CS5691: Pattern Recognition and Machine Learning

Programming Assignment # 1

Sourav Sahoo, EE17B040 and G Abhilash, EE17B007

March 14, 2020

Contents

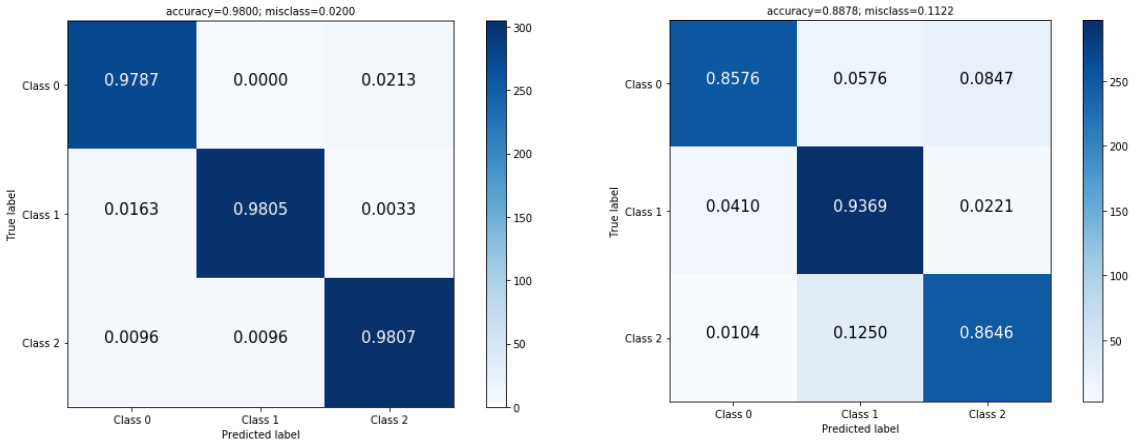
List of Figures

List of Tables

Table 1: Train and test accuracy of different models on datasets 1 and 2

	Dataset 1		**Dataset 2**	
	Train	**Test**	**Train**	**Test**
Model 1	0.9625	0.9778	0.8450	0.8278
Model 2	0.9628	0.9778	0.8568	0.8400
Model 3	0.9628	0.9778	0.8811	0.8622
Model 4	0.9631	0.9778	0.8556	0.8378
Model 5	**0.9631**	**0.9800**	**0.8983**	**0.8878**

where the values are in fraction out of 1 and the values in bold represent the model with best accuracy.



(a) Confusion matrix for dataset 1

(b) Confusion matrix for dataset 2

Figure 1: Confusion Matrix on the test datasets 1 and 2 for the best model

1 Bayesian Classifier

In this problem, we develop a Bayesian Classifier for the given datasets. As a data preprocessing step, we normalize all the datapoints by subtracting the empirical mean and dividing by the standard deviation. This is done so that in the later steps, we can avoid memory overflows and machine precision errors while calculating exponentials of numbers in order of 100s. *It is to be noted that the shape of the decision boundaries remain unaffected after data normalization.*

1.1 Classification Accuracy and Confusion Matrix

We first consider a Naive Bayes Classifier, where we assume that each of the features of input x are *independent and equally important*. We compute the individual empirical means μ_1 and μ_2 of x_1 and x_2 respectively where $x = [x_1, x_2]^\top$. The covariance matrix of class i is $\Sigma_i = \begin{pmatrix} \sigma_{i1}^2 & 0 \\ 0 & \sigma_{i2}^2 \end{pmatrix}$ where σ_1^2 and σ_2^2 are the variances of the individual features. When we assume that the covariance matrices are same for all the classes, the common covariance matrix is obtained by optimizing the Equation 1.

$$\min_{\Sigma} \sum_{i=1}^{3} ||\Sigma - \Sigma_i||_F^2 \tag{1}$$

where $|.|_F$ is the Frobenius norm. The solution to above is given by the *mean of covariance matrix* of the three classes. The dataset is split in 80:20 as training and test sets and the accuracy results are given for both the datasets 1 and 2 is given in Table 1. The confusion matrix for the best model in both the cases is given in Figure 1.

1.2 Decision Boundary and Contour Curves

We then plot the decision surface for the three classes for dataset 1 in Figure 2 and for dataset 2 in Figure 3. We superimpose the training data on the surface to observe the its actual distribution. We further plot the contour curves of the Gaussian distribution from which the data points came from along with corresponding eigen vectors of the covariance matrices. The plots are shown for dataset 1 in Figure 4 and for dataset 2 in Figure 5. It is to be noted that the the x-axis and y-axis are in normalized coordinates for better visualization of the plots.

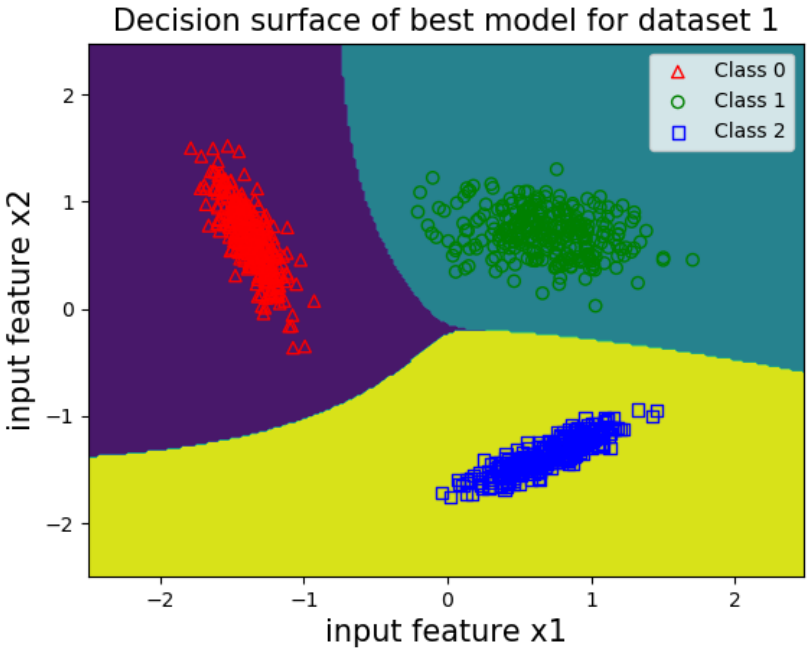


Figure 2: Decision surface and boundary for dataset 1

1.3 Inferences

From the plots and Table 1, we infer several points:

- From the table, we can see that model accuracy in case of dataset 2 is much less as compared to dataset 1, which is also evident from Figure 2 and 3. In the first case the classes are linearly separable but not so in the second case.
- Model 5 achieves the best accuracy out of all of the models. The reason behind it is that we did *not impose artificial constraints* that the covariance matrices of the classes to be same or that features are independent and equally important as in the case of Naive Bayes Classifier.
- It can be also seen in Table 1 that the *differences in the model accuracies is significantly evident only in dataset 2*, where the data points are not linearly separable.

2 Bayesian Estimation

In this problem, we do a Bayesian estimation of the mean of a one-dimensional Gaussian dataset. We assume the prior of the mean as $\mathbf{P}(\mu) = \mathcal{N}(\mu_0, \sigma_0^2)$. The Bayesian estimation of μ is given in Equation 2.

$$p(\mu|\mathcal{D}) \propto \prod_{i=1}^{n} p(x_i|\mu)p(\mu) \tag{2}$$

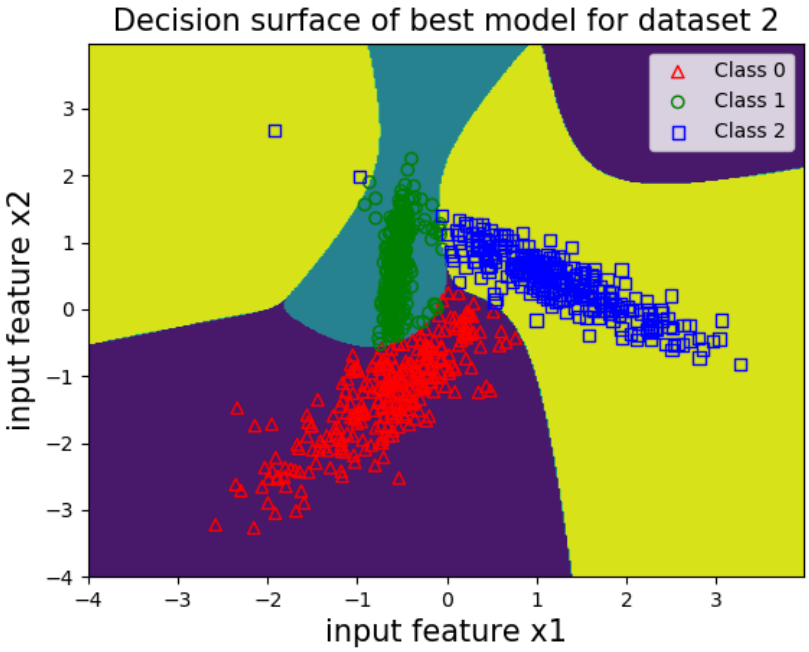


Figure 3: Decision surface and boundary for dataset 2

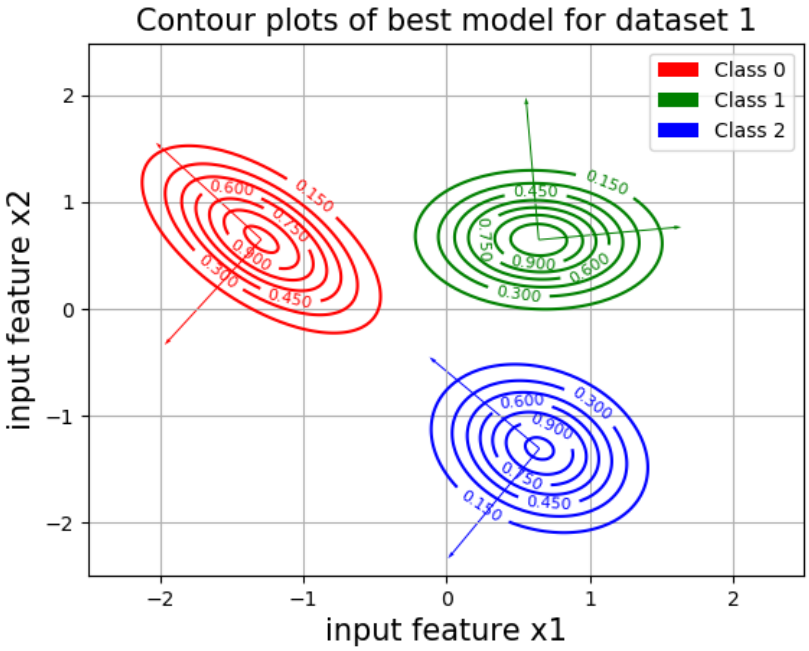


Figure 4: Contour Plots and eigen vectors of covariance matrix for dataset 1

As both the prior and likelihood as Gaussian distribution, the posterior distribution is also a Gaussian distribution given be $\mathcal{N}(\mu_n, \sigma_n^2)$. The values of μ_n and σ_n^2 can be analytically calculated and are given in Equation 3 and 4.

$$\mu_n = \frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2}\bar{x}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2}\mu_0 \tag{3}$$

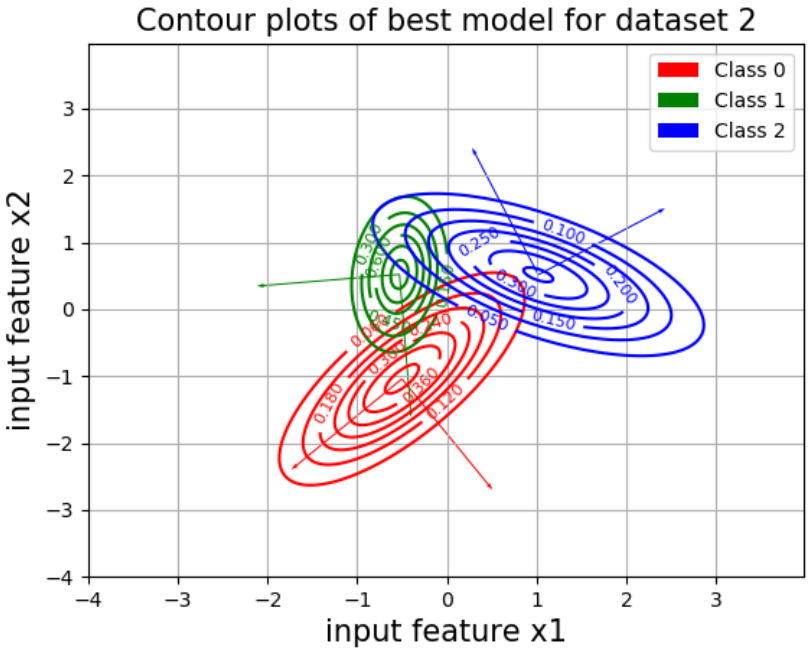


Figure 5: Contour Plots and eigen vectors of covariance matrix for dataset 2

$$\sigma_n^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2} \tag{4}$$

2.1 Estimating σ

It is to be noted that the value of σ is not explicitly given, so we need to calculate its value. We first compute μ_n for the given rates i.e. σ^2/σ_0^2 and then use maximum likelihood estimation to estimate the value of σ. As the given $\sigma^2/\sigma_0^2 \ll N$, where N is the number of data points, $\mu_n \approx \bar{x}_n$. Using the calculated value of μ_n, we get $\boxed{\sigma = 1.976}$

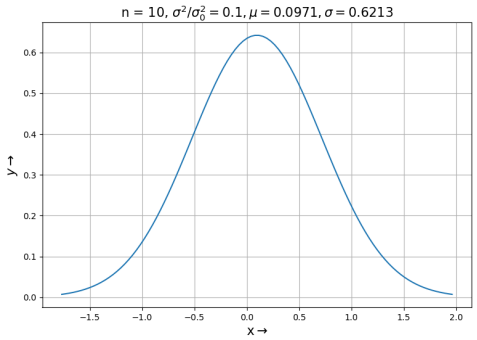
2.2 Calculating Posterior Probability Density

Once the value of σ is estimated, we compute the posterior probability density $p(\mu|\mathcal{D})$ for $N = [10, 100, 1000]$ points from the dataset and the given values of σ^2/σ_0^2. The plots of $p(\mu|\mathcal{D})$ are given in Figure 6, 7 and 8.

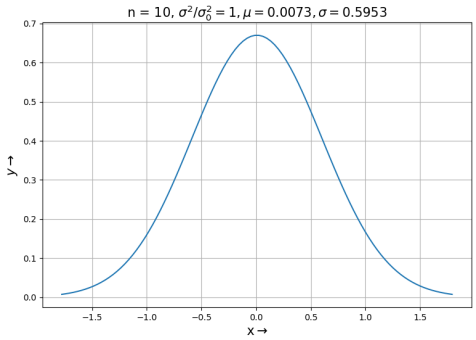
2.3 Inferences

From the plots of $p(\mu|\mathcal{D})$, we infer several points as listed:

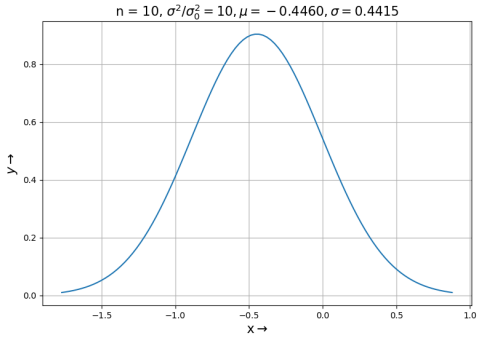
- As the value of N increases from 10 to 1000, the mean of $p(\mu|\mathcal{D})$ approaches the sample mean of the dataset which is obvious as more the number of data points sampled, more "information" about the dataset is captured. Similarly, we also observe that the variance of the distribution reduces because as more data points are sampled, the uncertainty in mean reduces.
- As the value of $r = \sigma^2/\sigma_0^2$ increases from 0.1 to 100, $p(\mu|\mathcal{D})$ shifts towards the prior mean, $\mu_0 = -1$, because the estimated mean is a convex combination of sample mean and prior mean and as the value of r increases, the prior mean is given more weightage as shown in Equation 3.



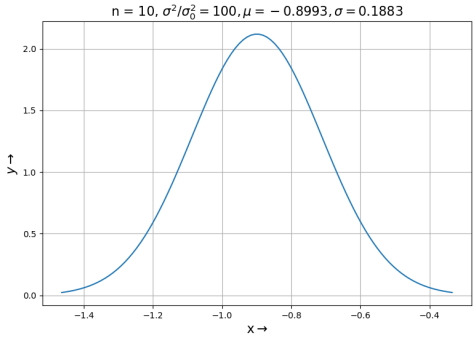
(a) $\sigma^2/\sigma_0^2 = 0.1$



(b) $\sigma^2/\sigma_0^2 = 1$



(c) $\sigma^2/\sigma_0^2 = 10$



(d) $\sigma^2/\sigma_0^2 = 100$

Figure 6: Plots of $p(\mu|\mathcal{D})$ when $N = 10$ points are sampled.

3 Maximum Likelihood Estimation

In this problem, we generate points from three Gaussian distributions and estimate the mean and covariance matrix of the generated points using Maximum Likelihood Estimation and then we shrink the covariance matrix using the Equation 5 ($0<\alpha<1$) and classify the points using the new covariance matrices.

$$\Sigma_i(\alpha) = \frac{(1-\alpha)n_i\Sigma_i + \alpha n\Sigma}{(1-\alpha)n_i + \alpha n} \tag{5}$$

3.1 Estimating Mean and Covariance

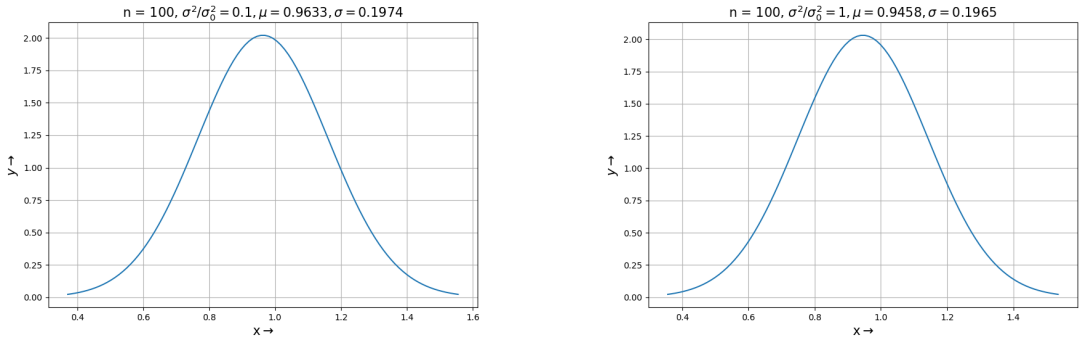
We are given three Gaussian distributions and we take 20 data points from each of the distributions, and estimate mean and covariance matrix of those 60 points. The best estimate would be Maximum likelihood estimate and the ML estimate of the mean of 60 points is given in Equation 6 and the ML estimate of the covariance matrix is given in Equation 7. We can use the same equations to find the best estimate of mean and covariance matrix for the 20 points each separately.

$$\mu_n = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6}$$

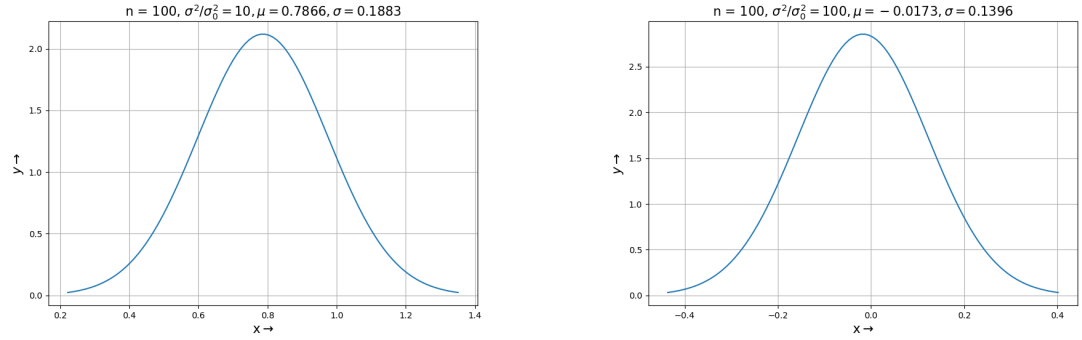
$$\Sigma_n = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_n)(x_i - \mu_n)^T \tag{7}$$

3.2 Plot of Classification error Vs α for training data

For a fixed value of α between 0 and 1, we can find the three new shrinked covariances matrices and assuming the 0-1 loss function, given that the prior is equal for the three distributions, we can



(a) $\sigma^2/\sigma_0^2 = 0.1$ (b) $\sigma^2/\sigma_0^2 = 1$



(c) $\sigma^2/\sigma_0^2 = 10$ (d) $\sigma^2/\sigma_0^2 = 100$

Figure 7: Plots of $p(\mu|\mathcal{D})$ when $N = 100$ points are sampled.

perform a Bayes classification on the 60 points generated using the means and the new covariance matrices. We see that for a 0-1 loss function and equal prior, the Bayes classification turns out in such a way that the point belongs to the distribution whose likelihood probability is more compared to the other two. Using this condition on Bayes classification, we can find the error occured in the Mis-classification of the data point for the given α. Now, if we vary the values of α from 0 to 1 and plot the graph against Mis-classification error is given in figure 9a.

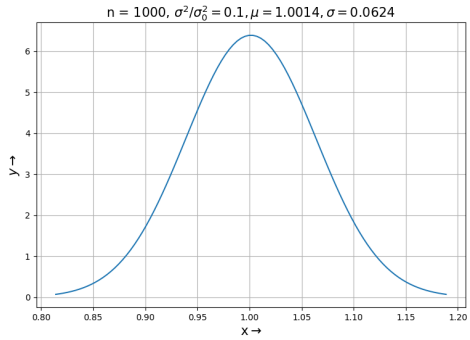
3.3 Plot of Classification error Vs α for test data

The test data is generated from the original Gaussian distributions and we generate 50 points from each distribution and now we plot the classification error Vs α for test data using the estimated means and covariance matrices for the training data. Plot of Classification error Vs α for test data is given in figure 9b.

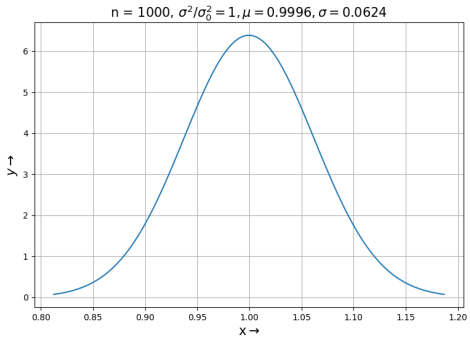
From the graphs, we observe that training error is always less than test error because we are using means and covariances of the training data to classify the test data but the test points are from the original distribution which makes more errors than training data for a particular value of alpha.

4 Singular Value Decomposition

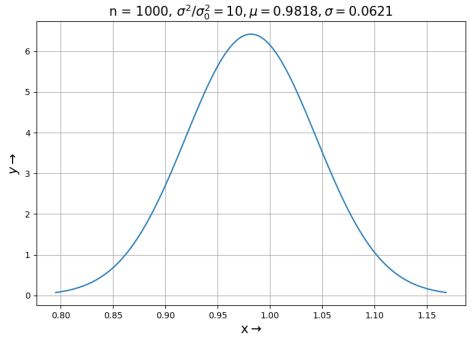
In this problem, we are going to generate two matrices where one matrix has highly correlated columns and the other matrix with statistically independent columns, we find the percentage of data captured by i singular vectors out of n for both the cases using Singular Value Decomposition.



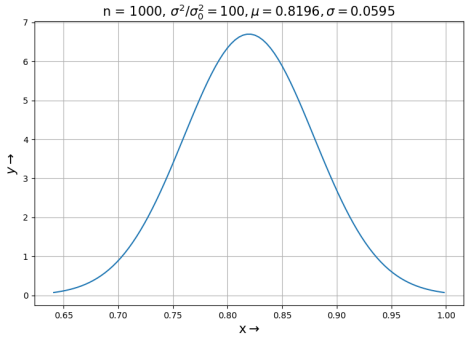
(a) $\sigma^2/\sigma_0^2 = 0.1$



(b) $\sigma^2/\sigma_0^2 = 1$

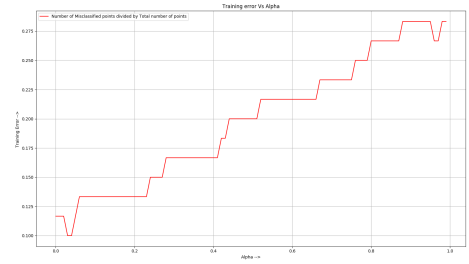


(c) $\sigma^2/\sigma_0^2 = 10$

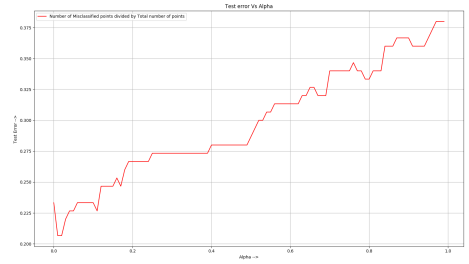


(d) $\sigma^2/\sigma_0^2 = 100$

Figure 8: Plots of $p(\mu|\mathcal{D})$ when $N = 1000$ points are sampled.



(a) Classification error vs α for training data



(b) Classification error vs α for test data

Figure 9: Plot of Classification error vs α for train and test data

4.1 Creating a 100 x 100 Matrix

Method for highly correlated columns:-

Generating the specified matrix can be done by using a Multivariate Gaussian normal of 100 feature vectors with high correlation coefficients and by taking mean as 0.5 and variance to be very small (between 0 and 0.1), we can generate numbers between 0 and 1, and correlation coefficient high (between 0.9 and 1) , we can generate a 100 x 100 Matrix with elements between 0 and 1 and highly correlated columns. The generated matrix changes from simulation to simulation and one such value of Frobenius norm of a simulation is $\boxed{\text{Frobenius Norm} = 93.32}$.

Method for statistically independent columns:-

Generating the specified matrix can be done directly by using a uniform generator between 0 and 1 which gives us a matrix elements between 0 and 1 and statistically independent columns. The generated matrix changes from simulation to simulation and one such value of Frobenius norm of a

Table 2: Fraction of Frobenius captured by singular vectors

Type	N	Fraction
Highly correlated	Top 10%	0.9999%
Highly correlated	Random 10%	0.0037%
Statistically Independent	Top 10%	0.9079%
Statistically Independent	Random 10%	0.1561%

Table 3: Number of singular vectors required to capture data

Type	Percentage of data	Number of vectors
Highly correlated	50%	1
Highly correlated	75%	1
Highly correlated	95%	1
Statistically Independent	50%	1
Statistically Independent	75%	1
Statistically Independent	95%	25

simulation is $\boxed{\text{Frobenius Norm} = 57.64}$.

4.2 Fraction of Frobenius norm captured by singular vectors and Plots

For finding the fraction of the Frobenius norm captured singular vectors, we need to find the data captured by those singular vectors and then find the norm of the data obtained and divide it by norm of the original matrix. The fraction of Frobenius captured by singular vectors is listed in table 2.The number of vectors required to get [50% 75% 95%] for both the matrices is listed in the table 3. Plot of number of singular vectors required to capture data for highly correlated matrix is given in figure 10, and for statistically independent matrix is given in figure 11.

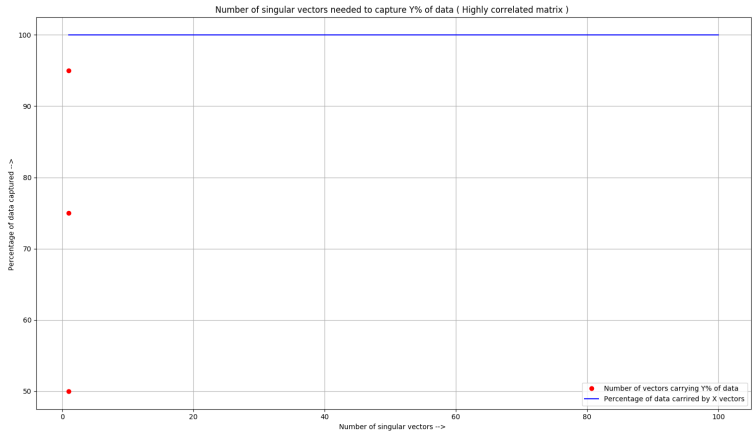


Figure 10: Number of singular vectors needed to capture Y% of data (Highly correlated matrix)

4.3 Observations

We observe the following regarding the relationship between data captured and singular vectors :

- For a matrix with highly correlated columns, the whole data is captured by a single vector and if we look at the graph, there is straight blue line indicating that the whole data is captured by the top first singular vector.

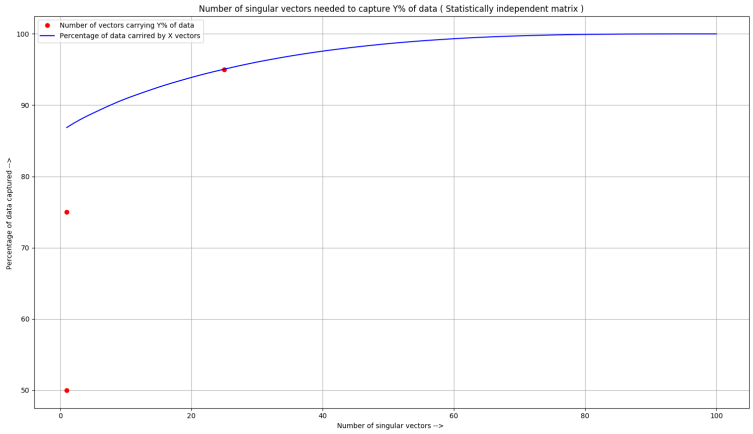


Figure 11: Number of singular vectors needed to capture Y% of data (Statistically independent)

Table 4: Quality of reconstructed images as a function of chosen principal components

N	Quality
Top 10%	91.22%
Top 25%	95.21%
Top 50%	97.94%
Random 10%	56.56%

- For a matrix with highly correlated columns, the percentage captured by random vectors is almost zero as the whole data is present in top one singular vector.
- For a matrix with statistically independent columns, we cannot capture the whole data in a single vector as each column feature is independent and from the graph we observe that the whole data is captured if we take around top 60 singular vectors.
- For a matrix with statistically independent columns, the percentage captured by random vectors is little high when compared to highly correlated as the data is spread between the singular vectors and each vector has some unique information than others.

5 Principal Component Analysis

In this problem, we apply principal component analysis on an image reconstruction problem. The image in gray scale format is presented in Figure 12.

5.1 Reconstruction using top N principal components

The image is reconstructed using top $N = [10, 25, 50]$ % of the principal components. The reconstructed images along with error based on Frobenius norm is presented in Figure 13. The quality of the reconstructed image is listed in Table 4. It is seen in Figure 13, that with as low as top 10% of the principal components, we can get the original image with reasonable accuracy.

5.2 Reconstruction using random principal components

The image is reconstructed using random 10% of the principal components. The reconstructed images along with error based on Frobenius norm is presented in Figure 14. The quality of the reconstructed image is listed in Table 4. Although by using top 10% of the principal components we can almost get back the actual image as shown in Figure 13, using random 10% of the principal components yields an indiscernible image in Figure 14.

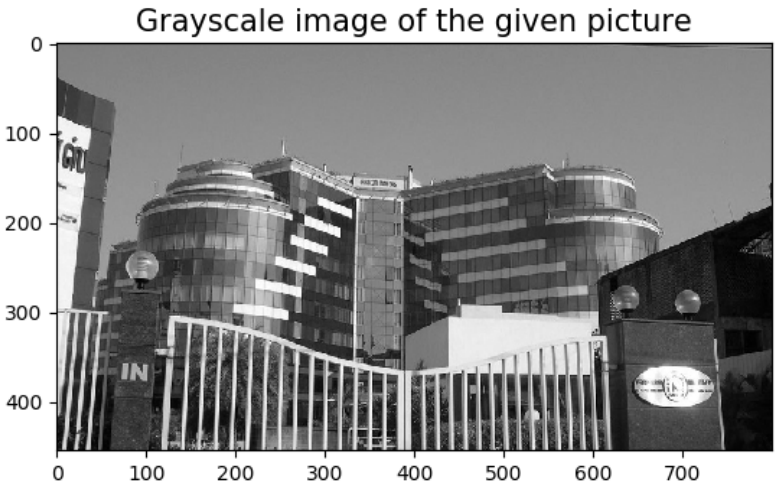


Figure 12: Grayscale format of the given image

5.3 Reconstruction error vs top N principal components

We investigate the quality (or equivalently the reconstruction error), of the reconstructed image using top $N\%$ of the principal components. The reconstruction loss as a function of N is plotted in Figure 15. As it is evident from the graph, with as less as top 10% of the principal components, we are able to get an image with is more than 90% similar to the original image.

The image is reconstructed using top $N = [10, 25, 50]$ % of the principal components. The reconstructed images along with error based on Frobenius norm is presented in Figure 13. The quality of the reconstructed image is listed in Table 4. It is seen in Figure 13, that with as low as top 10% of the principal components, we can get the original image with reasonable accuracy.

6 Regression and Bias Variance

In this problem, we analyse the bias-variance trade off for polynomial regression. $N = 100$ data points are generated according to the expression given in Equation 8:

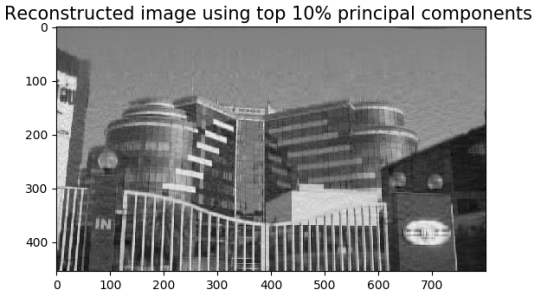
$$f(x) = e^{sin(2\pi x)} + \epsilon \tag{8}$$

where $\epsilon \sim \mathcal{N}(0, 0.2)$. The entire dataset is split into 80:20 (train:test) and polynomial regression is performed.

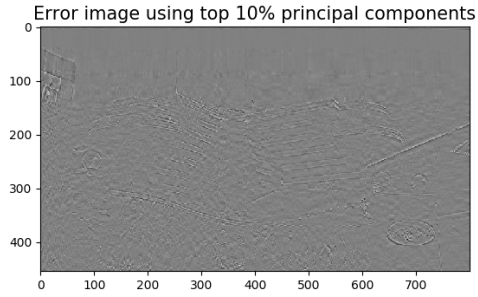
6.1 Polynomial Regression

The corresponding plots are presented in Figure 16 and the coefficients are given in Table 5. It can be seen in Table 5 that the coefficients explode as the degree of polynomial increases from $M = 1$ to $M = 9$.

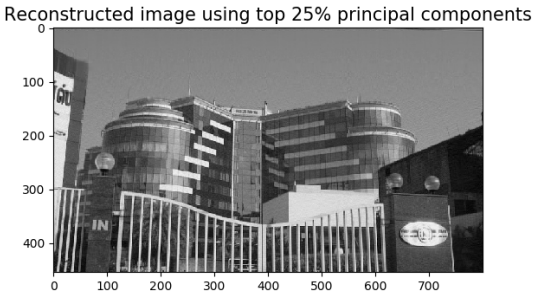
It is observed that for a given model complexity, as the number of data points increases, the issue of overfitting reduces. This can be explained as more the number of data points taken into consideration, we can have a more complex function to fit the data points. We try to fit $N = 10$ and $N = 80$ data points for a sixth degree polynomial and the results along with training error are shown in Figure 17.



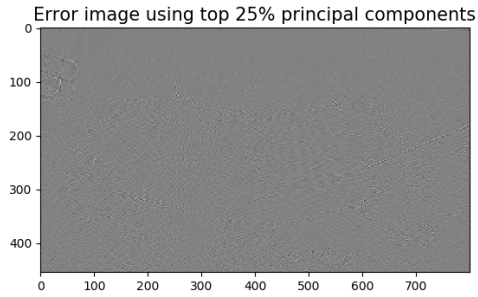
(a) Reconstructed image for top $N = 10\%$



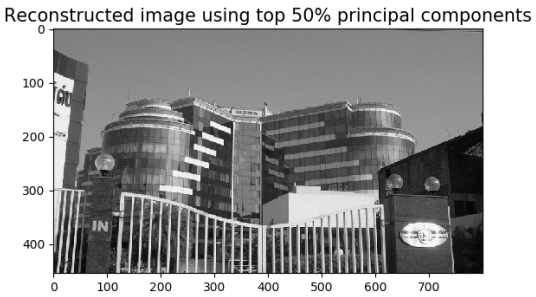
(b) Error image for top $N = 10\%$



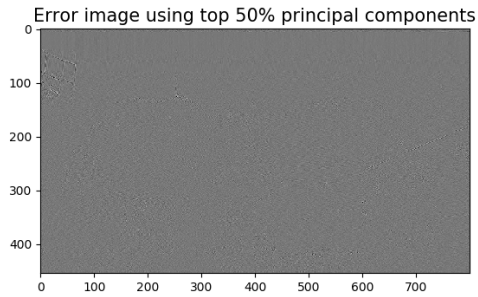
(c) Reconstructed image for top $N = 25\%$



(d) Error image for top $N = 25\%$



(e) Reconstructed image for top $N = 50\%$



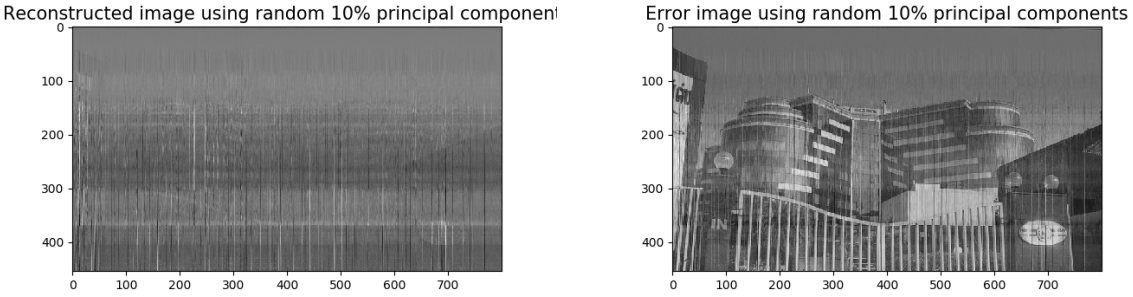
(f) Error image for top $N = 50\%$

Figure 13: Reconstructed images along with error images using top $N = [10, 25, 50]\%$ of principal components

We tried fitting polynomial with degrees $M = [1, 3, 6, 9]$ and the *degree 6 polynomial has the least generalization error*. The scatter plots for the training and test data along with errors is presented in Figure 18. The training and test error are almost equal like it should be in the ideal case.

6.2 Root-Mean-Square (RMS) error vs degree of polynomial

The root-mean-square (RMS) error of training data and testing data is plotted as a function of the degree of the polynomial M in Figure 19. It can be inferred from the plot that the training error continuously decreases as degree of polynomial increases but the generalization error decreases upto



(a) Reconstructed image for random 10%

(b) Error image for random 10%

Figure 14: Reconstructed images along with error images using random 10% of principal components

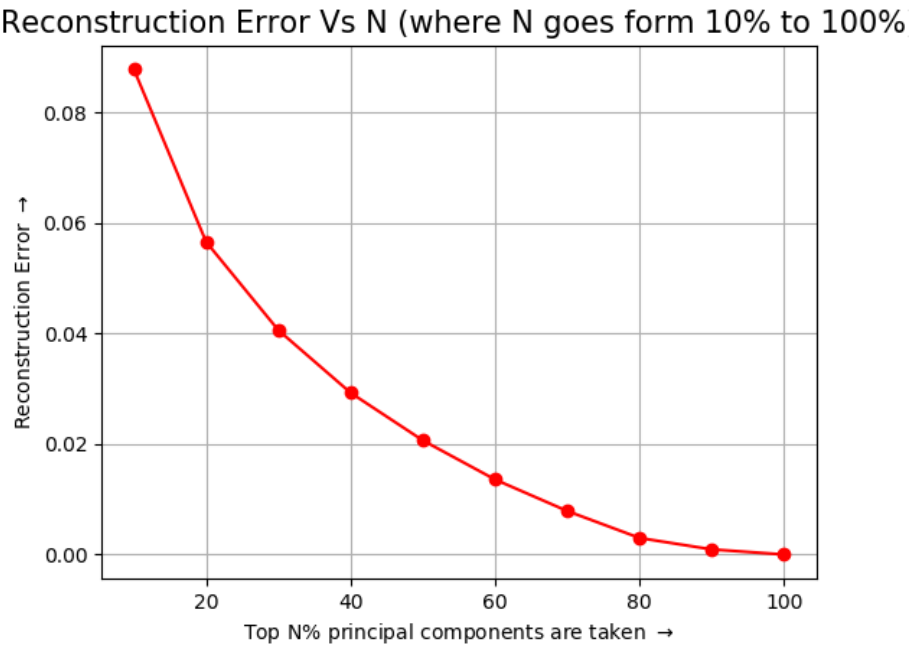


Figure 15: Reconstruction loss vs N

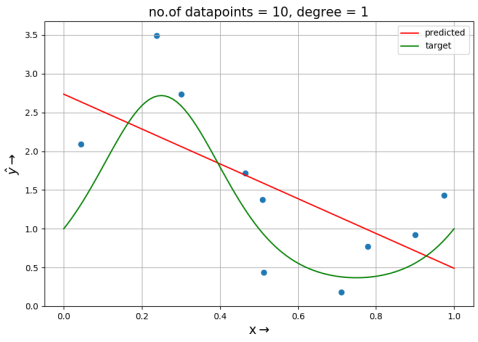
a certain extent and then increases which agrees with bias-variance trade-off theory [1].

References

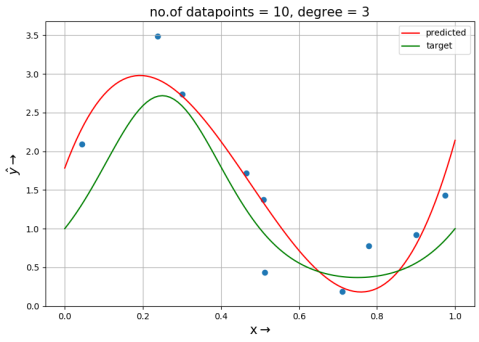
[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Table 5: Table of coefficients w^* of the polynomials

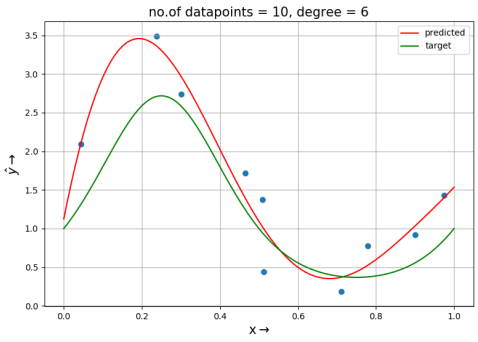
	$M=1$	$M=3$	$M=6$	$M=9$
w_0^*	2.736	1.783	1.125	7385.341
w_1^*	-2.246	13.553	25.318	-291655.847
w_2^*		-44.079	-66.827	3643567.62
w_3^*		30.883	-57.450	-2261486.6
w_4^*			307.152	81359258.4
w_5^*			-304.234	180715147
w_6^*			96.451	251698888
w_7^*				-2141412344
w_8^*				101728629
w_9^*				-20674174.2



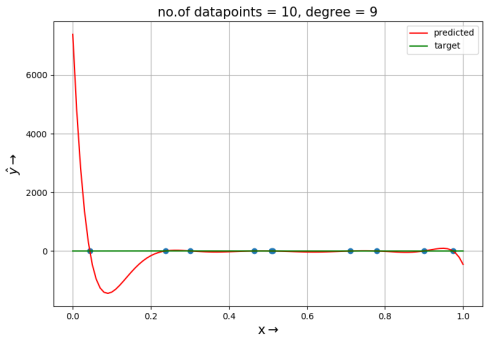
(a) $M = 1$



(b) $M = 3$

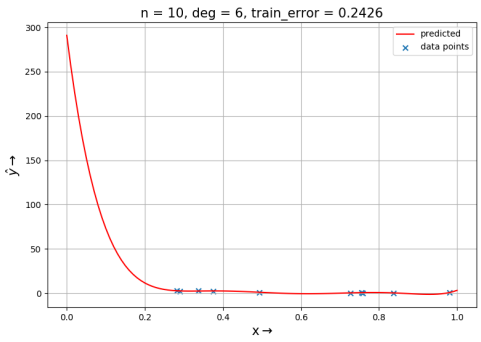


(c) $M = 6$

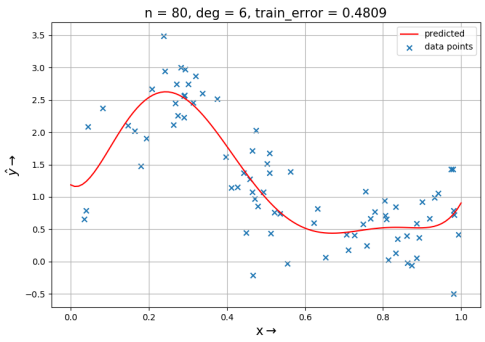


(d) $M = 9$

Figure 16: Plot of polynomials with different $M = [1, 3, 6, 9]$ fitted to $N = 10$ data points

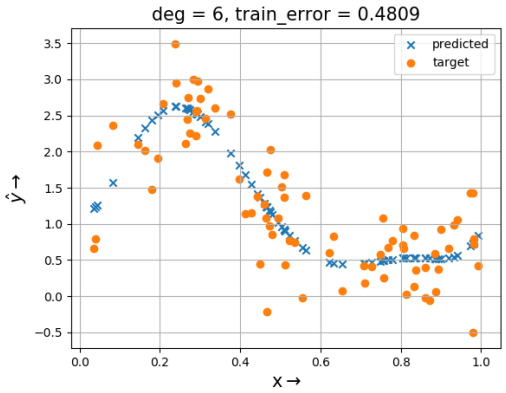


(a) $N = 10$ data points

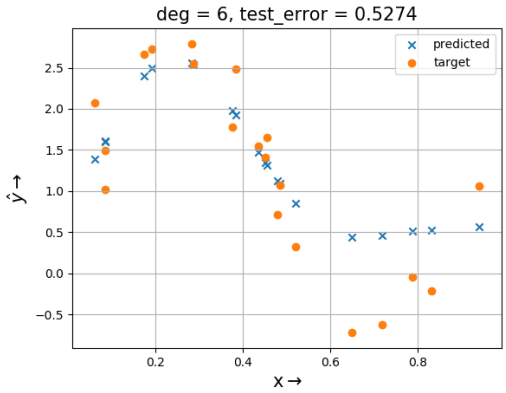


(b) $N = 80$ data points

Figure 17: Analyzing the effect of overfitting for a particular model complexity for $N = 10$ and $N = 80$ data points.



(a) Training data



(b) Test data

Figure 18: Scatter plots for polynomial with degree $M = 6$ for the training and test data

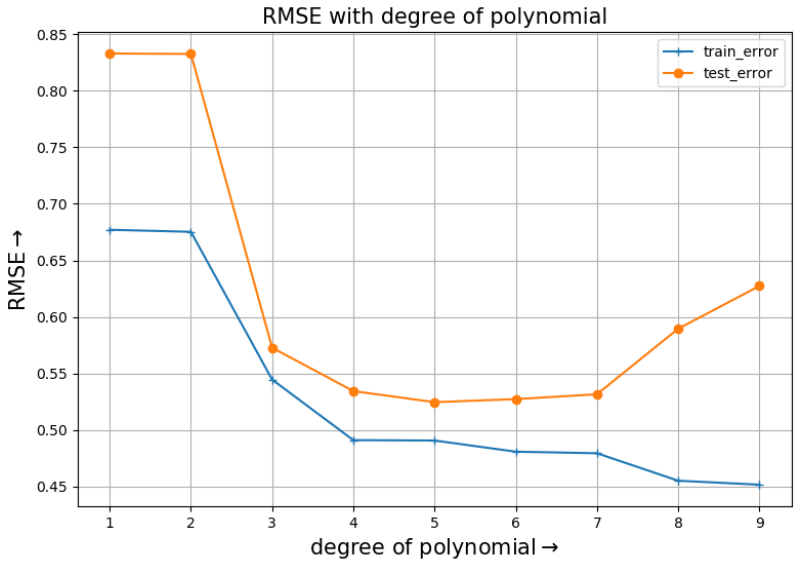


Figure 19: RMSE Values of training and testing errors vs degree of the polynomial