Pattern Recognition
Assignment 1

ED13B013 Sai Bhargav Ramu
CS17S008 Nitesh Methani
Group Number : 36

August 31, 2017

Contents

1 Image Reconstruction

Given an image (which is stored in the form of the matrix), we have to decompose the matrix using Singular Value Decomposition(SVD) and Eigen Value Decomposition(EVD) respectively. Using the singular values and eigen values, we have to reconstruct the original matrix (and hence original image) using the fewer singular values and eigen values. We also have to calculate the error and plot a line graph which has Number of Singular/ Eigen Values on X-axis and Error on Y-axis. The following sections decompose the matrix and reconstruct it using both the techniques and the observations made while experimenting are also discussed at the end of the section.

1.1 Singular Value Decomposition on both Square and Rectangular images

Using SVD an image matrix A_{mxn} can be expressed as the product of three matrices U, sigma and V where U is a mxm matrix, V is a nxn matrix and sigma is mxn matrix. U and V are orthogonal matrices. The columns of U and V are the eigen vectors of the the matrix A * A^t and A^t * A respectively. sigma is a diagonal matrix whose elements are the signular values of imageMatrix. Sngular Values are calculated as the square root of the eigen values. Once we have U, V and sigma we will reconstruct the imageMatrix and find its error image. Reconstruction of the imageMatrix can be done using the formula: [

$$reconstructedImageMatrix = U * sigma * V'$$

Matrix for Error image can be calculated as
$errorMatrix = originalImageMatrix - reconstructedImageMatrix$
The mathematical implementation for SVD is done in file mySVD.m

Now instead of using all the singular values. We have to reconstruct the matrix using top few singular values only. The following section describe this job in three different parts of the subsections.

1.1.1 By Converting the Image to Grayscale

Steps performed :

1. Read the image (Square/Rectangle) and convert it into Grayscale image.
2. Perform Singular value decomposition using the mySVD() function and store the value in U, sigma and V matrix respectively.
3. $experiment_on_Singular Vals()$ function will modify the sigma by taking only the top k singular values for different k and will reconstruct the image using this modified sigma.
   It will also return the error metric for different values of k.
4. Plot the graph between the number of singular values taken and the corresponding error metric.

Reconstructing using top N singular values

Reconstruct the images by taking only few singular values.
In our first iteration, we will take top 4 singular values and make others zero. Then reconstruct the image using this modified sigma matrix using the formula $modified_image = U * sigmaModified * V'$. Show the reconstructed image using imshow() function of Matlab. Calculate the error as
$error = abs(sum(sum(original_image - modifiedImage)))$;

In the second iteration, take top 29 singular values and repeat the above stps to get the modified sigma and calculate the error after taking 29 singular values.

Repeat the same procedure for some k singular values where $1 <= k <= 256$

In the second iteration, top 54 singular values in third iteration and so on....
Following figures shows the output for k= 4,29,.. singular values:

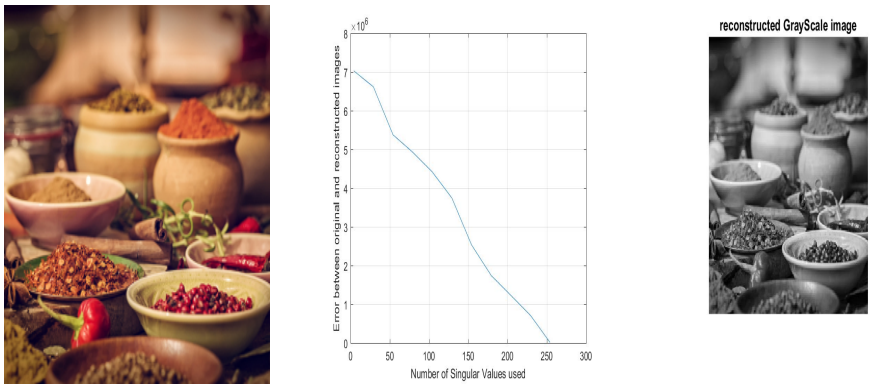
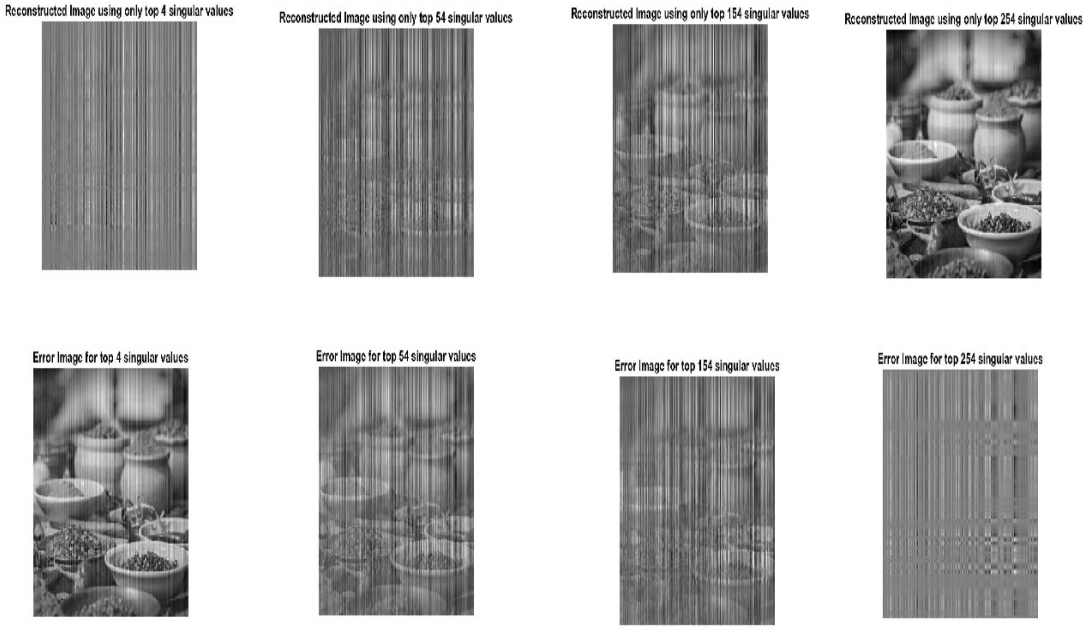


Figure 1: Original and Reconstructed Grayscale Images and Error Graph

Figure 2: Reconstructed images with their corresponding error images.



1.1.2 Seperately on each color band

Steps performed :

1. Read the image (Square/Rectangle) and extract its r, g and b channel.
2. Perform Singular value decomposition using the mySVD() function and store the value in U, sigma and V matrix respectively.
3. $experiment_on_Singular Vals()$ function will modify the sigma by taking only the top k singular values for different k and will reconstruct the image using this modified sigma.
   It will also return the error metric for different values of k.
4. Plot the graph between the number of singular values taken and the corresponding error metric.

Reconstructing using top N singular values

Reconstruct the images by taking only few singular values.
In our first iteration, we will take top 4 singular values and make others zero.

Then reconstruct the image using this modified sigma matrix using the formula $modified_image = U * sigmaModified * V'$. Show the reconstructed image using imshow() function of Matlab. Calculate the error as
$error = abs(sum(sum(original_image - modifiedImage)))$;

In the second iteration, take top 29 singular values and repeat the above stps to get the modified sigma and calculate the error after taking 29 singular values.

Repeat the same procedure for some k singular values where $1 <= k <= 256$

In the second iteration, top 54 singular values in third iteration and so on.... Following figures shows the output for k= 4,29,.. singular values:

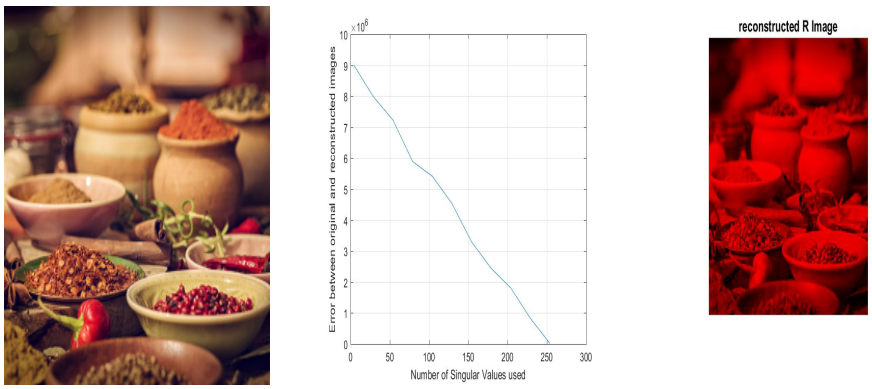
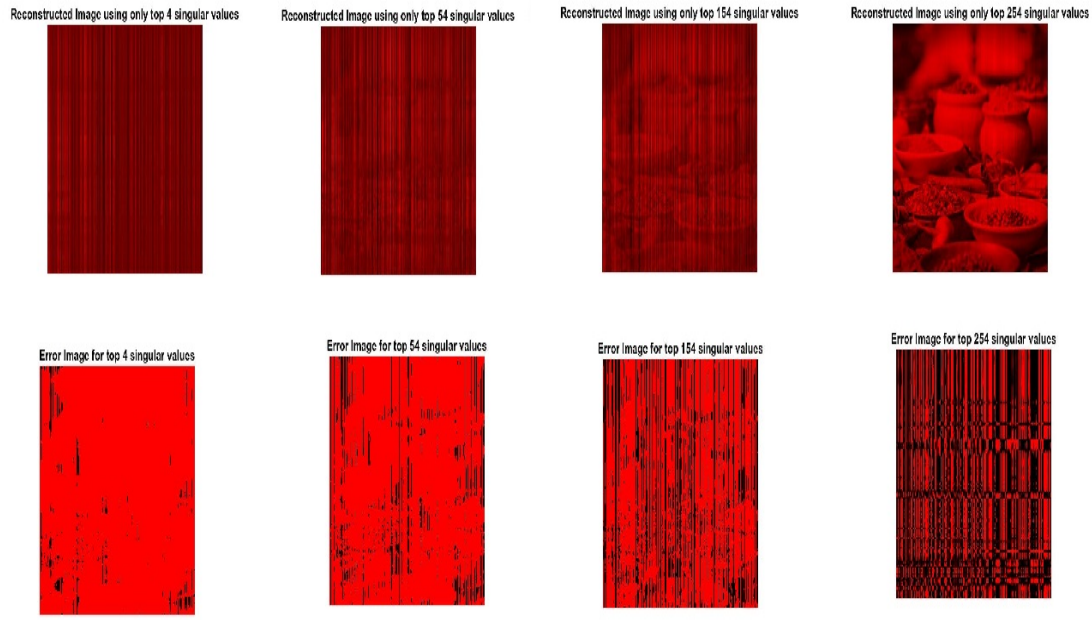


Figure 3: Original and Reconstructed R Channel Images and Error Graph

Repeating the same steps for each channel we get the following output for G and B channels respectively.

Figure 4: Reconstructed R Channel images with their corresponding error images.



1.1.3 After Concatenating the 8-bit R,G,B channel to form a 24-bit number

Reconstructing using top N singular values

Steps performed:

1. Read the image (Square/Rectangle) and extract its r, g and b channel.
2. Concatenate the 8 bit channels and form a 24 bit image Matrix call it as newImage.
3. Perform Singular value decomposition using the mySVD() function and store the value in U, sigma and V matrix respectively.
4. $experiment_on_Singular Vals()$ function will modify the sigma by taking only the top k singular values for different k and will reconstruct the image using this modified sigma.
   It will also return the error metric for different values of k.

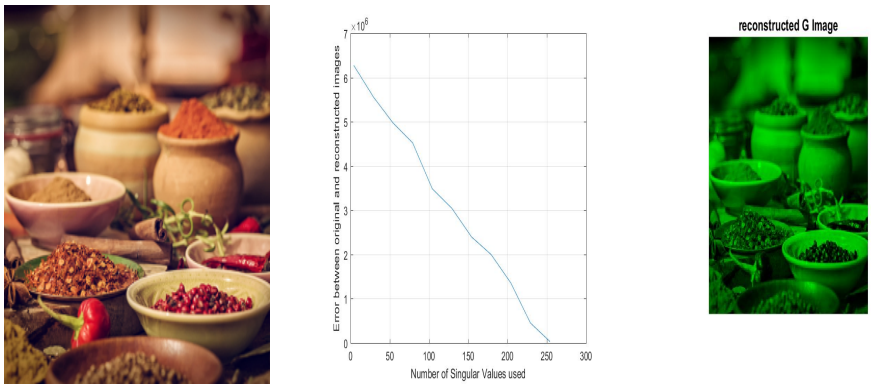
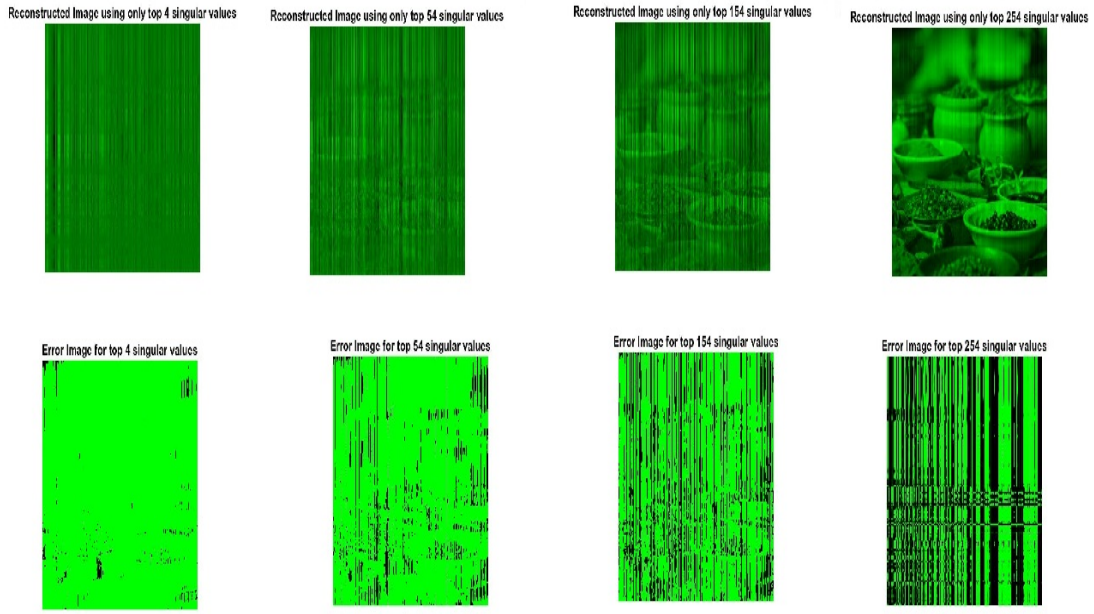


Figure 5: Original and Reconstructed G Channel Images and Error Graph

Figure 6: Reconstructed G Channel images with their corresponding error images.



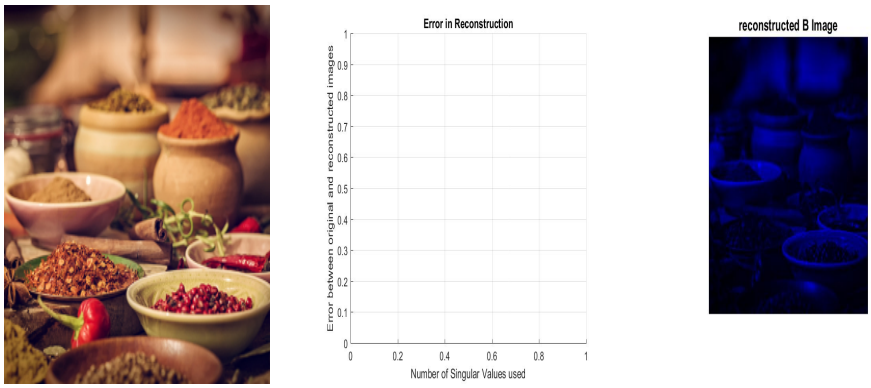


Figure 7: Original and Reconstructed B Channel Images and Error Graph

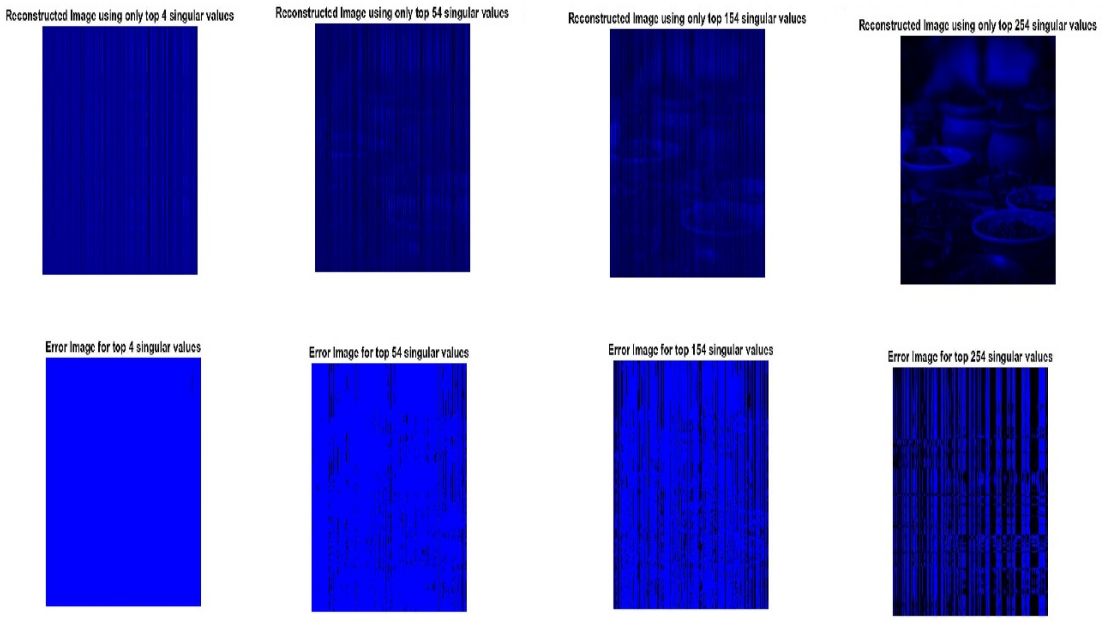


Figure 8: Reconstructed B channel images with their corresponding error images.

5. Plot the graph between the number of singular values taken and the corresponding error metric.

6. Then repeat the same set of experiments we have done above for the grayScale image.

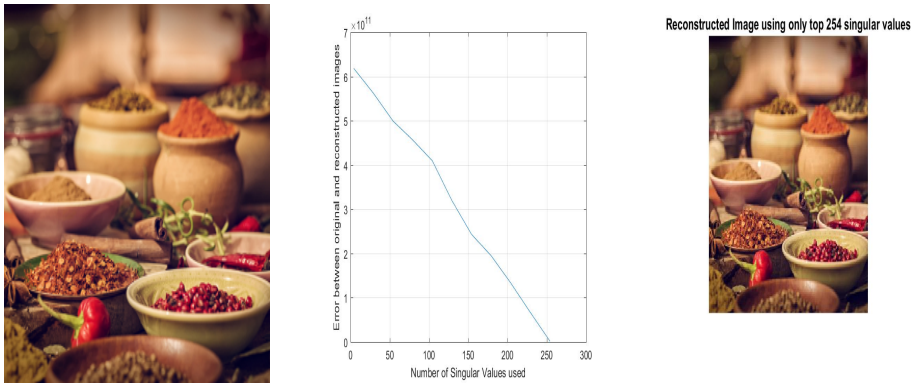
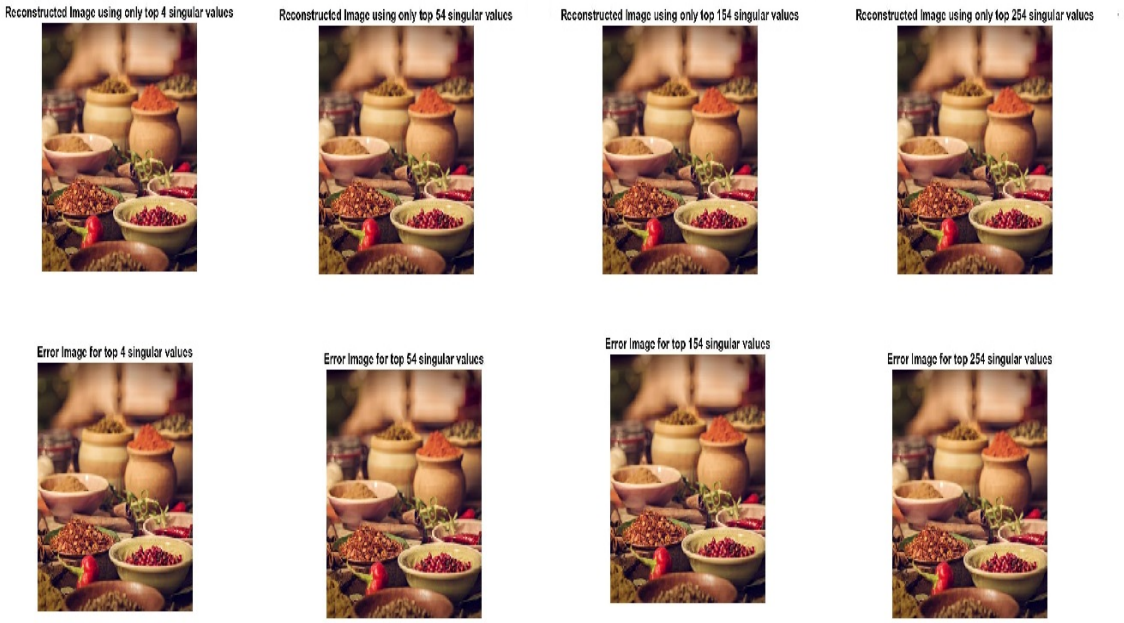


Figure 9: Original and Reconstructed Images and Error Graph

Figure 10: Reconstructed images with their corresponding error images.



1.2 Eigen Value Decomposition on both Square and Rectangular images

Using EVD an image matrix A_{mxn} can be expressed as the product of three matrices V, lambda and inverse of V. The columns of V are the eigen vectors of A. lambda is a diagonal matrix whose elements are the eigen values of imageMatrix. Once we have V and lambda we will reconstruct the imageMatrix and find its error image. Reconstruction of the imageMatrix can be done using the formula: [

$$reconstructedImageMatrix = V * lambda * inv(V)$$

Matrix for Error image can be calculated as
$errorMatrix = originalImageMatrix - reconstructedImageMatrix$

Now instead of using all the eigen values. We have to reconstruct the matrix using top few eigen values only. The following section describe this job in three different parts of the subsections.

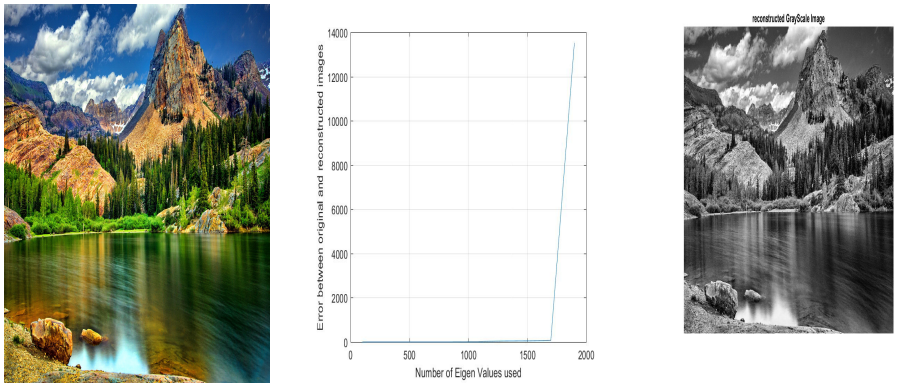


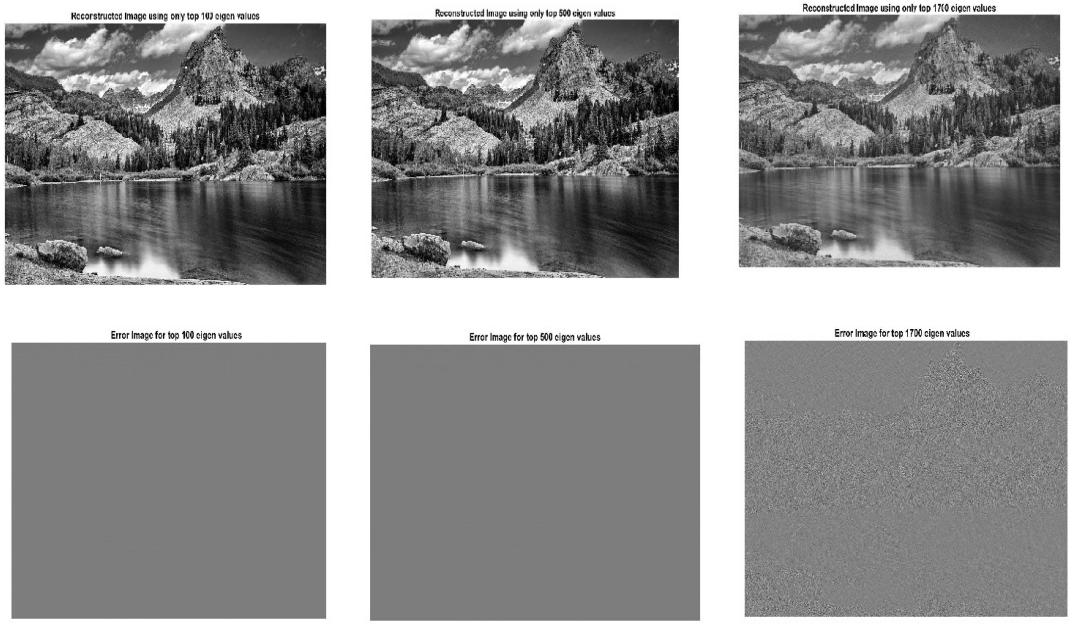
Figure 11: Original and Reconstructed Images and Error Graph

1.2.1 By Converting the Image to Grayscale

Steps performed :

1. Read the image (Square/Rectangle) and convert it into Grayscale image.
2. Perform Eigen value decomposition using the eig() function and store the value in lambda and V matrix respectively.
3. $experiment_on_Eigen Vals()$ function will modify the sigma by taking only the top k eigen values for different k and will reconstruct the image using this modified lambda.
   It will also return the error metric for different values of k.

Figure 12: Reconstructed images with their corresponding error images.



4. Plot the graph between the number of Eigen values taken and the corresponding error metric.

Reconstructing using top N Eigen values

Reconstruct the images by taking only few Eigen values.
In our first iteration, we will take top 4 Eigen values and make others zero. Then reconstruct the image using this modified sigma matrix using the formula $modified_image = V * lambdaModified * inv(V)$. Show the reconstructed image using imshow() function of Matlab. Calculate the error as
$error = abs(sum(sum(original_image - modifiedImage)))$;

In the second iteration, take top 29 Eigen values and repeat the above stps to get the modified lambda and calculate the error after taking 29 Eigen values. Repeat the same procedure for some k singular values where $1 <= k <= 256$

Take top 54 Eigen values in third iteration and so on....
Following figures shows the output for k= 4,29,.. Eigen values:

1.2.2 Seperately on each color band

Steps performed :

1. Read the image (Square/Rectangle) and extract its r, g and b channel.
2. Perform Eigen value decomposition using the eig() function and store the value in lambda and V matrix respectively.
3. $experiment_on_Eigen Vals()$ function will modify the lambda by taking only the top k eigen values for different k and will reconstruct the image using this modified lambda.
   It will also return the error metric for different values of k.
4. Plot the graph between the number of Eigen values taken and the corresponding error metric.

Reconstructing using top N Eigen values

Reconstruct the images by taking only few Eigen values.
In our first iteration, we will take top 4 Eigen values and make others zero. Then reconstruct the image using this modified sigma matrix using the formula $modified_image = V * lambdaModified * inv(V)$. Show the reconstructed image using imshow() function of Matlab. Calculate the error as
$error = abs(sum(sum(original_image - modifiedImage)))$;

In the second iteration, take top 29 Eigen values and repeat the above stps to get the modified sigma and calculate the error after taking 29 singular values.

Repeat the same procedure for some k Eigen values where $1 <= k <= 256$

In the second iteration, top 54 Eigen values in third iteration and so on....
Following figures shows the output for k= 4,29,.. Eigen values:

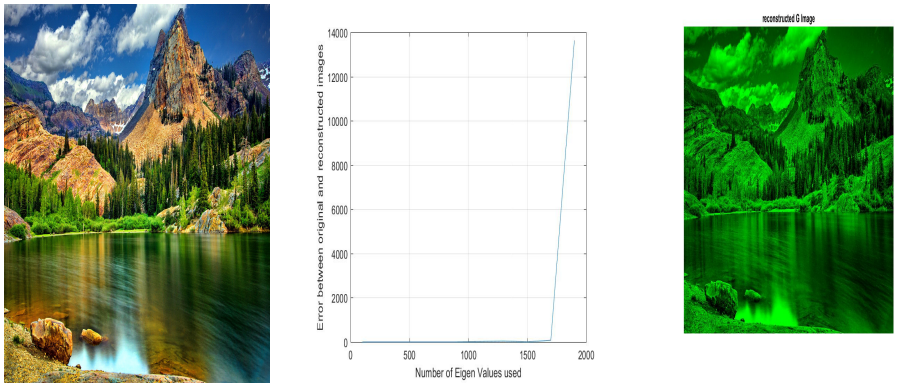
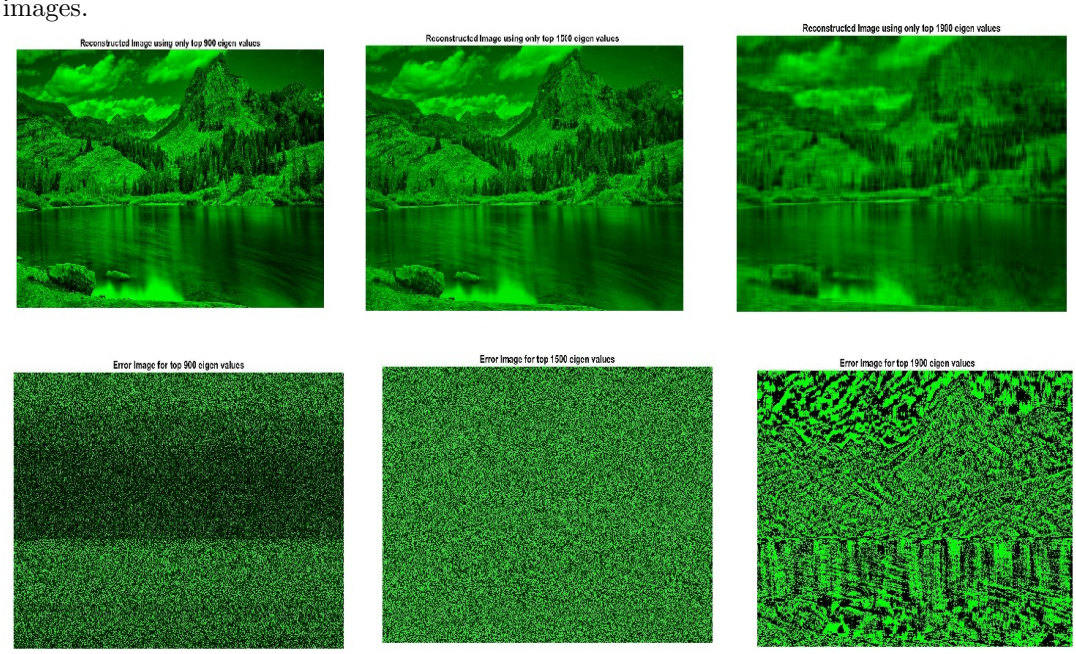


Figure 13: Original and Reconstructed Green Channel Images and Error Graph

Figure 14: Reconstructed Green Channel images with their corresponding error images.



Repeating the same steps for each channel we will get the following output.

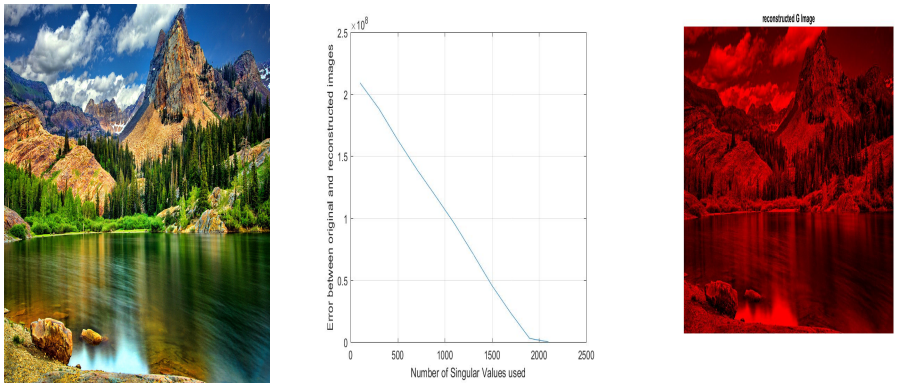
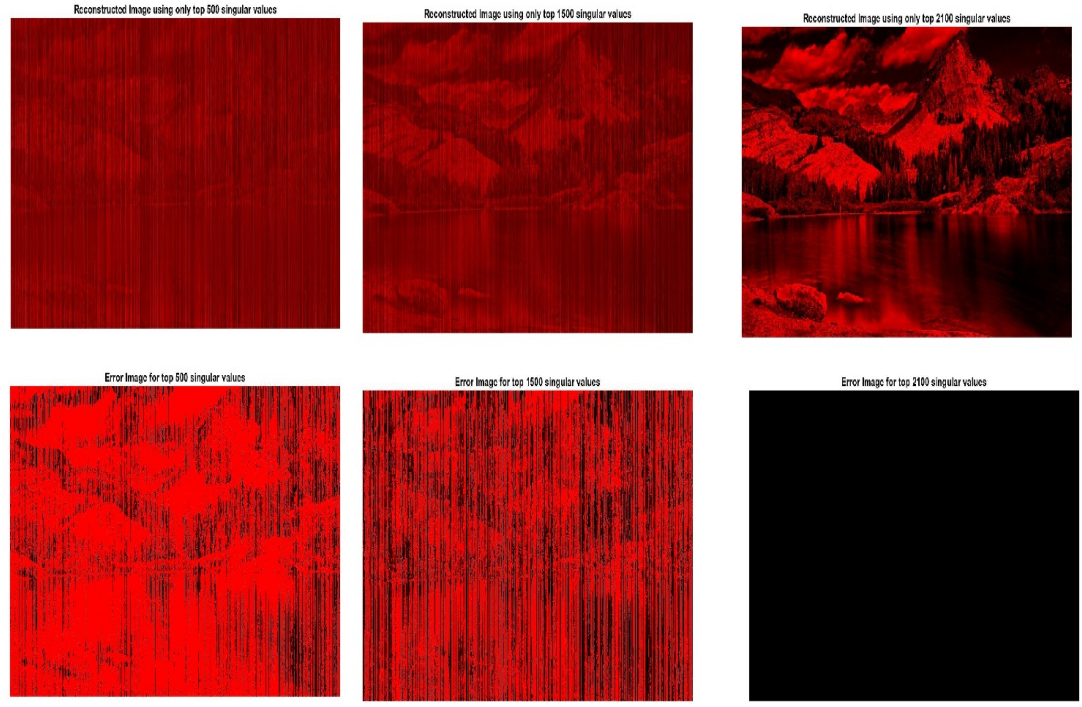


Figure 15: Original and Reconstructed Red Channel Images and Error Graph

Figure 16: Reconstructed Red Channel images with their corresponding error images.



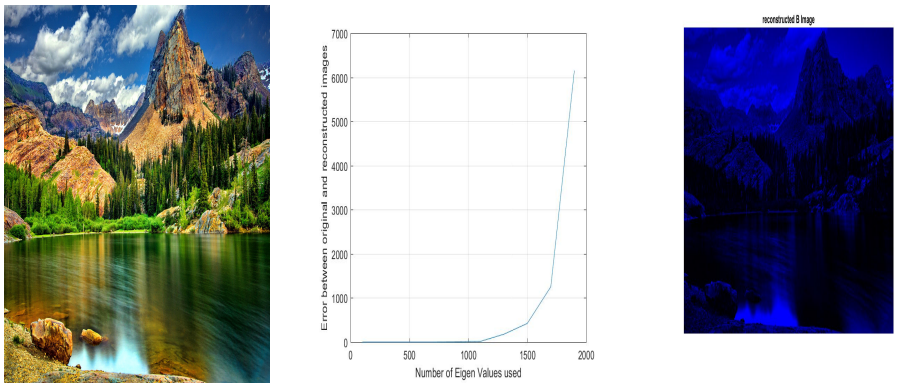


Figure 17: Original and Reconstructed Blue Channel Images and Error Graph

1.2.3 After Concatenating the 8-bit R,G,B channel to form a 24-bit number

Reconstructing using top N singular values

Steps performed:

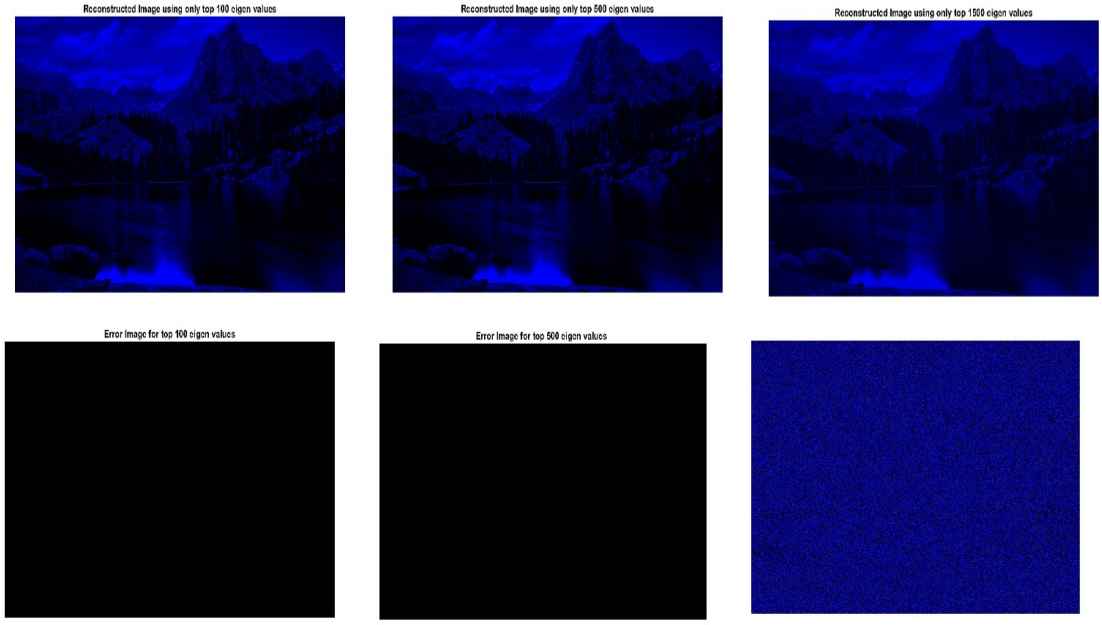


Figure 18: Reconstructed Blue Channel images with their corresponding error images.

1. Read the image (Square/Rectangle) and extract its r, g and b channel.
2. Concatenate the 8 bit channels and form a 24 bit image Matrix call it as newImage.
3. Perform Eigen value decomposition using the eig() function and store the value in lambda and V matrix respectively.
4. $experiment_on_Eigen Vals()$ function will modify the lambda by taking only the top k eigen values for different k and will reconstruct the image using this modified lambda.
   It will also return the error metric for different values of k.
5. Plot the graph between the number of Eigen values taken and the corresponding error metric.
6. Then repeat the same set of experiments we have done above for the grayScale image.

In this problem we learned how to reconstruct the image using its eigen/ singular values. The storage space required to store the reconstructed image is far less than the storage space required by the original image. The dimensions of the image are reduced without compromising the quality of the image. Less dimensions means less computing and hence faster execution.

2 Polynomial Regression

70% of the data is used for training (x_{train}), 20% of data for validation (x_{valid}) and 10% of data for testing (x_{test}).

2.1 1-dimensional data

For 1-Dimensional data we have x,y as

$$x = [x_1, \cdots, x_n]^T, y = [y_1, \ldots, y_n]^T$$

Let us modify it as,

$$A = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

$$A\theta = y$$

2.1.1 Different order of polynomials and their best fit

By using 70% of x data we calculate the estimate of θ i.e $\hat{\theta}$. Using method of least squares we get,

$$\hat{\theta} = (A^T A)^{-1} A^T y$$

We can decide the best fit based on SSE and R^2 parameters

$$SSE = \sum((A\hat{\theta})_i - y_i)^2$$

$$R^2 = 1 - \frac{SSE}{\sum(y_i - \bar{y})}$$

We can notice that as degree of the curve is increased after some point SSE starts increasing and R^2 starts dropping for validation data. The increase in R^2 for x_{train} and decrease of the same for x_{valid} is due to over fitting of data. So, from the above plot we can conclude that degree 4 is best fit for the given dataset.

On x_{test} we got $SSE = 14.63$ and $R^2 = 0.903$

Figure 19: Curve fitting for different degree on all given dataset

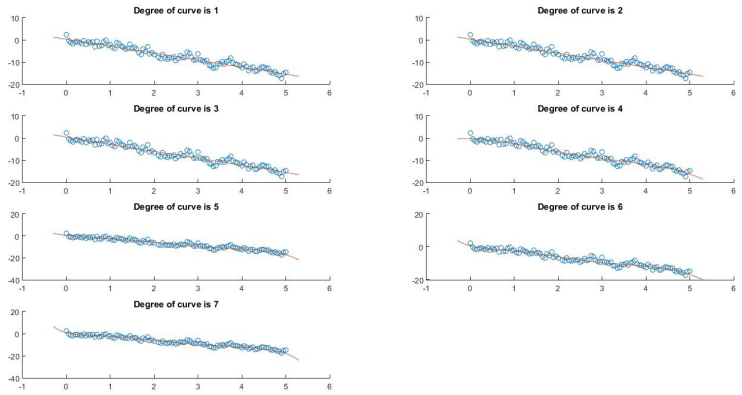
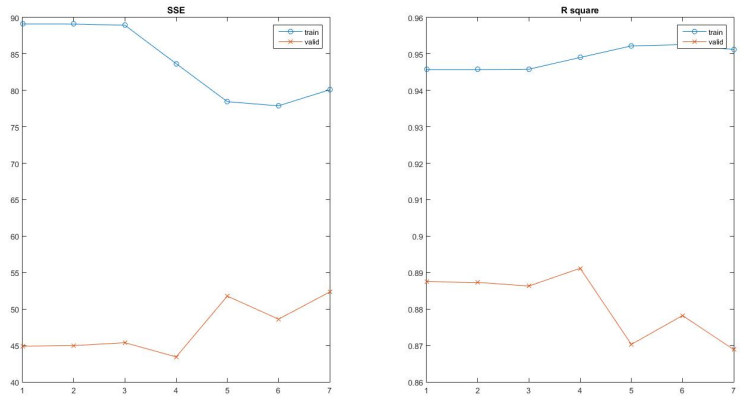


Figure 20: SSE and R^2 error for different degree of the polynomial



2.1.2 Ridge regression

Normal regression gives you unbiased regression coefficients Ridge and lasso regression allow you to shrink coefficients. This means that the estimated coefficients are pushed towards 0, to make them work better on new data-sets. This allows us to use complex models and avoid over-fitting at the same time.

Figure 21: Final polynomial fit with its residuals

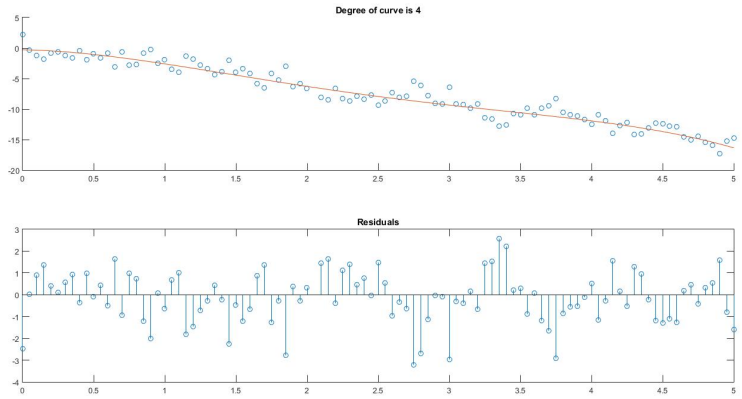
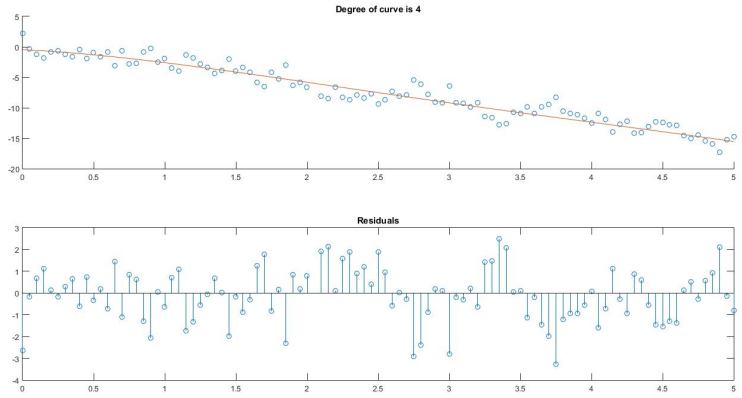


Figure 22: For $\lambda = 0.01$



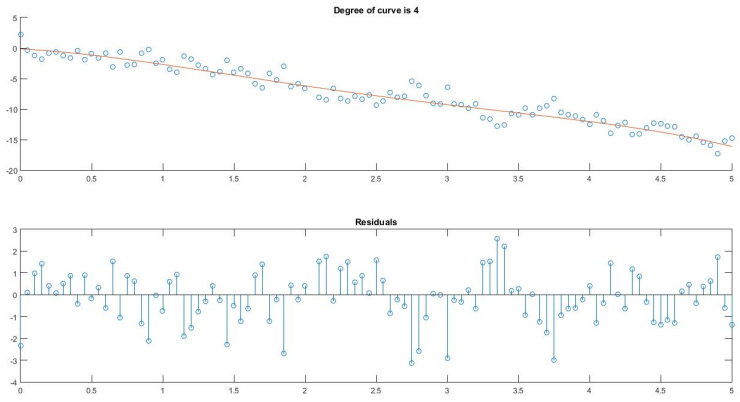
Using regression the equation to solve changes as,

$$\hat{\theta} = (A^T A - n\lambda I)^{-1} A^T y$$

On x_{test}, for $\lambda = 0.01$ we got $SSE = 15.26$ and $R^2 = 0.900$
On x_{test}, for $\lambda = 0.10$ we got $SSE = 16.65$ and $R^2 = 0.890$

Finally without ridge regression, $f(x) = -0.451 - 1.158x^1 - 1.247x^2 + 0.289x^3 - 0.022x^4$
This might be a best polynomial fit but from residues we can observe that there is some sinusoidal component left out. By considering sin, cos terms with appropriate frequency with be even better fit.

Figure 23: For $\lambda = 0.1$



2.2 2-dimensional data

2.2.1 Different order of polynomials and their best fit

We are given a dataset with 2 columns x_1, x_2 and their corresponding outcome is given in 3rd column as y.

$$x = \begin{bmatrix} 1 & x_1^1 & x_2^1 \\ \vdots & \vdots & \vdots \\ 1 & x_1^n & x_2^n \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

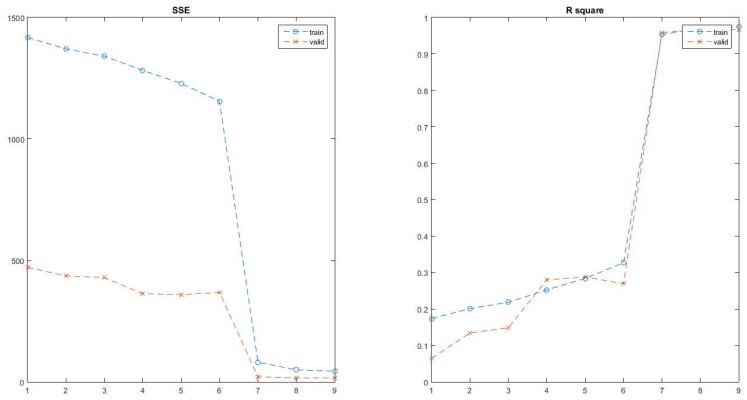
Now, we construct A from x_1 and x_2 with different combinations.

$$A = \begin{bmatrix} 1 & (x_1^1)^p (x_2^1)^0 & (x_1^1)^{p-1} (x_2^1)^1 & \cdots & \cdots & (x_1^1)^0 (x_2^1)^q \\ \vdots & \vdots & \vdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \cdots & \cdots & \cdots \\ 1 & (x_1^n)^p (x_2^n)^0 & (x_1^n)^{p-1} (x_2^n)^1 & \cdots & \cdots & (x_1^n)^0 (x_2^n)^q \end{bmatrix}$$

Above p, q are maximum degree for x_1 and x_2 respectively With method of least squares we try to estimate the value of $\hat{\theta}$ from $\hat{\theta} = (A^T A - n\lambda I)^{-1} A^T y$, where $\lambda = 0$ when ridge regression is not applied.

From SSE and R^2 parameters we try to find best surface plot for the data points.

Figure 24: SSE and R^2 error for different degree of the polynomial

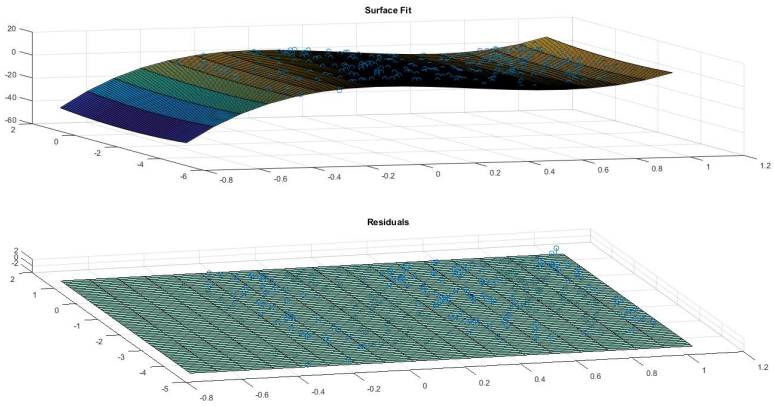


index	x degree	y degree
1	1	1
2	1	2
3	1	3
4	1	4
5	2	1
6	2	2
7	2	3
8	2	4
9	3	1
10	3	2
11	3	3
12	3	4
13	4	1
14	4	2
15	4	3
16	4	4

Models with index 10-12 performed well and SSE for training data increased for index 13, which maybe due to under fitting.

$$f(x_1, x_2) = 2.96 + 0.92x_2 + 0.16x_2^2 - 4.69x_1 - 0.33x_1x_2 - 0.06x_1x_2^2 - 43.33x_1^2 + 0.59x_1^2x_2 + 0.20x_1^2x_2^2 + 50.72x_1^3 + 0.21x_1^3$$

Figure 25: Surface Fit for given data points



2.2.2 Ridge Regression

Figure 26: For $\lambda = 0.1$

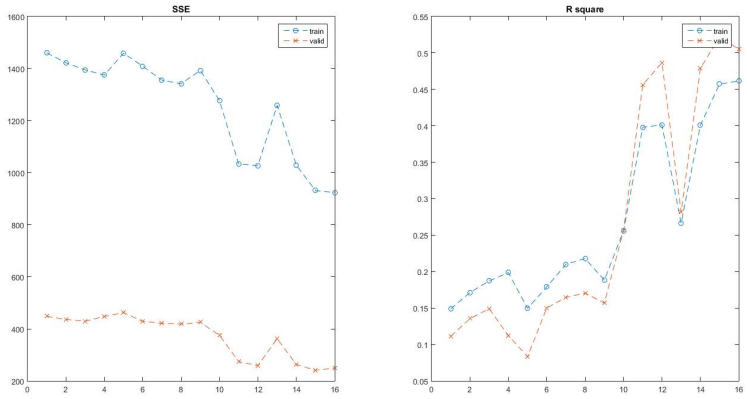
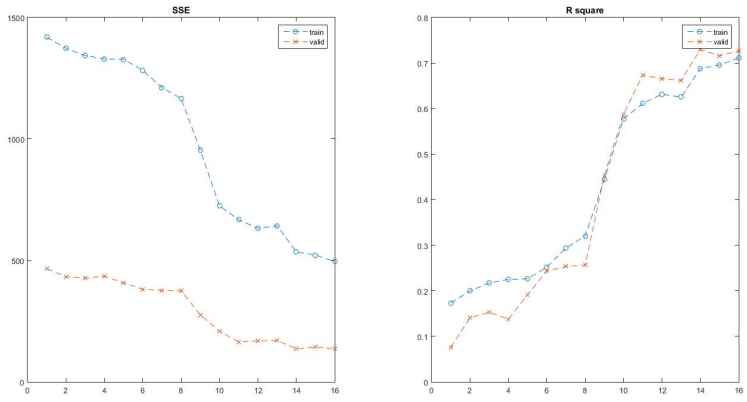


Figure 27: For $\lambda = 0.01$



index	x degree	y degree
1	1	1
2	1	2
3	1	3
4	1	4
5	2	1
6	2	2
7	2	3
8	2	4
9	3	1
10	3	2
11	3	3
12	3	4
13	4	1
14	4	2
15	4	3
16	4	4

By introducing ridge regression SSE and R^2 parameters got better for higher degree polynomials.

2.3 Multi dimensional data

By observing the given data we dropped off a repeating column,
$x = [1, x_1, x_2, \cdots, x_8], y = [y_1]$
We performed least square solution to the dataset and observed $SSE = 0$ and $R^2 = 1$.

$\hat{\theta} = [4.98e{-}16, 5.16e{-}17, -5.74e{-}17, -1.71e{-}16, -9.00e{-}16, -1.98e{-}15, 1.00]^T$

3 Reference

1. Introduction to Linear Algebra, Gilbert Strang
2. Pattern Recognition and Machine Learning, Christopher M. Bishop
3. http://home.iitk.ac.in/ crkrish/MLT/PreRequisites/linalgWithSVD.pdf
4. www.cs.utah.edu/ piyush/teaching/6-9-print.pdf