CSE 474/574
Introduction to Machine Learning
Programming Assignment 2
Classification and Regression

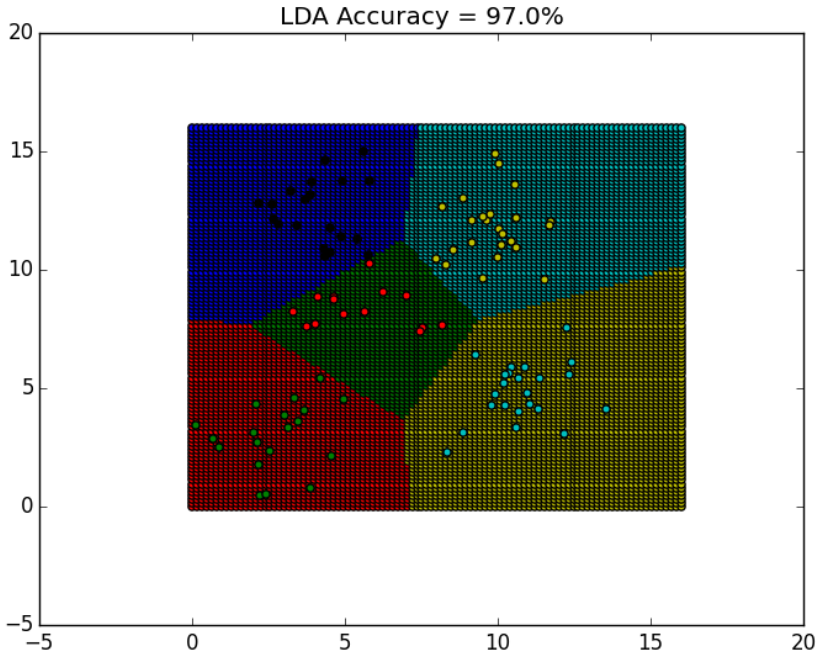
Team 73: *Harsh Harwani*, *Ankit Kapur*,
Harishankar Vishwanathan

Experiment 1: Gaussian discriminators

In this problem we experimented with two Gaussian discriminators:

1. Linear Discriminant Analysis

The plot below shows the the LDA classification.We achieved an accuracy of **97.0%** on the test data.

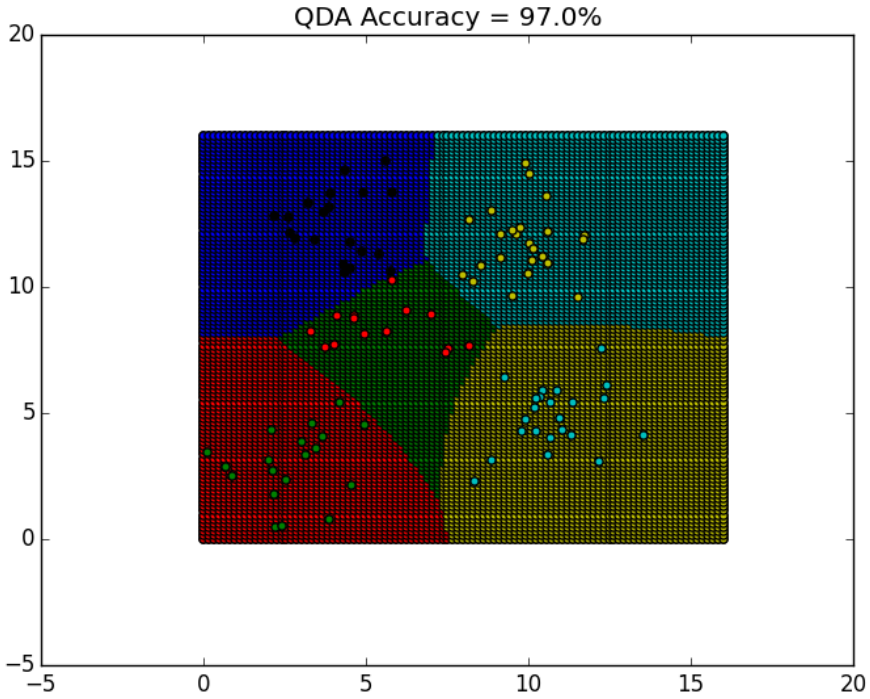


We calculated the mean of each class, and the covariance matrix of the entire sample dataset. In LDA we assume that the Gaussian distributions for different classes share the

same covariance. By making this assumption the classifier becomes linear. The decision boundary of various classes is linear (we learn lines in case of LDA).

2. Quadratic Discriminant Analysis

The plot below shows the the QDA classification.We achieved an accuracy of **97.0%** on the test data.



We calculated the mean of each class, and the covariance matrix of each class. In QDA we do not assume that the Gaussian distributions for different classes share the same covariance structure,each class has its own covariance matrix. As a result the classifier becomes quadratic in nature. The shape of the decision boundary is determined by a quadratic function which results in decision boundaries bein curves instead of lines.

In LDA decision boundaries are lines and in case of QDA they are curves. The difference in both the boundaries is because of the covariance matrix. In case of LDA as we consider the same covariance matrix, for points having the same probability, the common terms in the probability density function get cancelled out and we get a linear function in contrast to QDA where we get a quadratic function.

Experiment 2: Linear Regression

In this experiment we implemented the *Linear Regression* function, which basically learned the weight vector using the training data matrix x and the corresponding labels y. The weights are learned by implementing the ordinary least squares method by minimizing the squared loss function.

The squared loss function

$J = \frac{1}{2}(y - w^T X)^T (y - w^T X)$

was minimized by taking the derivative with respect to w and equating it to 0. The w obtained was:

$w = (X^T X)^{-1} X^T y$

Data Type(Training/Test)	Intercept(Y/N)	RMSE value
Training	Y	3.00630212
Training	N	8.88388057
Test	Y	4.30571724
Test	N	23.10577434

As shown in the above table, the RMSE is lower when an intercept is used (both in case of training and test data). This is as expected, because without the intercept, the hypothesis line learned must pass through the origin (and is restricted to rotation only). When an intercept is introduced it gives the line the ability to rotate as well as translate. So the hypothesis learned is much closer to the true concept - which is why the errors with the intercept are much lesser. Hence, **the accuracy with intercept is much higher** (5.3 times higher) than that without intercept.

Experiment 3: Ridge Regression

In this experiment we implemented the *learnRidgeRegression* function, which learned the weight vector using the training data matrix x and the corresponding labels y. A regularization factor λ was included as part of the learning function, in order to control the impact of outliers on the weights learned, and to avoid overfitting.

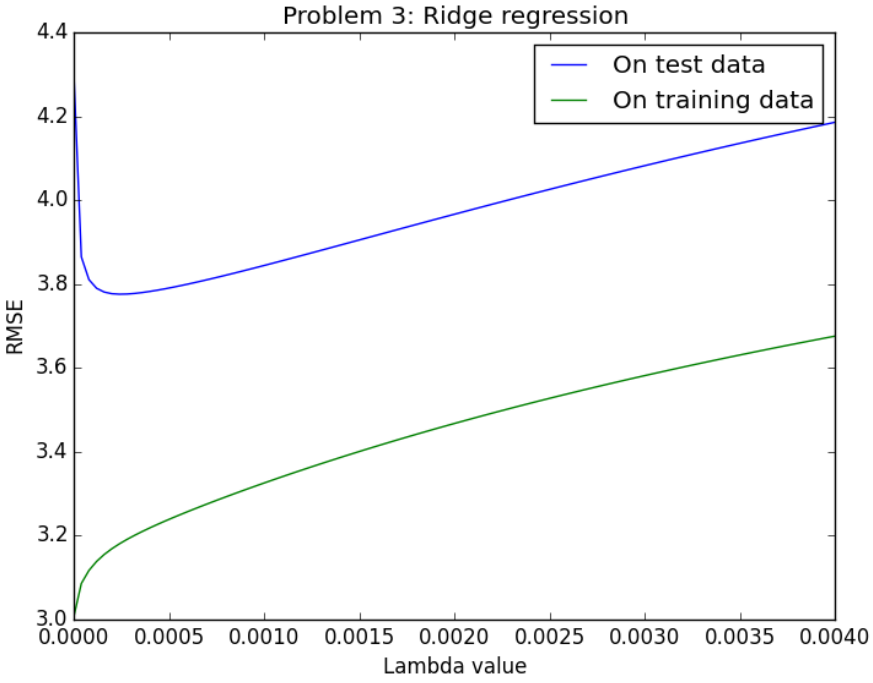
The regularized squared loss function

$$J = \frac{1}{2N}(y - Xw)^T(y - Xw) + \frac{1}{2}\lambda w^T w$$

was minimized by taking the derivative with respect to w and equating it to 0. The w obtained was:

$$w = (X^T X + \lambda N I)^{-1} + X^T Y$$

Optimum λ value observed: **0.00024**. This was the λ value that gave the lowest RMSE error of 3.77582332



Observations:

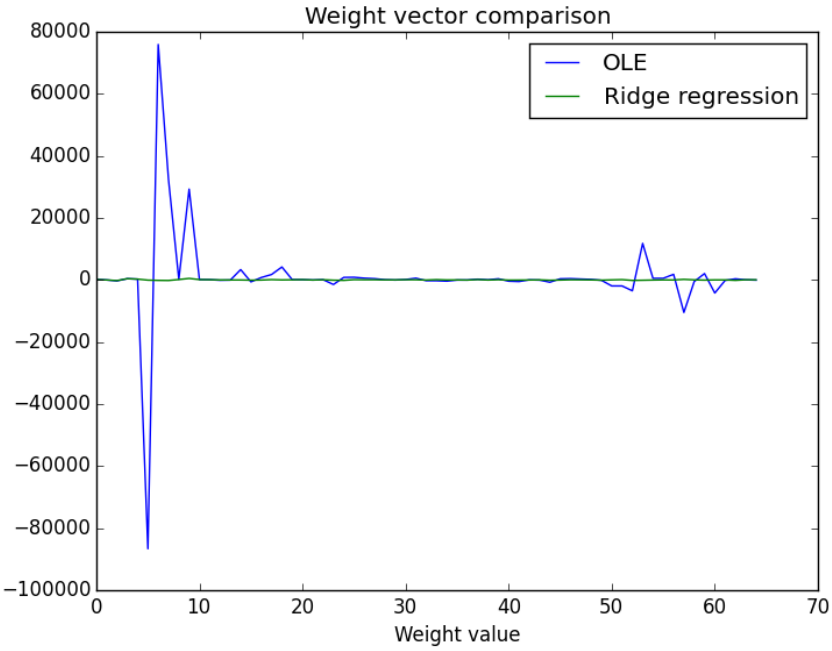
By varying the λ value, it was observed that RMSE steeply decreases initially with an

increasing λ value. This was expected behavior, as when there was no regularization, at $\lambda=0$, the $\frac{1}{2}\lambda w^T w$ term in the objective function J(w) would have become 0. So the learning function would be unable to control the weights in order to avoid **overfitting** on the training data. As the λ value was increased beyond 0, the $\frac{1}{2}\lambda w^T w$ term now became part of the minimization, thus allowing the weights to be kept in control. This reduced overfitting on the training data, and reduced errors on the test data - leading to lower RMSE values.

However, as λ goes increases further, beyond 0.0002, it was observed the errors began increasing. This was expected behavior as well, because the purpose of the regularization factor λ is to keep a grip on the weight vector, and to not allow it to grow too much. When the λ value is too high it puts a very tight restriction on the weight vector from growing during learning. As a result, an **underfitting** problem occurs, causing errors on the test data - as is visible in the growing RMSE values.

Weight vector comparisons

It was observed that the learning phase in OLE led to a much higher weight magnitudes as compared to the weights in the Ridge regression learning process.



While the total sum of the the weight vector obtained in Ridge regression was 2092.60, the sum observed for OLE was much higher at 57382.49. It was also observed that the variance in the OLE weight vector was much higher, as compared to Ridge regression:

Variance of the OLE weight vector: 237806821.76
Variance of the Ridge regression weight vector: 13333.80

The reason for the vast difference between the weight vector variances can be attributed to the fact that regularization was used in the Ridge regression function. The $\frac{1}{2}\lambda w^T w$ term that was incorporated into the objective function J(w) became part of the minimization, thus allowing the weights to be kept in control. **This is the reason why the weight vectors observed after the learning process in Ridge regression show such limited magnitudes.**

Experiment 4: Using Gradient Descent for Ridge regression Learning

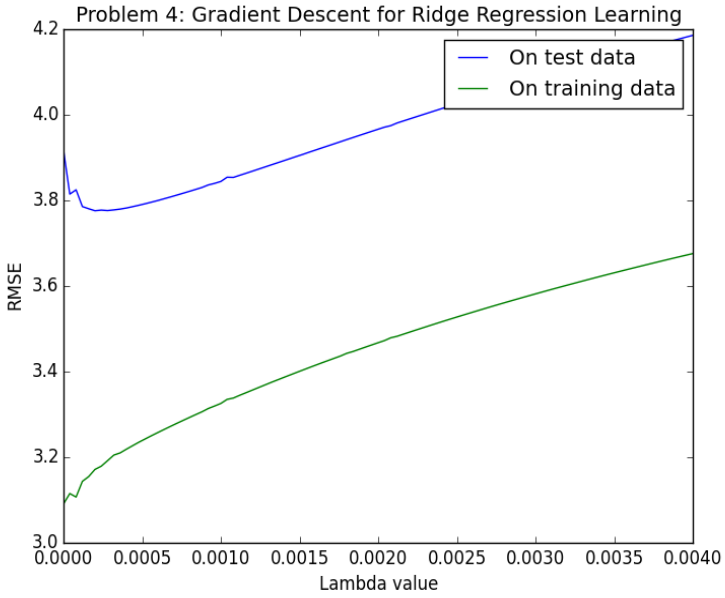
In this experiment we implemented the *regressionObjVal* function, which, given the data X and y and the regularization parameter λ, produced a squared error value using the regularized squared loss function:

$$J = \frac{1}{2N}(y - Xw)^T(y - Xw) + \frac{1}{2}\lambda w^T w$$

and the gradient of the squared error with respect to w:

$$\frac{dJ}{dw} = \frac{1}{N}(w^T(X^T X) - y^T X) + \lambda w^T$$

Here, unlike problem 3, the gradient descent method was used to minimize the loss function, by passing *regressionObjVal* to a *minimize* function which learned the weights.



Plotted above is the graph for RMSE vs λ using gradient descent. We observe behaviour similar to what we saw in Problem 3, where the derivative was used for the learning the weights. RMSE steeply decreases initially with increase in λ. As λ goes increases further, the error began to increase, as expected.

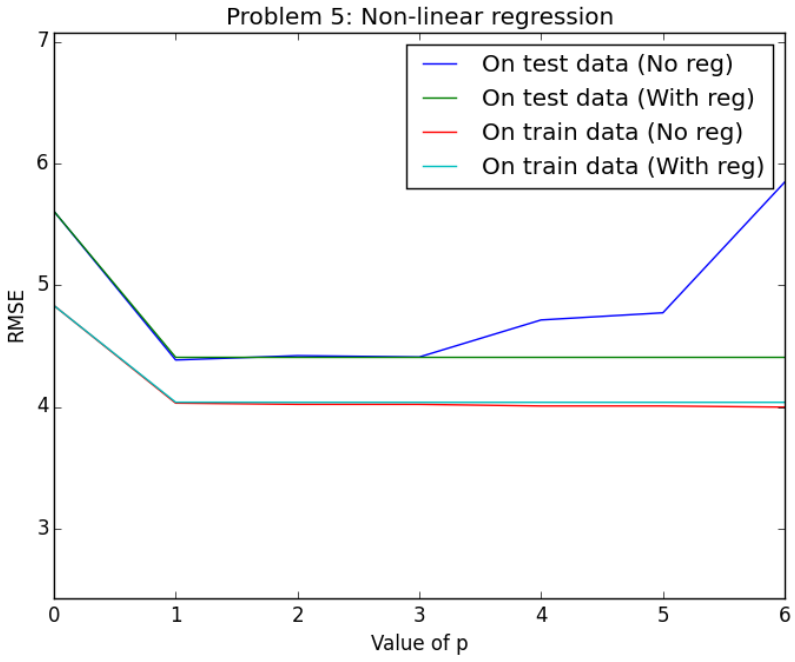
But while using the derivative optimization technique for ridge regression, if the number of features is large, the inverse of the covariance matrix becomes unstable. We may also get a singular covariance matrix, in which case finding an inverse is not possible. In these cases, the Gradient Descent technique is useful. Gradient descent gives satisfactory results, similar to what we saw using the derivative method.

Experiment 5: Non-linear Regression

In this experiment we implemented the *non-linear regression* function, in which we trained ridge regression weights using the non-linear mapping of the data. We experimented with as well as without regularization. The regularization factor used was the optimum value $\lambda = 0.0002$ obtained in the previous experiments. The function *mapNonLinear* was implemented to convert the single attribute x into a vector **Xd** of p attributes: $[x^0, x^1, x^2, \ldots x^p]$

This transformation allows us to use **linear regression** to learn non-linear curves. The learning process is exactly the same as that used in Ridge regression, except that instead of using the actual data vector **X** we use the transformed matrix **Xd** (obtained using the *mapNonLinear* function). We then obtained the predictions on the data by multiplying Xd by the weight vector W obtained from the learning function. The predicted label for each data point x is calculated as $y = w_0x^0 + w_1x^1 + w_2x^2 \ldots + w_px^p$

The RMSE errors observed for various values of p are noted below.



Observations

It was observed that **when p=0**, the value of RMSE is the high. This is expected because in this case the regression line will be a horizontal line and no learning is done from the training data - as the transformation Xd of the X vector is only made of ones (because the

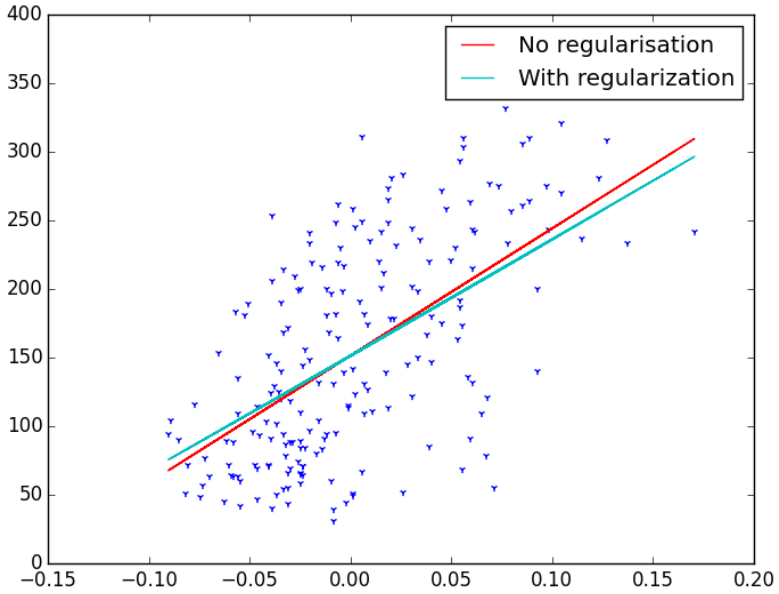
data points are raised to the power of zero).

The RMSE value begins decreasing beyond p=0 which is expected as weights are trained using ridge linear regression. When p=3 and λ=0 the RMSE value increases steeply, reason being there is no regularization factor, causing **overfitting** of data. The fact that there is overfitting is proven because when p=3 and λ=0.0002 the value of RMSE does not increase, as the additional λ term provides **regularization** and avoids **overfitting** of data. On the training data, the RMSE value continues to be low for values beyond p=3 because the training was done on this data itself. The optimum values of p observed were:

p=1 with no regularization (when λ=0)

p=2 with no regularization (when λ=0.0002)

The curves obtained for the optimum p values were plotted, both for λ=0 and λ=0.0002, and are shown in the figure below. It should be noted that since the curve for λ=0.0002 should ideally be a parabola since p=2, so the predicted label function will be quadratic $y = w_0x^0 + w_1x^1 + w_2x^2$. However the regularization factor makes it a straight line, which is why both curves appear as straight lines in the figure below.



Experiment 6: Interpreting the results

In order to choose the best setting using regression for predicting the diabetes level we can consider accuracy, performance and stability. Accuracy for each of the regression techniques can be calculated by considering their RMSE error value for test data. The table follows. Note that the RMSE for only the optimal λ value was selected while considering the RMSE for Ridge Regression and Gradient Descent Ridge Regression techniques.

Regression Technique (type)	**RMSE Error Value**	**Time taken (sec)**
Ordinary Linear Regression	4.30571724	0.00682497024536
Ridge Regression (Derivative Technique)	3.775832332	0.0364651679993
Gradient Descent Ridge Regression	3.777583231	6.00428509712
Non-Linear Regression	4.41854359939	0.032438993454

If accuracy is important, both the **Ridge Regression** and the **Gradient Descent Ridge Regression** seem to be a good choice; they have the lowest RMSE value for test data. But we must also note that while using the derivative optimization technique for ridge regression, if the number of features is large, the inverse of the covariance matrix becomes unstable. We may also get a singular covariance matrix, in which case finding an inverse is not possible. In this case, we can choose the Gradient Descent technique.

We can also see that the Gradient Descent Ridge Regression technique takes the maximum amount of time, while the Ordinary Linear Regression technique takes the least. Thus, if the taken time has to be minimized, but accuracy can be compromised, we suggest the Ordinary Linear Regression Technique.