

# CS6690 Pattern Recognition

## Assignment-1 (SVD, EVD, Linear Regression)

**Group-29:** Arulkumar S (CS15S023), Shitanshu Kusmakar (ED15F003)

August 31, 2015

### 1 Singular Value Decomposition

Singular value decomposition (SVD) is the process of factorizing a matrix  $A$  ( $m \times n$ ) into three parts as shown below.

$$A = U \Sigma V^T$$

$U$  = unitary matrix of size  $m \times m$

$\Sigma$  = rectangular diagonal matrix of size  $m \times n$  (diagonal elements are singular values)

$V$  = unitary matrix of size  $n \times n$

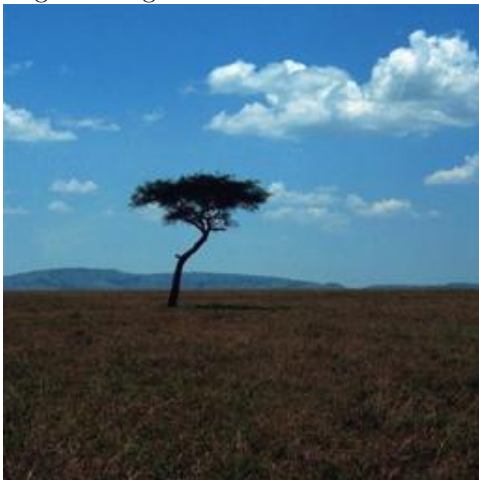
The main interest of this assignment is, to find out the least number of singular values needed to reconstruct the given image.

#### 1.1 Small image: 29.jpg (256 x 256 pixels)

The image is SVD factorized into 3 components using the inbuilt method from *python*'s linear algebra package.

The image is reconstructed using Top most  $N$  singular values, as well as random  $N$  singular values and the properties of reconstructed image are observed.

original image



##### 1.1.1 Observations on Top-N singular values

- \* The singular values are considered based on its magnitude to reconstruct the image. i.e., the higher magnitude component is considered first.

\* It is noticed that the image consists of multiple redundant singular value components.

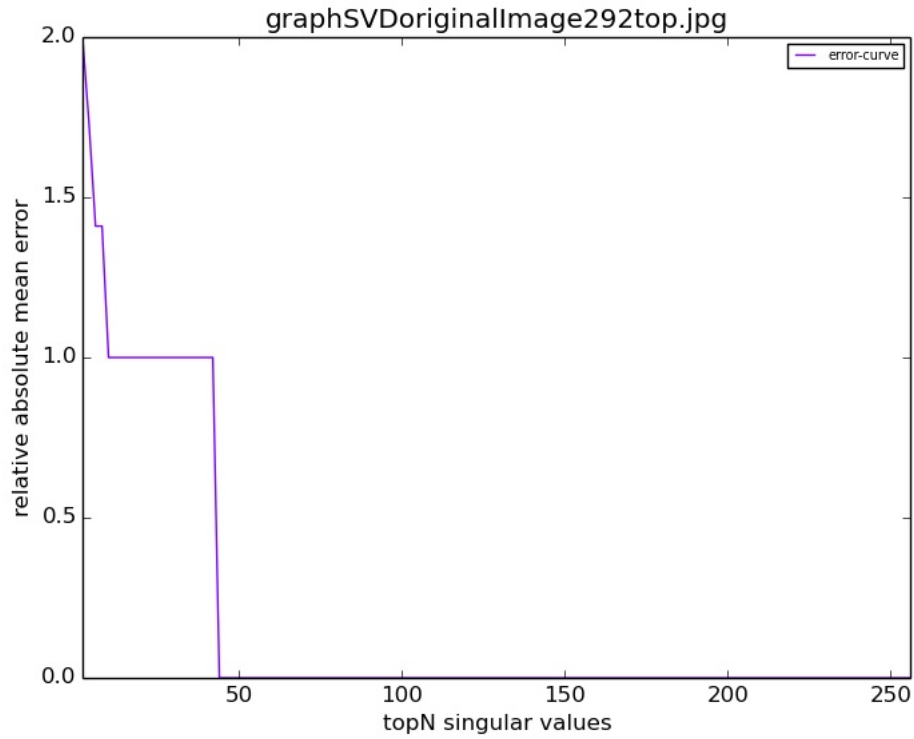
for example, the color channels (R, G, B) require only  $\tilde{50}$  singular value components out of 256 singular values to reconstruct the particular channel without any error.  
(only  $\tilde{25}$  components are needed to reconstruct the image without any visual appearance flaws)

\* It is interesting to see that it would be possible to compress the image once these high influential components are found, so that we can eliminate all the other redundant components.

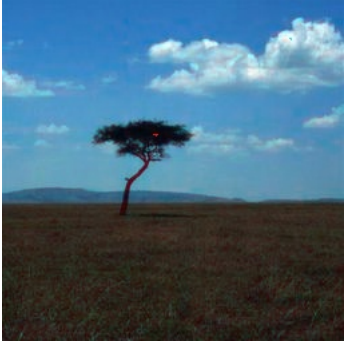
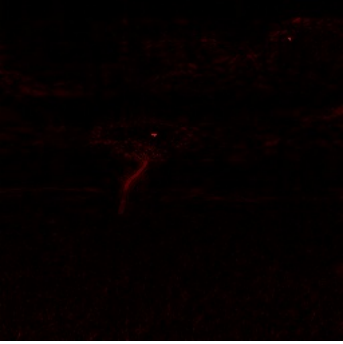







### 1.1.2 Red channel

The error reduction graph for red channel per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



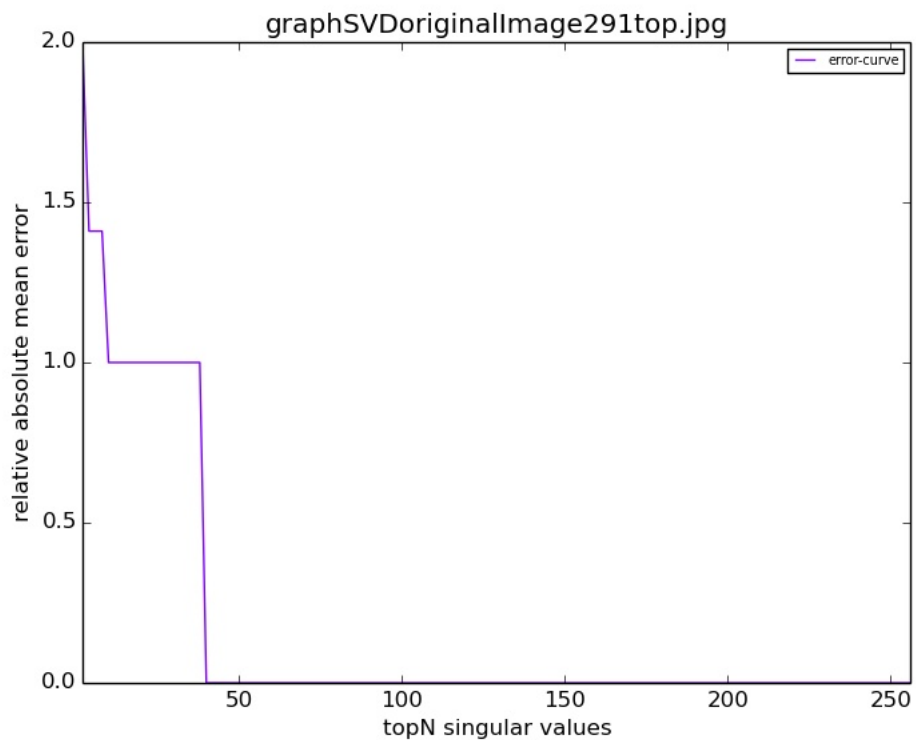
The various images that were constructed using top-N singular values are shown below

top most N singular values	reconstructed image	relative error image	relative absolute error
Top most 10			1.0
Top most 20			1.0
Top most 30			1.0
Top most 40			1.0
Top most 50			0.0


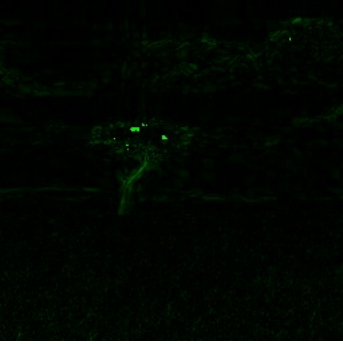


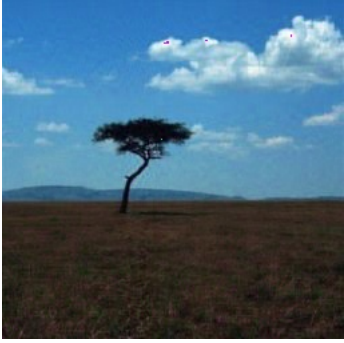

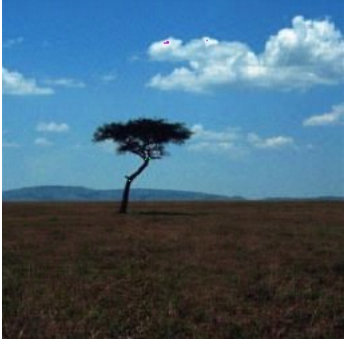
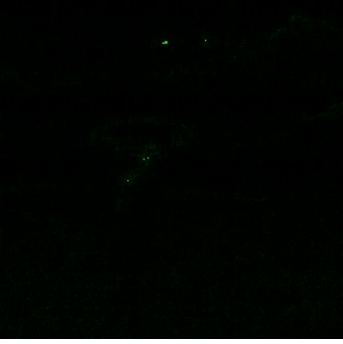

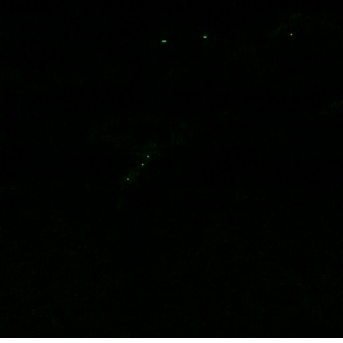
### 1.1.3 Green channel

The error reduction graph for Green channel per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



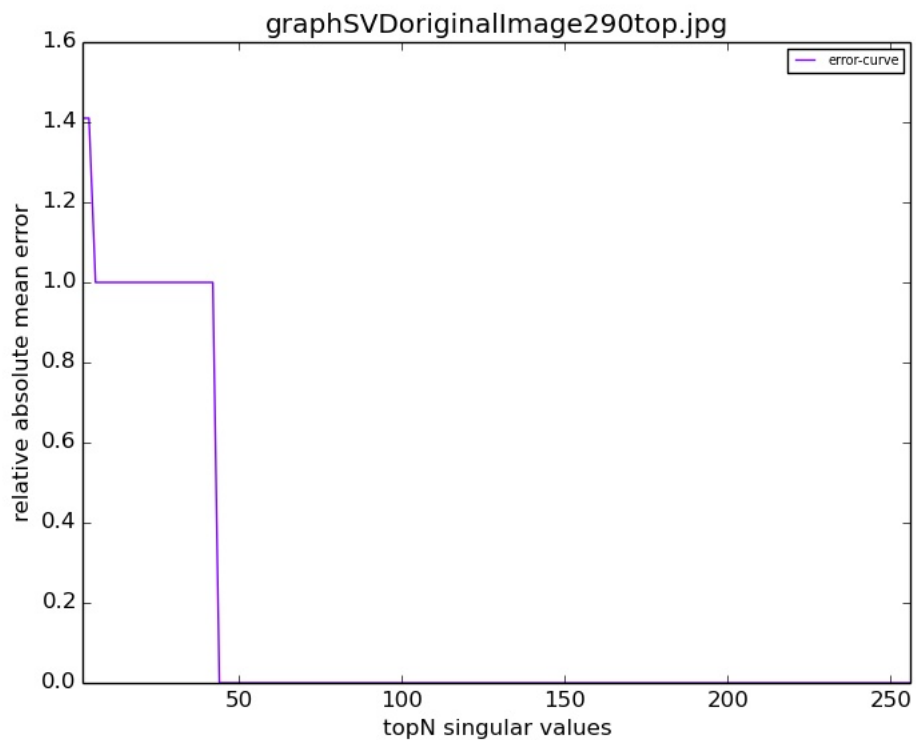
The various images that were constructed using top-N singular values are shown below

top most N singular values	reconstructed image	relative error image	relative absolute error
Top most 10			1.0
Top most 20			1.0
Top most 30			1.0
Top most 40			0.0
Top most 50			0.0




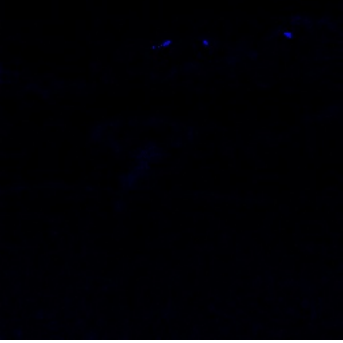
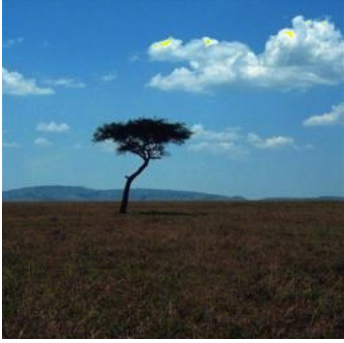





#### 1.1.4 Blue channel

The error reduction graph for Blue channel per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



The various images that were constructed using top-N singular values are shown below

top most N singular values	reconstructed image	relative error image	relative absolute error
Top most 10			1.0
Top most 20			1.0
Top most 30			1.0
Top most 40			0.0
Top most 50			0.0

### 1.1.5 Gray image








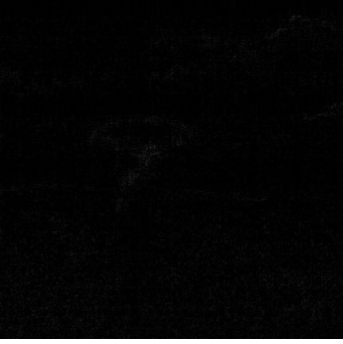

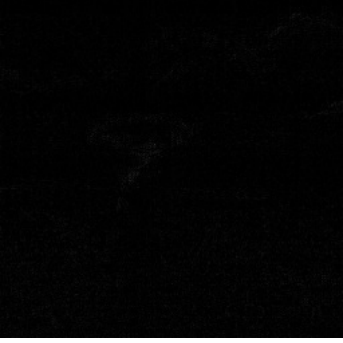
The error reduction graph for Gray image per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



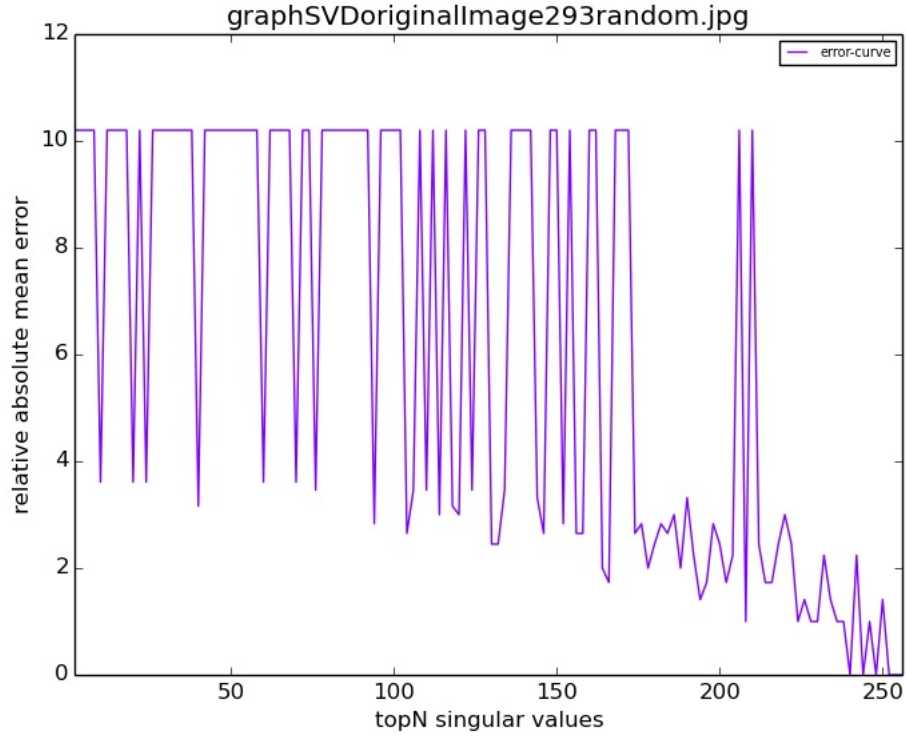
The various images that were constructed using top-N singular values are shown below


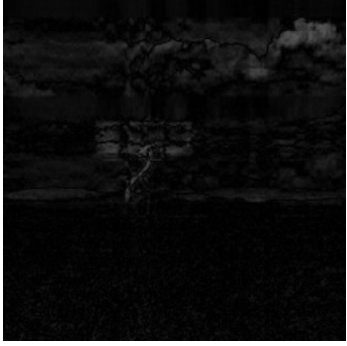

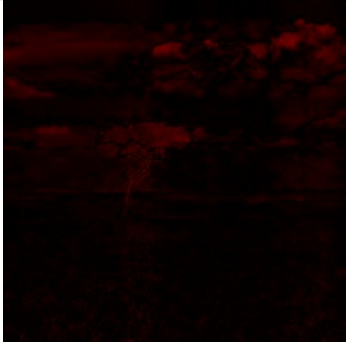

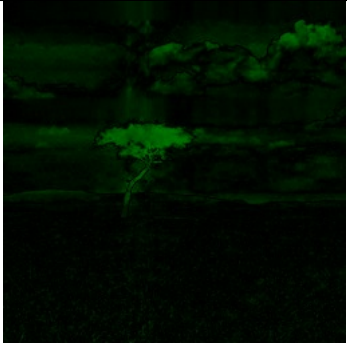



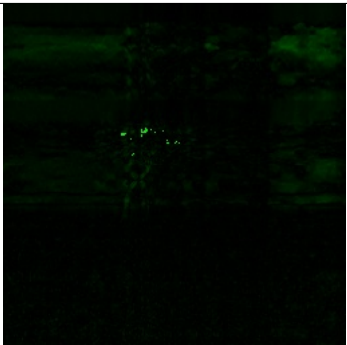


top most N singular values	reconstructed image	relative error image	relative absolute error
Top most 10			2.24
Top most 20			2.0
Top most 30			1.73
Top most 40			1.41
Top most 50			1.41

### 1.1.6 Observations on Random singular values

- \* In this analysis, The singular values are considered randomly without any order.
- \* It is noticed that adding high number of random components yields better results than less number of random components.
- \* The error reduction rate graph along with number of singular values included is shown in the pictures below. From the graph, we can see that the error is more when we include less number of random components. It is noticed that it is not always guaranteed that adding more number of random components reduce error.



random N singular values	reconstructed image	relative error image	relative absolute error
random 120 (Gray image)			3.0
random 80 (Red channel)			2.0
random 60 (Green channel)			2.0
random 100 (Blue channel)			1.0
random 190 (Green channel)			1.41

## 1.2 Big image: snakerock.jpg (3264 x 2448 pixels)

The image is SVD factorized into 3 components using the inbuilt method from *python*'s linear algebra package.

The image is reconstructed using Top most N singular values, as well as random N singular values and the properties of reconstructed image are observed.

original image



### 1.2.1 Observations on Top-N singular values

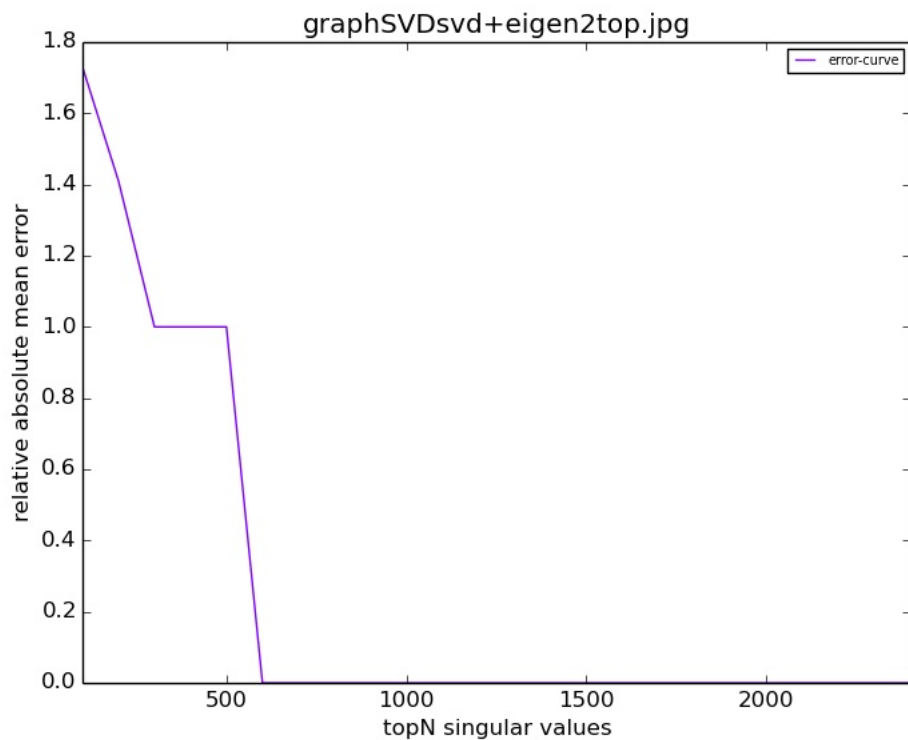
- \* The singular values are considered based on its magnitude to reconstruct the image. i.e., the higher magnitude component is considered first.
- \* There are considerably same amount of redundant components (2000 out of 2448) present in the Big image (as of smaller image (200 out of 256))
- \* visually, it is noticed that Top 100 - 200 singular value components are sufficient for recreating the picture
- \* The error reduction rate graph along with number of top most singular values included is shown in one of the pictures below. From the graph, we can see that the error reduction happens in multiple steps.
- \* There are some singular value components which influences the most and reduces the error by long margin. also, there are some components which keeps the error rate same after including them for reconstruction.

In this case, we could possibly remove such components to compress the image.

### 1.2.2 Red channel


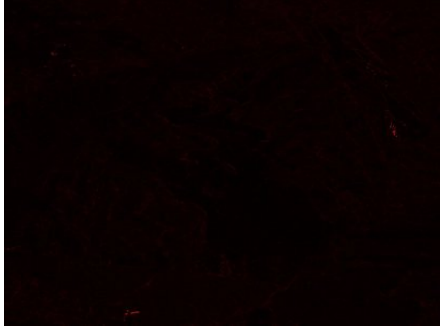







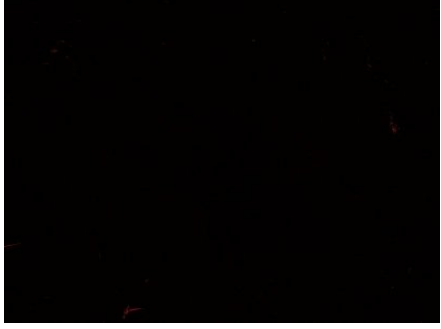
The error reduction graph for red channel per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



The various images that were constructed using top-N singular values are shown below

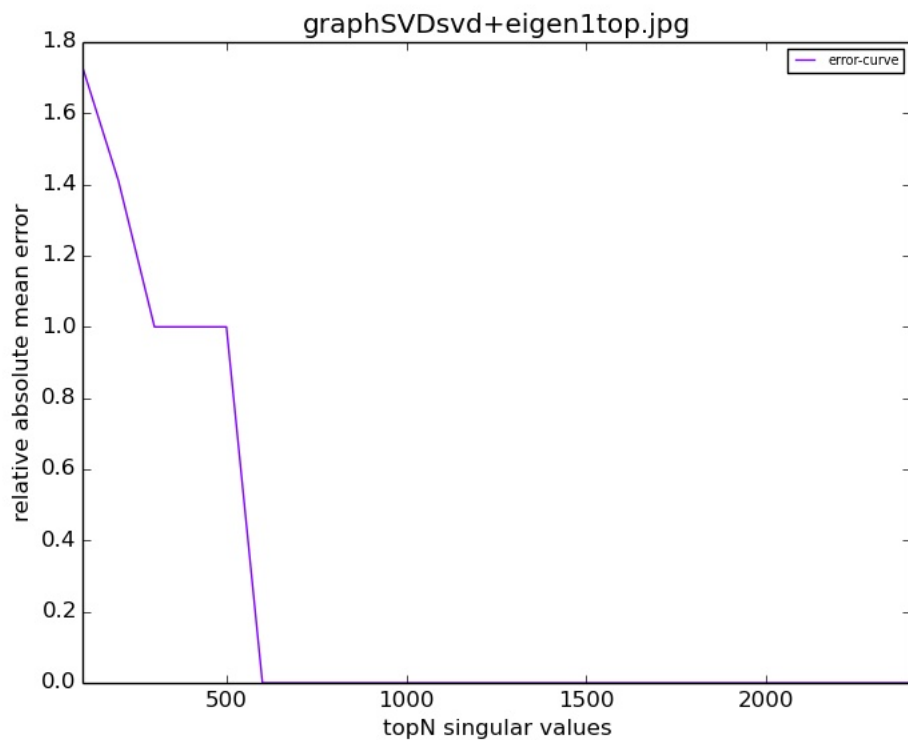


top most N singular values	reconstructed image	relative error image	relative al
Top most 100			1.73
Top most 200			1.41
Top most 300			1.0
Top most 400			1.0
Top most 500			1.0


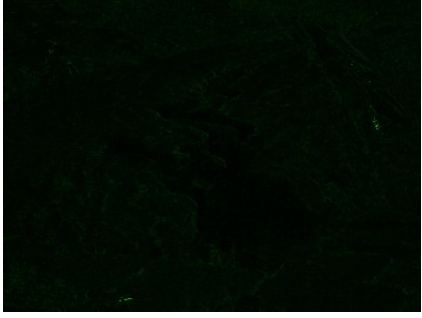

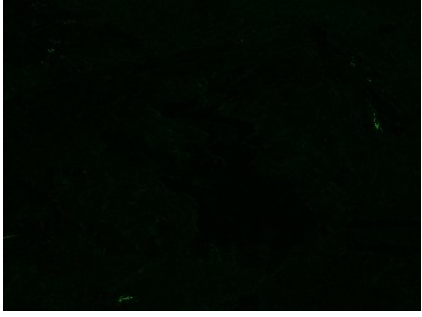






### 1.2.3 Green channel

The error reduction graph for Green channel per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



The various images that were constructed using top-N singular values are shown below

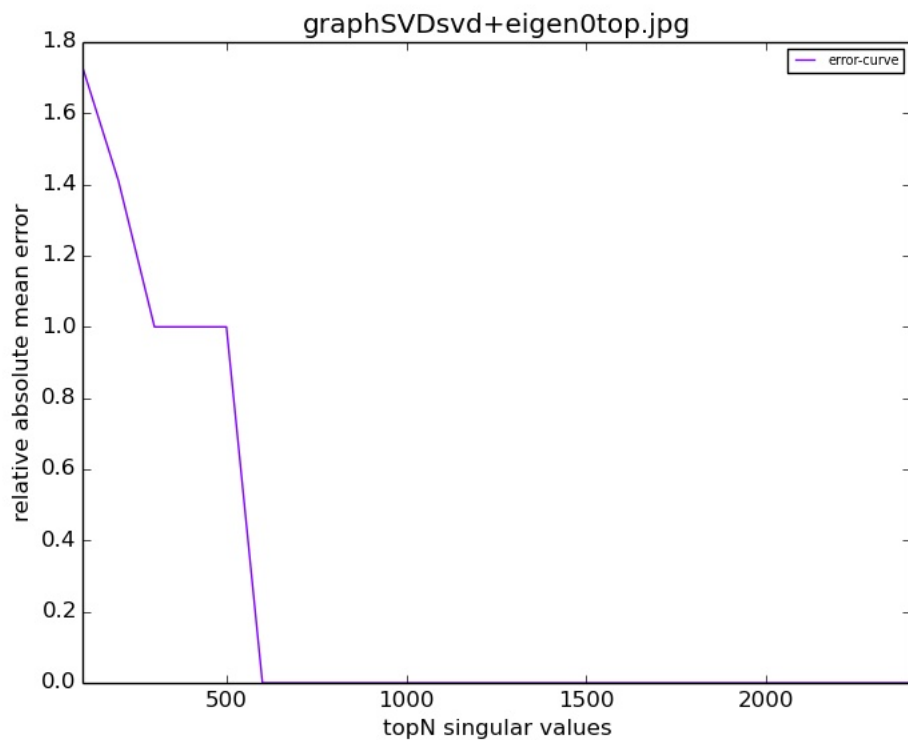
top most N singular values	reconstructed image	relative error image	relative absolute
Top most 100			1.73
Top most 200			1.4
Top most 300			1.0
Top most 400			1.0
Top most 500			1.0













### 1.2.4 Blue channel

The error reduction graph for Blue channel per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



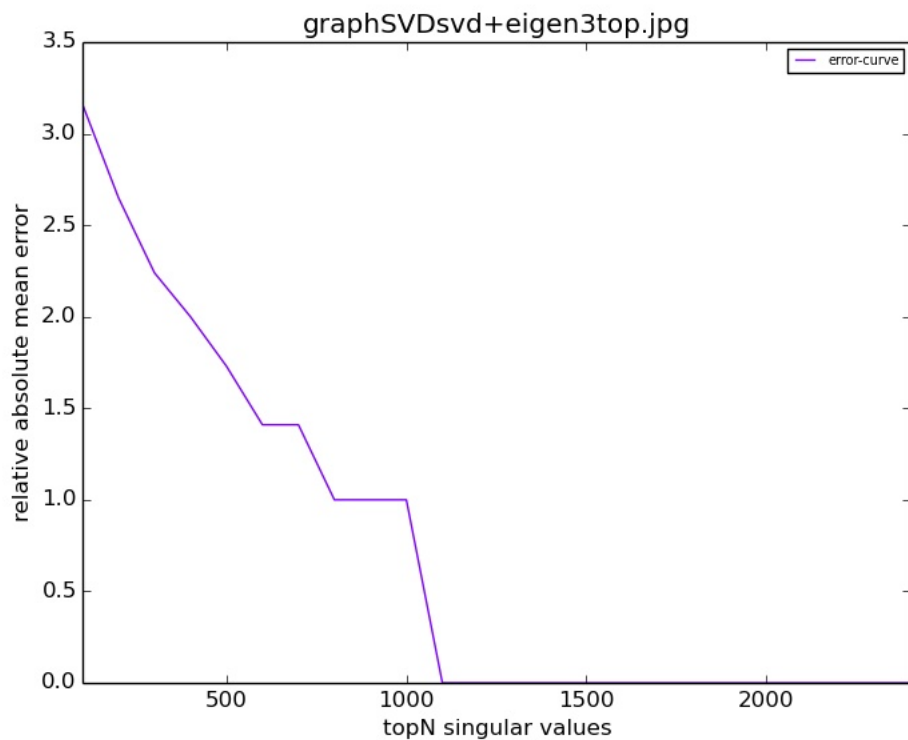
The various images that were constructed using top-N singular values are shown below

top most N singular values	reconstructed image	relative error image	relative absolute
Top most 100			1.73
Top most 200			1.4
Top most 300			1.0
Top most 400			1.0
Top most 500			1.0


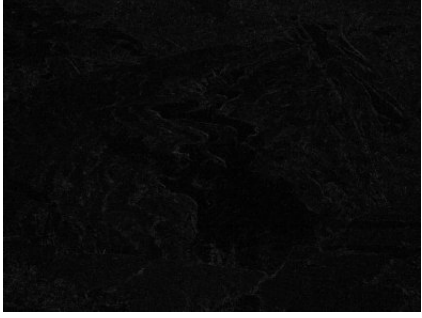

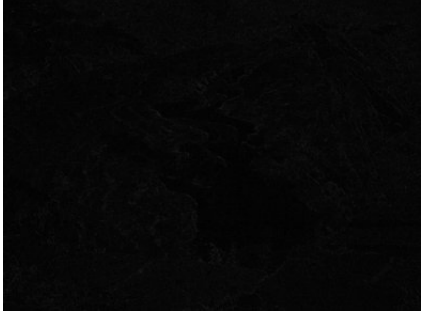

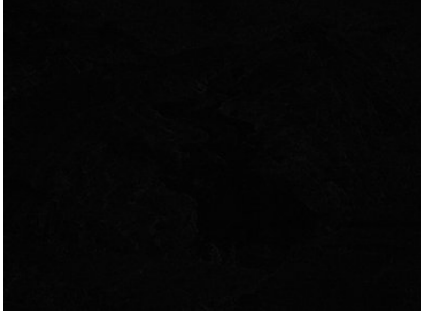

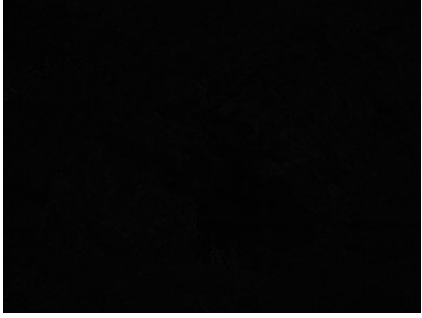

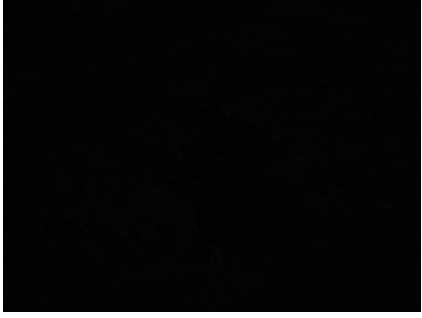
### 1.2.5 Gray image

The error reduction graph for Gray image per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



The various images that were constructed using top-N singular values are shown below

top most N singular values	reconstructed image	relative error image	relative absolute
Top most 100			3.16
Top most 200			2.24
Top most 300			2.0
Top most 400			1.73
Top most 500			1.41

## 2 Eigen Value Decomposition

### 2.1 Small image: 29.jpg (256 x 256 pixels)

Eigen value decomposition (or) Spectral decomposition is to factorize the image (matrix A) into 3 components as below.

$$A = Q\Upsilon Q^{-1}$$

Q = the square ( $NN$ ) matrix whose  $i^{th}$  column is the eigenvector  $q_i$  of A

$\Upsilon$  = the diagonal matrix whose diagonal elements are the corresponding eigenvalues of A

for this, the inbuilt method from *python*'s linear algebra package is used.

The image is reconstructed using Top most N eigen values and the properties of reconstructed images are observed.

original image



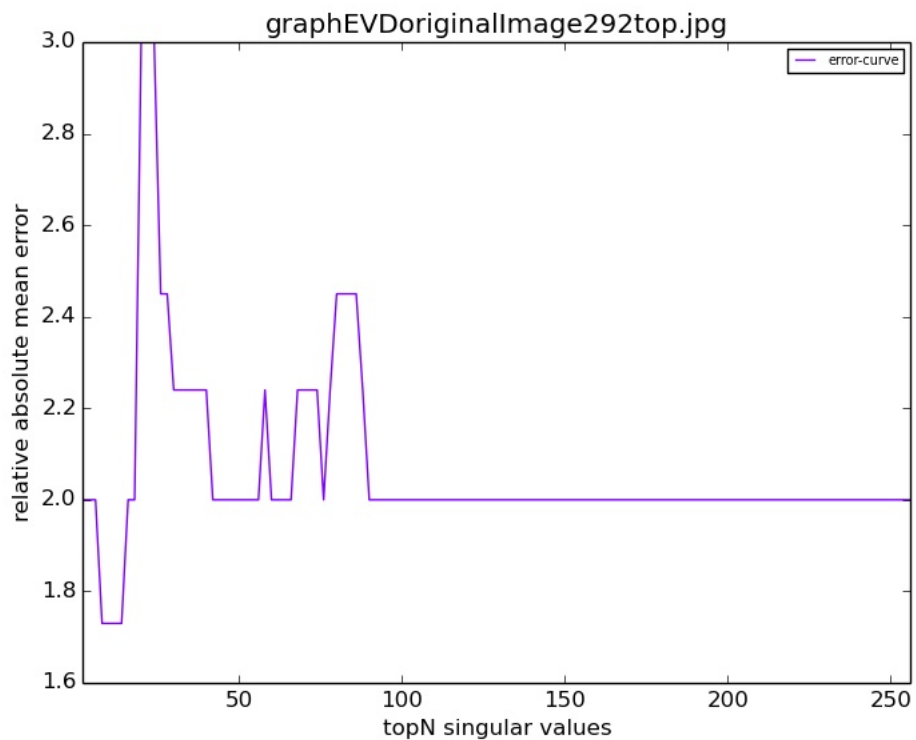
#### 2.1.1 Top-N eigen values

\* The eigen values are considered based on it's magnitude to reconstruct the image. i.e., the higher magnitude component is considered first.

### 2.1.2 Red channel


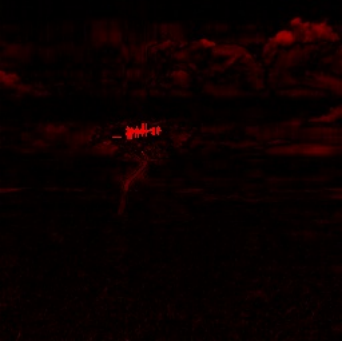

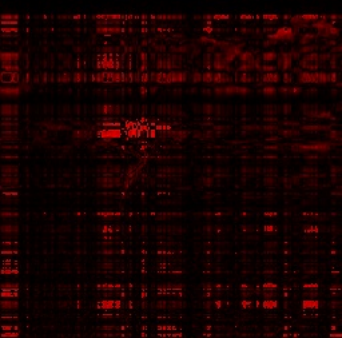

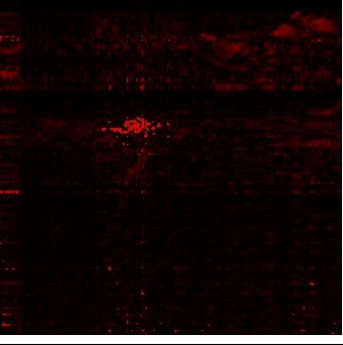
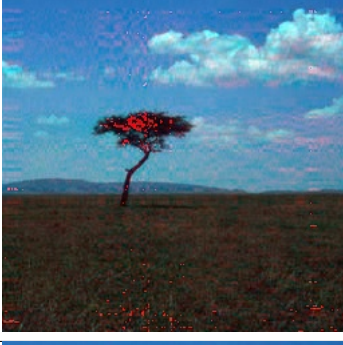
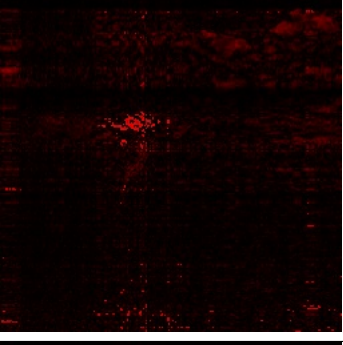
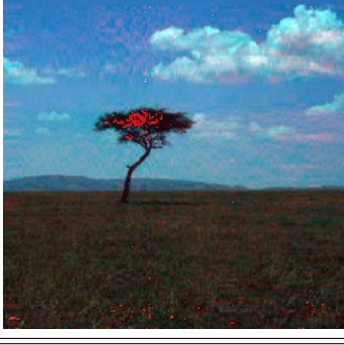
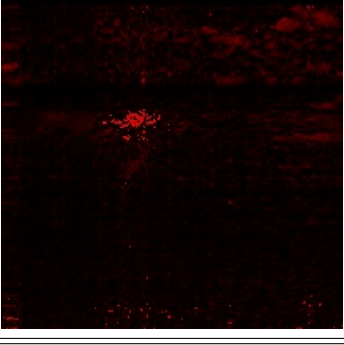
The error reduction graph for red channel per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



The various images that were constructed using top-N singular values are shown below

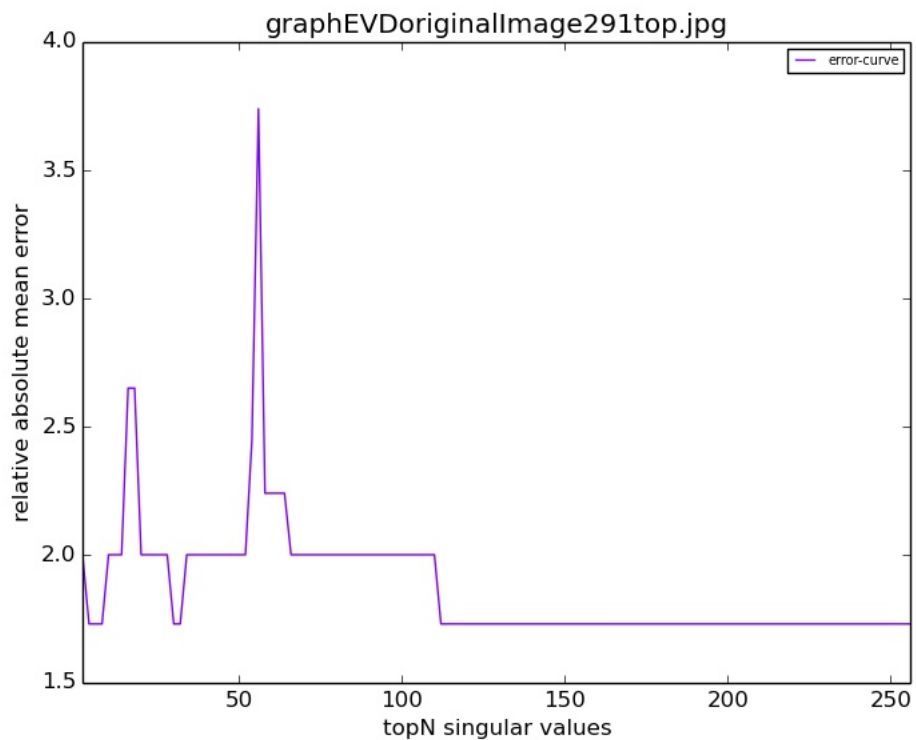


top most N eigen values	reconstructed image	relative error image	relative absolute error
Top most 10			1.73
Top most 20			3.0
Top most 30			2.24
Top most 40			2.24
Top most 50			2.0

### 2.1.3 Green channel


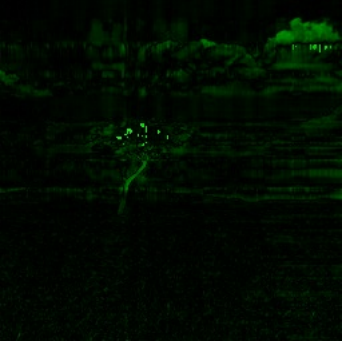

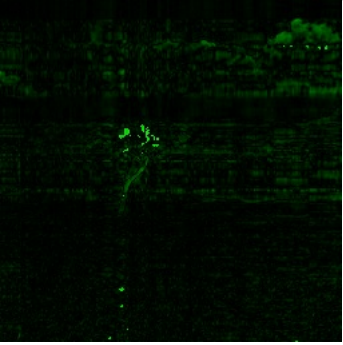



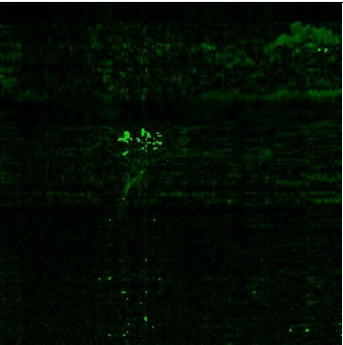

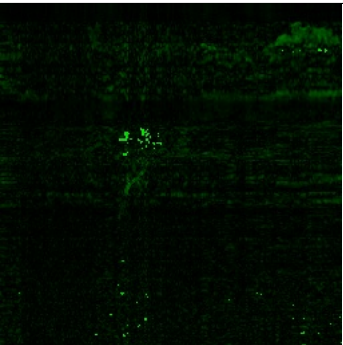
The error reduction graph for Green channel per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



The various images that were constructed using top-N singular values are shown below

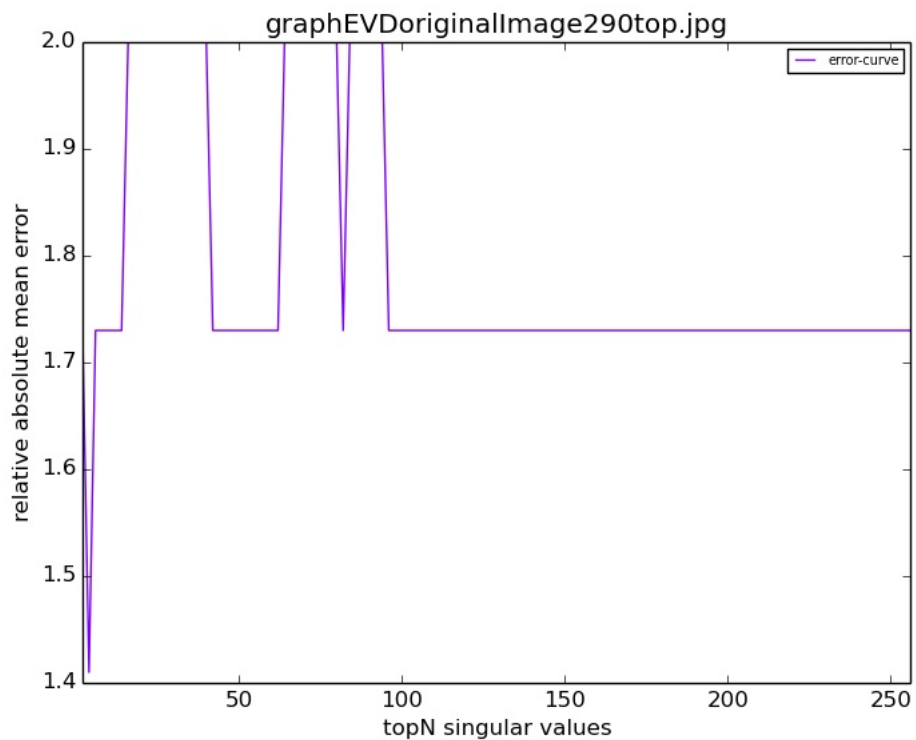


top most N eigen values	reconstructed image	relative error image	relative absolute error
Top most 10			2.0
Top most 20			2.0
Top most 30			1.73
Top most 40			2.0
Top most 50			2.0


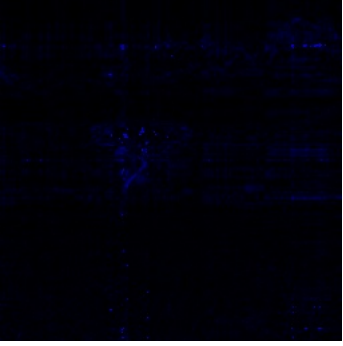

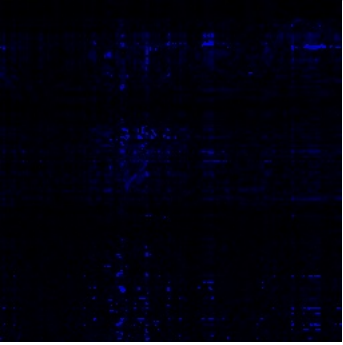

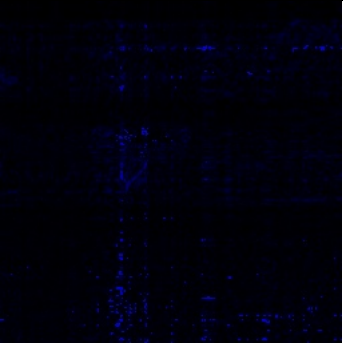

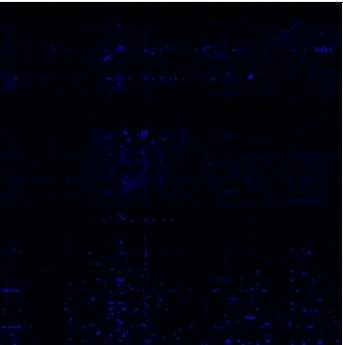

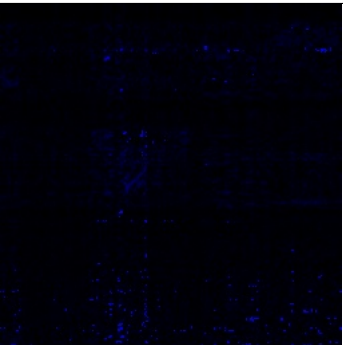
#### 2.1.4 Blue channel

The error reduction graph for Blue channel per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.



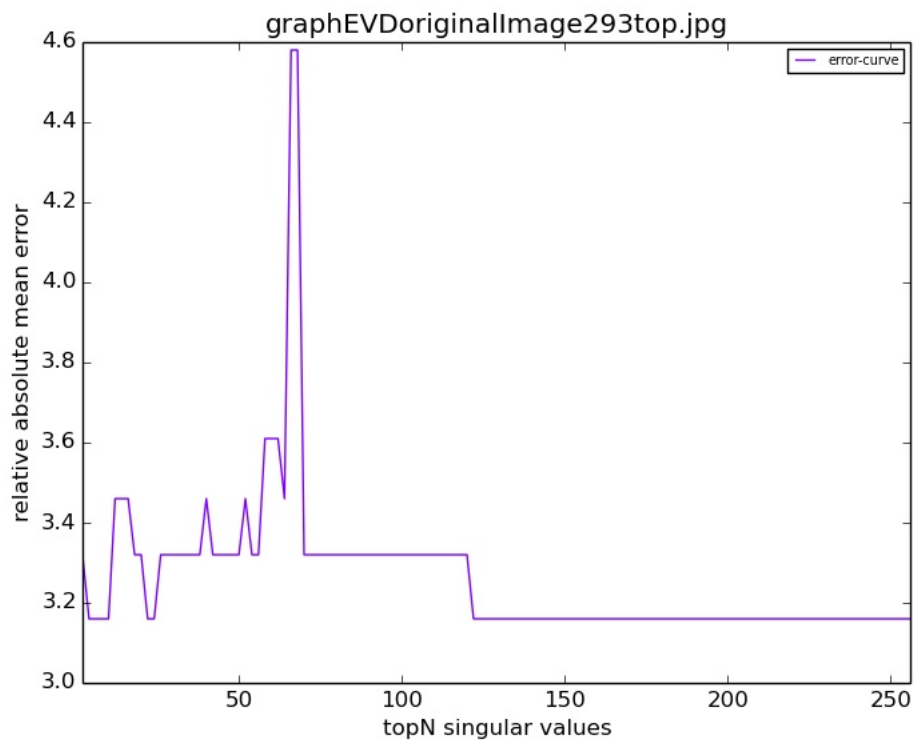
The various images that were constructed using top-N singular values are shown below

top most N eigen values	reconstructed image	relative error image	relative absolute error
Top most 10			1.73
Top most 20			2.0
Top most 30			2.0
Top most 40			1.73
Top most 50			1.73

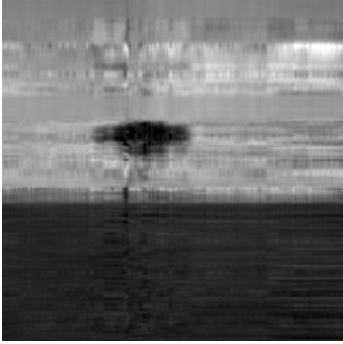


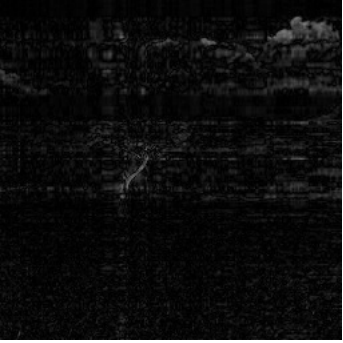



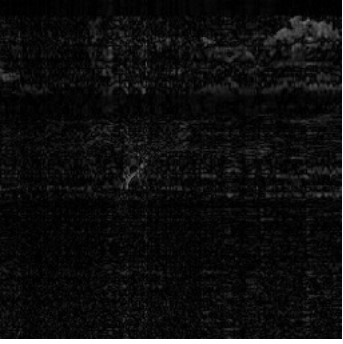

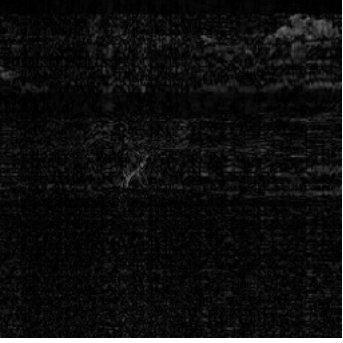
### 2.1.5 Gray image

The error reduction graph for Gray image per component inclusion is given below.

here, error corresponds to the Mean absolute error between reconstructed & original images.

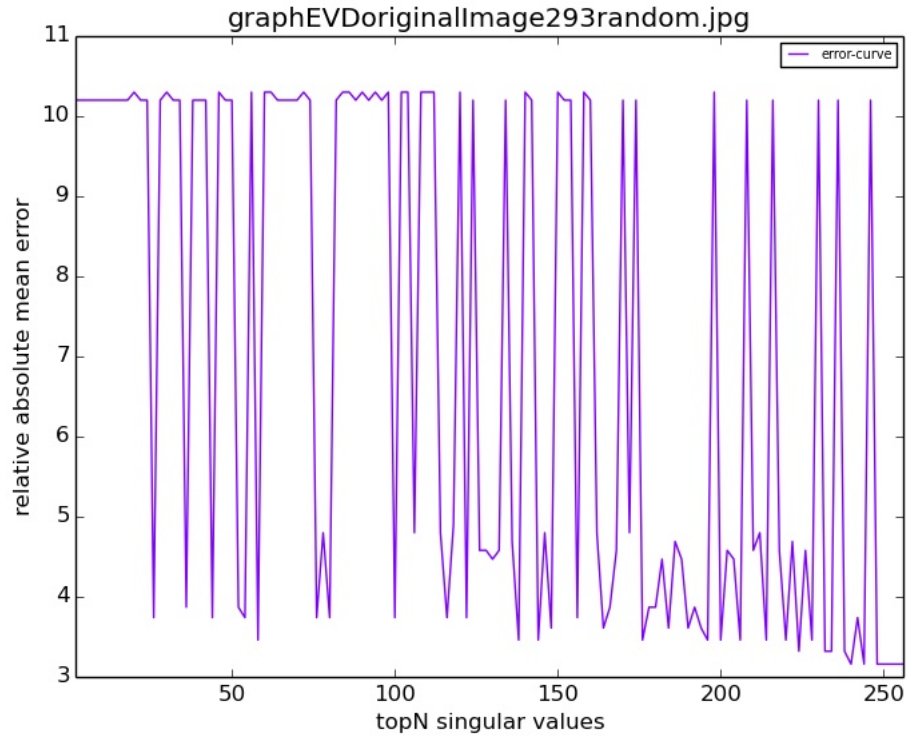


The various images that were constructed using top-N singular values are shown below

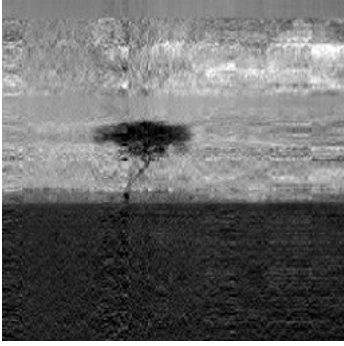
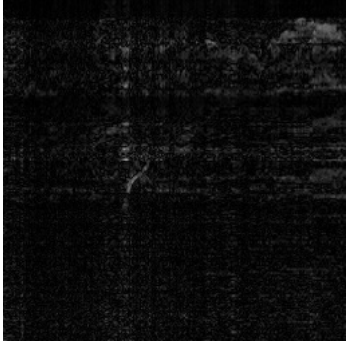

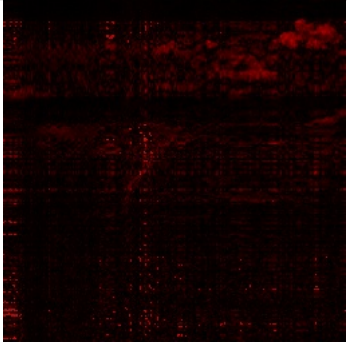

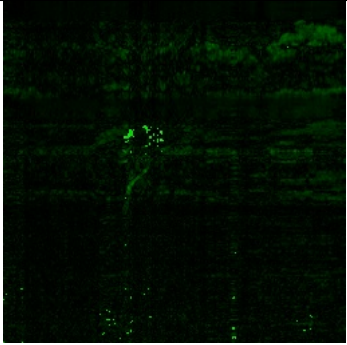

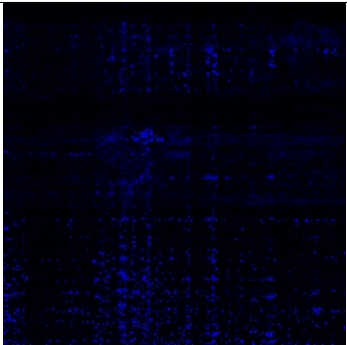
top most N eigen values	reconstructed image	relative error image	relative absolute error
Top most 10			3.46
Top most 20			3.16
Top most 30			3.32
Top most 40			3.32
Top most 50			3.46

### 2.1.6 Observations on Random Eigen values

- \* In this analysis, The Eigen values are considered randomly without any order.
- \* It is noticed that adding high number of random components yields better results than less number of random components.





random N eigen values	reconstructed image	relative error image	relative absolute error
random 190 (Gray image)			3.61
random 80 (Red channel)			2.45
random 240 (Green channel)			2.0
random 160 (Blue channel)			2.65

## 3 Linear Regression

### 3.1 Steps followed

- \* The given data set file (Data\_X\_r0\_sXX) is read into memory and split into 3 parts.
  1. Training set
  2. Validation set
  3. Test set
- \* For every data set file, I have tried to fit the model with polynomials upto order-8.

The models used are given below:

$$\text{Linear} = \text{bias} + X \quad (1)$$

$$\text{Quadratic} = \text{bias} + X + X^2 \quad (2)$$

$$\text{Cubic} = \text{bias} + X + X^2 + X^3 \quad (3)$$

$$\text{Degree} - 4 = \text{bias} + X + X^2 + X^3 + X^4 \quad (4)$$

$$\text{Degree} - 5 = \text{bias} + X + X^2 + X^3 + X^4 + X^5 \quad (5)$$

$$\text{Degree} - 6 = \text{bias} + X + X^2 + X^3 + X^4 + X^5 + X^6 \quad (6)$$

$$\text{Degree} - 7 = \text{bias} + X + X^2 + X^3 + X^4 + X^5 + X^6 + X^7 \quad (7)$$

$$\text{Degree} - 8 = \text{bias} + X + X^2 + X^3 + X^4 + X^5 + X^6 + X^7 + X^8 \quad (8)$$

$$\text{Sinusoid} = \text{bias} + X + X^2 + \sin(X) \quad (9)$$

- \* Each feature in the data set is normalized (scaled) using the formula  $X_{\text{norm}-i} = \frac{(X_i - \mu_i)}{\sigma_i}$
- \* The bias term (1) is added to all the examples of the scaled data set
- \* For the purpose of optimization of parameters ( $\omega_i$ ), there are two methods experimented in this assignment.

#### 1. Gradient descent method:

Initially, All the parameters ( $\omega_i$ ) are considered as 0.

$$\text{Cost function } (X, Y, \omega_i) = \frac{1}{(2 * \text{total\_examples})} (\sum_{i=1}^{\text{total\_examples}} (X_i \omega_i - Y_i)^2 + \lambda * \sum_{j=1}^{\text{total\_features}} \omega_j^2)$$

$\lambda$  = Regularization constant parameter = 5

On each iteration of the gradient descent, the parameters are adjusted, so that the Mean Squared Error (MSE) of the training set is minimized.

$$\omega_i := \omega_i - \frac{\alpha}{(\text{total\_examples})} (\sum_{i=1}^{\text{total\_examples}} ((X_i \omega_i - Y_i) * X_i) + \lambda * \sum_{j=1}^{\text{total\_features}} \omega_j)$$

$$\alpha = \text{Learning factor (coefficient)} = 0.05$$

#### 2. Linear Algebra method:

The given pattern recognition problem belongs to an overdetermined system. hence, to find the solution for the parameters, we can make use of *Moore – Penrose* Inverse.

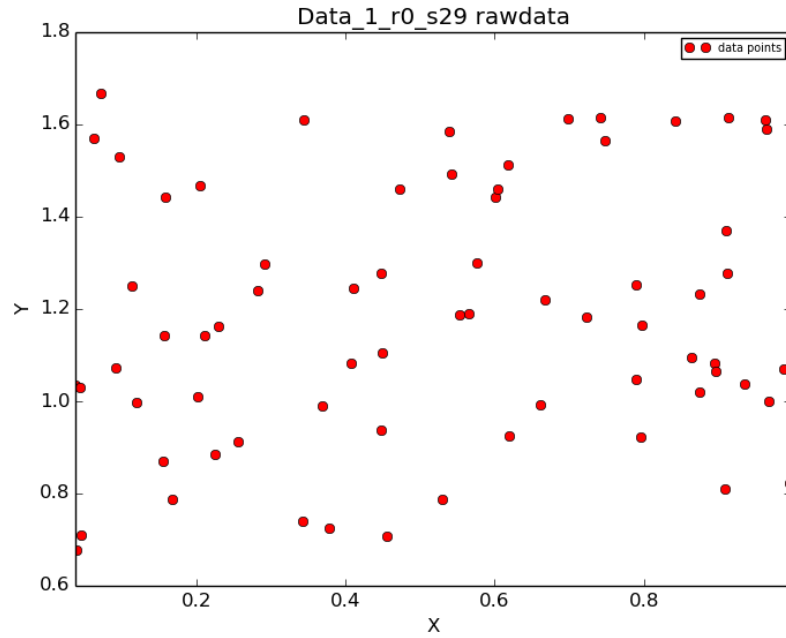
$$\text{Parameters } (\omega_i) = (X^T X)^{-1} X^T y$$

- \* After the models has been trained, All the models are compared using their performance in Cross-Validation set & the model which is performing the best will be chosen to predict the test data set results.



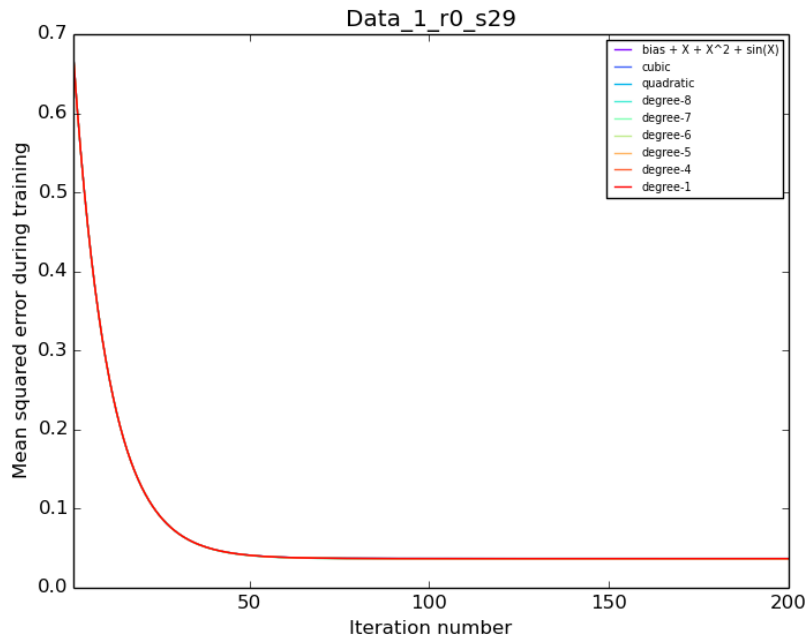
### 3.2 dataset: Data\_1\_r0\_s29

The plot of given dataset looks as below



The models mentioned in the previous section are trained one-by-one and the relevant feature weights (parameters) are obtained.

During the gradient descent process, the rate of convergence for various models is shown in the picture below:

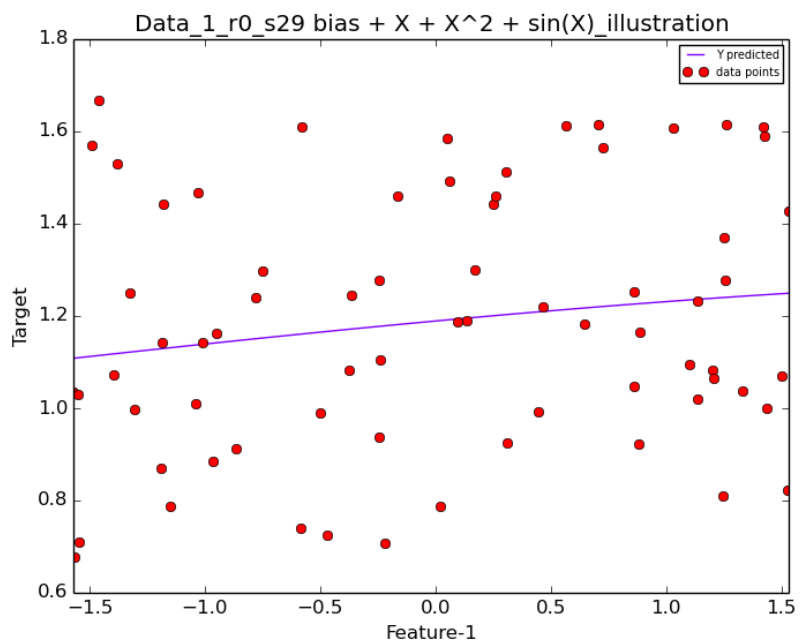


$X^2 + \sin(X)$  gives the parameters as,

$$\omega = [1.18525045, 0.02035862, -0.0044357, 0.02984357]^T$$

The model  $bias + X +$

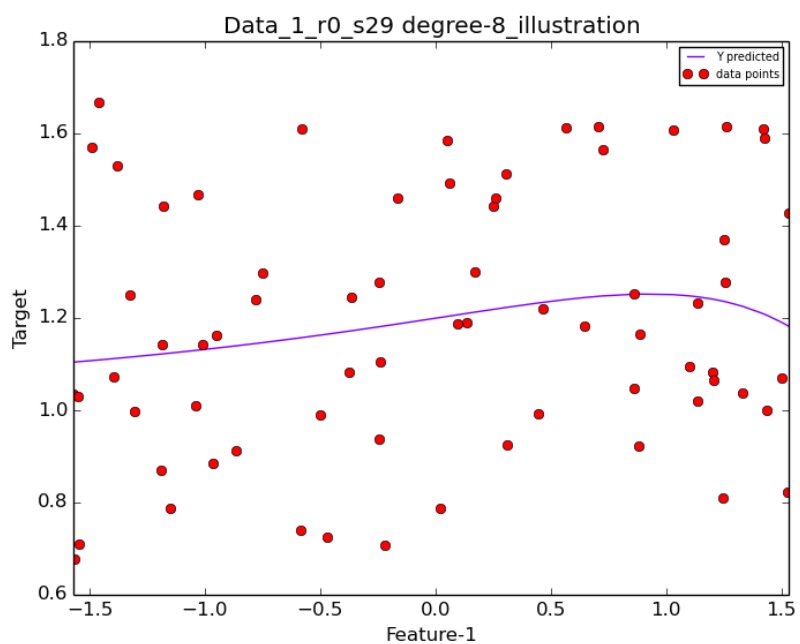
The corresponding plot will be,



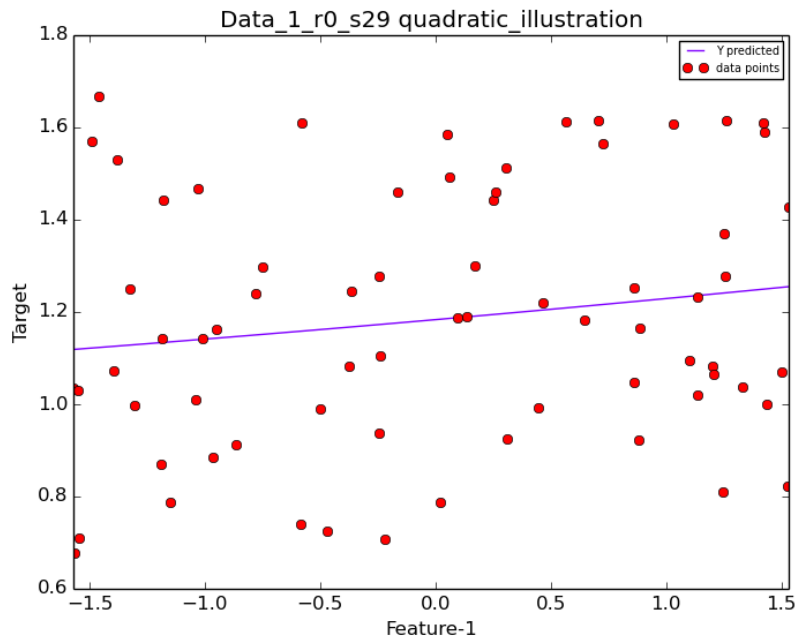
Similar to the above model, all the other models are trained for the given training data.

some of the plots are given below:

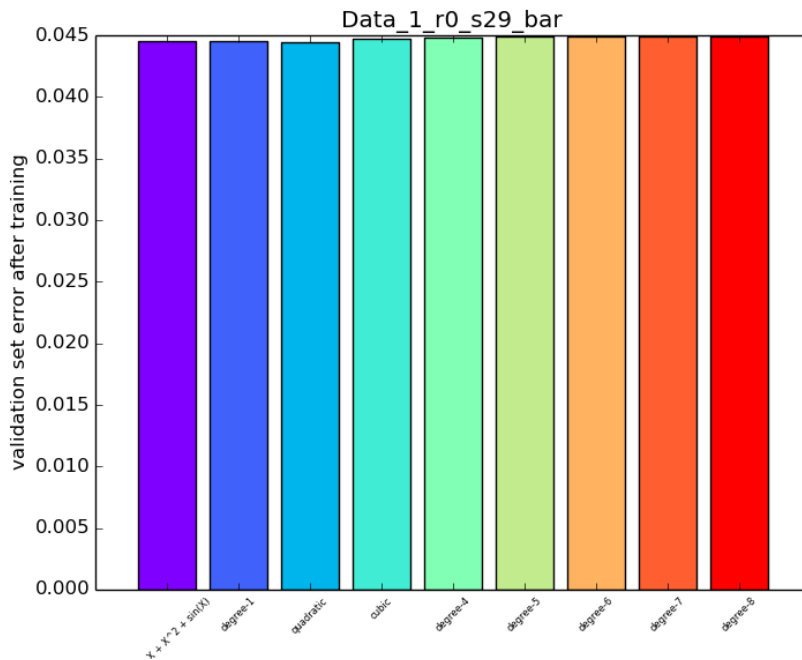
$$\text{bias} + X + X^2 + X^3 + X^4 + X^5 + X^6 + X^7 + X^8$$



$$\text{Quadratic} = \text{bias} + X + X^2$$



After the training of all the models, the performance is compared using Cross validation data. below is the comparison plot for the models:



As we can see in the bar chart, the hypothesis models

- \*  $\text{bias} + X + X^2 + \sin(X)$

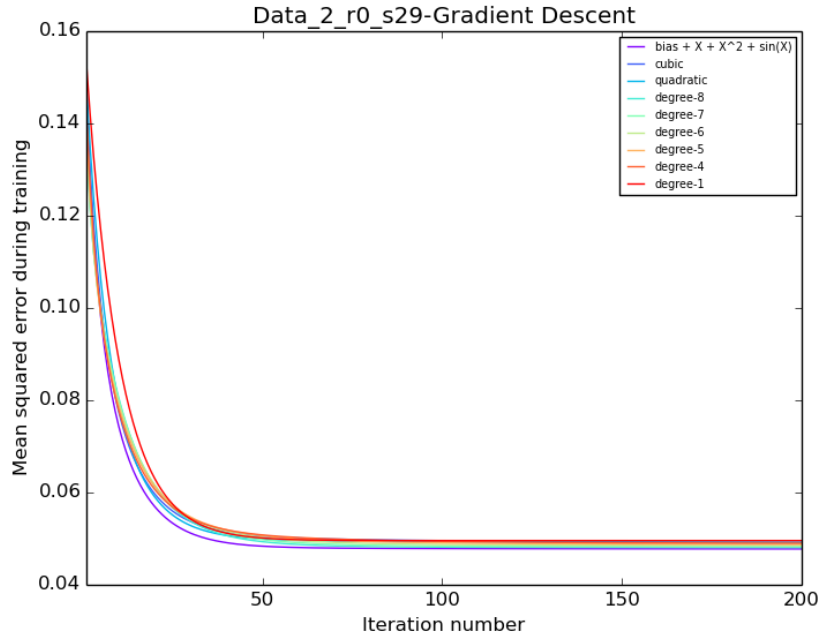
- \*  $\text{bias} + X$

- \*  $\text{bias} + X + X^2$

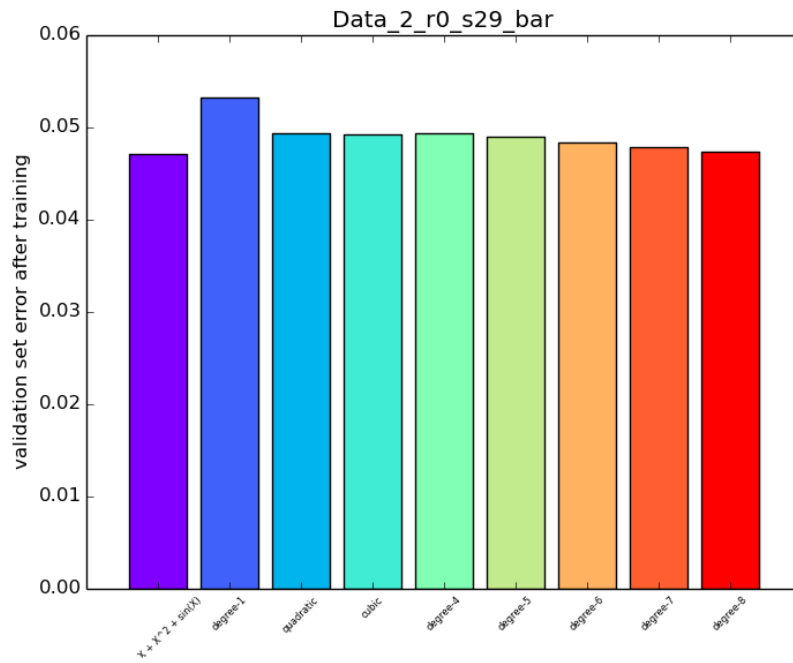
are performing better than other models.

### 3.3 dataset: Data\_2\_r0\_s29

The convergence of various models belonging to this dataset is show below

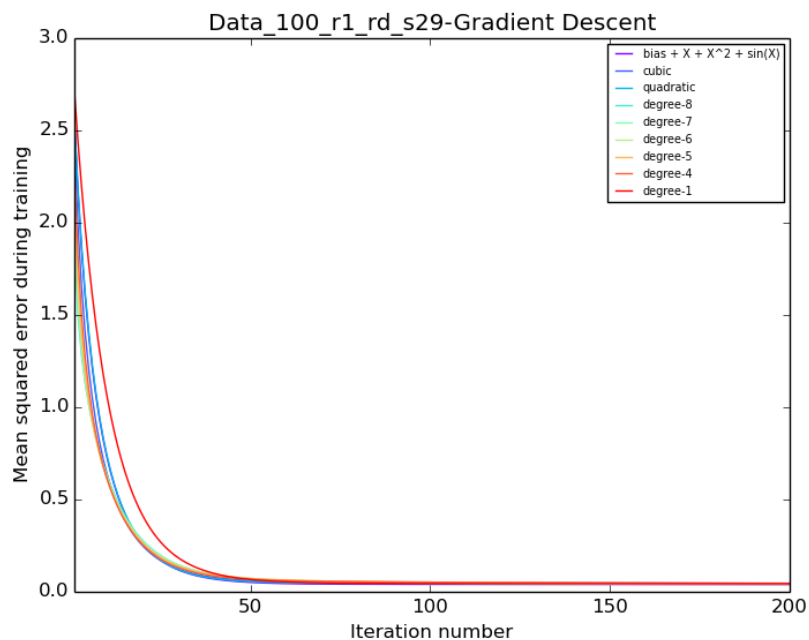


According to the performance in the cross-validation set, the model  $\text{bias} + X + X^2 + \sin(X)$  is selected to be the best fit for this data set.



### 3.4 dataset: Data\_100\_r1\_rd\_s29

The convergence of various models belonging to this dataset is show below



According to the performance in the cross-validation set, the quadratic model ( $bias + X + X^2$ ) is selected to be the best fit for this data set.

