ENPM 673

Perception for Autonomous Robots

Homework Report

[Software Used: Python]

Members: 1. Chayan Kumar Patodi (UID : 116327428)
2. Prasanna Marudhu Balasubramanian (UID : 116197700)
3. Saket Seshadri Gudimetla Hanumath (UID : 116332293)

For Problem Statement 1:

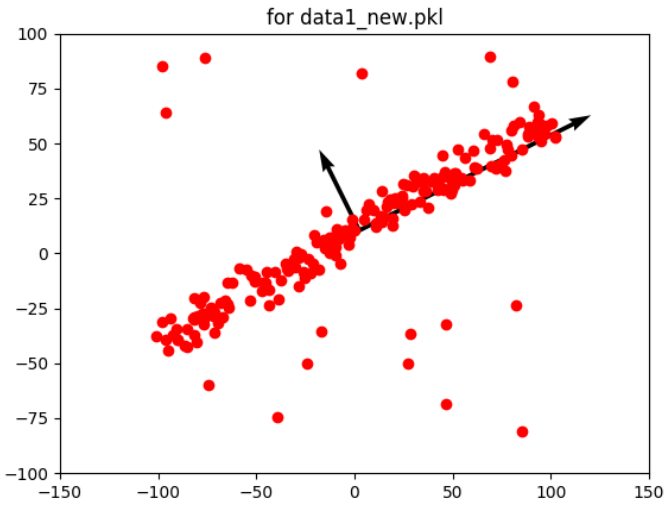
We first extracted the data from the given data points. We got some help from the read_data file. The trick was to get the proper syntax , as we were only allowed to use Numpy & matplotlib. We used the inbuilt function np.cov() to calculate the Covariance Matrices. Then we used another inbuilt function np.linalg.eig() on the covariance matrices to get the eigen values and eigen vectors respectively.

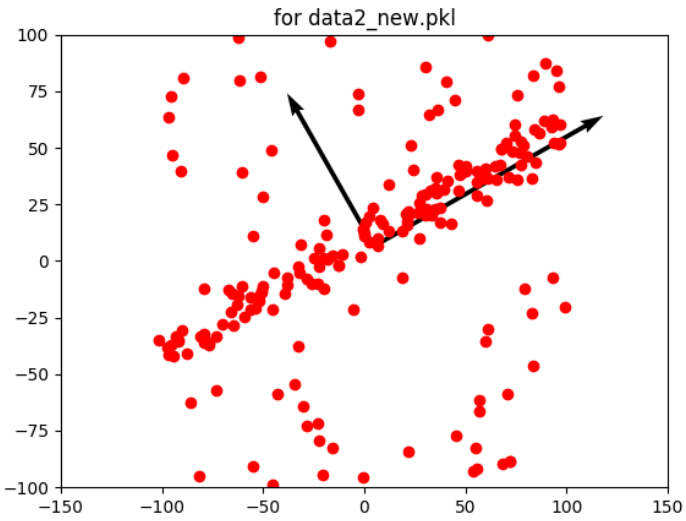
Our previous knowledge of eigen values and vectors was that , A scalar λ is called an eigenvalue of the $n \times n$ matrix A is there is a nontrivial solution x of $Ax = \lambda x$. Such an x is called an eigenvector corresponding to the eigenvalue λ , but after reading the mentioned links in the pdf , we also understood the relation between the covariance matrices and eigen values and vectors. Eigen Decomposition is a important concept from the point of the course.

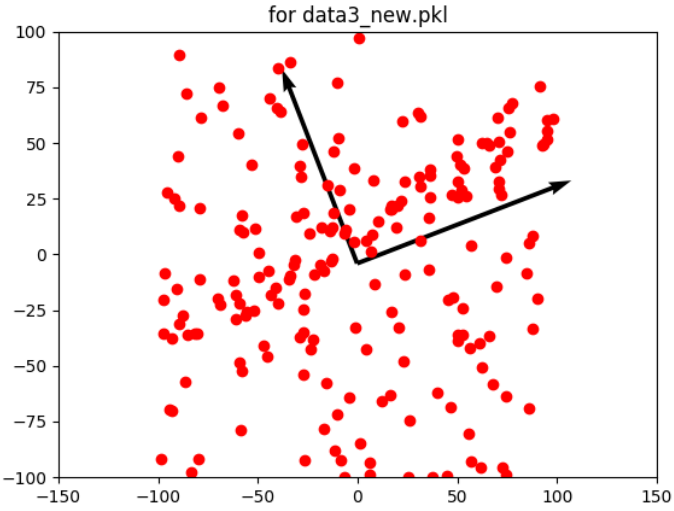
```
patodichayan@Chayan: ~/Downloads/HW1_data
patodichayan@Chayan:~/Downloads/HW1_data$ python3 P1.py
CovarianceMatrix1:

 [[3407.3108669  1473.06388943]
 [1473.06388943 1169.53151003]]
CovarianceMatrix2:

 [[3418.29090234 1039.75748627]
 [1039.75748627 2177.72593476]]
CovarianceMatrix3:

 [[3086.14137209  326.06544037]
 [ 326.06544037 2369.3509072 ]]
EigenValue1:

 [4138.24045932  438.60191761]
EigenValue2:

 [4008.72968704 1587.28715007]
EigenValue3:

 [3212.27252081 2243.21975848]
EigenVector1:

 [[ 0.89578578 -0.44448604]
 [ 0.44448604  0.89578578]]
EigenVector2:

 [[ 0.86957598 -0.49379917]
 [ 0.49379917  0.86957598]]
EigenVecto3:

 [[ 0.93265255 -0.36077585]
 [ 0.36077585  0.93265255]]
patodichayan@Chayan:~/Downloads/HW1_data$
```

Then we plotted 3 different figures for 3 different datasets , just to clearly see the direction and orientation of the quiver's with respect to the dataset. We used plt.quiver() to generate the plot.

The problems we initially faced was b/w understanding the relation b/w variance and eigenvalue and vectors , but with the help of supplemental readings , we got through the problems and presented a solution using the program.







Understanding of Eigen decomposition concept:

The data in the given data file is scattered in the xy-plane.

The variance value provides information about the range of data spread with respect to the major and minor axes in a given area or space. For instance the value of Variance(y, y) shows the level of data spread in the y-direction.

The relation between the level of spread of data in one axis or direction with respect to other direction or axes is provided by the co-variance value. This value depicts the co-relation between the axial spreading of data.

The covariance values can be computed into a matrix. This values tells whether there is a positive or negative correlation between the axial spread of data in space. In 2-D space, the covariance value shows that there is a diagonal spread of data

(either +ve or –ve co-relation) and when there is no covariance value then it means that there is only an axial spread of data.

The Eigen values and Eigen vectors can be computed from this covariance matrix values. The Eigen vector shows the direction of the data spread and it helps in predicting the future direction of spreading of new data. Whereas, the Eigen value is a value which shows the level or amount of data spread in a particular direction basically the magnitude of the variance.

The Eigen Vector is plotted in the scatter plot with respect to the level of variance in each direction. The Python code is attached in P1.py file.

For Problem Statement 2:

The real challenge in this problem was to find the correct formula for the slope of the line. We wanted the values of the slope and intercept, that gives the minimum error for the given dataset. The steps that are followed in Least Square method is:
1. Find the Quadratic Loss function.
2. Squaring the Loss Function and Finding its partial derivative.
3. Equating that to Zero and Solving for Slope.

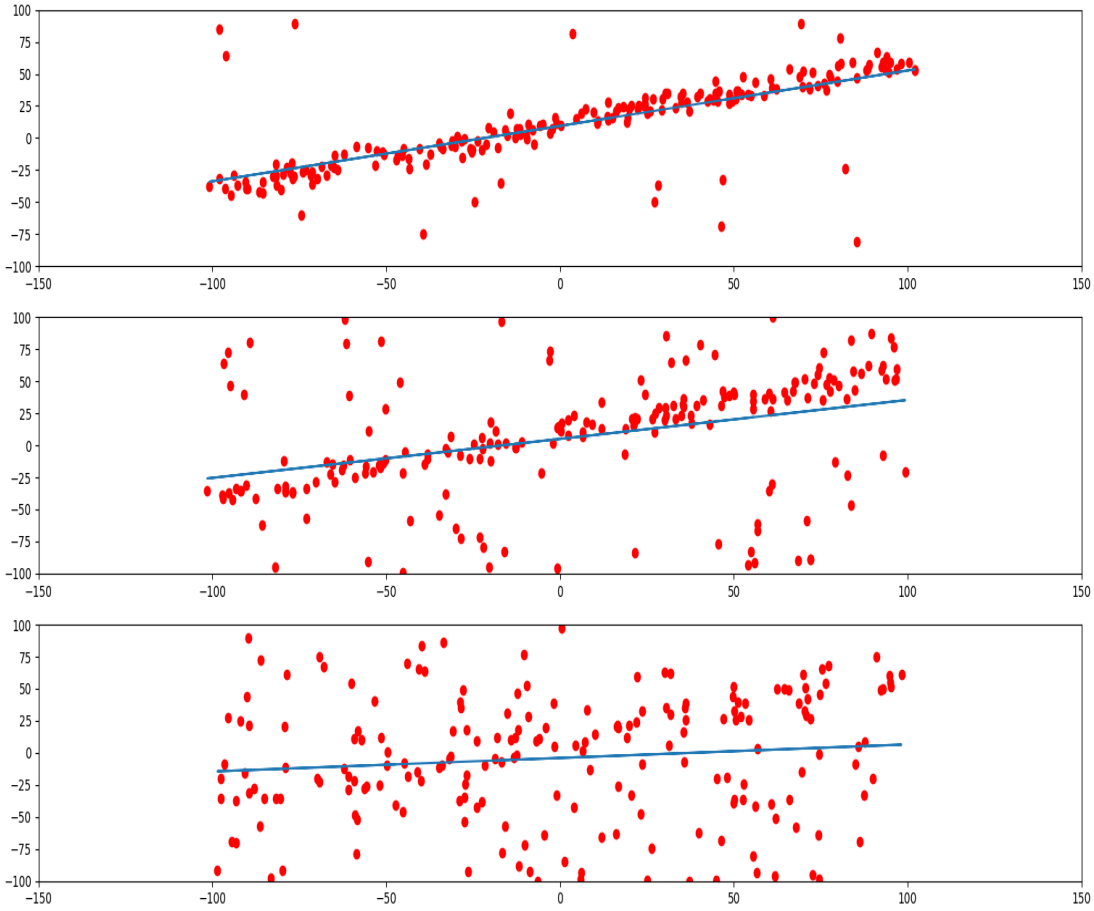
Intercept is dependent on the Slope. Using the equations, we create a function and write the code to solve for line fitting using least squares (for both vertical offsets as well as orthogonal offsets.)

(a). Vertical Distance:

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$c = \bar{y} - m\bar{x}$$

Output:



(b). Orthogonal Distance:

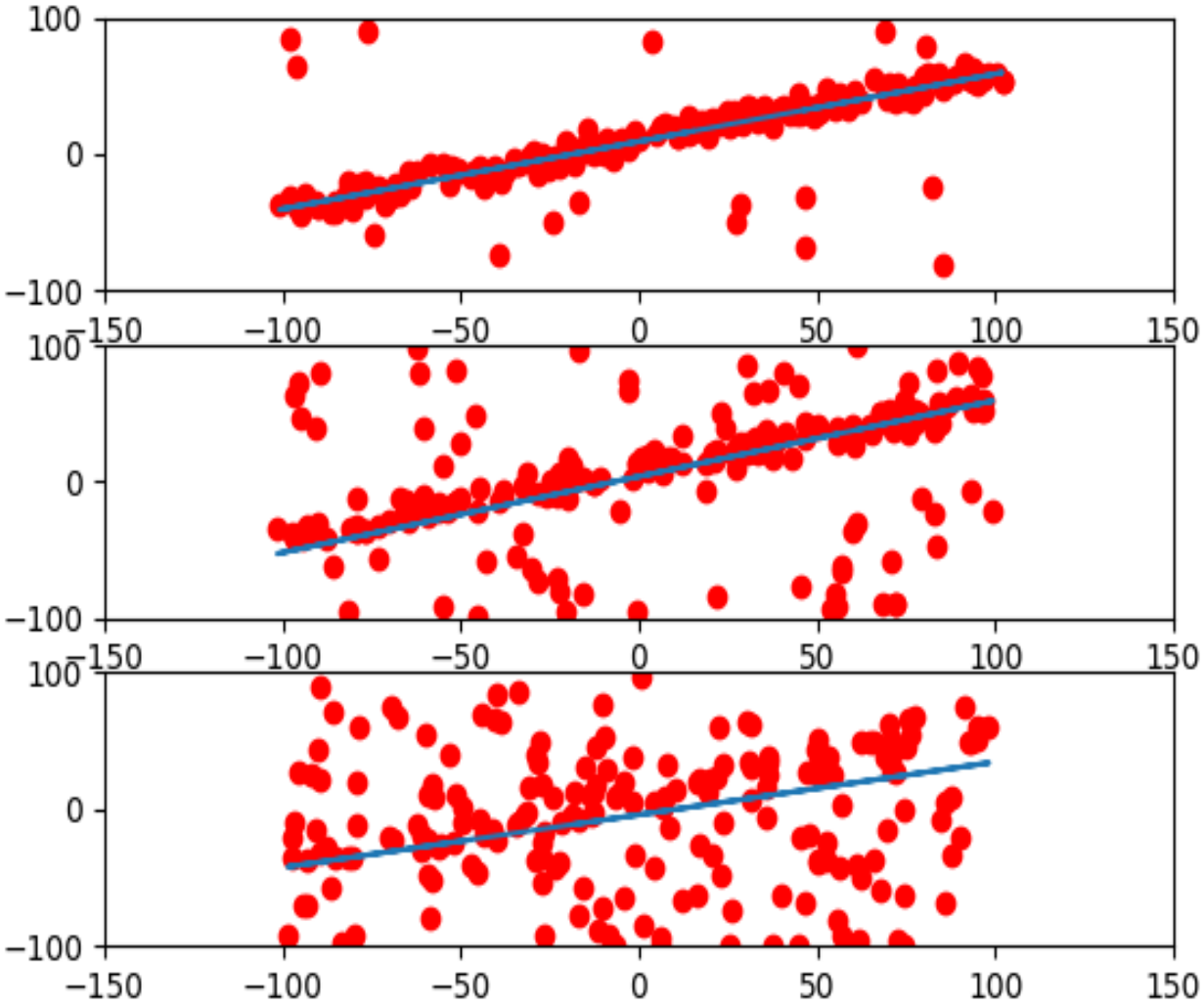
$$B \equiv \frac{1}{2} \frac{\left[\sum_{i=1}^{n} y_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} y_i\right)^2\right] - \left[\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right]}{\frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i y_i}$$

$$= \frac{1}{2} \frac{\left(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2\right) - \left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)}{n\bar{x}\,\bar{y} - \sum_{i=1}^{n} x_i y_i},$$

$$b = -B \pm \sqrt{B^2 + 1},$$

Where b = slope.

Output:



Orthogonal Regression for Data1, Data2 and Data3

Implementation based on understanding:

In case of **simple** linear regression method, we aim in developing the code to minimize the sum of the squared *vertical* distances between the *y* data values and the corresponding *y* values on the fitted line.

On the other hand, in case of **orthogonal regression,** the aim was to develop the algorithm to minimize the orthogonal (perpendicular) distances from the data points to the fitted line.

The Python code was developed for both the Simple and Orthogonal regression and plotted in the graph. The code is available in the P2a.py and P2b.py file.

For Problem Statement 3:

The difficult part in this problem was to decide the outlier rejection technique. To run the code for each dataset and finalizing one. We understood a lot about the techniques and why and how we choose certain methods.

Steps: 1. The outlier rejection technique is adopted in this to eliminate the influence of the outliers in fitting a line for points in the space

2. The rejection technique for each of the datasets is analyzed and implemented.

Dataset 1:

Method Adopted: RANSAC

The outlier rejection technique adopted for this dataset is RANSAC method. We generally adopt RANSAC algorithm to be used with other parameter estimation methods in order to obtain robust models with a certain probability when general assumptions about noise fail. It is used particularly because it determines (with an acceptable accuracy) certain function parameters even in the presence of gross-erroneous samples that can mislead the parameter estimation. In RANSAC method the algorithm splits the datasets into outliers and inliers, the outliers being the prime reason for increasing the prediction error. It does a pretty good job of rejecting the outliers that contribute no information about the model at all.

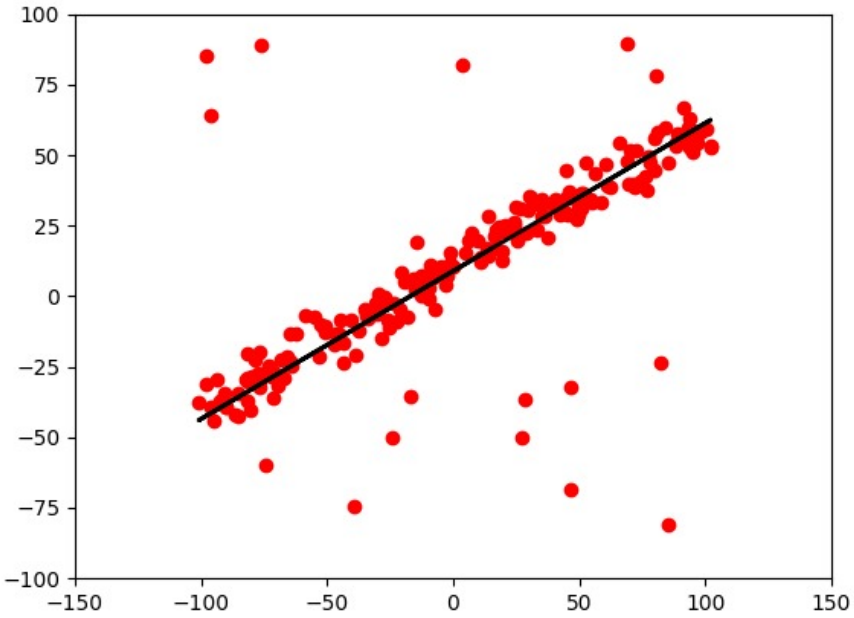
RANSAC algorithm begins with selection of a small set of random data points that are used to build a rough model (like a line for eg.). Then it determines which samples are within a certain error tolerance of the generated model. The samples are considered as agreed with the generated model and called as consensus set of the chosen samples. Data samples in the consensus are called inliers and the rest are called outliers. If the count of the samples in consensus is high enough then the algorithm trains the final model of the consensus using them. This process is repeated for a calculated number of iterations and the result of this is given in the form of a model that has the smallest average error among the generated models.

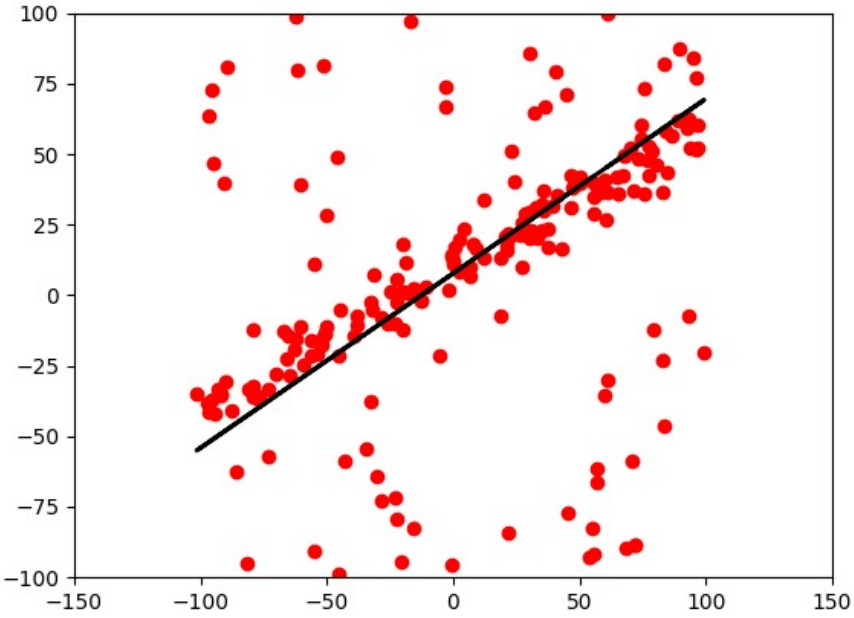
Advantage of RANSAC

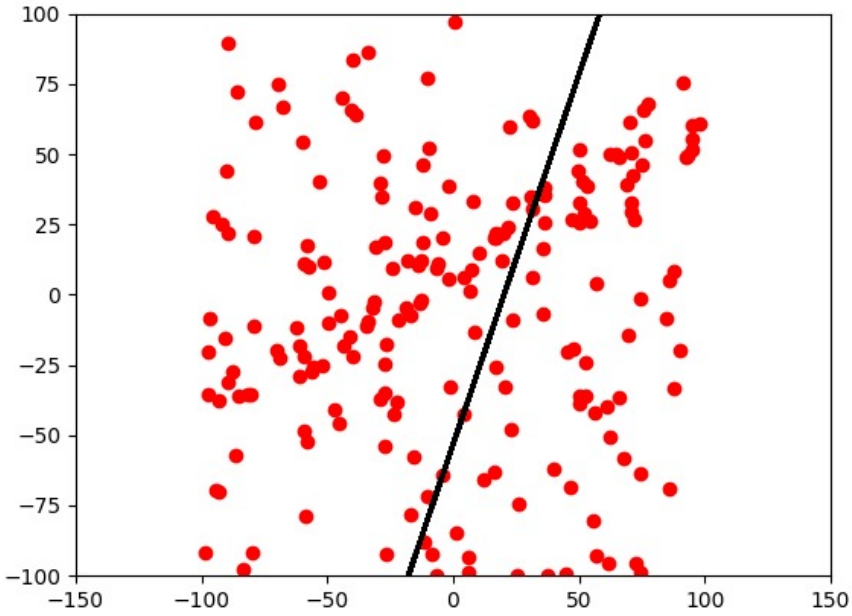
1. Robust against outliers
2. It is modular

Drawbacks of RANSAC:

1. Requires prior knowledge of data.
2. Number of iterations increases logarithmically with outlier percentage.


100
75
50
25
0
−25
−50
−75
−100
−150
−100
−50
0
50
100
150


100
75
50
25
0
−25
−50
−75
−100
−150
−100
−50
0
50
100
150



The python code for this problem statement is attached with the report, and is named P3R1.py, P3R2.py and P3R3.py. All of them are done for the respective datasets.

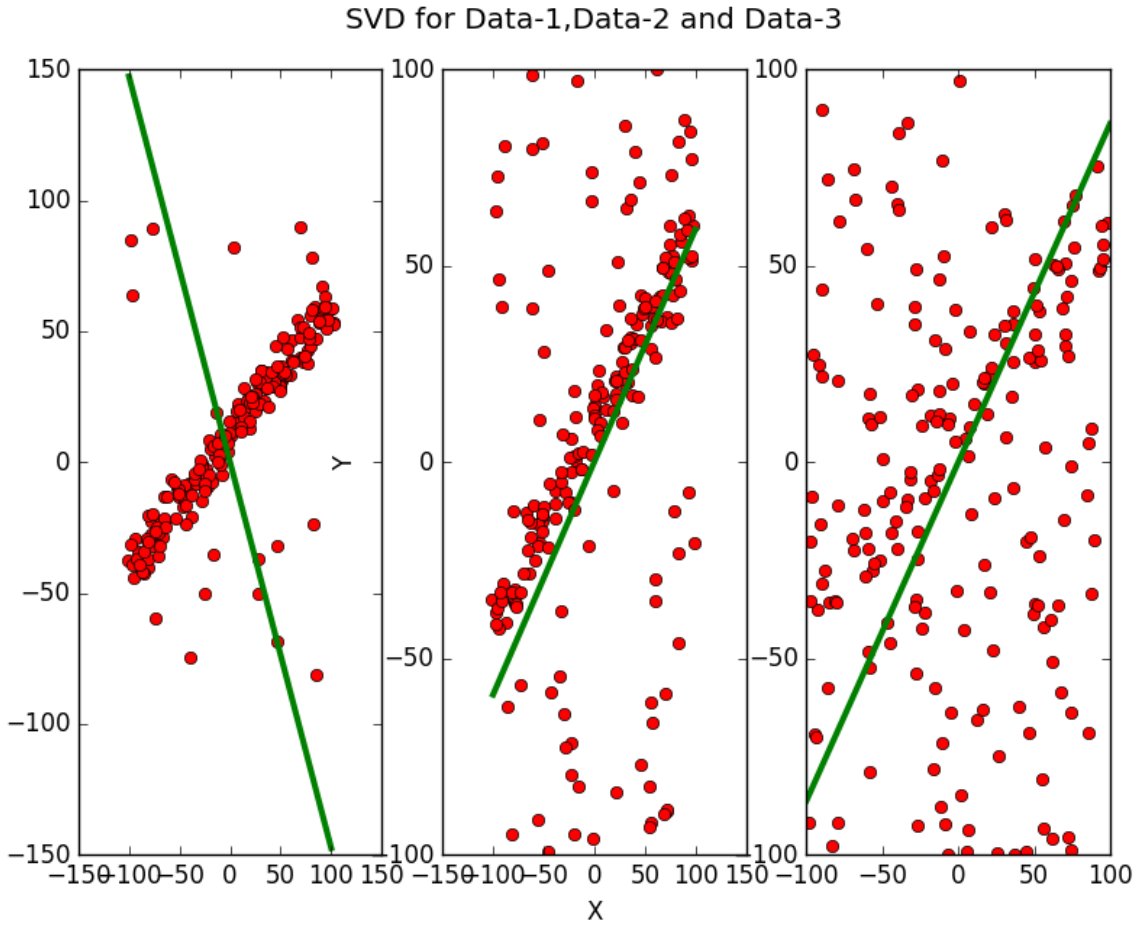
Dataset 2 & 3:

Method Adopted: Singular Value Decomposition.

In this Single Value Decomposition method, the given data from the dataset is formulated into a matrix and then the SVD decomposition is done to obtain the Unitary Matrix, Diagonal matrix and Unitary Transpose matrix were found and the slope and intercepts value for the best fit line was obtained from these matrix.

For the Dataset 2 and Dataset 3 we were able to obtain the plot of good fit comparing to the RANSAC method for the same datasets.

Since SVD is non random method, there is less calculation and computation process comparative to the RANSAC and other method. Also the number of iteration is less and processing duration for SVD method.


SVD for Data-1,Data-2 and Data-3
150
100
50
0
−50
−100
−150
−150 −100 −50 0 50 100 150
Y
100
50
0
−50
−100
−150 −100 −50 0 50 100 150
X
100
50
0
−50
−100
−100 −50 0 50 100