Next time, please print on both sides. :)

Sandipan Dey
Homework – 1
Data Mining (691)

59/60

1. a) An *m X n* Matrix *A can be viewed as a* **linear transformation** $A: R^m \to R^n$.

Column Space (the space spanned by the column vectors) or **Image** of A is defined by the set $\{Ax \mid x \in R^m\}$. **Column Rank** of A is defined by the dimension of the column space (dim (Image (A)), i.e., (maximal) **number of linearly independently column vectors** of the matrix A.

Similarly, Row Space of matrix A is defined by the space spanned by the row vectors of A and **Row Rank** of A is defined by the dimension of the Row Space, i.e., (maximal) **number of linearly independent row vectors** of the matrix A.

If the matrix A is transformed to its **row-reduced echelon** form, then the **number of pivots** (leading 1 in columns) indicate its column rank, while **number of non-zero rows** indicate its row rank. Since number of pivots = number of non-zero rows, we have, for any matrix A, **Rank (A) = Row Rank (A) = Column Rank (A)**.

Another definition of a rank of a matrix is given by the rank of **largest non-vanishing minor,** i.e., the rank of the largest (square) non-singular (with non-zero determinant, hence invertible) sub-matrix . The previous definition also follows from this definition: **the largest non-vanishing minor** has the **maximal number of linearly dependent (row or column) vectors** of the matrix.

b) *Det (B) = 4.1.4 = 16 (non zero), B* is 3X3 non-singular matrix, hence B is a full-rank matrix, i.e., rank is 3.

If $a_1, a_2, a_3 \in R,\ a_1\begin{bmatrix}4\\-2\\5\end{bmatrix}+a_2\begin{bmatrix}0\\1\\3\end{bmatrix}+a_3\begin{bmatrix}0\\0\\4\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}\Rightarrow\begin{bmatrix}4a_1\\-2a_1+a_2\\5a_1+3a_2+4a_3\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}\Rightarrow a_1=a_2=a_3=0,$

3 column vectors of *B* are linearly independent, hence Rank of B = 3.

Also, using Gaussian elimination method (by elementary matrix transformation), the row-reduced-echelon form of the matrix B. Calculation is shown below:

$$B=\begin{bmatrix}4 & 0 & 0\\ -2 & 1 & 0\\ 5 & 3 & 4\end{bmatrix}$$

$$\leftrightarrow\begin{bmatrix}1 & 0 & 0\\ -2 & 1 & 0\\ 5 & 3 & 4\end{bmatrix}\left(R1\leftarrow\frac{R1}{4}\right)$$

$$\leftrightarrow\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 3 & 4\end{bmatrix}\left(R2\leftarrow R2+R1\times 2,\ \ R3\leftarrow R3-R1\times 5\right)$$

$$\leftrightarrow\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & \frac{3}{4} & 1\end{bmatrix}\left(R3\leftarrow\frac{R3}{4}\right)$$

$$\leftrightarrow\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\left(R3\leftarrow R3-R2\times\frac{4}{3}\right)$$

It can easily be seen that both the number of pivots (leading 1's in columns) and the number of non-zero rows of A in row-reduced-echelon form = 3, hence rank (B) = 3.

B is a lower triangular matrix. Being a triangular matrix, the Eigen values of B are same as the diagonal elements of *B*, hence Eigen values are 4, 1, 4. Corresponding Eigen vectors are $\begin{bmatrix}0\\0\\1\end{bmatrix},\begin{bmatrix}0\\1\\-1\end{bmatrix},\begin{bmatrix}0\\0\\1\end{bmatrix}$ respectively.

It shows that geometric multiplicity of and Eigen value (dimension of the corresponding Eigen space) is always less than its algebraic multiplicity (number of zeros of the characteristic polynomial at that Eigen value). For example, the Eigen value 4 here has algebraic multiplicity 2, but geometric multiplicity 1, since the Eigen space corresponding to the Eigen value 4 consists of only one Eigen vector $\begin{bmatrix}0\\0\\1\end{bmatrix}$, hence has dimension 1. So, there are exactly 2 distinct Eigen values corresponding to which there are exactly 2 linearly independent Eigen vectors (easy to see by definition of linear independence).

Calculations follow:

$$B=\begin{bmatrix}4 & 0 & 0\\ -2 & 1 & 0\\ 5 & 3 & 4\end{bmatrix}$$

$$\det(B-\lambda I)=0 \Rightarrow \begin{vmatrix}4-\lambda & 0 & 0\\ -2 & 1-\lambda & 0\\ 5 & 3 & 4-\lambda\end{vmatrix}=0$$

$$\Rightarrow (4-\lambda)\begin{vmatrix}1-\lambda & 0\\ 0 & 4-\lambda\end{vmatrix}=0 \Rightarrow (4-\lambda)(1-\lambda)(4-\lambda)=0 \Rightarrow \lambda=4,1,4$$

Let $X=\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}$ be an Eigen vector. (Since $X \in NullSpace(B-\lambda I)$, to have non-trivial solution in X,

We must have $\det(B-\lambda I)=0$ as before).

$$BX=\lambda X \Rightarrow \begin{bmatrix}4x_1\\ -2x_1+x_2\\ 5x_1+3x_2+4x_3\end{bmatrix}=\begin{bmatrix}\lambda x_1\\ \lambda x_2\\ \lambda x_3\end{bmatrix} \Rightarrow \begin{array}{c}(4-\lambda)x_1=0\\ -2x_1+(1-\lambda)x_2=0\\ 5x_1+3x_2+(4-\lambda)x_3=0\end{array}$$

$$\lambda=4 \Rightarrow \begin{bmatrix}0=0\\ -2x_1-3x_2=0\\ 5x_1+3x_2=0\end{bmatrix} \Rightarrow x_1=x_2=0, x_3=c\in R\ (c\neq 0) \Rightarrow X=\begin{bmatrix}0\\ 0\\ c\end{bmatrix}$$

$$\lambda=1 \Rightarrow \begin{bmatrix}x_1=0\\ -2x_1=0\\ 5x_1+3x_2+3x_3=0\end{bmatrix} \Rightarrow x_1=0, x_2=-x_3=c\in R\ (c\neq 0) \Rightarrow X=\begin{bmatrix}0\\ c\\ -c\end{bmatrix}$$

We can choose any c (except 0), hence we choose c = 1.

c) A matrix has zero Eigen value $\Leftrightarrow$ it's singular, hence not invertible.

(Since $A^{-1}=\dfrac{Adj(A)}{\det(A)}$, $\exists A^{-1} \Leftrightarrow \det(A)\neq 0$).

The above can directly be proved from the <u>Invertible Matrix Theorem</u> from which we get that for a given A, the following statements are **both true or both false together**.

(1) *A* is an invertible matrix.

(2) The number *0* is not an Eigen value of *A*.

Proof:

Assume *A* is an *n x n* square matrix. If λ is an Eigen value of *A* and X is an Eigen vector (nonzero) corresponding to the Eigen value, we have, $AX = \lambda X \Leftrightarrow (A - \lambda I)X = 0$

$\Rightarrow$ (If)

If an Eigen value is 0,

putting $\lambda = 0$ in above equation, we have, $(A - 0.I)X = 0 \Leftrightarrow A.X = 0$.

Since an Eigen vector is by definition a non-zero vector, we search for the **non-trivial solution** of the above **homogeneous system of equations.** Now having $A = [a_1 \ a_2 \ldots a_n]$: a set of column vectors and

$X = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}$ the system of homogeneous equations $A.X = 0$ takes the form $x_1 a_1 + x_2 a_2 + \cdots + x_n a_n = 0$.

To have a non-trivial solution (not all zero) in scalars x_i we must have the vectors a_i linearly dependent (by definition of linear independence). It implies that in matrix ***A*** all the columns can't be linearly independent; ***A*** is not a full-rank matrix.

Hence, *det (A) = 0*, i.e., ***A*** is singular and **inverse does not exist.**

Another proof (indirect proof)

If an Eigen value is 0,

putting $\lambda = 0$ in above equation, we have, $(A - 0.I)X = 0 \Leftrightarrow A.X = 0$.

Let's assume to the contrary that $\exists A^{-1} \Rightarrow A^{-1}AX = A^{-1}0 = 0 \Rightarrow I_n X = X = 0$, a contradiction, since by definition Eigen vector is nonzero. Hence, our initial assumption was wrong $\Rightarrow \neg\exists A^{-1}$, i.e.,

Hence, ***A* is not invertible.**

$\Leftarrow$ (Only if)

If A is non-invertible, i.e., *det (A) = 0*

let's assume ***n*** Eigen values (latent roots of the characteristic polynomial) of the matrix ***A*** are $\lambda_1, \lambda_2, \cdots, \lambda_n$. Then we have the characteristic polynomial (degree- n monic polynomial in λ) as:

$\det(A - \lambda I_n) = (\lambda - \lambda_1)(\lambda - \lambda_2)\cdots(\lambda - \lambda_n)$

Put $\lambda = 0$ in the above equation to get, $\det(A) = (-1)^n \lambda_1 \lambda_2 \cdots \lambda_n$

$$\det(A) = 0 \Rightarrow (-1)^n \lambda_1 \lambda_2 \cdots \lambda_n = 0 \Rightarrow \exists \lambda_i = 0$$

Hence, ***A* has an Eigen value *0*.**

2.

a) Correlation Coefficient Matrix

You missed 2&6.

r Matrix: each entry is correlation coefficient $r_{C_2C_1} = \dfrac{\operatorname{cov}(C_1, C_2)}{\sqrt{\operatorname{var}(C1)\operatorname{var}(C_2)}}$.

	A	B	C	D	E	F	G	H	I	J
A	1.0000	0.4518	0.6306	-0.2164	0.4915	0.4898	0.2910	0.4741	-0.0252	0.0392
B	0.4518	1.0000	0.4581	-0.1721	0.4893	0.8174	-0.0469	-0.4285	0.0115	0.2226
C	0.6306	0.4581	1.0000	-0.2874	0.4910	0.6431	0.1012	0.0749	0.0729	0.1472
D	-0.2164	-0.1721	-0.2874	1.0000	-0.2461	-0.1999	-0.1202	-0.0515	-0.3807	-0.0886
E	0.4915	0.4893	0.4910	-0.2461	1.0000	0.6404	-0.0984	-0.0954	0.1249	0.2190
F	0.4898	0.8174	0.6431	-0.1999	0.6404	1.0000	-0.0158	-0.3091	0.0133	0.1419
G	0.2910	-0.0469	0.1012	-0.1202	-0.0984	-0.0158	1.0000	0.2591	0.0377	0.1174
H	0.4741	-0.4285	0.0749	-0.0515	-0.0954	-0.3091	0.2591	1.0000	-0.0478	-0.1987
I	-0.0252	0.0115	0.0729	-0.3807	0.1249	0.0133	0.0377	-0.0478	1.0000	0.1339
J	0.0392	0.2226	0.1472	-0.0886	0.2190	0.1419	0.1174	-0.1987	0.1339	1.0000

-1

[r, p] = corrcoef (X) % X is the data matrix loaded from the data set ***[MATLAB]***

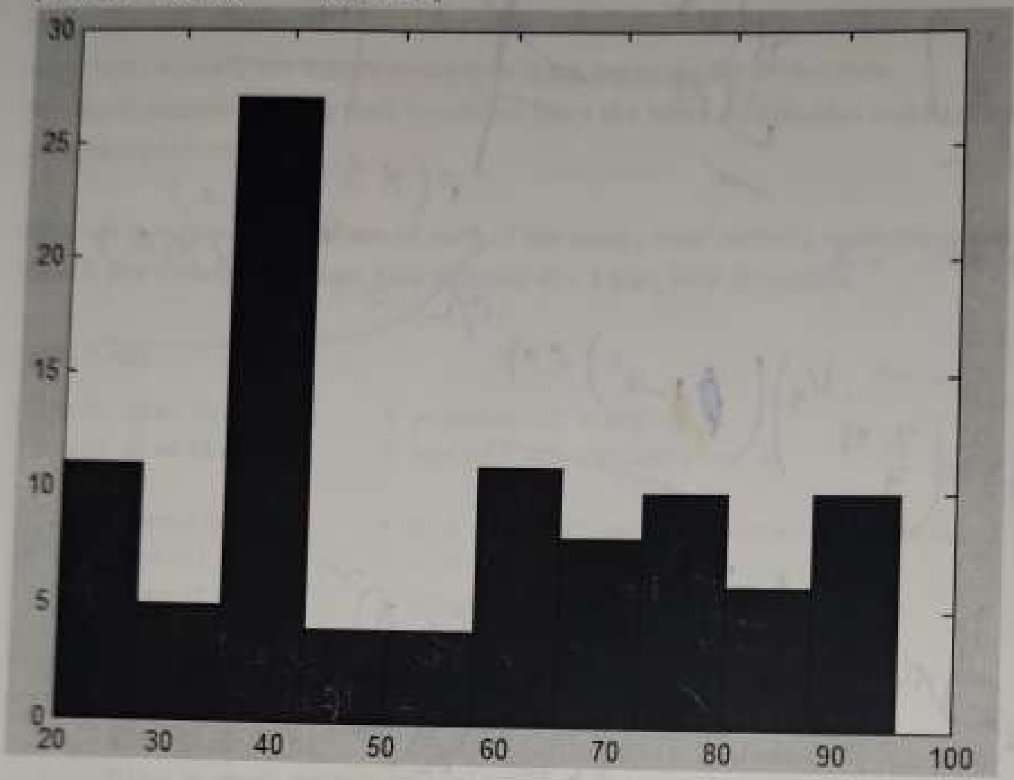
From the Matrix it's clear that group of features that are closely (linearly) related to each other (having correlation coefficient > 0.6) are (A, C), (C, F) and (E, F).

b) Range 20.0000 -27.5290 refers to the **half open interval** [20.0000, 27.5290). We do equal width binning (divide into 10 equal groups)

Range	Bin-Mean	Frequency	Tally Marks
20.0000 -27.5290	23.7645	11	卌 卌 |
27.5290 - 35.0580	31.2935	5	卌
35.0580 - 42.5870	38.8225	27	卌 卌 卌 卌 卌 ||
42.5870 - 50.1160	46.3515	4	||||
50.1160 - 57.6450	53.8805	4	||||
57.6450 - 65.1740	61.4095	11	卌 卌 |
65.1740 - 72.7030	68.9385	8	卌 |||
72.7030 - 80.2320	76.4675	10	卌 卌
80.2320 - 87.7610	83.9965	6	卌 |
87.7610 - 95.2900	91.5255	10	卌 卌
Total		96	

A = X(:,3);   % Select 3rd column from the left

[n, xout] = hist(A);   **[MATLAB]**



hist(A)   **[MATLAB]**

c) **Mahalanobis Distance** between two data vectors **X, Y** (of the same distribution, with covarinace matrix Σ is defined by: $\sqrt{(x-y)^T \Sigma^{-1} (x-y)}$. This distance takes into account the correlation of the data set and is scale invariant.

This distance measure has a huge application, e.g., to test whether a sample (multivariate) data tuple belongs to a population, it's not only sufficient to find the (Euclidean) distance of the point from the centroid (mean) of the population, but one must take care of the dispersion of the data along different directions and dimensions , covariance matrix takes care of it. One can use the normalized variate $\frac{X-\mu}{\sigma}$ where μ is the mean and σ is the standard deviation of the population to measure the distance of the sample point from the centroid of the population (taking care of dispersion), but this assumes that variation in population is spherically symmetric, where in reality it may not be so (may be ellipsoid along different feature directions), the covariance matrix in Mahalanobis distance takes care of this. Also, covariance matrix can be used to detect outliers in data more reliably for the similar reason.

Given data set (loaded into matrix **A**), since it's not explicitly mentioned whether to calculate the distance matrices in between every two data tuples or to calculate the matrices from the mean, let's first calculate the Mahalnobis and Euclidean distance between (a) <u>each ~~touple~~ and the mean</u>.

tuple

(b) **every two data tuples** (row vectors). Then we calculate both the distances between

(a)

Let's fisrt calculate the distance matrices **from mean vector** of the data.
We again assume that the data tuples are from the same distribution and hence share the same covariance matrix.

Here we calculate the distance of each of the tuples (row vectors) from the entire data (mean).
Hence, the distance matrices now become m x 1 (i.e., 96 x 1) vectors.

[MATLAB]

```
m = size(A, 1);         % number of rows of A
n = size(A, 2);         % number of coulmns of A

M = zeros(m, 1);        % m x m Mahalanobis distance matrix
C = cov(A);             % n x n covariance matrix
Ci = inv(C);            % inverse of covariance matrix

% Compute Mahalanobis distance matrix
for i = 1 : m
    X = (A(i,:));    % i-th data touple: row-vector (1 x n)
    M(i) = sqrt(mahal(X, A)); % A: entire data (m x n)
end

E = zeros(m, 1);      % m x m Mahalanobis distance matrix

% Compute Euclidean distance matrix
for i = 1 : m
    X = (A(i,:));    % i-th data touple: x row-vector (1 x n)
    E(i) =  sqrt(sum((X-mean(A)).^2, 2));    % A: entire data (m x n)
end

'Mahalanobis Distance Matrix:'
M   %   (m x m) matrix

'Euclidean Distance Matrix:'
E   %   (m x m) matrix
```

Mahalanobis Distance Matrix

$M = \underbrace{[d_i]}_{m \times 1}$, where $d_i = \sqrt{(x_i - \mu)\Sigma^{-1}(x_i - \mu)^T}$, where x_i : i^{th} **row vector** of matrix **A** (with n attributes) and μ is the mean (row vector) of matrix A. The element d_i in the matrix **M** denotes the Mahalanobis distance between i^{th} row vector and the mean of the matrix A.

Euclidean Distance Matrix

$E = \underbrace{[d_i]}_{m \times 1}$, where $d_i = \sqrt{(x_i - \mu)(x_i - \mu)^T}$, where x_i : i^{th} **row vector** of matrix **A** (with n attributes) and

μ is the mean (row vector) of matrix A. The element d_i in the matrix **E** denotes the Euclidean distance between i^{th} row vector and the mean of the matrix A.

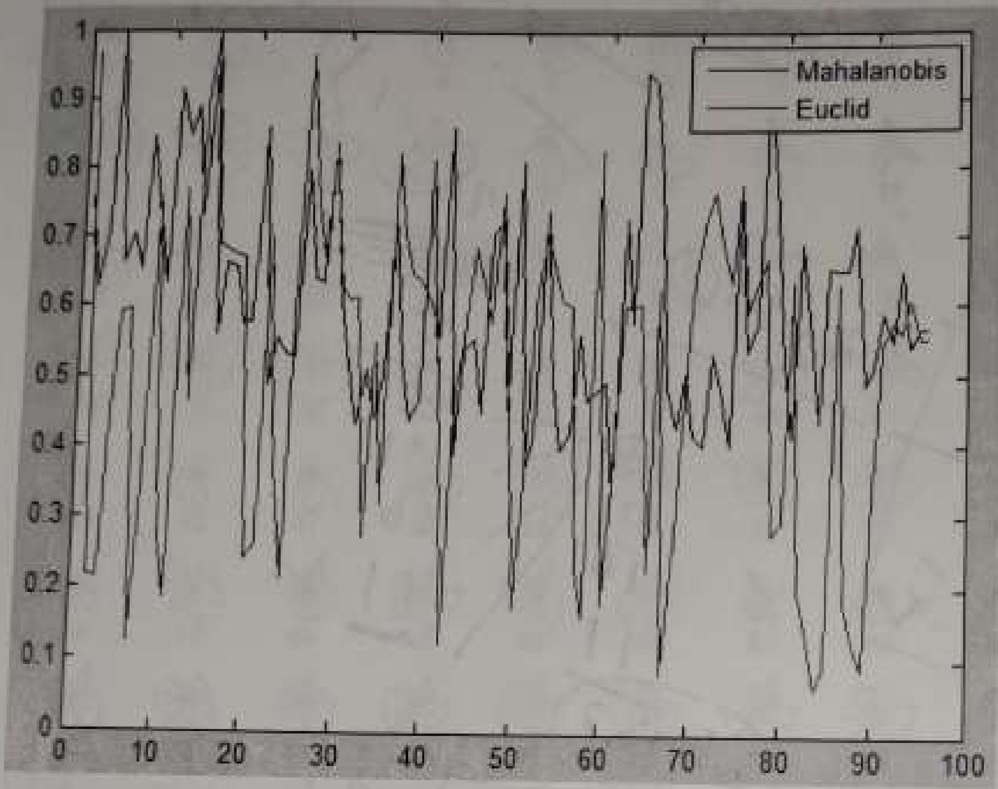
M

1	3.96	25	2.76	49	2.38	73	2.63
2	3.29	26	4.17	50	4.24	74	2.11
3	3.69	27	3.35	51	1.94	75	3.94
4	5.24	28	3.32	52	2.15	76	2.81
5	3.47	29	4.37	53	3.86	77	3.01
6	3.68	30	3.18	54	2.60	78	4.80
7	3.43	31	2.70	55	2.06	79	4.10
8	4.41	32	2.26	56	2.21	80	2.73
9	3.78	33	2.67	57	2.93	81	2.17
10	3.31	34	2.31	58	2.44	82	3.61
11	4.79	35	2.63	59	2.51	83	2.92
12	4.43	36	3.75	60	2.68	84	2.28
13	4.66	37	2.73	61	1.75	85	3.43
14	3.83	38	2.29	62	3.81	86	3.41
15	5.01	39	2.46	63	3.03	87	3.40
16	2.94	40	4.25	64	4.92	88	3.73
17	3.47	41	2.74	65	4.86	89	2.56
18	3.44	42	4.49	66	4.11	90	2.69
19	3.02	43	2.06	67	2.53	91	3.08
20	3.02	44	2.82	68	2.24	92	2.87
21	4.50	45	2.90	69	2.65	93	3.39
22	2.56	46	2.36	70	2.18	94	2.83
23	2.90	47	3.70	71	2.11	95	2.96
24	2.79	48	3.79	72	2.80	96	2.96

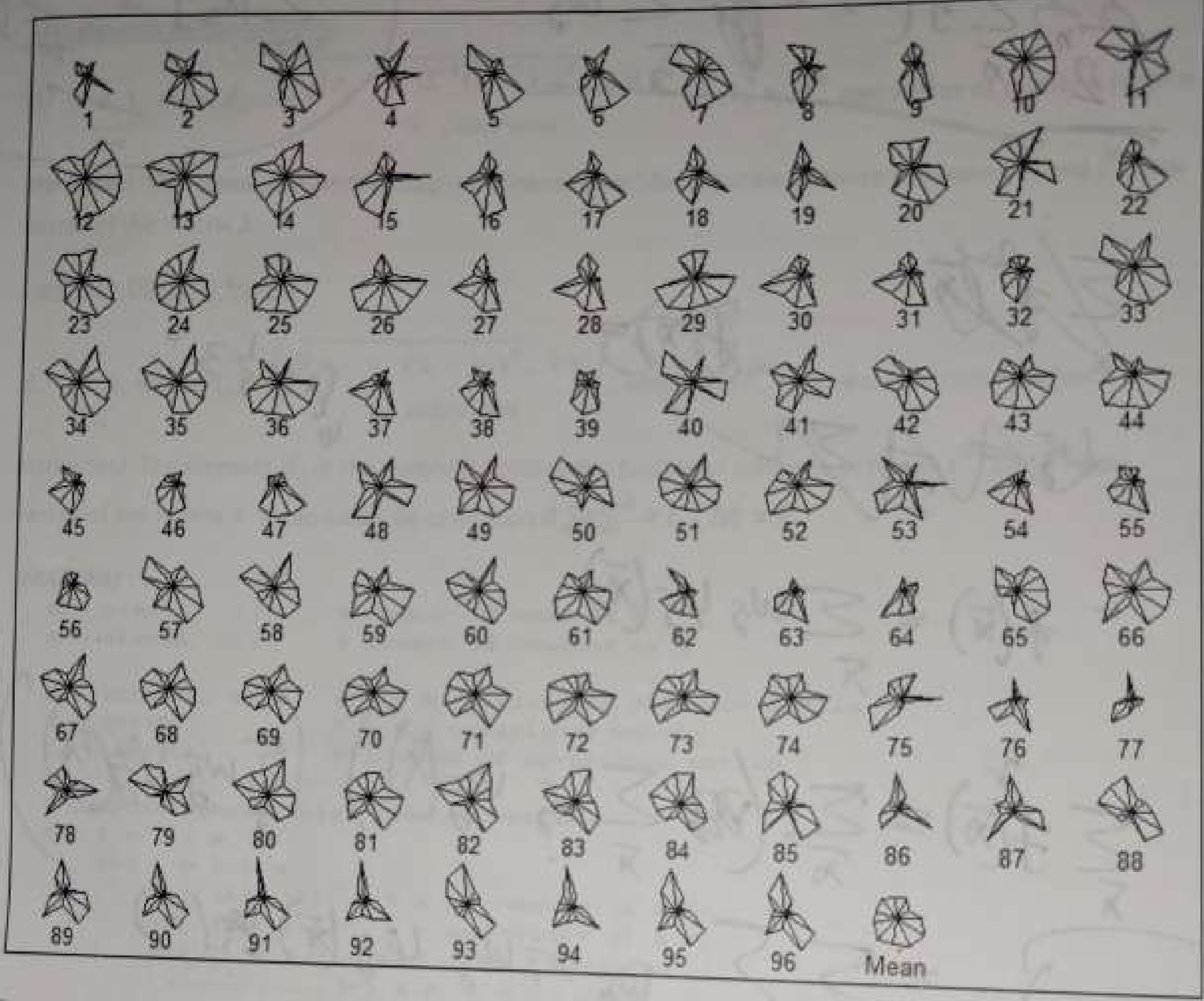
E

1	769.94	25	525.84	49	398.19	73	545.53
2	171.10	26	758.77	50	136.40	74	496.85
3	168.74	27	589.54	51	285.87	75	610.03
4	348.60	28	527.14	52	475.42	76	466.33
5	465.84	29	647.54	53	554.16	77	500.49
6	467.12	30	494.13	54	514.08	78	525.96
7	98.62	31	478.94	55	479.82	79	217.95
8	280.22	32	482.68	56	471.60	80	231.31
9	586.82	33	214.22	57	183.44	81	498.99
10	211.61	34	432.17	58	128.75	82	157.43
11	145.00	35	252.34	59	650.29	83	104.14
12	603.28	36	644.97	60	140.58	84	48.53
13	363.42	37	553.88	61	281.28	85	73.54
14	698.61	38	510.38	62	472.98	86	496.08
15	786.57	39	499.91	63	473.68	87	143.96
16	542.00	40	481.19	64	475.32	88	101.20
17	537.34	41	452.50	65	178.16	89	69.31
18	529.17	42	98.35	66	491.27	90	268.61
19	528.26	43	377.05	67	66.47	91	425.51
20	188.99	44	436.91	68	214.00	92	451.79
21	206.03	45	540.29	69	374.62	93	440.57
22	550.03	46	502.47	70	507.41	94	476.98
23	290.47	47	453.67	71	574.26	95	433.72
24	169.22	48	600.45	72	601.58	96	433.72

If the above distance matrices are normalized, so that the largest distance is 1, to see the relative distances, we get the following plot (data tuple index vs. normalized distance of the tuple from mean):



If we plot each data tuple using some plot (e.g., **glyphplot**, which shows every data tuple along its 11 dimensions), the data tuples look like the following:



From the above two plots we can see that Mahalanobis distance is a better distance measure than Euclidean. For instance, compare the Mahalanobis and Euclidean distance for the 60th and the 61st data tuple. As it can be seen from the glyph plot, 61st data point is closer to the mean vector than the 60th, this is supported by Mahalanobis distance, since it takes care of the direction of variation, but Euclidean distance reports to the contrary (refer to vectors M and E computed above).

(b)

First we compute both the distances (Mahalnobis and Euclid) between <u>**every two data tuples**</u> (**row vectors**). We assume that the data tuples are from the same distribution and hence share the same covariance matrix.

Notice that in the above definition X, Y are column vectors, if they are row vectors, then the definition will change to: $\sqrt{(x-y)\Sigma^{-1}(x-y)^T}$. We use this definition to compute the Mahalanobis distance matrix.

Mahalanobis Distance Matrix

$M = \underbrace{[d_{ij}]}_{m \times m}$, where $d_{ij} = \begin{cases} \sqrt{(x_i - x_j)\Sigma^{-1}(x_i - x_j)^T}, & i \neq j \\ 0, & otherwise \end{cases}$, where x_i : i^{th} row vector of matrix **A** (with n attributes). The element d_{ij} in the matrix **M** denotes the Mahalanobis distance between i^{th} and j^{th} row vector of the matrix A.

Euclidean Distance Matrix

$E = \underbrace{[d_{ij}]}_{m \times m}$, where $d_{ij} = \begin{cases} \sqrt{(x_i - x_j)(x_i - x_j)^T}, & i \neq j \\ 0, & otherwise \end{cases}$, where x_i : i^{th} row vector of matrix **A** (with n attributes). The element d_{ij} in the matrix **E** denotes the Euclidean distance between i^{th} and j^{th} row vector of the matrix A. It can easily be seen than if $\boldsymbol{\Sigma = \Sigma^{-1} = I_n}$, $\boldsymbol{M = E}$.

[MATLAB]

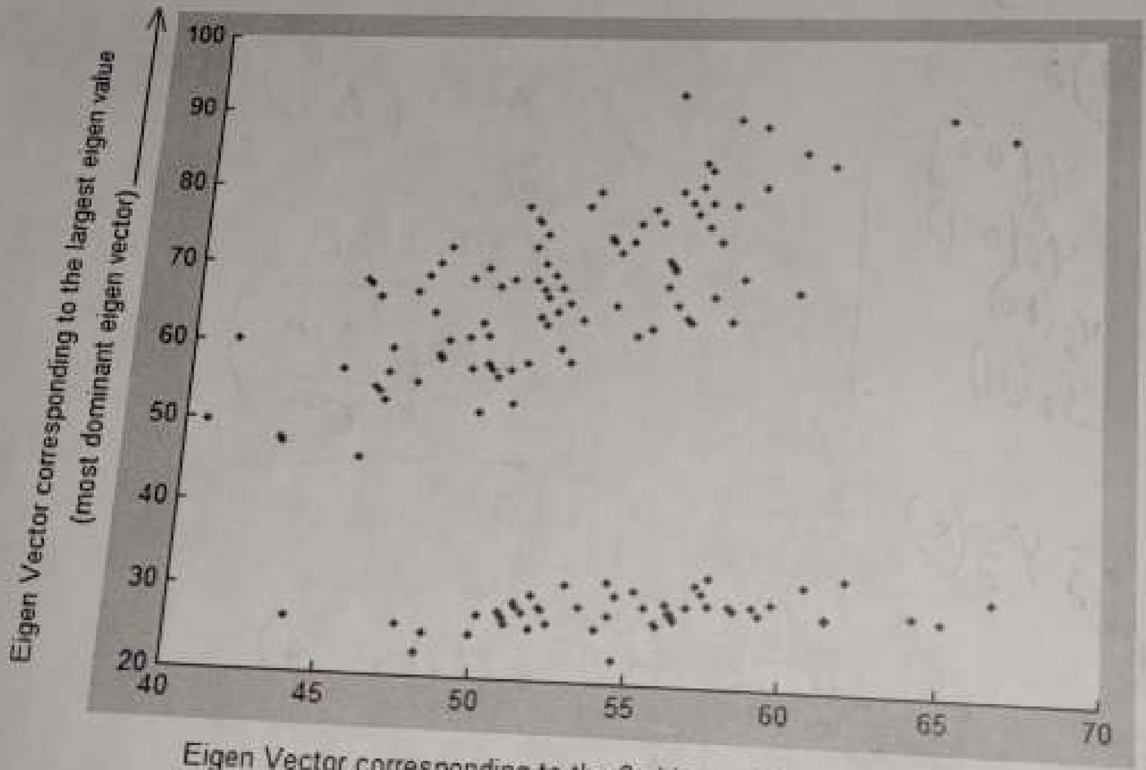
```
m = size(A, 1);       % number of rows of A
n = size(A, 2);       % number of coulmns of A

M = zeros(m, m);      % m x m Mahalanobis distance matrix
C = cov(A);           % n x n covariance matrix
Ci = inv(C);          % inverse of covariance matrix

% Compute Mahalanobis distance matrix
for i = 1 : m
    for j = 1 : m
        x = (A(i,:));    % x row-vector (1 x n)
        y = (A(j,:));    % y row-vector (1 x n)
        M(i,j) = sqrt((x-y) * Ci * (x-y)');
                        % 1 x n  n x n  n x 1
    end
end

E = zeros(m, m);      % m x m Mahalanobis distance matrix

% Compute Euclidean distance matrix
for i = 1 : m
    for j = 1 : m
        x = (A(i,:));    % x row-vector (1 x n)
        y = (A(j,:));    % y row-vector (1 x n)
        E(i,j) = sqrt((x-y) * (x-y)');
                        % 1 x n   n x 1
    end
end

'Mahalanobis Distance Matrix:'
M   %   (m x m) matrix

'Euclidean Distance Matrix:'
E   %   (m x m) matrix
```

-0.3688	-0.5697	0.7294	-0.0858
0.3558	0.558	0.658	0.3593

Dominant Eigen values: 24.5812 (2nd largest), 425.6304 (largest).
Corresponding **dominant Eigen vectors**: vectors 3 (2nd most dominant) & 4 (most dominant).

Projection (across dominant Eigen vectors 3 & 4)

*P = A * V (:, 3:4)* *% project across two dominant eigenvectors.*



Eigen Vector corresponding to the 2nd largest eigen value
(most dominant eigen vector)

scatter(P(:,1),P(:,2), 3, [1,0,0], 'filled') ***[MATLAB]***

It can be seen that there are 2 distinct directions of maximum variation in the feature space.