CS306: Data Analysis and Visualization
Assignment 1

Name: *Pratvi Shah* Student ID: *201801407*

1. **Least squares (Question 1)**

(a) Performing ordinary least squares on the data to fit a line $y_{pred} = \beta_0 + \beta_1 x$ involves finding the coefficients β_0 and β_1 such that the error in predicting the linear fit is minimum.

$$minimize(\Sigma_{i=0}^{n}(y_{pred} - y)^2)$$

For solving the above requirement:

- we substitute y_{pred} in terms of β_0 , β_1 and x.
- Take partial derivative with respect to β_0 and β_1 and equate those expressions to zero as first derivative of the maxima/minima is zero.
- We now have two equations and two unknowns which upon solving would give us β_0 and β_1.

Finally we get :

$$\beta_1 = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\Sigma(x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

We have $\bar{x}$ = 3.375 and $\bar{y}$= 4.625.
Thus, substituting the above values we get β_0 =1.67364 and β_1 =0.87448. Which gives us:

$$y_{pred} = 1.67364 + 0.87448x$$

RMSE between the fit and the original data came out to be 0.61535.
R squared value was observed to be 0.882926 which is fairly near 1 suggesting a good fit.

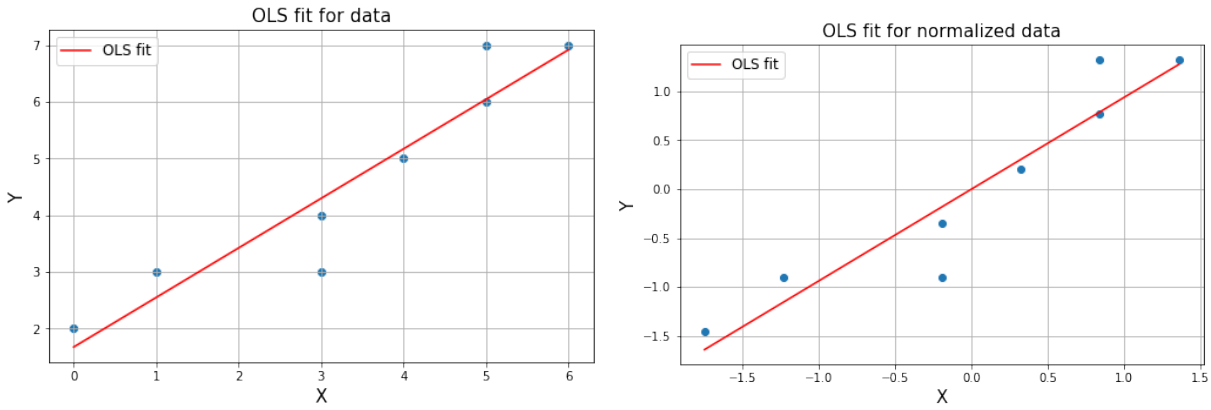

Figure 1: Linear fit of data using OLS

(b) After normalizing the data and applying the same method as mentioned in part a, we have $\hat{x}$=$\hat{y}$=0 and we get β_0 =0 and β_1 =0.93964. Which gives us:

$$y_{pred} = 0.93964x$$

RMSE between the fit and the normalized data came out to be 0.34216.
R squared values came out to be 0.882926 which is same as without normalization.

The RMSE of normalised data is lower than what we found in original data due to the scaling involved while standardizing/normalizing the data. RMSE shows the absolute error while R squared is a better measure for the goodness of fit and the value for the same comes out to be constant with and without normalization suggesting that there is **no influence of normalization on OLS fit**.

Normalization is useful in most of the cases(but not OLS as it is invariant) as it makes the data dimensionless and keeps a common scale for all the features. This is very useful while we do regression because if the data of one feature has a range which is significantly larger than that of other features then that feature will have a greater influence on how the regression output will fit the data. So, normalizing the data columns to zero mean and one standard deviation brings down the data columns to a common range while maintaining the inherent difference within the column.

(c) Outliers are the data-points that lie significantly away from the other data-points. This level of significance is subjective and is determined by the data set available. For this question outliers will be the data-points which do not lie within the range of $(\mu - 3\sigma, \mu + 3\sigma)$, where σ is the standard deviation and μ is the mean of the data.

The outlier for this data can be any point having $x \notin (\mu - 3\sigma, \mu + 3\sigma) = (-2.42236, 9.172359)$ and $y \notin (\mu - 3\sigma, \mu + 3\sigma) = (-0.77031, 10.02031)$ simultaneously.

So for the given dataset (x,y) = (12,-100) was chosen as the outlier. This gives us the data as:

x = [0, 1, 3, 3, 4, 5, 5, 6, 12]
y = [2, 3, 3, 4, 5, 6, 7, 7, -100]

(d) RANdom Sample Consensus is an iterative algorithm which is based on consensus or a probabilistic approach towards achieving a good fit to the data and is robust to the outliers thereby making it a outlier detection algorithm. It is an algorithm in which the inliers decide the fit and the outliers are considered to be the points which do not fit the model as they do not lie within the permissible error tolerance region.

Following is the psuedo code for the same:

```
itr_max= specify the number of iterations
n= 2 # >=2 elements needed to get a linear fit
init_p= initial probability to tolerate outliers
err= error tolerance for outliers

do:

   temp_data= select n random samples out of the data
   temp_fit= get fit of temp_data i.e., slope and intercept in case of line
   predicted= predicted values of data using temp_fit paramenters
   inliers=[]
   inlier_count=0

   for sample in data:
       if error between sample and predicted < err:
           sample is an inlier
           add sample to inliners
           inlier_count++
       end if
   end for
```

```
    fit_err = (data_size-inliner_count)/data_size
    if fit_err < init_p:
        init_p= fit_err
        good_fit= temp_fit
        iter_count++
     end if

 while iter_count<itr_max

 ans = good_fit
```

(e) For a common case, the definition of the outlier mainly decides on how we detect the outlier and remove them. But for our particular case of RANSAC algorithm we remove the outliers based on the *err* mention in the above pseudo code. The error(*err*) will decide the maximum permissible distance/cost of a data point from the fit produced. If the calculated metric is within the *err* limit then that data point will not be removed otherwise it will be removed and hence, the removal of outlier is facilitated.
In RANSAC algorithm, once we obtain a fit we try to build a better fit with the inliers obtained from the current fit. In the case when the $init_p$ probability to tolerate outliers for new fit is smaller than the previously obtained fit then we continue building up on the same set of inliers disregarding the outliers completely. But when there is no better fit available from the inliers we restart the algorithm from the beginning and consider every data point for random selection.

Thus, removal of outlier depends upon the *err* metric and the further steps to obtain the best possible fit are taken with respect to the inlier only thereby outlier are completely ignored in RANSAC algorithm. On applying the above algorithm and using the inbuilt RANSAC regressor of python we can obtain the following fit:

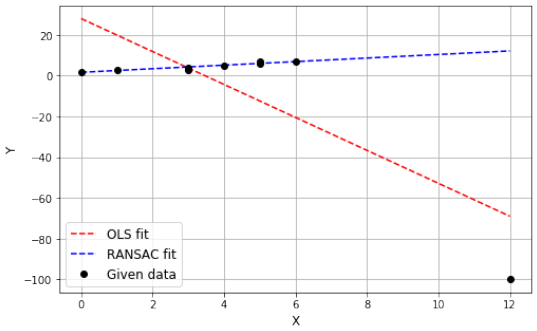

Figure 2: Comparison of linear regression and RANSAC algorithm

The data was the one obtained in the part (c) after introducing the outliers.Following code helped produce the above result:

```
from sklearn.linear_model import LinearRegression,RANSACRegressor
x=[0, 1, 3, 3, 4, 5, 5, 6, 12]
y=[2, 3, 3, 4, 5, 6, 7, 7, -100]

x_axis=np.array(x)
x_axis=x_axis.reshape(-1, 1)

plt.figure(figsize=[8,5])
l_reg = LinearRegression()
ypred_lr = l_reg.fit(x_axis,np.array(y))
ypred_lr=ypred_lr.predict(x_axis)
plt.plot(x,ypred_lr,'r--',label='OLS fit')
ransac= RANSACRegressor()
```

```
ypred_ransac = ransac.fit(x_axis,np.array(y))
ypred_ransac = ypred_ransac.predict(x_axis)
plt.plot(x_axis,ypred_ransac,'b--',label='RANSAC fit')
plt.plot(x,y,'ko',label='Given data')
plt.legend(fontsize=12)
plt.xlabel('X',fontsize=12)
plt.ylabel('Y',fontsize=12)
plt.grid()
```

The above plot clearly shows how RANSAC algorithm ignores the outliers and produces the best fit with the inliers.

(f) For carrying out ordinary least square to fit a parabola we need to minimize the error in the predicted values i.e.,

$$minimize\|\mathbf{e}\|^2 = \|\mathbf{b} - A\mathbf{x}\|^2$$

where,

$$A = [x^2 \;\; x \;\; \mathbf{1}]$$
$$x = [x_1 \;\; x_2 \;\; x_3 \ldots \;\; x_n]^T$$

and the last column of A is a 1xn dimension column of ones.For our case n=8=length(x)

We need to solve the system of equations given by $Ax = b$. For least square approach we can use method of normal equations to compute the best fit. For this we follow the given steps:

- Multiply both sides by A^T giving us $A^TAx = A^Tb$.
- Now, keep x on the left hand side and compute the solution by finding inverse of A^TA.

So, the final solution comes out to be:

$$\hat{x} = (A^TA)^{-1}A^Tb$$

where the term $(A^TA)^{-1}A^T$ is the pseudo-inverse of the matrix A. The solution $\hat{x}$ gives us the required parameters. This approach gives us the least square fit as:

$$\begin{aligned}\|b - Ax\|^2 &= (b - Ax) * (b - Ax)^T \\ &= b^Tb - 2x^TA^Tb + x^TA^TAx\end{aligned}$$

upon taking partial derivative of the above equation with respect to x we can equate that partial derivative to zero as the first derivative of a local minima is zero. When we do this we get:

$$A^TAx = A^Tb$$

this verifies the solution found above [1].

Furthermore, we can also get the solution using 3 differential equations and solving them for β_0,β_1 and β_2 just like in part (a).

Using the above method the parameters obtained are:
β_0= 2.148818
β_1= 0.292651
β_2= 0.098412
with equation of parabola as

$$y = 2.148818 + 0.292651x + 0.098412x^2$$

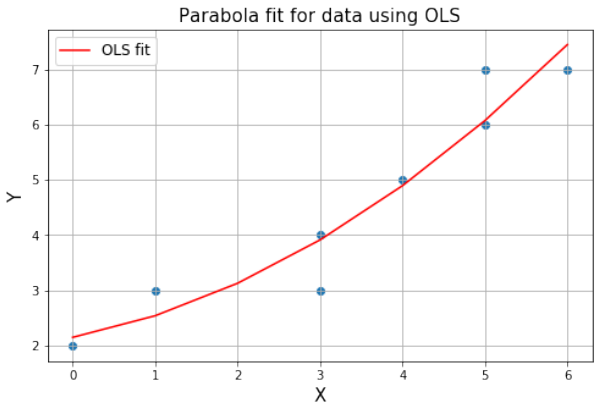

Figure 3: Parabola fit using OLS

2. **SVD and eigen decomposition (Question 2)**

We have the following matrix for our question X= $\begin{bmatrix} 12 & -4 & 0 & 0 & 0 \\ 4 & 12 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$

(a) The features in a general data matrix are the column values for a particular row. So, the features of the 5 samples are:

- Features of Sample 1 : $[12\ ,\ -4\ ,\ 0\ ,\ 0\ ,\ 0]$
- Features of Sample 2 : $[4\ ,\ 12\ ,\ 0\ ,\ 0\ ,\ 0]$
- Features of Sample 3 : $[0\ ,\ 0\ ,\ 2\ ,\ 0\ ,\ 0]$
- Features of Sample 4 : $[0\ ,\ 0\ ,\ 0\ ,\ 1\ ,\ -1]$
- Features of Sample 5 : $[0\ ,\ 0\ ,\ 0\ ,\ -1\ ,\ 1]$

(b) The columns of the matrix X represents the features so to find the feature covariance matrix we perform $\Sigma = X^T X$.

$$\Sigma = \begin{bmatrix} 12 & 4 & 0 & 0 & 0 \\ -4 & 12 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} * \begin{bmatrix} 12 & -4 & 0 & 0 & 0 \\ 4 & 12 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 160 & 0 & 0 & 0 & 0 \\ 0 & 160 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & -2 & 2 \end{bmatrix}$$

(c) From the above step we got the feature covariance matrix Σ and to find the eigen values and eigen vectors we solve

$$\Sigma \mathbf{x} = \lambda \mathbf{x}$$
$$(\Sigma - \lambda I)\mathbf{x} = 0$$

where λ will consists of the eigen values and x will consists of the corresponding eigen vectors.

$$(\Sigma - \lambda I) = \begin{bmatrix} 160-\lambda & 0 & 0 & 0 & 0 \\ 0 & 160-\lambda & 0 & 0 & 0 \\ 0 & 0 & 4-\lambda & 0 & 0 \\ 0 & 0 & 0 & 2-\lambda & -2 \\ 0 & 0 & 0 & -2 & 2-\lambda \end{bmatrix}$$

The above matrix directly gives $[\lambda_1, \lambda_2, \lambda_3] = [160, 160, 4]$ while the problem reduces to solving :

$$\begin{bmatrix} 2-\lambda & -2 \\ -2 & 2-\lambda \end{bmatrix} = 0$$

$$(2-\lambda)^2 - 4 = 0$$
$$\therefore [\lambda_4, \lambda_5] = [0, 4]$$

Finally we get all the eigen values as $[\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5] = [160, 160, 4, 0, 4]$
We need to substitute the above λ values to find the eigen vectors i.e., the columns of the matrix U.

Substituting values of λ one by one we observe the following values of x:

- $\lambda_1 = 160$: [1 ,0 ,0 ,0 ,0]
- $\lambda_2 = 160$: [0 ,1 ,0 ,0 ,0]
- $\lambda_3 = 4$: [0 ,0 ,0 ,-0.70710678 ,0.70710678]
- $\lambda_4 = 0$: [0 ,0 ,0 ,0.70710678 ,0.70710678]
- $\lambda_5 = 4$: [0 ,0 ,1 ,0 ,0]

Finally for $\lambda = [4, 0, 160, 160, 4]$ we have the eigen vector matrix U (with eigen vectors as the columns of U)as :

$$U = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -0.7071678 & 0.7071678 & 0 & 0 & 0 \\ 0.7071678 & 0.7071678 & 0 & 0 & 0 \end{bmatrix}$$

We can retrieve back Σ by firstly performing matrix multiplication of λ and U as $(\Sigma_i \lambda_i U_{.,i})$ then right multiplying by U^{-1} i.e., getting the inverse transform to bring the points back to the original vector space.

We can also retrieve back the original matrix by taking each eigen vector and multiplying it with its transpose to form a nxn matrix where n=dimension of eigen vector. After doing so we incorporate the weights of each eigen vector i.e., multiply each eigen vector with the corresponding eigen value and performing the same for all eigen vectors and summing them up to find the original matrix Σ.

Thus, the above values of eigen vectors U corresponding to eigen values λ satisfy the equation: $\Sigma = \Sigma_i \lambda_i U_i U_i^+$.

(d) For SVD I have made use of the following code:

```
def find_svd():
    m=[[12 ,-4, 0, 0 ,0],[4 ,12 ,0 ,0 ,0],[0 ,0 ,2 ,0 ,0]
                         ,[0 ,0 ,0 ,1 ,-1],[0 ,0 ,0 ,-1 ,1]]
    U,S,Vt=numpy.linalg.svd(m)
    V=np.transpose(Vt)
    return U,S,V
```

The return values of the above function are:

S: Singular values of the matrix X
S : $[s_1, s_2, s_3, s_4, s_5] = [12.649, 12.649, 2, 2, 0]$

U: columns have Left eigen vectors(LEV) of the matrix X corresponding to above singular values are:

- LEV_1 = [-0.31623, 0.94868 , 0 , 0 , 0]
- LEV_2 = [-0.94868 , -0.31623 , 0 , 0 , 0]
- LEV_3 = [0 , 0 , 0 , -0.707106 , 0.707106]
- LEV_4 = [0 , 0 , 1 , 0 , 0]

- LEV_5 = [0 , 0 , 0 , 0.707106 , 0.707106]

V: columns have Right eigen vectors(REV) of the matrix corresponding to S are:

- REV_1 = [0 , 1 , 0 , 0 , 0]
- REV_2 = [-1 , 0 , 0 , 0 , 0]
- REV_3 = [0 , 0 , 0 , -0.707106 , 0.707106]
- REV_4 = [0 , 0 , 1 , 0 , 0]
- REV_5 = [0 , 0 , 0 , -0.707106 , -0.707106]

Finally,

$$U = \begin{bmatrix} -0.31623 & -0.94868 & 0 & 0 & 0 \\ 0.94868 & -0.31623 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -0.707106 & 0 & 0.707106 \\ 0 & 0 & 0.707106 & 0 & 0.707106 \end{bmatrix}$$

(e) Values:

- From the above values of SVD and EVD we can see that the singular values are the square root of the eigen values as observed in the above case. Singular values need to be real whereas eigen values can be complex also.

Vectors:

- The eigen vectors are orthogonal and form the orthonormal basis in the same vector space as the points and thus it is a mapping from $\mathbf{R^n} \rightarrow \mathbf{R^n}$. Singular vector matrices U and V are orthogonal to each other and lead to transformation from $\mathbf{R^n} \rightarrow \mathbf{R^m}$ for a matrix $\mathbf{A}$ of dimension mxn. V and U form the basis for the domain and range in such a case and hence, the dimension of V is nxn and that of U is mxm.

3. **Analysis of Variance (Question 3)**

We have the following data for this question:

$$\begin{bmatrix} & M1 & M2 & M3 \\ 1 & 47 & 55 & 54 \\ 2 & 53 & 54 & 50 \\ 3 & 49 & 58 & 51 \\ 4 & 50 & 61 & 51 \\ 5 & 48 & 55 & 50 \\ 6 & 46 & 52 & 49 \end{bmatrix}$$

For this question, the average number of products produced by each machine is given by:
$\mu_{M1} = 48.83$
$\mu_{M2} = 55.83$
$\mu_{M3} = 50.83$
$\mu_{all} = 51.83$

(a) To find variance amongst the 3 sample means we use the following equation where n_i represents the number of elements in each column(=6):

$$S_x^2 = \sum_{i=1}^{3} n_i(\mu_{Mi} - \mu_{all})^2$$
$$S_x^2 = 156$$

(b) To find the residual variance(variance within each column) we used the following equation:

$$S_p^2 = \Sigma_{i=1}^{3}\Sigma_{j=1}^{6}(x_{ij} - \mu_{Mj})^2$$
$$S_p^2 = 96.5$$

(c) Degree of freedom for F are:

$$dof(S_x^2) = 3 - 1 = 2 = \text{number of columns - 1}$$
$$dof(S_p^2) = \text{number of columns} * (len(Mi) - 1) = 3 * 5 = 15$$

For the above given degree of freedom (2,15) we get F ratio using:

$$F = \frac{S_x^2/dof(S_x^2)}{S_p^2/dof(S_p^2)}$$
$$= 12.12435$$

The corresponding p-value evaluated using inbuilt function comes out to be 0.0007361977.The following function was used to obtain the value:

```
p_value=stats.f_oneway(m1,m2,m3).pvalue
```

.

(d) For this case the null hypothesis H_0 can be defined as:

H_0 = Average productivity of all the machines is the same $(\mu_{M1} = \mu_{M2} = \mu_{M3})$

H_1 = The average productivity of atleast one machine is different from the rest

If the *p-value* comes out to be less than 0.01 like 0.0007361 in the above case, then we would reject the null hypothesis as we would not be making more than 1% (=*p-value**100) error if we reject the null hypothesis.
For our case we will incur an error of $\approx 0.073\%$ if we reject the null hypothesis so for $\alpha = 0.01$ we reject the null hypothesis for our case.

(e) This problem is two-way ANOVA without replication as there is only one instance of each observable. We perform ANOVA where there are 2 null hypothesis:

Hypothesis in terms of factor A = Row/Operators:
H_0= The average productivity of all the operators is same
H_1= The average productivity of atleast one operator is different from the rest

Hypothesis in terms of factor B = Columns/Machines:
H_0= The average productivity of all the machines is same
H_1= The average productivity of atleast one machine is different from the rest

So, for this case we use the following equations:

$$
\begin{aligned}
SST &= SSR + SSC + SSE \\
MSR &= \frac{SSR}{dof(R)} \\
MSC &= \frac{SSC}{dof(C)} \\
MSE &= \frac{SSE}{(dof(C) * dof(R))} \\
F_C &= \frac{MSC}{MSE} \\
F_R &= \frac{MSR}{MSE}
\end{aligned}
$$

where , R= Row, C=Column, E=Error

SSX represent the sum of squares of X and X$\in (R, C, E)$

$count(C) = n = 3$

$count(R) = m = 6$

dof(R)= $m - 1$

dof(C)= $n - 1$

dof(E)=dof(C)$*$dof(R)

T=$\Sigma_{i=1}^{m}\Sigma_{j=1}^{n} x_{ij}$

Now we calculate the neccessary sum of squares:

- $SST = \Sigma_{i=1}^{m}\Sigma_{j=1}^{n} x_{ij}^2 - \frac{T^2}{n * m}$

  $SST = 252.3$

- $SSR = \frac{\Sigma_{i=1}^{m}(\Sigma_{j=1}^{n} x_{i,j})^2}{m-1} - \frac{T^2}{n * m}$

  $SSR = 43.166$

- $SSC = \frac{\Sigma_{j=1}^{n}(\Sigma_{i=1}^{m} x_{ij})^2}{n-1} - \frac{T^2}{n * m}$

  $SSC = 156$

- $SSE = SST - SSC - SSR$

  $SSE = 53.333$

Now we calculate the F value from the above sumof squares when $MSE = 5.33333$:

- $MSR = 8.6333$ using the formula listed above
  $F_R = \frac{MSR}{MSE} = 1.61875$

- $MSC = 78$ using the formula listed above
  $F_C = \dfrac{MSC}{MSE} = 14.625$

With degree of freedom (5,10) for Row/Operator case we get $F_{cR} = 3.32583453$ for $\alpha = 0.05$

With degree of freedom (2,10) for Column/Machines elements we get $F_{critC} = 4.10282102$ for $\alpha = 0.05$.

Thus, from the above case we see that $F_{cR} > F_R$ so we cannot reject that null hypothesis while $F_{critC} < F_C$ so we reject this null hypothesis.

Finally we can conclude that there is no sufficient evidence that average productivity of the operators is not the same(not rejecting the null hypothesis) while there is sufficient evidence that average productivity of all the machines is not the same(rejecting the null hypothesis).

4. **Contraint optimization and modeling using linear algebra techniques (Question 4)**

(a) To compute the values of a, b, c, d, e, f in terms of the parameters of circle we can simplify the equation of the circle and later compare the equation with the general conic. So,

$$\begin{aligned}(x - x_0)^2 + (y - y_0)^2 &= r^2 \\ x^2 - 2xx_0 + x_0^2 + y^2 - 2yy_0 + y_0^2 - r^2 &= 0 \\ x^2 + y^2 - 2xx_0 - 2yy_0 + x_0^2 + y_0^2 - r^2 &= 0\end{aligned}$$

Comparing coefficients with general conic equation of $ax^2 + bxy + cy^2 + dx + ey + f = 0$ we get:

$$\begin{aligned} a = 1 \qquad b = 0 \qquad c = 1 \\ d = -2x_0 \quad e = -2y_0 \quad f = x_0^2 + y_0^2 - r^2\end{aligned}$$

(b) Direct linear transform involves solving the system of equations of type $\mathbf{AX} = \mathbf{0}$ to find $\mathbf{X}$.

$\mathbf{A} = \begin{bmatrix}\mathbf{x^2} & \mathbf{xy} & \mathbf{y^2} & \mathbf{x} & \mathbf{y} & \mathbf{1}\end{bmatrix}_{n\times 6}$ and $\mathbf{X} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}_{6\times 1}$

DLT Algorithm:

i. Normalize the data given.
ii. Compute design matrix $\mathbf{A}$ depending on the type of fit needed.
iii. Perform SVD on matrix $\mathbf{A}$ to get $A = UDV^T$ where U,V correspond to the left and right singular vectors and D consists of the elements of diagonal matrix representing the singular values of $\mathbf{A}$.
iv. The singular vector in $\mathbf{V}$ corresponding to the minimum singular value in D will be the solution.

The last column of $\mathbf{V}$ which corresponds to the last eigen value in $\mathbf{D}$ (as D has elements in descending order, the last singular value will hence be minimum) will be the solution to X.

$$\begin{aligned}\mathbf{UDV^T} &= SVD(\mathbf{A}) \\ \left\|\mathbf{UDV^TX} - \mathbf{b}\right\| &= 0 \\ \left\|\mathbf{DV^TX} - \mathbf{U^Tb}\right\| &= 0 \\ \because \mathbf{U^Tb} &= 0 \\ \therefore \left\|\mathbf{DV^TX}\right\| &= 0\end{aligned}$$

The above equation will me minimized when $\mathbf{DV^T}$ will take its minimum value which is achieved for the minimum eigen value of D. Hence, the previously suggested solution i.e., the column of $\mathbf{V}$ corresponding to the least eigen value will be the solution of $\mathbf{X}$.

(c) For our case, we will assume $\mathbf{x}$ to be the x coordinates of the data, and $\mathbf{y}$ to be the y coordinates of the data.

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

We can incorporate the conditions obtained for variables in part (a), in the general equation of the conic and try to find the constraint matrix:

$$\begin{aligned}
\therefore ax^2 + 0xy + ay^2 + dx + ey + f = 0 \\
\therefore a(x^2 + y^2) + dx + ey + f = 0 \\
x^2 + \frac{2dx}{2a} + (\frac{d}{2a})^2 - (\frac{d}{2a})^2 + y^2 + \frac{2ey}{2a} + (\frac{e}{2a})^2 - (\frac{e}{2a})^2 + \frac{f}{a} = 0 \\
(x - \frac{d}{2a})^2 + (y - \frac{e}{2a})^2 = -\frac{f}{a} + (\frac{d}{2a})^2 + (\frac{e}{2a})^2
\end{aligned}$$

We can now put the condition on the RHS i.e., $r^2 > 0$ as the square of the radius should always be greater than zero. Hence, the condition we get for circle fitting problem is:

$$-\frac{f}{a} + (\frac{d}{2a})^2 + (\frac{e}{2a})^2 > 0$$

$$d^2 + e^2 > 4af$$

From the above equation we can see that our design matrix can be reduces from having 6 features to 4 features (as we can combine x^2 and y^2 feature and neglect xy feature due to $b = 0$)and thus the new design matrix comes out to be:

$$\mathbf{A} = \begin{bmatrix}(\mathbf{x^2 + y^2}) & \mathbf{x} & \mathbf{y} & \mathbf{1}\end{bmatrix}_{n\times 4} \mathbf{X} = \begin{bmatrix}\mathbf{a}\\ \mathbf{d}\\ \mathbf{e}\\ \mathbf{f}\end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix}z_1 & z_2 & z_3 & z_4\\ z_5 & z_6 & z_7 & z_8\\ z_9 & z_{10} & z_{11} & z_{12}\\ z_{13} & z_{14} & z_{15} & z_{16}\end{bmatrix}$$

So, finally putting the constraint we have:

$$\mathbf{X^T C X} > 0$$

$$\begin{bmatrix}\mathbf{a} & \mathbf{d} & \mathbf{e} & \mathbf{f}\end{bmatrix} \begin{bmatrix}z_1 & z_2 & z_3 & z_4\\ z_5 & z_6 & z_7 & z_8\\ z_9 & z_{10} & z_{11} & z_{12}\\ z_{13} & z_{14} & z_{15} & z_{16}\end{bmatrix} \begin{bmatrix}\mathbf{a}\\ \mathbf{d}\\ \mathbf{e}\\ \mathbf{f}\end{bmatrix} > 0$$

Upon multiplying the above matrices we get:

$$\begin{aligned}
a(az_1 + dz_5 + ez_9 + fz_{13}) \\
+d(az_2 + dz_6 + ez_{10} + fz_{14}) \\
+e(az_3 + dz_7 + ez_{11} + fz_{15}) \\
+f(az_4 + dz_8 + ez_{12} + fz_{16}) > 0
\end{aligned}$$

Now, equating the coefficients of d^2 , e^2 and af for getting the condition $d^2 + e^2 > 4af$ satisfied:

$$z_6 = 1$$

$$z_{11} = 1$$

$$z_{13} = z_4 = -2$$

$$\text{other } \ z_i \ \text{ are equal to } 0$$

Hence, following is the required constraint matrix for circle fitting whent he design matrix is represented by A:

$$\mathbf{A} = \begin{bmatrix} (\mathbf{x^2} + \mathbf{y^2}) & \mathbf{x} & \mathbf{y} & \mathbf{1} \end{bmatrix}$$

$$\textit{Constraint matrix } \ \mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix}$$

5. **Maximum Likelihood Estimation (Question 5)**

   (a) To find the MLE of θ for a sample of n observations $(X_1, X_2, X_3 \ldots, X_n)$:

$$p(X \mid \theta) = \frac{e^{-\theta}\theta^X}{X!}$$

Assuming that all the n observations are independent of each other, the likelihood function is the product of their probability function given above. So,

$$p(X_1, X_2, X_3 \ldots, X_n \mid \theta) = \prod_{i=1}^{n} p(X_i \mid \theta)$$

As there is product on the right hand side we can take the log on both sides to convert it into log likelihood($L(\theta) = L(X_1, X_2, X_3 \ldots, X_n \mid \theta)$) form and easier calculation:

$$\begin{aligned} L(\theta) &= ln(\prod_{i=1}^{n} p(X_i \mid \theta)) \\ &= ln(\prod_{i=1}^{n} \frac{e^{-\theta}\theta^{X_i}}{X_i!}) \\ &= \sum_{i=1}^{n} ln(\frac{e^{-\theta}\theta^{X_i}}{X_i!}) \\ &= \sum_{i=1}^{n} \big(ln(e^{-\theta}) + ln(\theta^{X_i}) - ln(X_i!)\big) \\ &= \sum_{i=1}^{n} \big(-\theta + X_i ln(\theta) - ln(X_i!)\big) \\ &= -n\theta + ln(\theta)\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} ln(X_i!) \end{aligned}$$

Now, to find the maximum likelihood estimate we will take the derivative of the above log likelihood with respect to θ and equate it to zero as the first derivative for maxima is zero.

$$\frac{dL(\theta)}{d\theta} = 0$$

$$\frac{dL(\theta)}{d\theta} = \frac{d}{d\theta}\big(-n\theta + ln(\theta)\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} ln(X_i!)\big)$$

$$0 = -n + \frac{1}{\theta}\sum_{i=1}^{n} X_i$$

$$n = \frac{1}{\theta}\sum_{i=1}^{n} X_i$$

$$\theta = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Thus, we finally get the maximum likelihood estimate of θ to be equal to the sample mean.

(b) As stated above likelihood function is

$$p(X_1, X_2, X_3 \ldots, X_n \mid \theta) = \prod_{i=1}^{n} p(X_i \mid \theta)$$

while the log-likelihood function is

$$L(\theta) = \sum_{i=1}^{n} ln(p(X_i \mid \theta)) = \sum_{i=1}^{n} ln\big(\frac{e^{-\theta}\theta^{X_i}}{X_i!}\big)$$

Log likelihood is preferred over likelihood as the summation operation in log likelihood reduces the operations required to find the derivative to a very large extent as the product rule on n factor multiplication in likelihood is difficult.

Also, using log likelihood being monotonically increasing does not lead to loss of generality as the likelihood will be maximum at the same time as the log likelihood achieves its maxima.

(c) Sample data is given to be [15, 8, 13, 11, 7, 16, 25, 30]
To calculate the specific value of the data we use the result obtained in previous part:

$$\theta = \frac{1}{n}\sum_{i=1}^{n} X_i$$
$$= \frac{15 + 8 + 13 + 11 + 7 + 16 + 25 + 30}{8}$$
$$= 15.625$$

The specific value of θ = 15.625 hours.

References

1. http://mlwiki.org/index.php/Normal_Equation#Minimization
2. https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm
3. Class notes supplied by Prof. Pankaj Kumar were used for general reference.

To view the codes open the given link in google colab:
https://colab.research.google.com/drive/1qIbcw13Rwm_NWIqis6QRYVZM_7KydeZV?usp=sharing