MATH524 Final Exam: Solution by Zino Meyer ID 132611593

December 16, 2024

1. **SVD for space-time data matrix**

   Make an SVD analysis of the January standardized anomalies data of sea level pressure (SLP) at Darwin and Tahiti from 1961 to 2000. You can download the SLP data from this Final Exam on Canvas. The two data files are PSTANDdarwin.txt and PSTANDtahiti.txt.

   a. **Write an R code to organize the January data from 1961 to 2000 into a 2 ×40 space-time data matrix A. Put Darwin data in the first row and Tahiti in the second row. Explicitly print the first six columns of the data matrix A. Copy and paste the six columns of the data into your R code as comments indicated by #.**

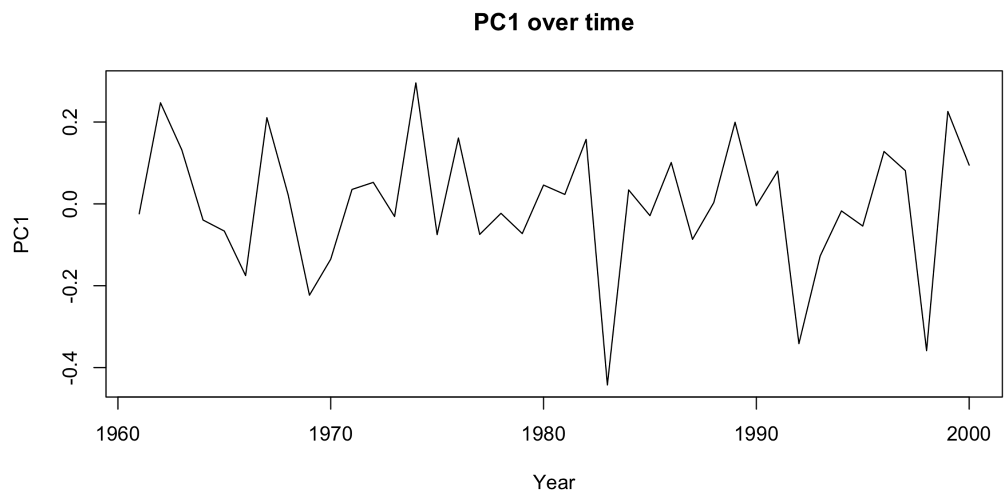
```
# a)
# Darwin data
darwin_data <- read.table("data/PSTANDdarwin.txt", header=F)
dim(darwin_data)
# [1] 65 13

darwin_jan <- darwin_data[11:50, 2] # rows for 1961-2000, Jan column

# Tahiti data
tahiti_data <- read.table("data/PSTANDtahiti.txt", header=F)
dim(tahiti_data)
# [1] 65 13

tahiti_jan <- tahiti_data[11:50, 2] # rows for 1961-2000, Jan column

# Bind to space-time data matirx
da_ta_jan = rbind(darwin = darwin_jan, tahiti = tahiti_jan)
da_ta_jan[, 1:6] # print first 6 columns
#        [,1] [,2] [,3] [,4] [,5] [,6]
# darwin  0.5 -1.4 -0.7  0.6 -0.3  0.7
# tahiti  0.1  1.8  1.0  0.0 -1.0 -1.5
```

b. **Make the SVD calculation for this space-time matrix A= UDVt, where Vt is the transpose matrix of V.**

```
# b)
svd_da_ta_jan <- svd(da_ta_jan)
svd_da_ta_jan
# $d
# [1] 9.231838 6.429866
#
# $u
#             [,1]      [,2]
# [1,] -0.6063979 0.7951614
# [2,]  0.7951614 0.6063979
#
# $v
#               [,1]          [,2]
# [1,] -0.024229499  0.0712643902
# [2,]  0.246998228 -0.0033763944
# [3,]  0.132112366  0.0077427595
# [4,] -0.039411301  0.0742001203
# [5,] -0.066426865 -0.1314096266
```

c. **Plot the first singular vector in the above temporal matrix V against time 1961 to 2000 as a time series curve, which is called the first principal component, denoted by PC1.**

```
# c)
plot(1961:2000,
    svd_da_ta_jan$v[, 1],
    type = "l",
    xlab = "Year", ylab = "PC1",
    main = "PC1 over time"
)
```



d. **Check the historical El Niño events between 1961 and 2000 from the Internet like https://origin.cpc.ncep.noaa.gov/products/analysis_monitoring/ensostuff/ONI_v5.php and interpret the three extreme values of PC1 in January 1983, January 1992, and January 1998 from the perspective of El Niño events. Here, extreme values mean either the local maxima or local minima for the PC1 curve in Part (c).**

   The three extreme values in 1983, 1992, and 1998 are all local minima in the PC1 curve. Comparing this with the historical data from El Niño, these points express above average sea surface temperatures in the Niño Region 3.4 (5°N - 5°S, 120° - 170°W), which corresponded with very strong El Niño periods. In January of ‘83, the historic data shows +2.2, +2.2, and +1.9 degrees anomalies for the 3-month averages that include January, on the basis of 30-year averages. For ‘92, the anomalies were +1.5, +1.7, and +1.6 degrees, and for ‘98, they were +2.4, +2.2, and +1.9 degrees, respectively. Thus, we can conclude that local minima of Darwin & Tahiti sea-level pressure data’s PC1 signal stark above average sea surface temperature anomalies in the Niño Region 3.4, and thus mostly result in an El Niño event. While PC1 captures a complex pattern, usually not expressible with words, these PC1 values could express correlation between Darwin & Tahiti SLP, meaning a local minima in the PC1 could be strong negative correlation between the SLP of the two locations, and thus capturing El Niño events.

e. **Interpret the first singular column vector in U, which is called the first empirical orthogonal function (EOF1), as weights of Darwin and Tahiti stations. *Hint: Read some website materials or books on El Niño and check if the two values of the EOF1 vector have the same sign or different signs. Here, different signs mean that one value is positive and another is negative; the same sign means that both values are positive or negative.***

   The first singular column vector in U is (-0.6063979, 0.7951614), so it’s roughly (-0.61, 0.8), which we could assign as weights to the Darwin and Tahiti station, respectively. This result shows that the pressure at these two stations varies inversely, which aligns with the known El Niño characteristics. During El Niño, the SLP in Tahiti tends to be lower than average, while in it is higher in Darwin. During La Niña, the opposite happens. Thus, the EOF1 successfully captures the spatial pattern: an increase in pressure at one station corresponds to a decrease at the other.

2. **The singular values of SVD and the eigenvalues of a covariance matrix**

   Let A be a space-time anomaly data matrix. Let C be a square covariance matrix defined by $C = AA^T$, where A^T is the transpose matrix of A. Let d_1 be the first singular value of A, and λ_1 be the first eigenvalue of C. Show that $\lambda_1 = d_1^2$.

   Given:

   - Definition of SVD:

   $$A = UDV^t$$

   - Definition of covariance matrix:

   $$C = AA^T$$

   Wanted:

   - Show that the first eigenvalue λ_1 of the covariance matrix C relates to the first singular value d_1 of A by $\lambda_1 = d_1^2$

   Proof:

   1. Rewriting covariance C in terms of SVD:
      - With definition of SVD, substituting A:

      $$C = \frac{1}{Y}(UDV^T)(UDV^T)^T$$

      - With properties of transpose: $(UDV^T)^T = (V^T)^T D^T U^T = VD^TU^T$:

      $$C = (UDV^T)VD^TU^T$$

      - Since D is diagonal, $D = D^T$:

      $$C = UDV^TVDU^T$$

- And since V is orthogonal, $V^TV = I$::

$$
\begin{aligned}
C &= UDIDU^T \\
&= UD^2U^T
\end{aligned}
$$

2. Show that diagonal values of D^2 are eigenvalues of C:

- Definition of eigenvalues λ & eigenvectors $\mathbf{v}$:

$$C\mathbf{v} = \lambda\mathbf{v}$$

- Considering columns $\mathbf{u}_i$ of U to be eigenvectors of C:

$$C\mathbf{u}_i = \lambda\mathbf{u}_i$$

- Substitute $C = UD^2U^T$, we can rewrite to:

$$
\begin{aligned}
C\mathbf{u}_i &= UD^2U^T\mathbf{u}_i \\
&= UD^2(U^T\mathbf{u}_i)
\end{aligned}
$$

- Since U is orthogonal, we get the standard basis vector e:

$$U^T\mathbf{u}_i = \mathbf{e}_i$$

- Substitute back:

$$
\begin{aligned}
C\mathbf{u}_i &= UD^2\mathbf{e}_i \\
&= U\left(d_i^2\mathbf{e}_i\right) \quad \text{(since } D^2 \text{ is diagonal)} \\
&= d_i^2U\mathbf{e}_i \\
&= d_i^2\mathbf{u}_i
\end{aligned}
$$

This shows that $\mathbf{u}_1$ is an eigenvector of C corresponding to the eigenvalue $\lambda_1 = d_1^2$

3. **Linear equations**

a. **Use R to solve the following linear equations $Ax = b$ to find the vector x, where A and b are given as follows:**

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 0 & 9 \\ 3 & 1 & 2 & 9 \end{bmatrix}$$

$$b = \begin{bmatrix} 2 \\ 5 \\ 0 \\ 1 \end{bmatrix}$$

```
# a)
A <- matrix(c(1, 2, 3, 4,
              4, 5, 6, 0,
              7, 8, 0, 9,
              3, 1, 2, 9), nrow = 4, byrow = TRUE)

b <- c(2, 5, 0, 1)

x <- solve(A, b)
x
# [1]  0.116402116  0.005291005  0.751322751 -0.095238095
```

b. **Using text, create your own word problem for two linear equations with two variables. *For example, I have two brothers, John and Mike. The difference of John's age minus Mike's is 4, and the sum of their ages is 20. What are the ages of John and Mike?***

Word problem 1:

Yoda has been training young Jedis three times as long as Luke has. The difference in their experience is 156. How long have Yoda and Luke trained Jedis, respectively?

Word problem 2:

Dylan spends three times minus 5 hours as long practicing surfing per week as Maya does. Together, the product of their weekly surfing practice is 120. How long does each practice surfing per week?

4. **Dot product, cross product, angle, and R plotting**

a. **Given the following two vectors $a = (2, 1)$, $b = (1, 2)$. Use hand calculation to find the dot product of these two vectors.**

$$a \cdot b = (2 \times 1) + (1 \times 2) = 2 + 2 = 4$$

b. **Use R and hand calculation to calculate the angle θ between the two vectors, as shown in Fig. 1. Use degrees, not radians, for the angle.**

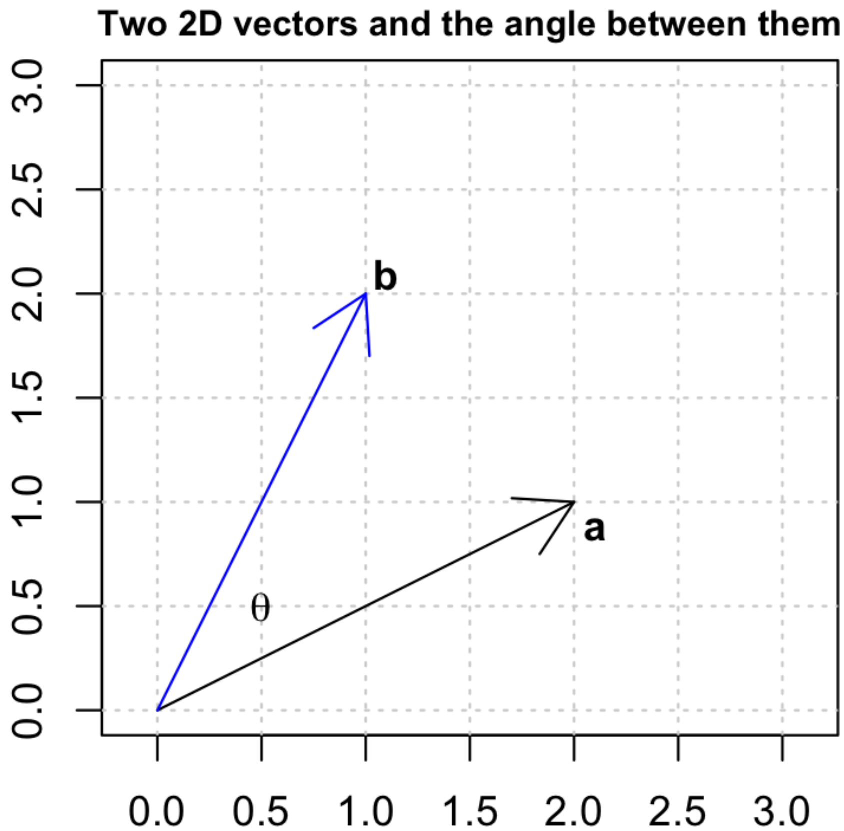
With definition of angles between two vectors $cos(\theta) = \frac{a \cdot b}{\|a\| \|b\|}$, we can calculate the angle theta:

$$cos(\theta) = \frac{4}{\sqrt{2^2 + 1^2} \times \sqrt{1^2 + 2^2}} = \frac{4}{5}$$

$$\theta = \cos^{-1}\left(\frac{4}{5}\right) = 36,8698976458^\circ$$

c. **Write an R code to reproduce Fig. 1**

```
# c) plot vectors
a <- c(2, 1)
b <- c(1, 2)

plot(c(0, 3), c(0, 3), type = "n",
     xlab = "", ylab = "", asp = 1)
grid()
title("Two 2D vectors and the angle between them",
      cex.main = 0.85, line = 0.5)

# vector a
arrows(0, 0, a[1], a[2], col = "black")
text(2.1, 0.7, "a", pos = 3, col = "black", font = 2)

# vector b
arrows(0, 0, b[1], b[2], col = "blue")
text(1.1, 1.9, "b", pos = 3, col = "black", font = 2)

# angle
text(0.5, 0.5, labels = expression(theta), col = "black", font = 2)
```



d. **Use hand calculation to find the cross product of the following two 3D vectors $u = (2,1,0)$, $v = (1,2,0)$. That is, find the vector $w = u \times v$. Use R to verify your result.**

With definition of cross product:

$$(u_1, u_2, u_3) \times (v_1, v_2, v_3) = (u_2 v_3 - u_3 v_2,\ u_3 v_1 - u_1 v_3,\ u_1 v_2 - u_2 v_1)$$

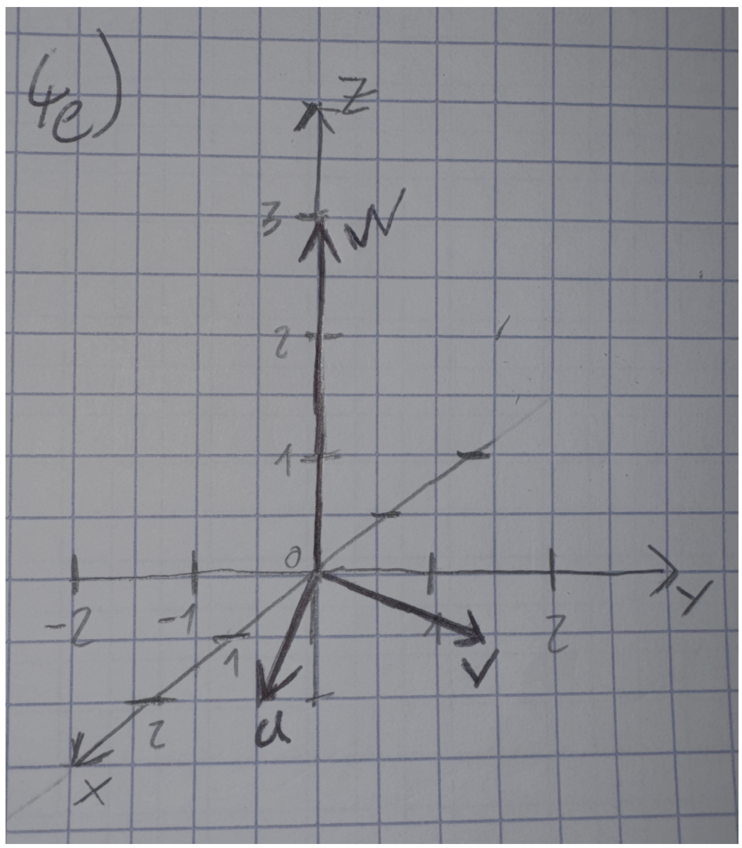
We can calculate it for u and v:

$$w = \left(1 \cdot\ 0 - 0 \cdot\ 2,\ 0 \cdot\ 1 - 2 \cdot\ 0,\ 2 \cdot\ 2 - 1 \cdot\ 1\right) = (0, 0, 3)$$

```
# 4 )
# d)
u <- c(2,1,0)
```

```
v <- c(1,2,0)

library(pracma)

cross_product <- cross(u, v)
cross_product
# [1] 0 0 3
```

e. **Use your hand to draw the three vectors u,v, and w.**



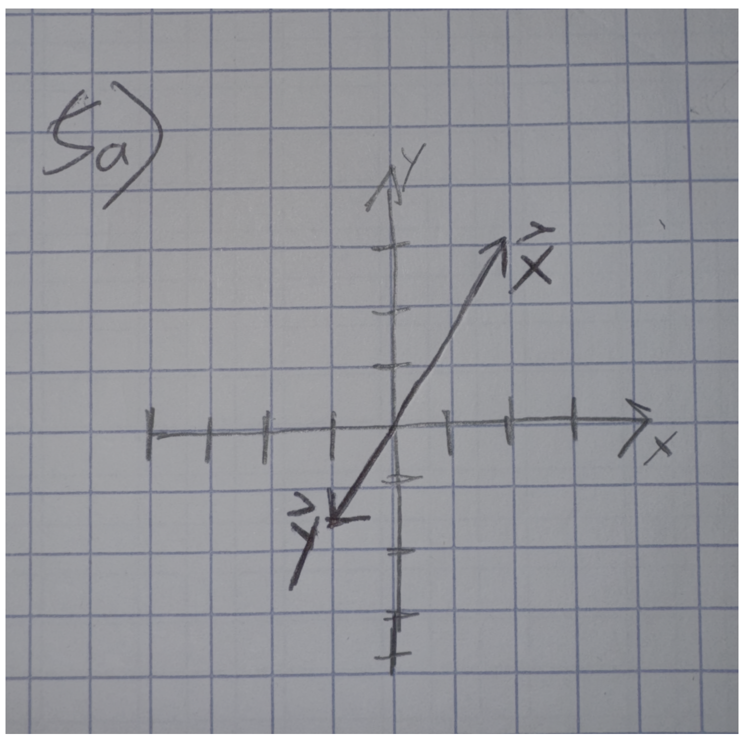
f. **Use text, the right-hand rule, and a real-life example to explain the meaning of the cross product of two vectors.**

The cross product of two vectors in 3D space is a vector perpendicular to both vectors. It can be thought of a pole perpendicularly sticking out of a plane formed by the two vectors. The right hand rule gives the direction of the resulting vector: if the index finger of the right hand (pointing straight) is the first vector, and the middle finger (pointing perpendicular to the index finger) is the second vector, then the extended thumb will give the direction of the resulting vector of the cross product. In our example above, u would be the index finger, v the middle finger, and w the thumb. The best real-life example is torque. The cross product of the position vector r (e.g. a wrench or the pedal of a bicycle, distance & direction from pivot point to where force is applied) with the force vector F (the magnitude & direction in which force is applied) results in the torque vector (direction & magnitude of rotational force).

5. **Concept problems**

   a. **Use your own words and hand-draw a diagram to explain what it means for two 2D vectors that are linearly dependent. Limited to 20-50 words.**

   In a 2D space, two vectors can only be represented by each other (be linearly dependant) if they point in the same or opposite direction, because there is a scalar s, for which $\mathbf{x} = s\mathbf{y}$.



   b. **If u is an eigenvector of a matrix A, and if v is also an eigenvector of the matrix A, then is the vector $u + v$ an eigenvector of the matrix A? Use a mathematical proof to justify your answer.**

   Let:

   - $\mathbf{u}$ be eigenvector of A with eigenvalue λ: $A\mathbf{u} = \lambda\mathbf{u}$

- $\mathbf{v}$ be eigenvector of A with eigenvalue μ: $A\mathbf{v} = \mu\mathbf{v}$

Hypothesis:

- Is there a $\mathbf{w} = \mathbf{u} + \mathbf{v}$, where $A\mathbf{w} = k\mathbf{w}$

Proof:

- Assume $\mathbf{w}$ is an eigenvector:

$$A(\mathbf{u} + \mathbf{v}) = k(\mathbf{u} + \mathbf{v})$$

- By distributive principle:

$$A\mathbf{u} + A\mathbf{v} \; = k\mathbf{u} + k\mathbf{v}$$

- Substitute eigenvalue definition for $\mathbf{u}$ and $\mathbf{v}$:

$$\lambda\mathbf{u} + \mu\mathbf{v} = k\mathbf{u} + k\mathbf{v}$$

- Bring on same side:

$$\begin{aligned} \lambda\mathbf{u} + \mu\mathbf{v} - (k\mathbf{u} + k\mathbf{v}) &= 0 \\ \lambda\mathbf{u} - k\mathbf{u} + \mu\mathbf{v} - k\mathbf{v} &= 0 \\ (\lambda - k)\mathbf{u} + (\mu - k)\mathbf{v} &= 0 \end{aligned}$$

- This expression can only be true if
  - the eigenvectors are linearly dependant:
    - → false, since eigenvectors with different eigenvalues are independent by definition
  - the coefficients are zero:
    - $\lambda - k = 0$ implies $\lambda = k$
    - $\mu - k = 0$ implies $\mu = k$
    - → thus, the eigenvalues are identical → $\lambda = \mu = k$
    - → thus, the eigenvectors correspond to the same eigenvalue

Concluding:

- If $\mathbf{w} = \mathbf{u} + \mathbf{v}$ should be an eigenvector of A, $\mathbf{u}$ and $\mathbf{v}$ have to correspond to the same eigenvalue,
- otherwise, $\mathbf{w}$ is ***<u>not</u>*** an eigenvector of A.

c. **Given that**

$$A = \begin{bmatrix} 5 & 1 \\ 0 & 3 \end{bmatrix}$$
$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

i. **Show by hand, not by R, that vector (1,1) is not an eigenvector of this matrix A.**

Test:

$$\begin{aligned} Av &= \lambda v \\ \begin{bmatrix} 5 & 1 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} &= \lambda \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \begin{bmatrix} (5 \times 1) + (1 \times 1) \\ (0 \times 1) + (3 \times 1) \end{bmatrix} &= \lambda \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 6 \\ 3 \end{bmatrix} &= \lambda \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned}$$

There is no scalar λ that satisfies this equation, thus the vector (1,1) is no eigenvector of the matrix A.

ii. **According to the following matrix expression $P(x_1, x_2) = x^t A^t A x$ write down by hand the second-order polynomial of x_1 and x_2 in the following form $P(x1, x2) = ax_1^2 + bx_1x_2 + cx_2^2$. You need to find a, b and c.**

- Calculate all values in $x^t A^t A x$:

$$A^t = \begin{bmatrix} 5 & 0 \\ 1 & 3 \end{bmatrix}$$

$$A^t A = \begin{bmatrix} 5 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 5 & 1 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 25 & 5 \\ 5 & 10 \end{bmatrix}$$

$$x^t = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

- Expand $x^t A^t A x$:

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 25 & 5 \\ 5 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Compute the polynomial with matrix multiplication:

$$
\begin{aligned}
P(x_1, x_2) &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 25 & 5 \\ 5 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\
&= \begin{bmatrix} 25x_1 + 5x_2 & 5x_1 + 10x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\
&= 25x_1^2 + 5x_1x_2 + 5x_1x_2 + 10x_2^2 \\
&= 25x_1^2 + 10x_1x_2 + 10x_2^2
\end{aligned}
$$

- With the form $P(x_1, x_2) = ax_1^2 + bx_1x_2 + cx_2^2$, we find:

$$
\begin{aligned}
a &= 25 \\
b &= 10 \\
c &= 10
\end{aligned}
$$

iii. **Use R to compute all the eigenvectors and eigenvalues of $A^t A$.**

```
# 5)
# c)
# iii)
A <- matrix(c(5, 1, 0, 3), nrow = 2, byrow = TRUE)
A
#      [,1] [,2]
# [1,]    5    1
# [2,]    0    3

ATA <- t(A) %*% A
ATA
#      [,1] [,2]
# [1,]   25    5
# [2,]    5   10

eigen_result <- eigen(ATA)

eigenvalues <- eigen_result$values
print(eigenvalues)
# [1] 26.513878  8.486122

eigenvectors <- eigen_result$vectors
print(eigenvectors)
#            [,1]       [,2]
```

```
# [1,] -0.9570920  0.2897841
# [2,] -0.2897841 -0.9570920
```

iv. **Let (x_1, x_2) be equal to the first unit eigenvector of A^tA computed in (ii). Use R to compute the numerical value of $P(x_1, x_2)$.**

```
# iv)
x <- eigenvectors[,1] # use eigenvectors value from above
x
# [1] -0.9570920 -0.2897841

x1 <- x[1]
x2 <- x[2]

# Calculate the polynomial
polyn <- 25 * x1^2 + 10 * x1 * x2 + 10 * x2^2
polyn
# [1] 26.51388
```

The numerical value of $P \approx 26.514$

6. **From a photo to a data matrix**

   a. **Use R package `imager` to read Sam's photo file `sam.png` and produce a grayscale data matrix `graydat`. Print the first four rows and first three columns of the data matrix. Put this part of the print out on your code as comments marked by #. The figure can be downloaded from the exam site on Canvas.**

```
# 6)
# a)
library("imager")
sam <- load.image("data/SamPhoto.png")
dim(sam)
# [1] 430 460   1   3

graydat <- grayscale(sam)
dim(graydat)
# [1] 430 460   1   1

print(graydat[1:4, 1:3])
```

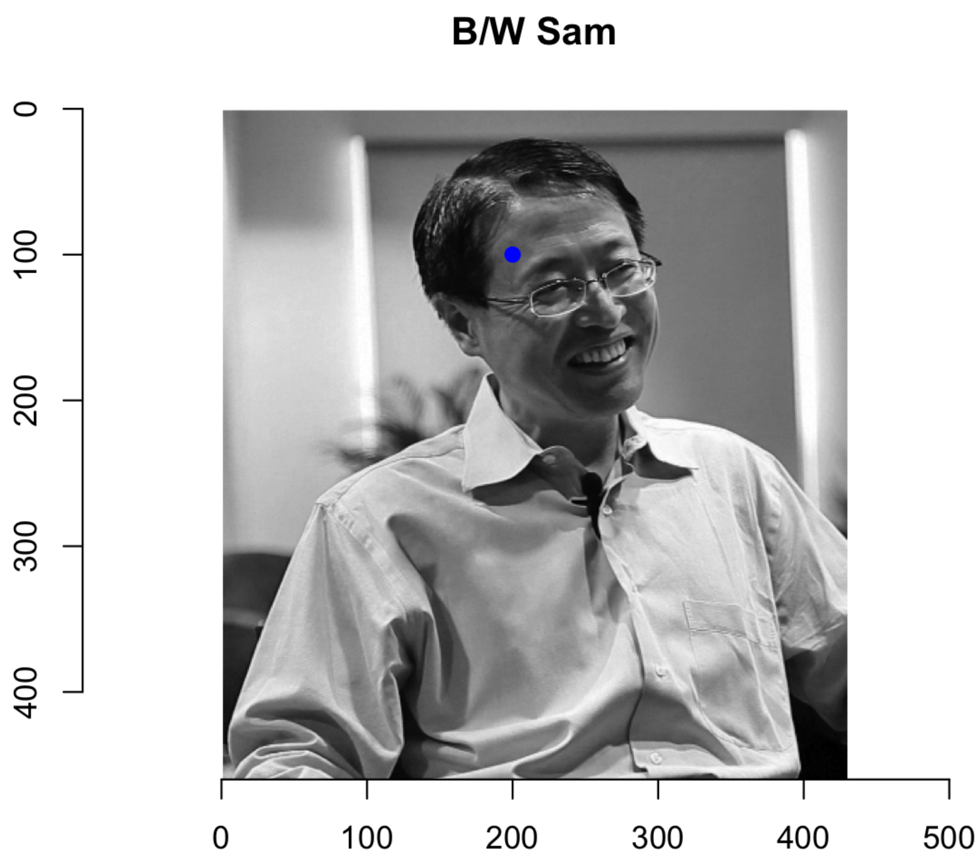
```
#            [,1]      [,2]      [,3]
# [1,] 0.6075294 0.6126667 0.6215686
# [2,] 0.6072549 0.6112549 0.6186667
# [3,] 0.6053725 0.6098431 0.6107843
# [4,] 0.6158431 0.6192941 0.6158431
```

b. **Use R to find the maximum and minimum values of `graydat`.**

```
# b)
print(max(graydat))
# [1] 1

print(min(graydat))
# [1] 0.1607843
```

c. **Plot a grayscale photo of Sam. According to the graydat matrix' rows and columns, use R command `points()` to mark the location of the 100th row and 200th column with a blue solid round point on the grayscale photo.**

```
# c)
plot(graydat,
     xlim = c(0, 430), ylim = c(460, 0),
     main = 'B/W Sam')
points(200, 100, col = "blue", pch = 19)
```



7. **Machine learning**

   a. **K-means: Based on the K-means principle of minimal tWCSS described in Section 3.1 of our textbook, use both hand-calculation and R to determine the two clusters from the following three points:** $P_1(1,1), P_2(1,0), P_3(3,4)$

   Hand calculation:

   - Find tWCSS for all cases:
     - Case 1: $C_1(P_1, P_2), C_2(P_3)$
       - cluster center 1: $\mu_1 = (\frac{1+1}{2}, \frac{1+0}{2}) = (1, 0.5)$
       - cluster center 2: $\mu_2 = (3, 4)$
       - tWCSS:

$$
\begin{aligned}
\mathrm{WCSS}_1 &= (P_1-\mu_1)^2+(P_2-\mu_1)^2\\
&= ((1-1)^2+(1-0.5)^2)+((1-1)^2+(0-0.5)^2)\\
&= 0.5\\
\mathrm{WCSS}_2 &= (P_3-\mu_2)^2\\
&= (3-3)^2+(4-4)^2\\
&= 0\\
\mathrm{tWCSS} &= \mathrm{WCSS}_1+\mathrm{WCSS}_2 = 0.5+0 = 0.5
\end{aligned}
$$

- Case 2: $C_1(P_1, P_3), C_2(P_2)$
  - cluster center 1: $\mu_1 = (\frac{1+3}{2}, \frac{1+4}{2}) = (2, 2.5)$
  - cluster center 2: $\mu_2 = (1, 0)$
  - tWCSS:

$$
\begin{aligned}
\mathrm{WCSS}_1 &= (P_1-\mu_1)^2+(P_3-\mu_1)^2\\
&= ((1-2)^2+(1-2.5)^2)+((3-2)^2+(4-2.5)^2)\\
&= 6.5\\
\mathrm{WCSS}_2 &= (P_2-\mu_2)^2\\
&= (1-1)^2+(0-0)^2\\
&= 0\\
\mathrm{tWCSS} &= \mathrm{WCSS}_1+\mathrm{WCSS}_2 = 6.5+0 = 6.5
\end{aligned}
$$

- Case 3: $C_1(P_2, P_3), C_2(P_1)$
  - cluster center 1: $\mu_1 = (\frac{1+3}{2}, \frac{0+4}{2}) = (2, 2)$
  - cluster center 2: $\mu_2 = (1, 0)$
  - tWCSS:

$$
\begin{aligned}
\mathrm{WCSS}_1 &= (P_2-\mu_1)^2+(P_3-\mu_1)^2\\
&= ((1-2)^2+(0-2)^2)+((3-2)^2+(4-2)^2)\\
&= 10\\
\mathrm{WCSS}_2 &= (P_1-\mu_2)^2\\
&= (1-1)^2+(1-1)^2\\
&= 0\\
\mathrm{tWCSS} &= \mathrm{WCSS}_1+\mathrm{WCSS}_2 = 10+0 = 10
\end{aligned}
$$

- Result: Case 1 has the smallest tWCSS of 0.5, thus we can choose its means as cluster centers:

- cluster center 1: $\mu_1 = (1, 0.5)$
- cluster center 2: $\mu_2 = (3, 4)$

R code calculation:

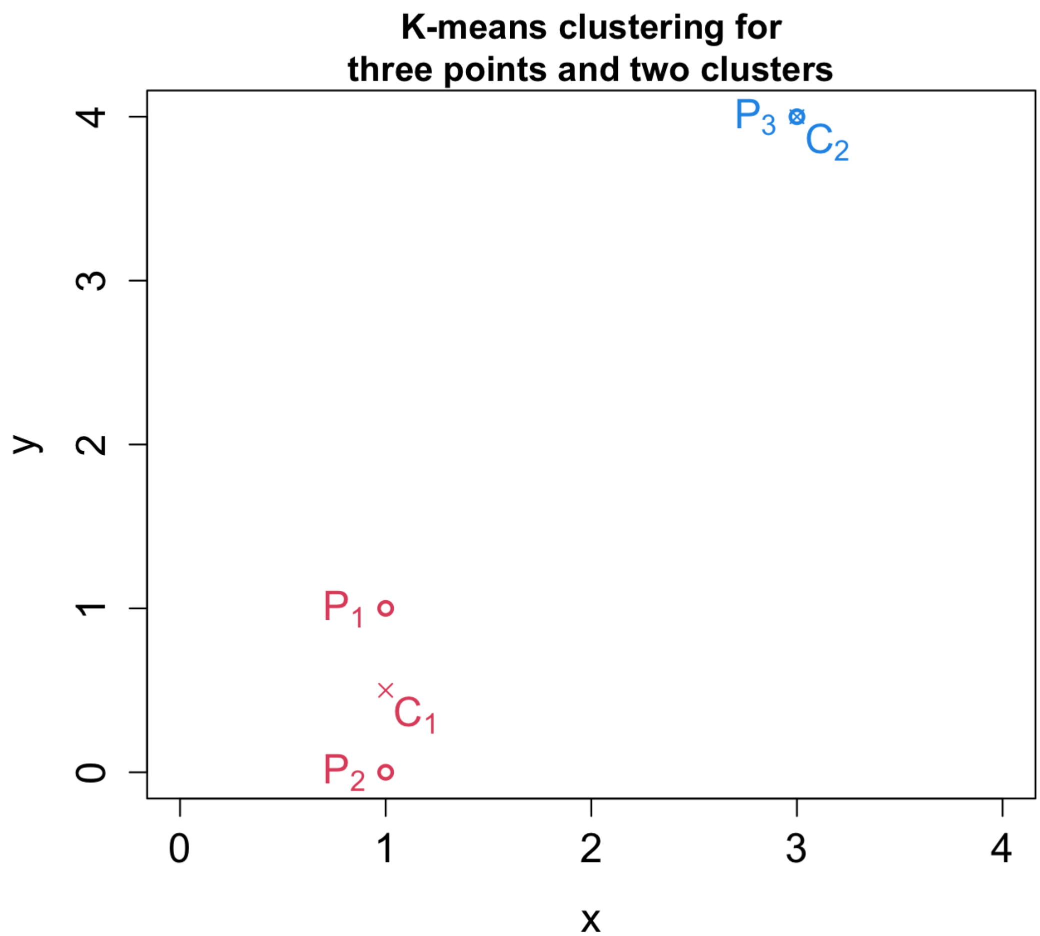
```
# a)
N = 3; K = 2
mydata <- matrix(c(1, 1, 1, 0, 3, 4),
                 nrow = N, byrow = TRUE)
p1 = mydata[1, ]
p2 = mydata[2, ]
p3 = mydata[3, ]

### Manual calculation
# Case 1: C1 = (P1, P2)
mu1_c1 = (p1 + p2)/2 # mean of points in C1
mu2_c1 = p3 # mean of points in C2
tWCSS_c1 = norm(p1 - mu1_c1, type = '2')^2 +
  norm(p2 - mu1_c1, type = '2')^2 +
  norm(p3 - mu2_c1, type = '2')^2
tWCSS_c1
# [1] 0.5

# Case 2: C1 = (P1, P3)
mu1_c2 = (p1 + p3)/2
mu2_c2 = p2
tWCSS_c2 = norm(p1 - mu1_c2, type = '2')^2 +
  norm(p3 - mu1_c2, type = '2')^2 +
  norm(p2 - mu2_c2, type = '2')^2
tWCSS_c2
# [1] 6.5

# Case 3: C1 = (P2, P3)
mu1_c3 = (p2 + p3)/2
mu2_c3 = p1
tWCSS_c3 = norm(p2 - mu1_c3, type = '2')^2 +
  norm(p3 - mu1_c3, type = '2')^2 +
  norm(p1 - mu2_c3, type = '2')^2
tWCSS_c3
# [1] 10
```

```
# Use first case to get optimal cluster means:
mu1_c1
# [1] 1.0 0.5
mu2_c1
# [1] 3 4

### Calculation using kmeans() function
Kclusters = kmeans(mydata, K)

Kclusters$tot.withinss
# [1] 0.5

centers <- Kclusters$centers
C1 <- centers[1, ]
C1
# [1] 1.0 0.5

C2 <- centers[2, ]
C2
# [1] 3 4
```

b. **Use R to plot the three points and the two centers on a figure similar to Fig. 3.1 in the textbook.**

```
# b)
cluster <- Kclusters$cluster
cluster
# [1] 2 2 1

# Plot
plot(mydata[, 1], mydata[, 2],
     lwd = 2,
     xlim = c(0, 4), ylim = c(0, 4),
     xlab = "x", ylab = "y",
     col = cluster * 2,
     main = "K-means clustering for
three points and two clusters",
     cex.lab = 1.4, cex.axis = 1.4
)
points(centers[, 1], centers[, 2],
       col = c(2, 4), pch = 4
```

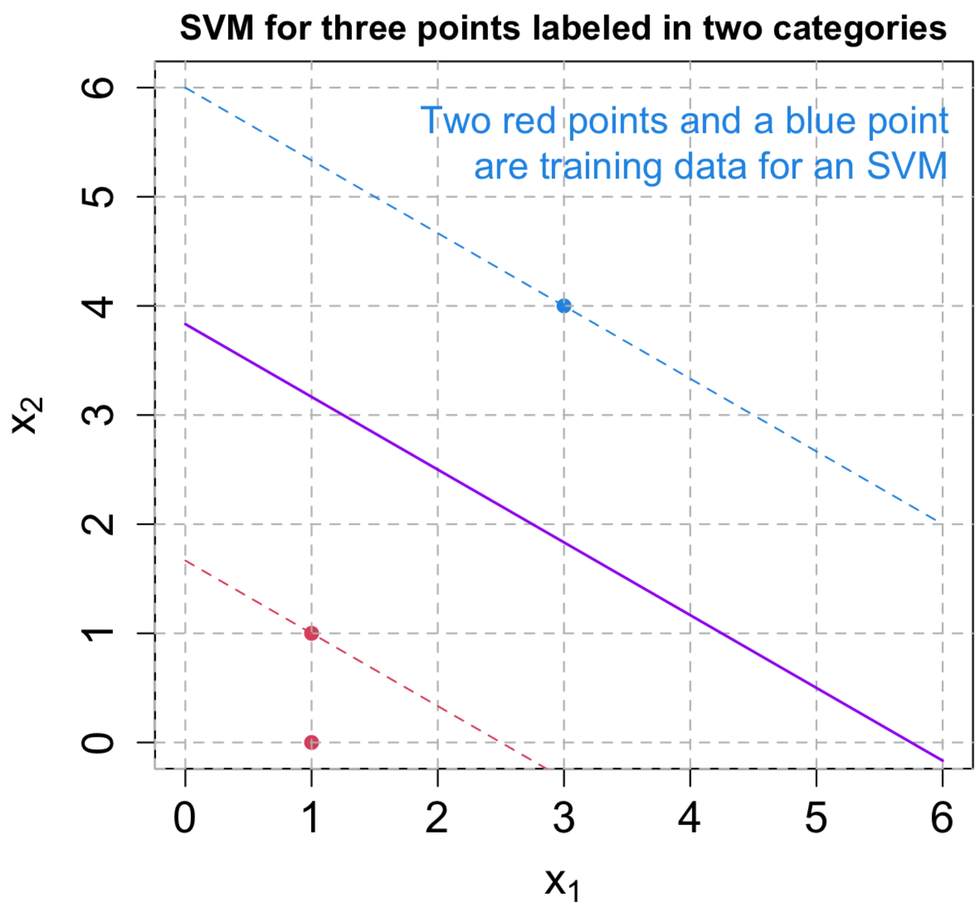
```
)
for (i in K:1) {
  text(centers[i, 1] + 0.15, centers[i, 2] - 0.15,
       bquote(C[.(i)]),
       cex = 1.4, col = i * 2)
}
for (i in 1:N) {
  text(mydata[i, 1] - 0.2, mydata[i, 2],
       bquote(P[.(i)]),
       cex = 1.4, col = cluster[i] * 2)
}

dev.off()
```



c. **Support vector machine: Use R to plot the two hyperplanes $w \cdot x - b = \pm 1$ and the separating plane $w \cdot x - b = 0$ for the maximum separation between the two clusters in (a). Use Fig. 3.6 in the textbook as a reference.**

```
# c)
x <- mydata
y <- Kclusters$cluster

# Train SVM
library(e1071)
svm3P = svm(y ~ ., data = data.frame(x, y = as.factor(y)),
            kernel = "linear", cost = 10,
            scale = FALSE,
            type = 'C-classification')

# Find hyper-planes
w = t(svm3P$coefs) %*% svm3P$SV
w
#              X1        X2
# [1,] -0.3076923 -0.4615385

b = svm3P$rho
b
# [1] -1.769231

# Calculate x1 and x2 for the hyper-planes
x1 = seq(0, 6, len = 31)
x20 = (b - w[1]*x1)/w[2]
x2p = (1 + b - w[1]*x1)/w[2]
x2n = (-1 + b - w[1]*x1)/w[2]

# Plot the SVM results
par(mar = c(4.5, 4.5, 2.0, 2.0))
plot(x, col = y * 2, pch = 19,
     xlim = c(0, 6), ylim = c(0, 6),
     xlab = bquote(x[1]), ylab = bquote(x[2]),
     cex.lab = 1.5, cex.axis = 1.5,
     main = 'SVM for three points labeled in two categories')
axis(2, at = (-2):8, tck = 1, lty = 2,
     col = "grey", labels = NA)
axis(1, at = (-2):8, tck = 1, lty = 2,
     col = "grey", labels = NA)

lines(x1, x20, lwd = 1.5, col = 'purple') # separating lane
lines(x1, x2p, lty = 2, col = 2) # hyper plane 1
```

```
lines(x1, x2n, lty = 2, col = 4) # hyper plane 2

text(4,5.5,
     "Two red points and a blue point
     are training data for an SVM ",
     cex = 1.3, col = 4)
```



d. **What are the supporting vectors in (c)? Calculate the distance D_m between the positive hyperplane and the negative hyperplane according to the formula $D_m = 2/|w|$**

```
# d)
x1_sv = svm3P$SV

# support vector 1
x1_sv1 = x1_sv[1]
x2p_sv = (1 + b - w[1]*x1_sv1)/w[2]
sv1 <- c(x1_sv1, x2p_sv)
```

```
sv1
# [1] 1 1

# support vector 2
x1_sv2 = x1_sv[2]
x2n_sv = (-1 + b - w[1]*x1_sv2)/w[2]
sv2 <- c(x1_sv2, x2n_sv)
sv2
# [1] 3 4

# maximum margin of separation
d_m = 2/norm(w, type ='2')
d_m
# [1] 3.605551
```