LFDA'])
plt.grid()
plt.show()
```



Advanced Data Analysis
Homework Week - 12

Aswin Vijay

Question 2

We are asked to derive the following pairwise expressions for Local Fischer Discriminant Analysis, given the number of samples as n and number of samples of class l being n_l. The within-class scatter matrix S^w and the between-class scatter matrix S^b are,

$$\begin{aligned} S^w &= \frac{1}{2}\sum_{i,i'=1}^{n} Q^w_{i,i'}(x_i - x_{i'})(x_i - x_{i'})^T \\ S^b &= \frac{1}{2}\sum_{i,i'=1}^{n} Q^b_{i,i'}(x_i - x_{i'})(x_i - x_{i'})^T \end{aligned} \tag{1}$$

where,

$$\begin{aligned} Q^w_{i,i'} &= \begin{cases} \frac{1}{n_l} > 0 & (y_i = y_{i'} = l) \\ 0 & (y_i \neq y_{i'}) \end{cases} \\ Q^b_{i,i'} &= \begin{cases} \frac{1}{n} - \frac{1}{n_l} > 0 & (y_i = y_{i'} = l) \\ \frac{1}{n} & (y_i \neq y_{i'}) \end{cases} \end{aligned} \tag{2}$$

Derivation

From Fischer Discriminant Analysis we have the following scatter matrices given total number of classes c,

$$\begin{aligned} S^w &= \sum_{l=1}^{c}\sum_{i:y_i=l}(x_i - \mu_l)(x_i - \mu_l)^T \\ S^b &= \sum_{i=1}^{c} n_l(\mu_l - \mu)(\mu_l - \mu)^T \end{aligned} \tag{3}$$

where μ_l is the smaple mean of class l,

$$\begin{aligned} \mu_l &= \frac{1}{n_l}\sum_{i:y_i=l} x_i \\ \mu &= \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}\sum_{l=1}^{c} n_l\mu_l \end{aligned} \tag{4}$$

Starting with S^w in Eq.3 and substituting the class sample mean μ_l we get,

$$
\begin{aligned}
S^w &= \sum_{l=1}^{c} \sum_{i:y_i=l} (x_i - \mu_l)(x_i - \mu_l)^T \\
&= \sum_{l=1}^{c} \sum_{i:y_i=l} \left(x_i - \frac{1}{n_l} \sum_{i':y_{i'}=l} x_{i'} \right) \left(x_i - \frac{1}{n_l} \sum_{i':y_{i'}=l} x_{i'} \right)^T \\
&= \sum_{i=1}^{n} x_i x_i^T - \sum_{l=1}^{c} \frac{1}{n_l} \sum_{i,i':y_i=y_{i'}=l} x_i x_{i'}^T \\
&= \sum_{i=1}^{n} \left(\sum_{i'=1}^{n} Q^w_{i,i'} \right) x_i x_i^T - \sum_{i,i'=1}^{n} Q^w_{i,i'} x_i x_{i'}^T \quad \text{Using Eq. 2} \\
&= \frac{1}{2} \sum_{i,i'=1}^{n} Q^w_{i,i'} (x_i x_i^T - x_i x_{i'}^T - x_{i'}^T x_i + x_{i'}^T x_{i'}^T) \\
&= \frac{1}{2} \sum_{i,i'=1}^{n} Q^w_{i,i'} (x_i - x_{i'})(x_i - x_{i'})^T
\end{aligned}
\tag{5}
$$

Using mixture scatter matrix formula we have,

$$
\begin{aligned}
S^m &= S^w + S^b \\
&= \sum_{i=1}^{n} (x_i - \mu)(x_{i'} - \mu)
\end{aligned}
\tag{6}
$$

Using 6 we derive S^b as,

$$
\begin{aligned}
S_b &= \sum_{i=l}^{n} x_i x_i^T - \frac{1}{n} \sum_{i,i'=1}^{n} x_i x_{i'}^T - S^w \\
&= \sum_{i=1}^{n} \left(\sum_{i'=1}^{n} \frac{1}{n} \right) x_i x_i^T - \sum_{i,i'=1}^{n} \frac{1}{n} x_i x_{i'}^T - S^w \\
&= \frac{1}{2} \sum_{i,i'=1}^{n} \left(\frac{1}{n} - Q^w_{i,i'} \right) (x_i - x_{i'})(x_i - x_{i'})^T \\
&= \frac{1}{2} \sum_{i,i'=1}^{n} Q^b_{i,i'} (x_i - x_{i'})(x_i - x_{i'})^T
\end{aligned}
\tag{7}
$$

Thus we derive both the required scatter matrices for LFDA.

References

- *Masashi Sugiyama. 2007. Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis. J. Mach. Learn. Res. 8 (5/1/2007), 1027–1061.*

Advanced Data Analysis
Homework Week - 12

Aswin Vijay

Question 3

We need to prove that,

$$\begin{aligned} \operatorname{rank}\left(S^b\right) &\leq c-1 \\ S_b &= \sum_{y=1}^{c} n_y \mu_y \mu_y^T \end{aligned} \tag{1}$$

It can be also written as,

$$S_b = \sum_{y=1}^{c} n_y \left(\mu_y - \mu\right)\left(\mu_y - \mu\right)^T \tag{2}$$

where μ_y denotes the mean of training samples in class y. μ is c is the number of classes.

$$\begin{aligned} \mu_y &= \frac{1}{n_y} \sum_{i:y_i=y} x_i \\ \mu &= \frac{1}{n} \sum_{i=1}^{n} x_i \end{aligned} \tag{3}$$

If we perform a rank analyis of Eq. 1, we see that S_b is of the form $\sum \mu_y \mu_y^T$. So the rank of S_b is rank of $\mu_y \mu_y^T$, where μ_y is a column vector. The sum in Eq.2 can be represented by the following matrix product,

$$\begin{aligned} S_b &= \mathrm{M}\mathrm{M}^T \text{ where} \\ \mathrm{M} &= \sqrt{n_y}[\mu_1 - \mu, \mu_2 - \mu, ..., \mu_c - \mu] \end{aligned} \tag{4}$$

Now we use the following property,
- For a given real matrix A, $\operatorname{rank}(A) = \operatorname{rank}\left(AA^T\right) = \operatorname{rank}\left(A^T A\right)$
- Rank of S_b is therefore rank of M.

Since there are c classes, the column space of M is contained within the c-dimensional space spanned by the c class means. However, the class means are not all linearly independent since the overall mean μ is already in the column space of M as $\mu_i - \mu$. Therefore, the maximum number of linearly independent vectors in the column space of M is $c - 1$.

Thus, the rank of S^b is at most $c - 1$. Thus proven.

ADA Homework Week 12 - Problem 4

Least Squares probabilistic classification based on a Gaussian kernel model

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1)
```

Make Data

In [ ]:
```
# Train data
n = 90
c = 3
y = np.repeat(np.arange(1, c + 1), n/c)

x = np.random.randn(n // c, c) + np.tile(np.linspace(-3, 3, c), (n // c, 1))
x = x.flatten(order='F').reshape(-1,1)

# Test data
N = 100
X = np.linspace(-5, 5, N).reshape(-1, 1)
```

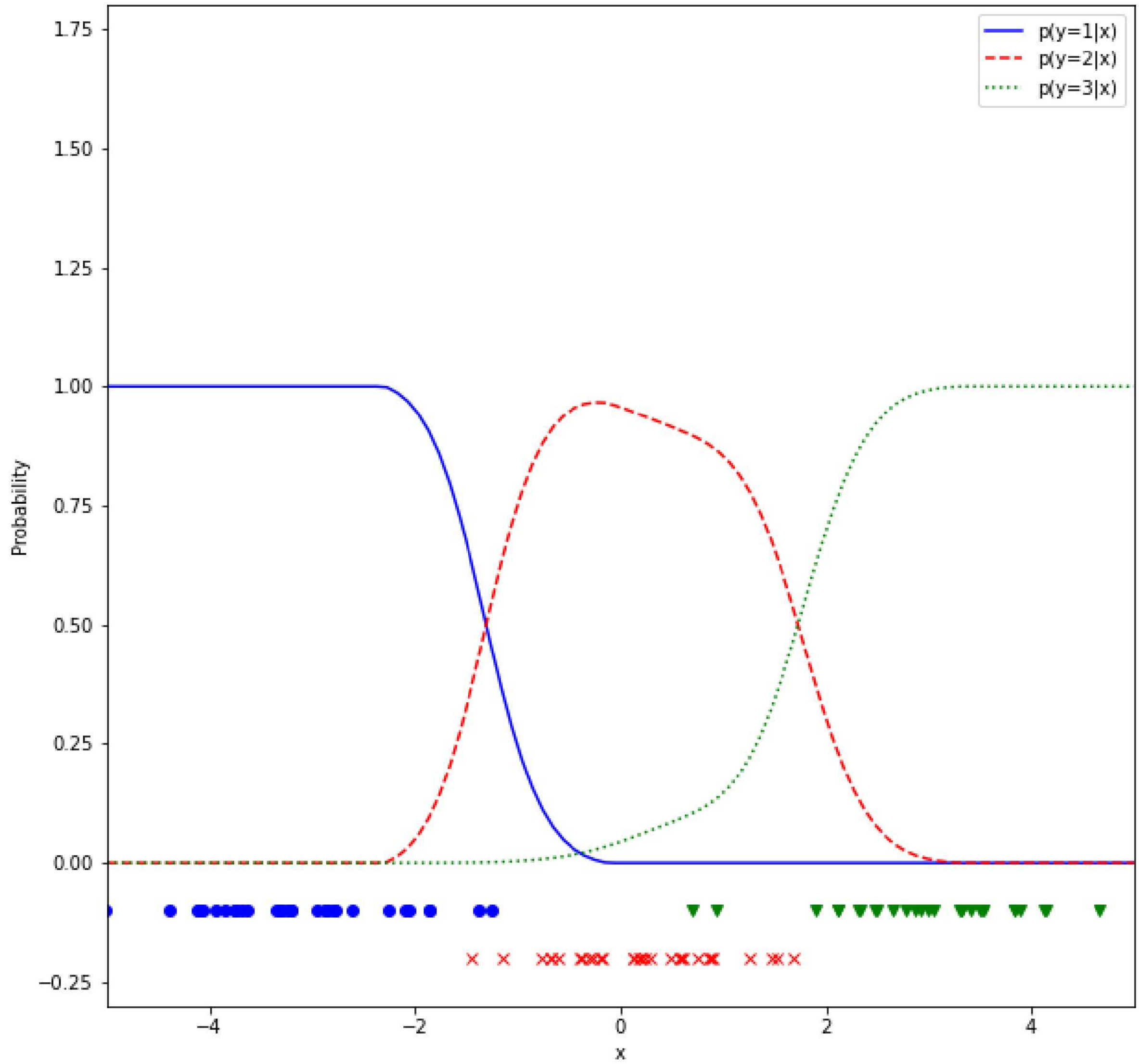
Compute the Least Squares Classification using Gaussian Kernel

In [ ]:
```
def LSPCG(x, y, X):

    # Reference: Masashi Sugiyama. 2015. Introduction to Statistical Machine Learn

    l = 0.1
    hh = 2 * 1 ** 2

    # Gaussian Kernel matrix for train and test data.
    k = np.exp(-np.sum((x[:, None] - x[None]) ** 2, axis=2))/hh
    K = np.exp(-np.sum((X[:, None] - x[None]) ** 2, axis=2))/hh

    Kt = np.zeros((N, c))

    for yy in range(1, c + 1):
        yk = y == yy
        ky = k[:, yk]
        # Compute Least Squares Solution
        ty = np.linalg.solve(ky.T @ ky + l * np.eye(np.sum(yk)), ky.T @ yk)
        # replace negative values with 0
        Kt[:, yy - 1] = np.maximum(0, K[:, yk] @ ty)

    # Compute probabilities by diviing prob with sum of prob.
    ph = Kt / np.tile(np.sum(Kt, axis=1), (c, 1)).T

    return ph
```

In [ ]:
```
ph = LSPCG(x, y, X)
```

Plot Solution

In [ ]:
```
plt.figure(figsize=(10,10))
plt.clf()
plt.axis([-5, 5, -0.3, 1.8])
plt.plot(X, ph[:, 0], 'b-')
plt.plot(X, ph[:, 1], 'r--')
plt.plot(X, ph[:, 2], 'g:')
plt.plot(x[y == 1], -0.1 * np.ones(n // c), 'bo')
plt.plot(x[y == 2], -0.2 * np.ones(n // c), 'rx')
plt.plot(x[y == 3], -0.1 * np.ones(n // c), 'gv')
plt.xlabel("x")
plt.ylabel("Probability")
plt.legend(['p(y=1|x)', 'p(y=2|x)', 'p(y=3|x)'])
plt.show()
```



Advanced Data Analysis
Homework Week - 12

Aswin Vijay

Question 5: PCA derivation using maximum variance fromulation

Consider the dataset of observations $\{x_i\}$, $i = 1, ..., n$ and has dimension d. We then define an orthogonal basis $\left\{t_j | t_j \in \mathbb{R}^d\right\}_{j=1}^{m}$, where $m \leq d$ such that,

$$t_j^T t_{j'} = \begin{cases} 1 & (j = j') \\ 0 & (j \neq j') \end{cases}$$
$$TT^T = I, T = (t_i, ..., t_m)^T \in \mathbb{R}^{m \times d} \tag{1}$$

.

The orthogonal projection of sample x_i on this basis is then given by,

$$\sum_{j=1}^{m} (t_j^T x_i) \cdot t_j = T^T T x_i \tag{2}$$

Let the sample mean of the observations be,

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3}$$

The mean of the projected data is then

$$\sum_{j=1}^{m} (t_j^T \overline{x}) \cdot t_j = T^T T \overline{x} \tag{4}$$

The variance of the projeted data is then given by,

$$\frac{1}{n} \sum_{i=1}^{n} \left(T^T T x_i - T^T T \overline{x}\right)^2 \tag{5}$$

Let us define the covariance matrix as,

$$C = \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^T \tag{6}$$

Then the projected variance becomes,

$$\frac{1}{n} \sum_{i=1}^{n} \left(T^T T x_i - T^T T \overline{x}\right)^2 = T^T C T \tag{7}$$

We now maximise the projected variance $T^T C T$ with respect to T. This has to be a constrained maximisation to prevent $\|T\| \to \infty$. The constraint comes fromt the normalisation condition

$TT^T = I$. We introduce lagranges multiplier λ_1 to enforce the condition, the maximization objective then becomes,

$$\max_{T \in R^{m \times d}} T^T CT + \lambda_1 (I - TT^T)$$
$$T_{\text{PCA}} = \text{argmax}_{T \in R^{m \times d}} \text{ tr}(T^T CT) \text{ s.t } TT^T = I_m \tag{8}$$

Which is equivalent to the objective formulated by minimizing sum of projected errors. Setting derivative with respect to T equal to zero we get,

$$CT = \lambda_1 T \tag{9}$$

multiplying LHS by T^T we get,

$$T^T CT = \lambda_1 \tag{10}$$

So the variance will be a maximum when we set T equal to the eigenvector having the largest eigenvalue λ_1. The solution to the PCA objective is then obtained by eigen value decomposition of the C matrix and is given by,

$$T_{\text{PCA}} = U(\varepsilon_1, ..., \varepsilon_m)^T \tag{11}$$

- $\varepsilon_1, ..., \varepsilon_m$: The eigenvectors corresponding to the eigenvalues $\lambda_1 \geq \lambda_2 \geq ...\lambda_d$ of the eigenvalue problem $C\varepsilon = \lambda\varepsilon$.
- U: any $m \times m$ orthogonal matrix, where $U^{-1} = U^T$