Lab-2 Report Chirag Khurana (2016CSB1037)

Linear Ridge Regression

Introduction

The function named **mylinridgereg(X, Y, lambda)** implemented with both **gradient descent** and **analytical** method that calculates the linear least squares solution with the ridge regression penalty parameter lambda (λ) and returns the regression weights.

- Data Standardization($\bar{u} = 0,\ \sigma = 1$):

$$x' = \frac{(x - \bar{x})}{\sigma}$$

- Cost Function:

$$J(w) = \sum_{i=1}^{N} (x_i^T w - y_i)^2 + \lambda ||w||_2^2$$

- Gradient Descent Learning parameter(α) = 0.0001
- Gradient Descent Stopping Criteria:

  i) Maximum iterations = 10^6

  ii) Difference between previous and current cost < 10^{-3}

- Analytical Method:

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

- Used analytical method in experiments to do faster calculations.

Experiments:

1. **The effect of λ on error change for size of the training set:**

   Training Data Partition Fraction Set = {0.03,0.06,0.1,0.2,0.5,0.8,1}

   λ Set = {0,0.1,0.5,1,2,4,8,16,20}

   For each combination of fraction and λ i.e. 7×9 linear regressions performed.

   Each regression combination is performed 100 times with randomized data partition and Average Mean Square Error over 100 repetitions is taken.

   Training/Testing Average Mean Square Error:

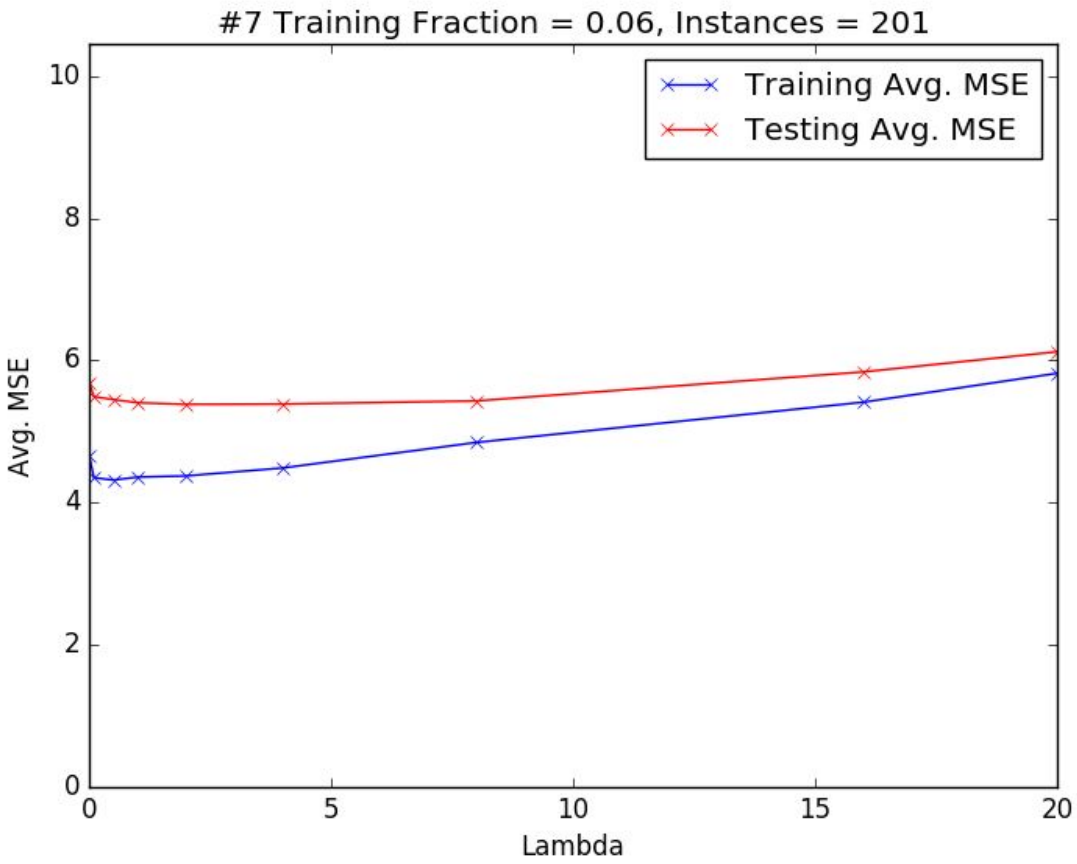
Frac → **λ ↓**	**0.03**	**0.06**	**0.1**	**0.2**	**0.5**	**0.8**	**1**
0	4.07 6.14	4.66 5.69	4.57 5.47	4.59 5.31	4.63 5.25	4.70 5.24	4.70 5.24
0.1	4.17 5.77	4.35 5.49	4.44 5.38	4.59 5.28	4.67 5.24	4.68 5.24	4.70 5.24
0.5	4.19 5.65	4.32 5.45	4.56 5.32	4.54 5.27	4.66 5.24	4.69 5.23	4.70 5.24
1	4.14 5.63	4.36 5.41	4.47 5.35	4.62 5.26	4.67 5.23	4.68 5.22	4.70 5.23
2	4.35 5.61	4.38 5.38	4.56 5.31	4.61 5.25	4.66 5.23	4.67 5.22	4.70 5.23
4	4.67 5.58	4.49 5.39	4.62 5.30	4.68 5.25	4.68 5.22	4.69 5.22	4.71 5.22
8	5.31 6.00	4.85 5.43	4.77 5.31	4.73 5.24	4.72 5.21	4.72 5.21	4.71 5.22
16	6.81 7.41	5.41 5.83	5.03 5.45	4.85 5.27	4.75 5.22	4.74 5.21	4.74 5.21
20	7.81 8.45	5.82 6.12	5.22 5.56	4.95 5.29	4.78 5.23	4.75 5.21	4.75 5.21

Average MSE vs Lambda

Fraction = 0.03

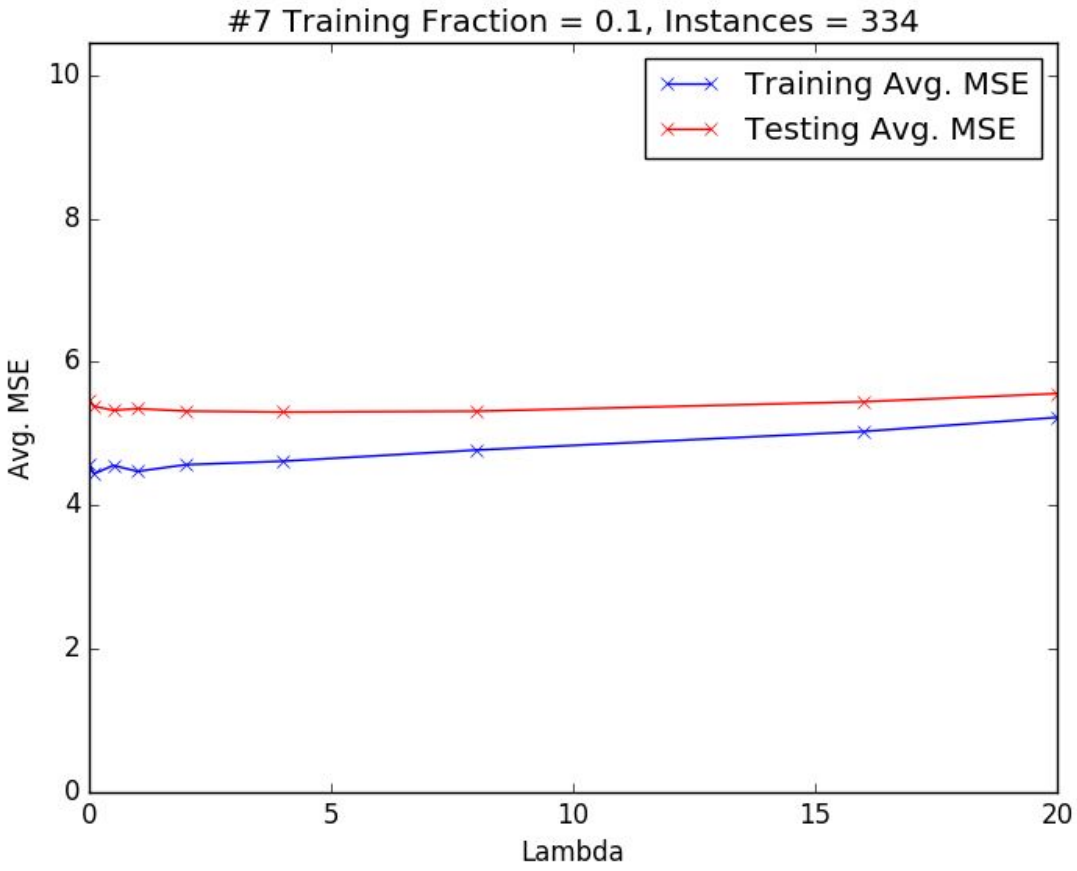
Observation: For such small fraction Training MSE increases as lambda increases, but Testing MSE decreases for small lambda.

Fraction = 0.06



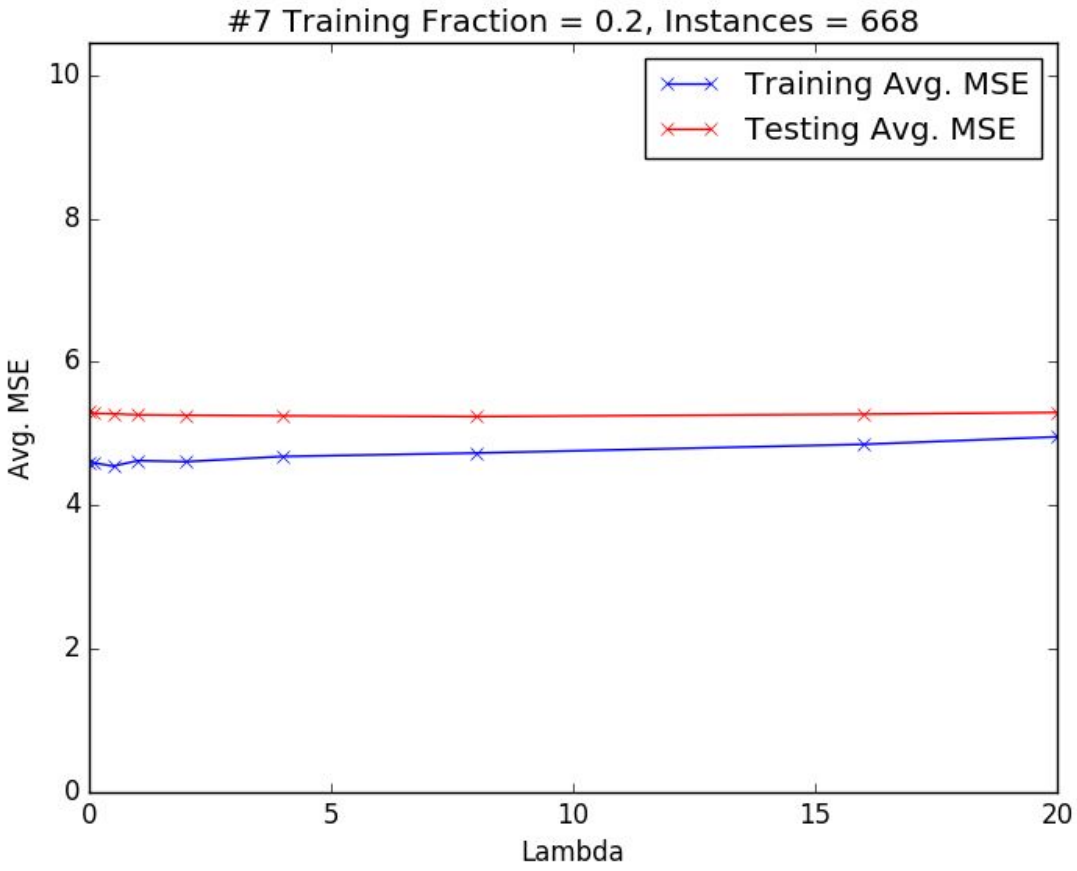
Observation: For small(not very small) fraction both Training and Testing MSE decreases for small lambda which means overfitting is reduced by penalizing weights.

Fraction = 0.1



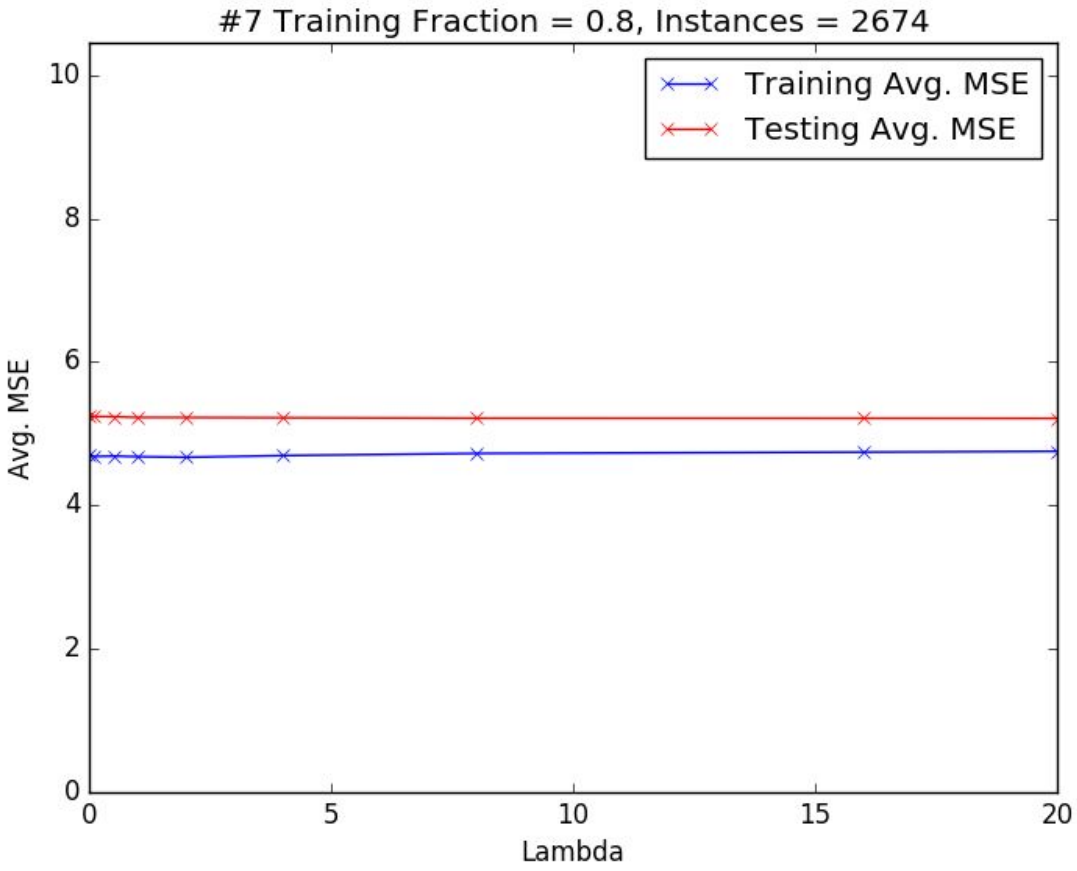
Observation: For small fraction both Training and Testing MSE decreases for small lambda which means overfitting is reduced by penalizing weights.

Fraction = 0.2



Observation: Both Training and Testing MSE remains almost constant for different λ.

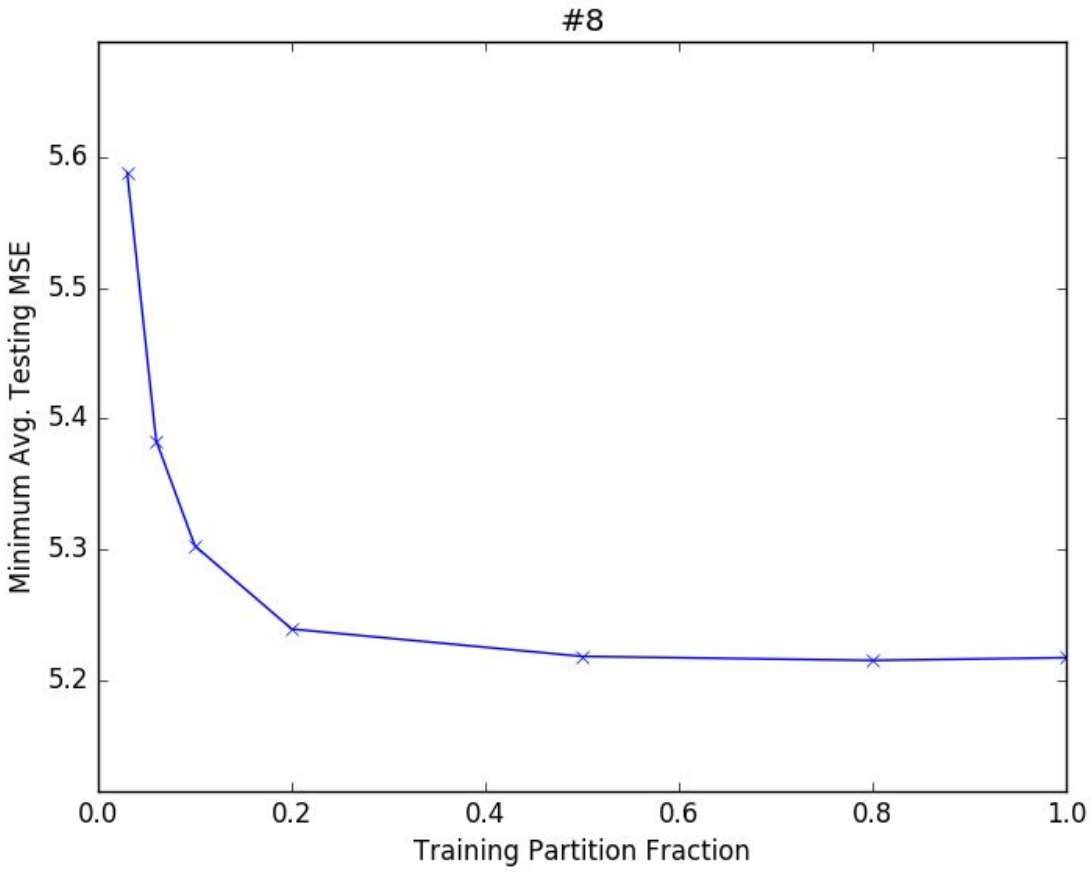
Fraction = 0.8



Observation: For large fractions both Training and Testing MSE remains almost constant for different λ.

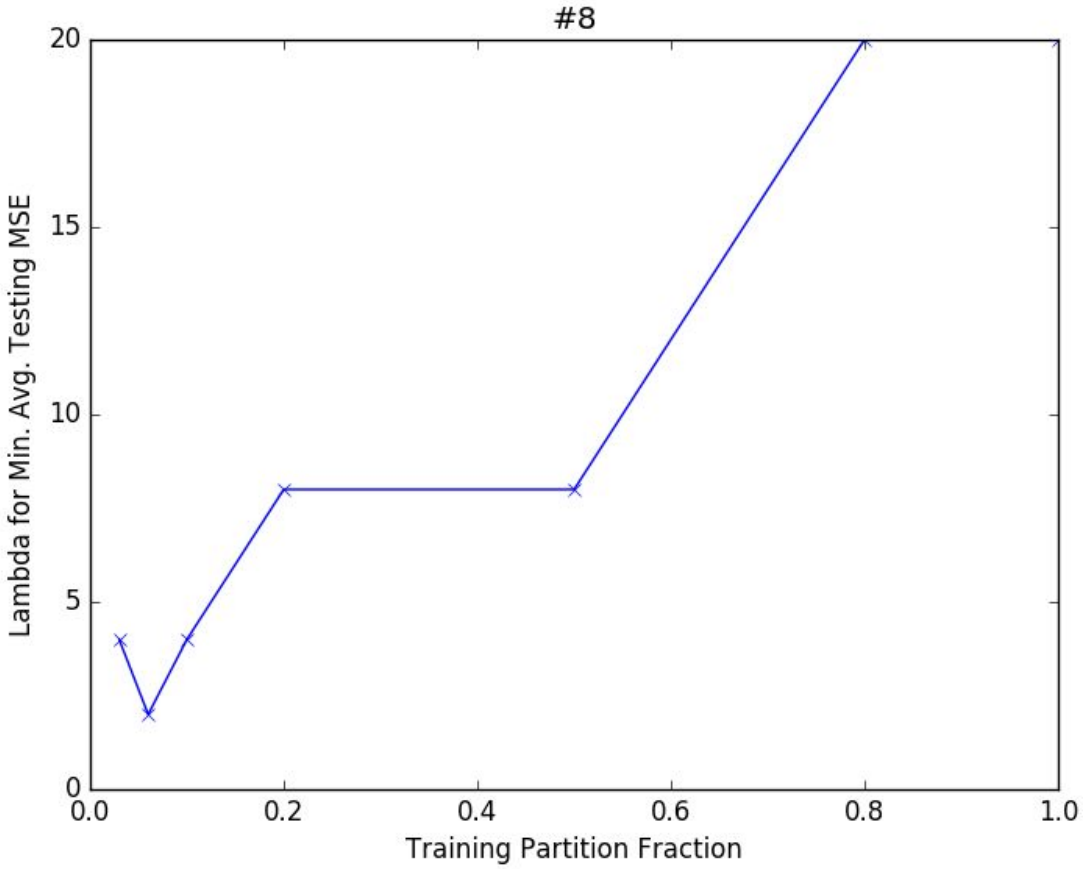
The above figures provide some insight, but is not very clear.

Graph - Minimum average mean squared testing error versus the training set fraction values



Observation: We can clearly see that as training partition fraction increases, Minimum Avg. Testing Error decreases.

Graph - The λ value that produced the minimum average mean squared testing error versus the training set fraction

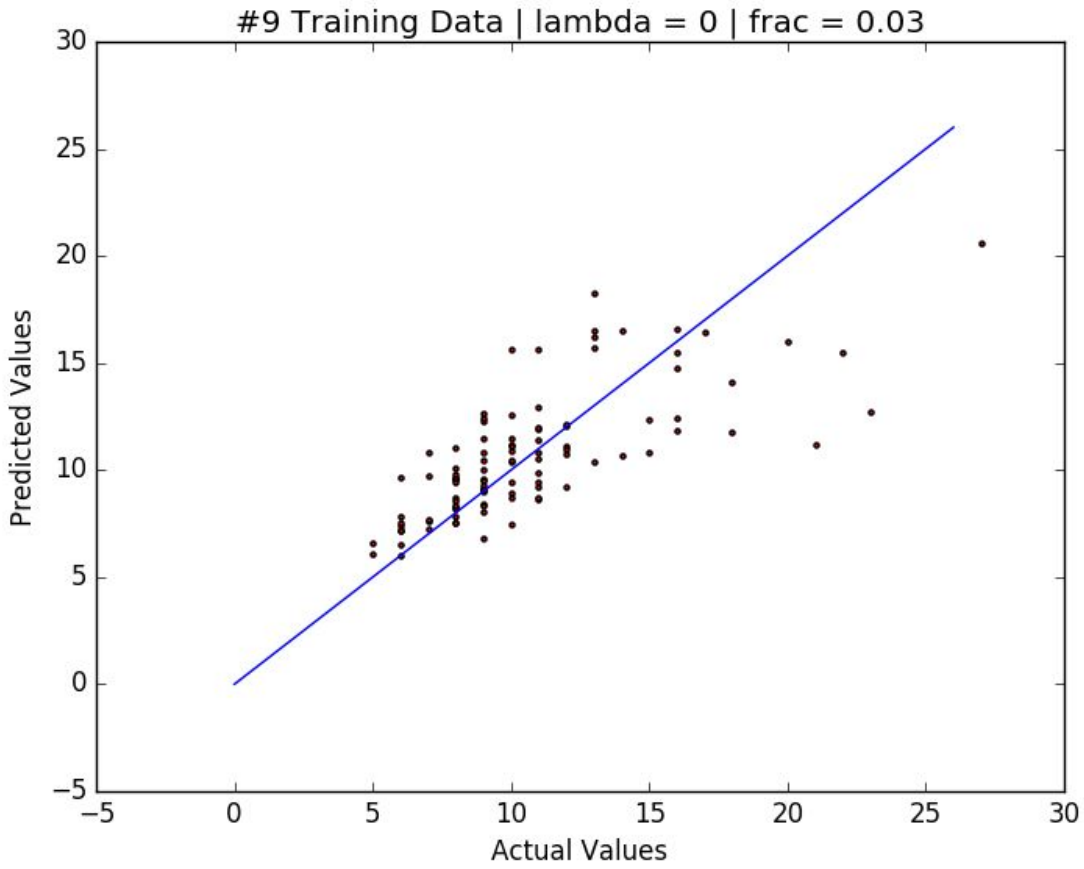


Observation: We can clearly see that the λ for Min. Avg. Testing MSE increases as partition fraction increases, because there is more penalty required for more training data to decrease the high weights.

To know if we have learned a good model:

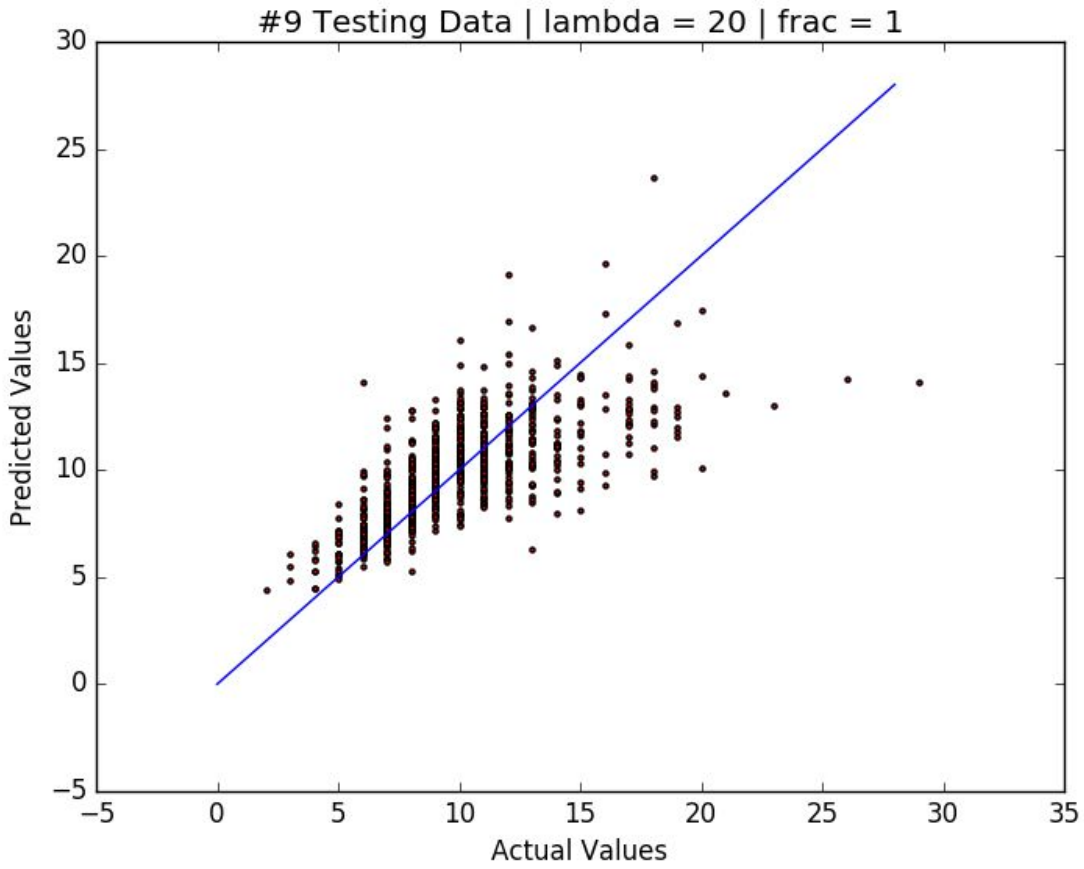
Graph - The predicted values vs the actual values for training data

Fraction and λ corresponding to min avg MSE taken.



Graph - The predicted values vs the actual values for testing data

Fraction and λ corresponding to min avg MSE taken.



If the model is good, then all the points will be close to a 45-degree line through the plot. But we can see there are many points which are very far from the line. Also all other points are not much close to the line. Therefore this model is not bad nor very good.

Regularized Logistic Regression

Introduction

Implemented regularized logistic regression with both gradient descent and newton raphson methods.

- Cost Function:

$$J(w) = \sum_{i=1}^{N} (y_i log(f(x_i) + (1 - y)_i log(1 - f(x_i)) + \lambda ||w||_2^2$$

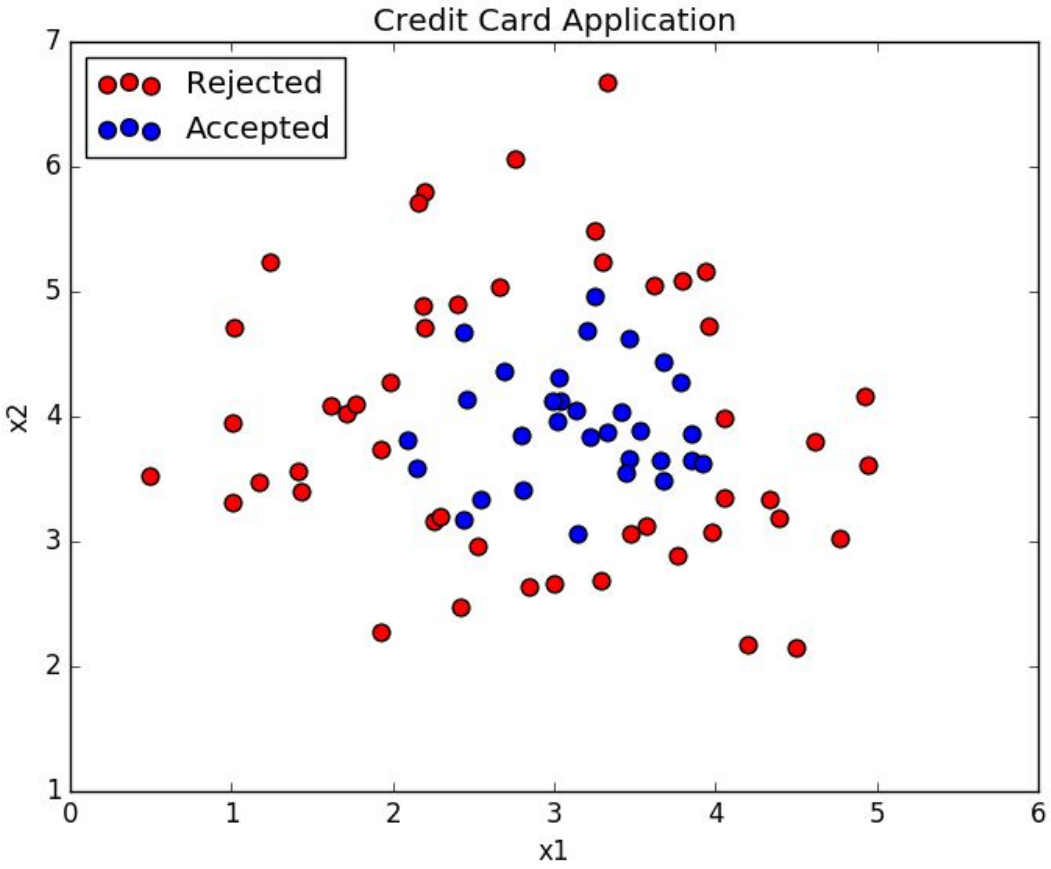
- Gradient Descent Learning parameter(α) = 0.0001
- Gradient Descent/Newton Raphson Stopping Criteria:

  i) Maximum iterations = 10^3

  ii) Difference between previous and current cost < 10^{-7}

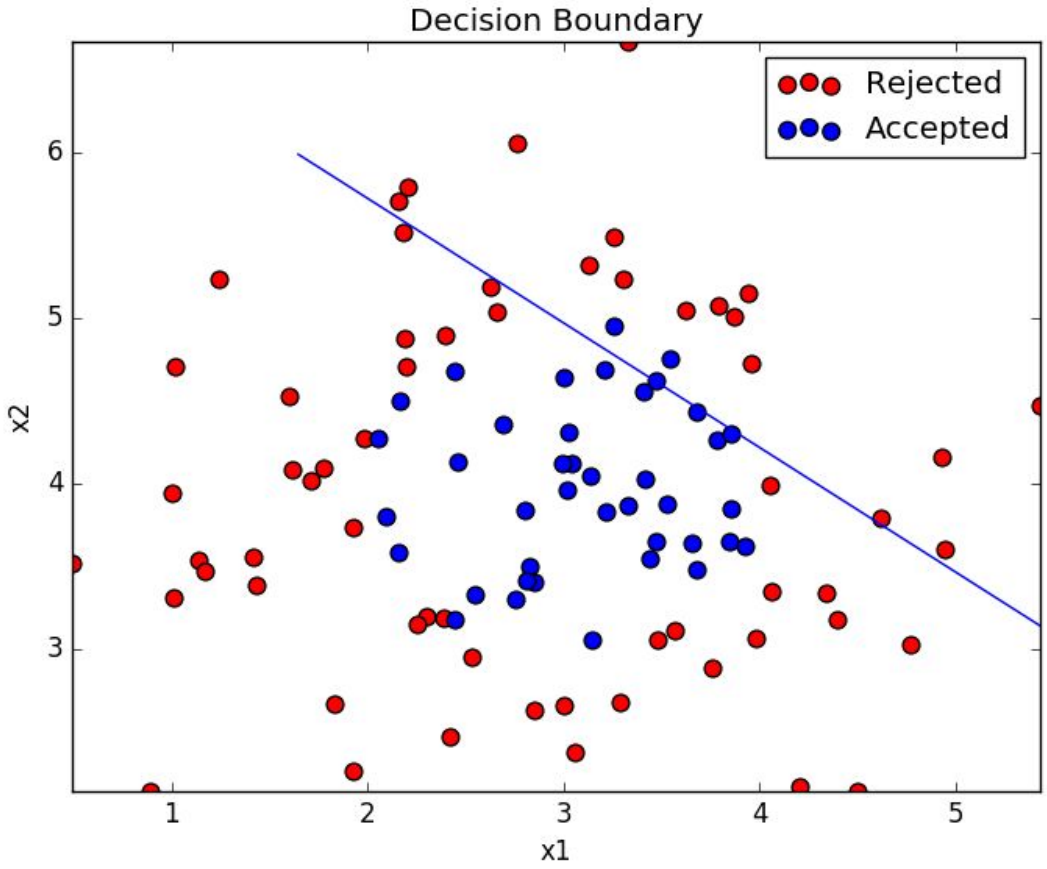
- Data partitioned into training and testing (80%-20%).

Training Dataset:



Observation: We can see data is not linearly separable.

Decision Boundary if directly logistic regression applied:



Newton Raphson with 3 iterations:

Training Accuracy: 50%

Testing Accuracy: 35%

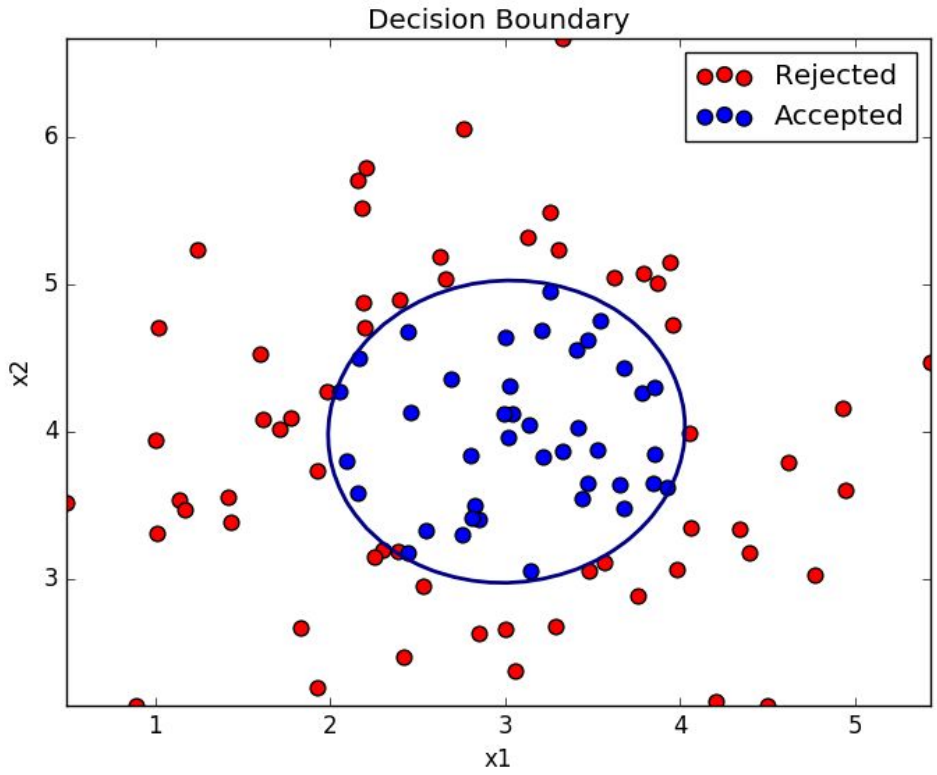
Gradient Descent with 1000 iterations:

Training Accuracy: 47.5%

Testing Accuracy: 40%

Observation: We can see data is not linearly separable.

After Feature Transform with degree = 2 and $\lambda = 0$:



Newton Raphson:

Training Accuracy: 100%

Testing Accuracy: 95%

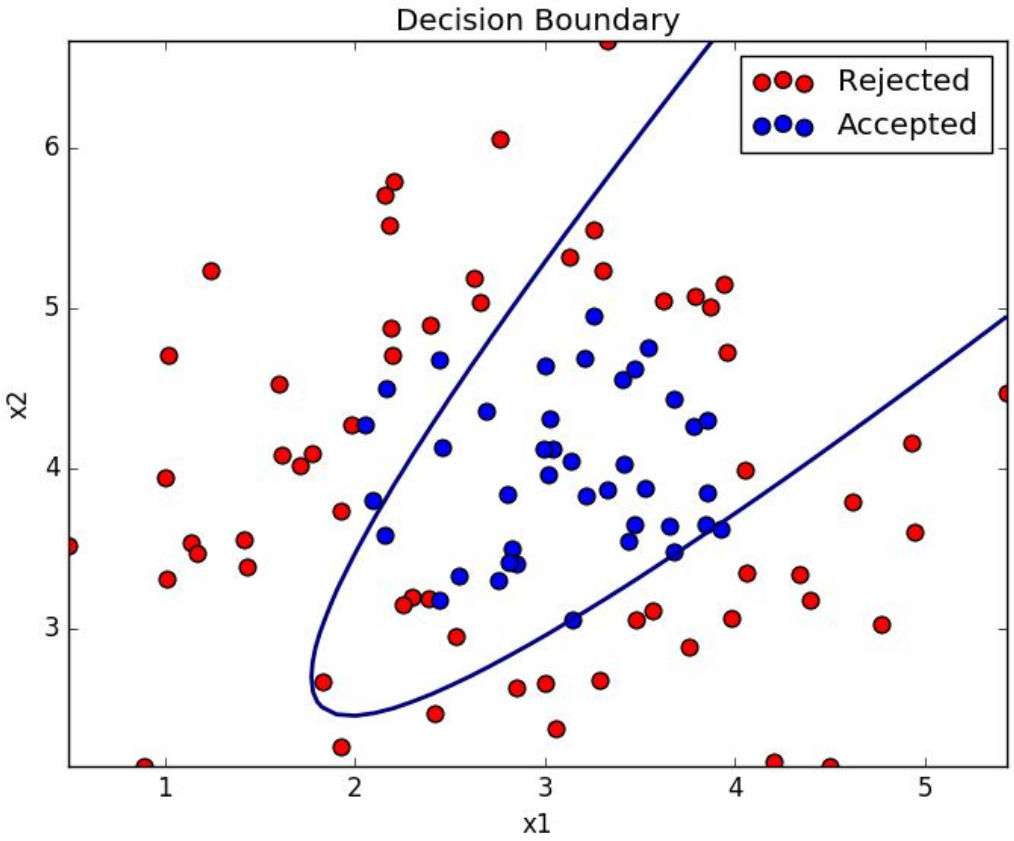
No. of iterations: 9

Gradient Descent:

Training Accuracy: 77.5%

Testing Accuracy: 75%No. of iterations = 1000

After Feature Transform with degree = 2 and $\lambda = 0.1$:



Newton Raphson:

Training Accuracy: 82.5%

Testing Accuracy: 70%

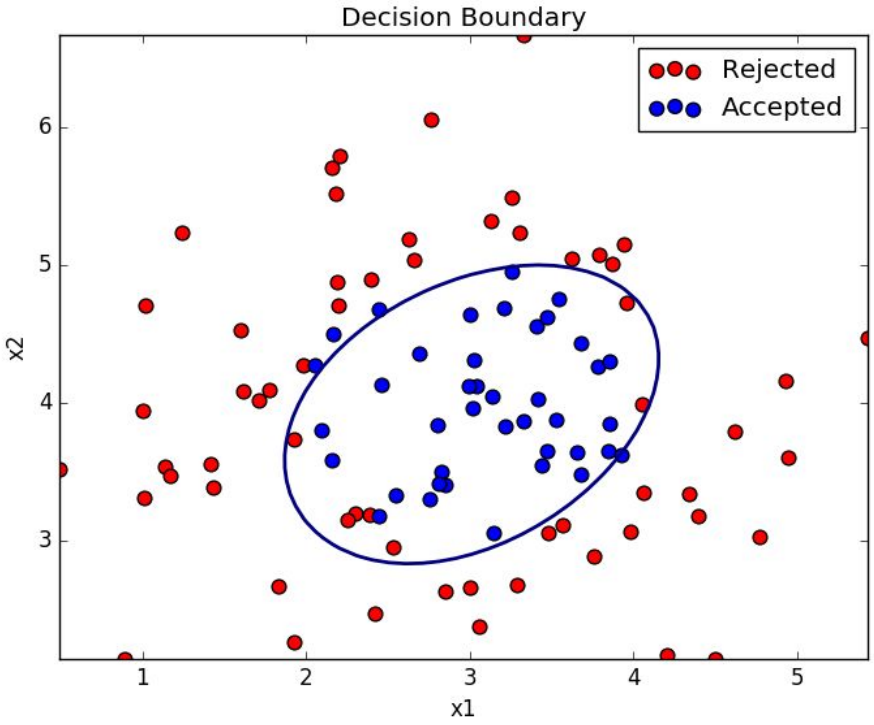
No. of iterations: 6

Gradient Descent:

Training Accuracy: 83.75%

Testing Accuracy: 70.0% No. of iterations = 1000

After Feature Transform with degree = 3 and $\lambda = 0.1$:



Newton Raphson:

Training Accuracy: 86.5%

Testing Accuracy: 100%

No. of iterations: 7

Gradient Descent:

Training Accuracy: 61.25%

Testing Accuracy: 85%No. of iterations = 1000

Observation: Overfitting can be avoided with regularized regression.

Newton Raphson vs Gradient Descent

Newton Raphson converges very fast than gradient descent and accuracy of Newton Raphson is also better than gradient descent.

Process to decide on the appropriate degree of the transformation:

Started with 1 degree and $\lambda = 0$

For degree = 2, $\lambda = 0$: Training accuracy is 100%, but testing accuracy is 95%.

For degree = 2, $\lambda = 0.1$: Training accuracy is 82.5% and testing accuracy is 70%

For degree = 3, $\lambda = 0.1$: Training accuracy is 86.5% and testing accuracy is 100%.

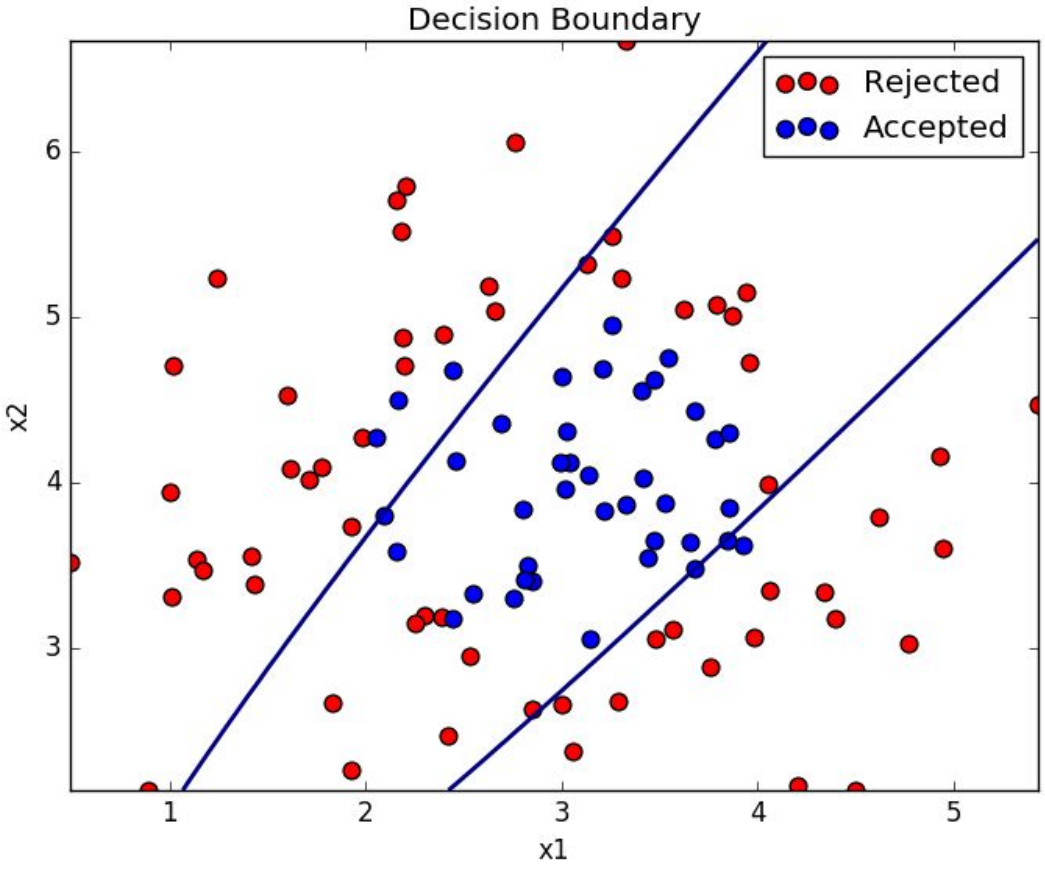
It means for degree = 2 and $\lambda = 0$ model is overfitted. Therefore degree = 3, $\lambda = 0.1$ is a better option.

Case of Underfitting:

Degree = 2 and $\lambda = 2$

Training Accuracy = 77.5%

Testing Accuracy = 75.0%



Case of Overfitting:

Degree = 2 and $\lambda = 0$

Training Accuracy = 100%

Testing Accuracy = 95%