Introduction to Machine Learning

Team: Sphoorthi Keshannagari (50372699) skeshann@buffalo.edu
Bharath Reddy Nallu (50376364) bnallu@buffalo.edu
Ishita Aggarwal (50431309) iaggarwa@buffalo.edu

Problem 1: Experiment with Gaussian Discriminators

Linear Discriminant Analysis (LDA) is used as a classification and visualization technique. Quadratic Discriminant Analysis (QDA) is a variant of LDA. LDA assumes that the feature covariance matrix of both classes is same, thereby resulting in a linear decision boundary. In QDA an individual covariance matrix is estimated for each class of the observations and leads to a quadratic decision boundary.

1.1 Linear Discriminant Analysis (LDA)

1.1.1 Code Explained:

For implementing LDA, we create a function ldaLearn(). We input the x & y matrices from the training data. The first step is to calculate the mean for each of the classes present in the training sample. We iterate through two loops i & j to pick data in each classes and store the calculated means of the same in a d x k matrix 'means'. We calculate a single d x d covariance matrix using cov function in numpy library.

The returned mean and covariance matrices along with the Xtest and ytest matrices are then passed into the function – ldaTest().We calculate the predicted labels by doing using the pdf of Gaussian distribution and returns the class with maximum pdf. We calculate the accuracy based on the number of predicted values that match with true values of y.

1.1.2 Accuracy:

The accuracy for LDA is: 97

1.2 Quadratic Discriminant Analysis (QDA)

1.2.1 Code Explained:

For implementing QDA, we create a function qdaLearn(). The means are calculated following the same steps as LDA. We calculate a total of k d x d covariance matrices by using numpy.cov function on submatrices of x based on classes.

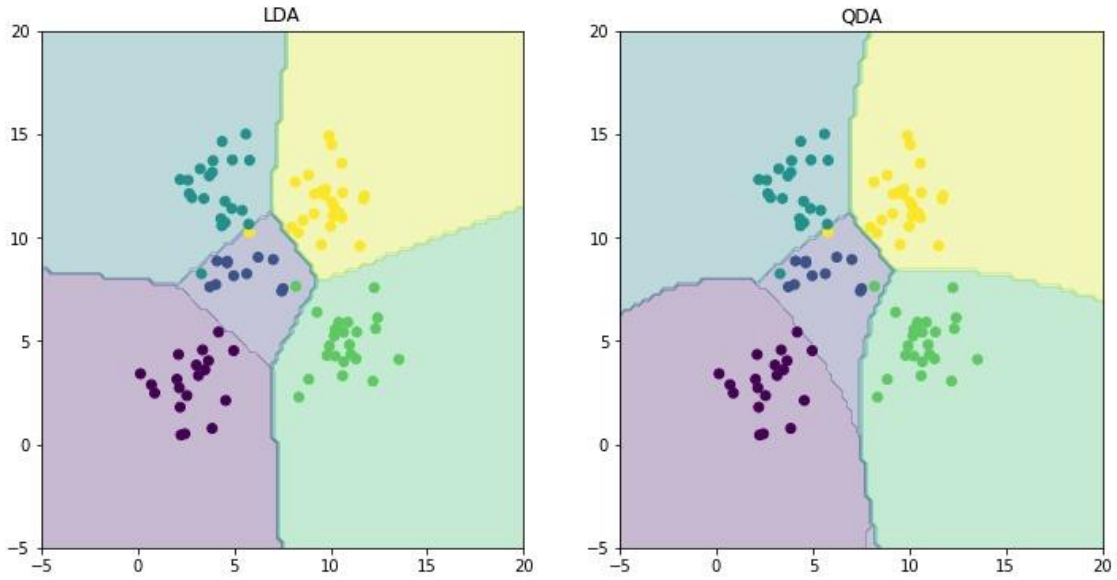
The returned mean and covariance matrices along with the Xtest and ytest matrices are then passed into the function – qdaTest(). We calculate the predicted labels by doing using the pdf of Gaussian distribution and returns the class with maximum pdf. We calculate the accuracy based on the number of predicted values that match with true values of y.

1.2.2 Accuracy:

The accuracy for QDA is: 97

1.3 **Plotting Discriminating Boundary:**

```
LDA Accuracy = 97.0
QDA Accuracy = 97.0
```



Based on the training data, we observe that the accuracy of both the methods is similar.

Problem 2: Experiment with Linear Regression

2.1 Objective

Our objective is to implement Linear Regression. Linear Regression is used in the field of predictive modelling and is aimed at minimizing the error of a model to make the most accurate predictions possible. Given: X is the input data matrix, y is the target vector, and w is the weight vector for regression.

We will calculate the mean squared error of the training data by implementing the below formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

2.2 Code Explained:

For implementing Linear Regression, using the x & y from training data into the function learnOLERegression() - we calculate the weight by implementing the below formula:

$\widehat{\mathbf{w}}_{MLE} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$. The returned weight value w along with (xi and yi) from the training data are passed into the function testOLERegression(). We calculate the aggregate mean squared value over N and return the same. We multiply the weight with each value in the xi column and subtract the same from the yi column and calculatethe MSE of the data.

2.3 Output:

We add an intercept to the X axis, to validate the MSE value with intercept.

MSE with intercept: 3707.840181476299

MSE without intercept: 106775.3615522326

Prediction with intercept is performing better than prediction without intercept.

Problem 3: Experiment with Ridge Regression

3.1 Objective:

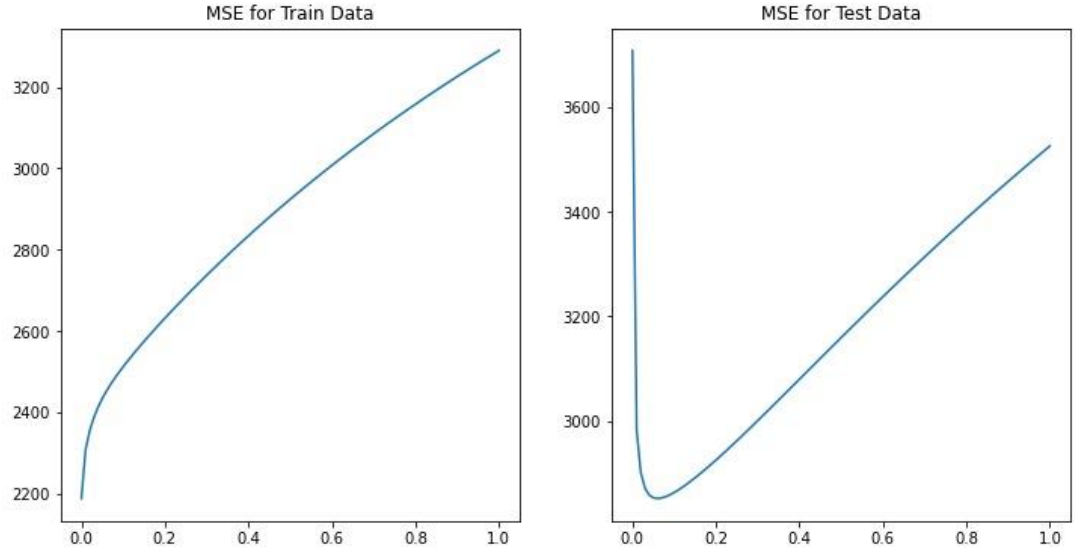
Our objective is to implement Ridge Regression. The term ridge stands for the extra influence (lambda) added to all points on a curve to maintain its overall shape. We will calculate the ridge regression by implementing the below formula: $\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$

3.2 Code Explained:

For implementing Ridge Regression, we will pass the (x,y) from the training data and a set of lambda value generated to calculate the weight. We will calculate the weight w by using the formula: $\hat{\mathbf{w}}_{MAP} = (\lambda \mathbf{I}_D + \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ and we will return the weight from the function learnRidgeRegression(). We will pass multiple values of lambda from 0 to 1 and calculate the MSE.

3.3 Output:

Plots of MSE vs Lambda for Training Data and Test Data:



Comparing the relative magnitudes of the weights learnt from OLE and Ridge Regression, we observe that the weight values are similar.

On comparing MSE for OLE and Ridge regression, we observe that the error in Ridge regression is lesser than the error in OLE.

The optimal value for lambda for test data is 0.06. It is the optimal value because the mean squared error is the minimum when lambda is 0.06.

The optimal value for lambda for training data is 0. It is the optimal value because the mean squared error is the minimum when lambda is 0.

Problem 4: Using Gradient Descent for Ridge Regression Learning

4.1 Objective:

Our objective is to implement gradient descent for Ridge Regression Learning. Ridge Regression solves the overfitting problem when MSE for test data is high. We use gradient descent to minimize the squared loss. The formula for calculating the error for the same is:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \frac{1}{2}\lambda||\mathbf{w}||_2^2$$

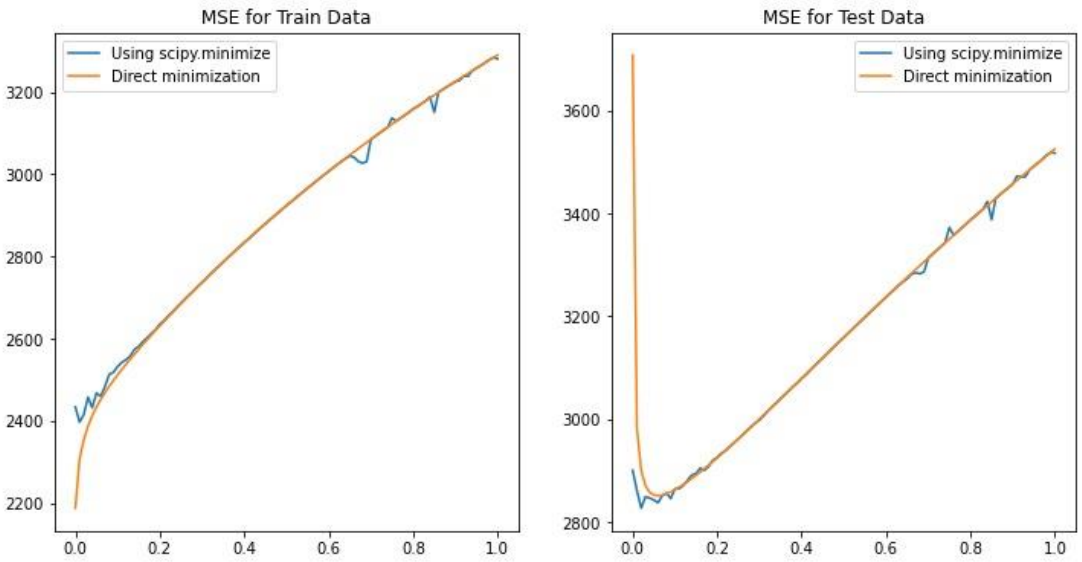
To calculate the error gradient, we will differentiate the loss function over w.

4.2 Code Explained:

For implementing the gradient descent for ridge regression learning, we will pass the following values to the function regressionObjVal() – weight (from problem 3), x & y from the training data and the lambda value. We will calculate the error by implementing the formula mentioned above. We will then calculate the error gradient and return the error and error gradient.

4.3 Output:

Plot of MSE vs lambda for Training Data and Test Data:



Comparing the values of results obtained in Problem 3 to that in Problem 4, we observe that the error is lesser when applying gradient descent for ridge regression learning.

Problem 5: Non-Linear Regression

5.1 Objective:

Our objective is to implement Non-Linear Regression. It is used in place when the data is depicting a curving trend. In this case, a linear regression would not give optimal results and we would need to implement non-linear regression techniques. Non-linear can be of any shape – exponential, logistic or logarithmic.

5.2 Code Explained:

For implementing non-linear regression, we pass a vector x and an integer p (having value 0-6) in the function mapNonLinear(). We will convert the single column matrix x into a matrix Xp of size N x (p+1) with columns having values 1, x, x square, x cube and so on. We will return this matrix Xp to be plotted.

5.3 Output:

	Training Data		Test Data	
p	MSE (lambda 0)	MSE (lambda 0.06)	MSE (lambda 0)	MSE (lambda 0.06)
0	5650.7105389	5650.71190703	6286.4049168	6286.88196694
1	3930.91540732	3951.83912356	3845.03473017	3895.85646447
2	3911.8396712	3950.68731238	3907.12809911	3895.58405594
3	3911.18866493	3950.68253152	3887.97553824	3895.58271592
4	3885.47306811	3950.6823368	4443.32789181	3895.58266828
5	3885.4071574	3950.68233518	4554.83037743	3895.5826687
6	3866.88344945	3950.68233514	6833.45914872	3895.58266872

Based on the above values, we can determine that the optimal values of p are as below:

Optimal p when lambda is 0 for Test Data: 1

Optimal p when lambda is 0.06 for Test Data: 4

Optimal p when lambda is 0 for Training Data: 6

Optimal p when lambda is 0.06 for Training Data: 6

Problem 6: Interpreting Results

Comparing the various approaches in terms of training and test data:

Technique	Training Data	Test Data
Linear Regression with intercept	2187.1602949303897	3707.840181476299
Linear Regression without intercept	19099.44684457091	106775.3615522326
Ridge Regression	2187.16029493	2851.33021344
Gradient Descent for Ridge Regression	2396.44210894	2826.9525654
Non-linear with lambda 0	3866.8834494460493	4443.327891813327
Non-linear with lambda 0.06	3895.582668719096	3895.8564644739627

Based on the above table, we determine that for Training Data the best setting is Ridge Regression.

And for Test Data the best setting is Gradient Descent for Ridge Regression.