Problem Set 3

NAME: Harsh Bhate GTID: 903424029

Problem	Points
Problem 1	
Problem 2	
Problem 3	
Problem 4	
Total	

Problem 1

Model	**Optimized Parameter**	**MSE on Test Set**	**Data** (Train, Test, Validate)
Linear	None	24.4836	(400, 106, 0)
Ridge	$\lambda = 10.7$	23.9856	(300, 106, 100)
LASSO	$\alpha = 0.1$	23.3346	(300, 106, 100)

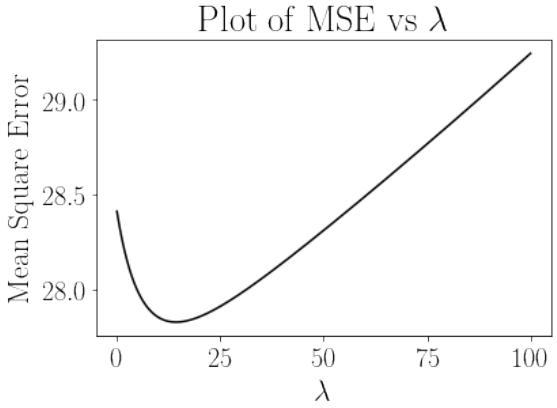
Non-zero Components in LASSO

The number of non-zero components in LASSO was found to be 11.

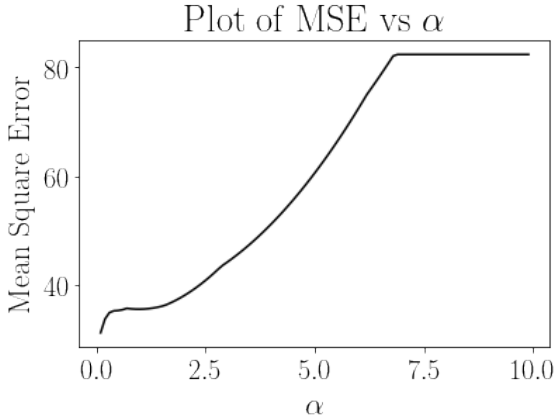
Optimization

To find the optimal parameters for Ridge Regression and LASSO, we split the training set to get a holdout or validation set. About 25% of the training set is allocated towards validation set. The optimal parameter was found by iterating over a range of parameter values. For each value, the model was trained on the training set and evaluated on validation set using MSE.The parameter value corresponding to lowest MSE on validation setwas considered optimal. The subsequent optimization curves are shown below:

Optimization Curve for Ridge Regression



Optimization Curve for LASSO Regression



Code

```
from sklearn.datasets import load_boston
import numpy as np
boston = load_boston()
x = boston.data
y = boston.target

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,
                                                    y,
                                                    train_size=400,
                                                    random_state=13)

from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(x_train)
x_train_std = scaler.transform(x_train)

X = np.concatenate([np.ones((x_train_std.shape[0], 1)), x_train_std], axis=1).astype(
X_test = np.concatenate([np.ones((x_test.shape[0], 1)), scaler.transform(x_test)], a


theta_opt = np.matmul(np.matmul(np.linalg.inv(np.matmul(np.transpose(X), X)), np.tran
                      y_train.astype(np.float))

from sklearn.metrics import mean_squared_error
pred = np.matmul(X_test, theta_opt)
error = mean_squared_error(y_test, pred)
print ("MSE for Least Square Model = %.4f"%(error))

def Gamma(d, Lambda):
    """
        Function to generate Gamma
    """
    I = np.identity(d)
    O_00 = 0
    O_01 = np.zeros((1,d))
    O_10 = np.zeros((d,1))
    gamma = np.block([[O_00, O_01],
                      [O_10, Lambda*I]]);
    return gamma

def ridgeRegression(X,Y, Lambda):
    """
        Function to find optimal weights for ridge regression
    """
    d = X.shape[1]-1
    gamma = Gamma(d, np.sqrt(Lambda))
    X_inv = np.linalg.inv(np.matmul(np.transpose(X), X) + np.matmul(np.transpose(gamm
```

```
48      theta_opt = np.matmul(np.matmul(X_inv, np.transpose(X)), Y)
49      return theta_opt
50
51  def evaluateRidgeRegression(X,Y,theta):
52      """
53          Function to evaluate performance of ridge Regression using MSE
54      """
55      pred = np.matmul(X, theta)
56      error = mean_squared_error(Y, pred)
57      return error
58
59  X_train, X_val, Y_train, Y_val = train_test_split(X,
60                                                    y_train,
61                                                    test_size=0.30,
62                                                    random_state=13)
63
64  errors = []
65  Lambdas = np.arange(0.1,100,0.1).tolist()
66  for Lambda in Lambdas:
67      theta = ridgeRegression(X_train, Y_train, Lambda)
68      error = evaluateRidgeRegression(X_val, Y_val, theta)
69      errors.append(error)
70
71  import matplotlib.pyplot as plt
72  from matplotlib import rc
73  rc('text', usetex=True)
74  rc('xtick', labelsize=20)
75  rc('ytick', labelsize=20)
76  font = {'family' : 'serif',
77          'size'   : 22}
78  rc('font', **font)
79
80  plt.figure()
81  plt.plot(Lambdas, errors, 'k')
82  plt.xlabel(r'$\lambda$')
83  plt.ylabel(r'Mean Square Error')
84  plt.title(r"Plot of MSE vs $\lambda$")
85  plt.show()
86
87  Lambda_opt = Lambdas[errors.index(min(errors))]
88  print ("The optimal value of lambda is %.4f"%Lambda_opt)
89  print ("The value of MSE corresponding to minimum Lambda is %.4f"%min(errors))
90  theta_opt = ridgeRegression(X_train, Y_train, Lambda_opt)
91  error = evaluateRidgeRegression(X_test, y_test, theta_opt)
92  print ("MSE for Ridge Regression Model = %.4f"%(error))
93
94
95  X_train, X_val, Y_train, Y_val = train_test_split(x_train_std,
```

```
                                                    y_train,
                                                    test_size=0.3,
                                                    random_state=13)

from sklearn import linear_model
errors = []
alphas = np.arange(0.1,10,0.1).tolist()
for alpha in alphas:
    reg = linear_model.Lasso(alpha)
    reg.fit(X_train,Y_train)
    pred = reg.predict(X_val)
    error = mean_squared_error(Y_val, pred)
    errors.append(error)
# Plotting the error
plt.figure()
plt.plot(alphas, errors, 'k')
plt.xlabel(r'$\alpha$')
plt.ylabel(r'Mean Square Error')
plt.title(r"Plot of MSE vs $\alpha$")
plt.show()
# Finding the Optimal alpha
alpha_opt = alphas[errors.index(min(errors))]
print ("The optimal value of alpha is %.4f"%alpha_opt)
print ("The value of MSE corresponding to minimum alpha is %.4f"%min(errors))
# Finding Optimal Model
reg_opt = linear_model.Lasso(alpha_opt)
reg_opt.fit(X_train,Y_train)

pred = reg_opt.predict(scaler.transform(x_test))
error = mean_squared_error(y_test, pred)
print ("MSE for LASSO = %.4f"%(error))
```

Problem 2

The LASSO model is described by:

$$\begin{aligned}\hat{\theta} &= \min_{\theta} ||y - X\theta||_2^2 + \lambda||\theta||_1 \\ &= \min_{\theta} y^T y - 2y^T X\theta + \theta^T X^T X\theta + \lambda||\theta||_1\end{aligned} \tag{1}$$

The Elastic-net regularizer is given by:

$$\begin{aligned}\hat{\theta} &= \min_{\theta} ||y - X\theta||_2^2 + \lambda(\alpha||\theta||_2^2 + (1-\alpha)||\theta||_1) \\ &= \min_{\theta} y^T y - 2y^T X\theta + \theta^T X^T X\theta + \lambda\alpha\theta^T\theta + \lambda(1-\alpha)||\theta||_1 \\ &= \min_{\theta} y^T y - 2y^T X\theta + \theta^T X^T X\theta + \theta^T(\lambda\alpha\mathbb{I})\theta + \lambda(1-\alpha)||\theta||_1 \\ &= \min_{\theta} y^T y - 2y^T X\theta + \theta^T X^T X\theta + \theta^T(\lambda\alpha\mathbb{I})\theta + \lambda||\theta||_1 - \lambda\alpha\mathbb{1}^T\theta \\ &= \min_{\theta} y^T y - (2y^T X + \lambda\alpha\mathbb{1}^T)\theta + \theta^T(X^T X + \lambda\alpha\mathbb{I})\theta + \lambda||\theta||_1 \\ &= \min_{\theta} \left|\left| \begin{pmatrix} y \\ 0 \end{pmatrix} - \begin{pmatrix} X \\ \sqrt{\lambda\alpha}\mathbb{I} \end{pmatrix} \theta \right|\right|_2^2 + \lambda||\theta||_1\end{aligned}$$

Thus, upon observing the results obtained above and those of [1], we conclude that elastic net is a stabilized version of *LASSO*. Mathematically,

$$\boxed{\hat{\theta}_{elastic} = \min_{\theta} ||\hat{y} - \hat{X}\theta||_2^2 + \lambda||\theta||_1 \text{ where } \hat{y} = \begin{pmatrix} y \\ 0 \end{pmatrix} \text{ and } \hat{X} = \begin{pmatrix} X \\ \sqrt{\lambda\alpha}\mathbb{I} \end{pmatrix}}$$

Problem 3

(a).

Ridge Regression

λ	**Slope**	**Intercept**
1.25	0.304	0.259

(b).

Linear Fit using Huber Loss

ϵ	α	**Slope**	**Intercept**
1.0000	0.0010	0.3276	0.2608
1.0000	0.0258	0.3069	0.2613
1.0000	0.0505	0.3267	0.2608
1.0000	0.0753	0.3262	0.2608
1.0000	0.1000	0.3256	0.2608
1.0625	0.0010	0.3411	0.2580
1.0625	0.0258	0.3408	0.2581
1.0625	0.0505	0.3406	0.2582
1.0625	0.0753	0.3403	0.2583
1.0625	0.1000	0.3401	0.2584
1.1250	0.0010	0.3683	0.2479
1.1250	0.0258	0.3679	0.2481
1.1250	0.0505	0.3675	0.2483
1.1250	0.0753	0.3672	0.2484
1.1250	0.1000	0.3668	0.2486
1.1875	0.0010	0.3726	0.2474
1.1875	0.0258	0.3723	0.2476
1.1875	0.0505	0.3719	0.2477
1.1875	0.0753	0.3716	0.2479
1.1875	0.1000	0.3712	0.2481
1.2500	0.0010	0.3582	0.2577
1.2500	0.0258	0.3578	0.2578
1.2500	0.0505	0.3575	0.2580
1.2500	0.0753	0.3572	0.2582
1.2500	0.1000	0.3568	0.2583

(c).

Support Vector Regression

C	ϵ	**Slope**	**Intercept**
1.0000	0.0010	0.3095	0.2603
1.0000	0.0258	0.3206	0.2704
1.0000	0.0505	0.3618	0.2562
1.0000	0.0753	0.3956	0.2361
1.0000	0.1000	0.3876	0.2354
1.0625	0.0010	0.3095	0.2603
1.0625	0.0258	0.3206	0.2704
1.0625	0.0505	0.3618	0.2562
1.0625	0.0753	0.3956	0.2361
1.0625	0.1000	0.3876	0.2354
1.1250	0.0010	0.3095	0.2603
1.1250	0.0258	0.3206	0.2704
1.1250	0.0505	0.3618	0.2562
1.1250	0.0753	0.3956	0.2361
1.1250	0.1000	0.3904	0.2332
1.1875	0.0010	0.3095	0.2603
1.1875	0.0258	0.3206	0.2704
1.1875	0.0505	0.3630	0.2555
1.1875	0.0753	0.3956	0.2361
1.1875	0.1000	0.3968	0.2280
1.2500	0.0010	0.3095	0.2603
1.2500	0.0258	0.3206	0.2704
1.2500	0.0505	0.3618	0.2562
1.2500	0.0753	0.3956	0.2361
1.2500	0.1000	0.4019	0.2264

Problem 4

(a). The log likelihood for logistic regression is given by:

$$l(\theta) = \sum_{i}^{N} y_i \theta^T x_i - log(1 + e^{\theta} x_i)$$

The error function in our case is:

$$E(\theta) = -l(\theta) + \frac{\lambda}{2}||\theta||_2^2$$

The gradient descent update rule for regularized logisic regression is:

$$\theta(t+1) = \theta(t) - \eta \nabla E(\theta)$$

We now compute $\nabla E(\theta)$ as follows:

$$\begin{aligned}
\nabla E(\theta) &= \nabla\Big(- l(\theta) + \frac{\lambda}{2}||\theta||_2^2\Big) \\
&= \nabla\Bigg(- \Big(\sum_{i}^{N} y_i \theta^T x_i - log(1 + e^{\theta} x_i)\Big) + \frac{\lambda}{2}||\theta||_2^2\Bigg) \\
&= -\nabla \sum_{i}^{N} y_i \theta^T x_i - log(1 + e^{\theta} x_i) + \frac{\lambda}{2}\nabla||\theta||_2^2 \\
&= -\Big(\sum_{i}^{N} y_i x_i - \frac{e^{\theta^T x_i}}{1 + e^{\theta^T x_i}} x_i) + \lambda\theta \\
&= \sum_{i}^{N} \frac{e^{\theta^T x_i}}{1 + e^{\theta^T x_i}} x_i - y_i x_i + \lambda\theta
\end{aligned} \tag{2}$$

Thus, the gradient update rule is:

$$\boxed{\theta^{t+1} = \theta^t - \eta\left(\sum_{i}^{N} \frac{e^{\theta^T x_i}}{1 + e^{\theta^T x_i}} x_i - y_i x_i + \lambda\theta \right)}$$

Or,

$$\boxed{\theta^{t+1} = (1 - \eta\lambda)\theta^t - \eta\left(\sum_{i}^{N} (g_\theta(x_i) - y_i) x_i \right)}$$

where $g(s)$ is the logistic function.

(b). The update rule using Newton-Rhapson method is:

$$\theta(t+1) = \theta(t) - \eta[\nabla^2(E(\theta))]^{-1}\nabla(E(\theta))|_{x=x_t} \tag{3}$$

The derivative of a logistic regression function is:

$$g'(s) = g(s)(1-g(s)) \tag{4}$$

We now compute the Hessian using results obtained in [2] and [4] as follows:

$$\begin{aligned} \nabla^2(E(\theta)) &= \nabla\Big(\nabla E(\theta)\Big) \\ &= \nabla\Big(\sum_i^N g(\theta^T x_i)x_i - y_i x_i + \lambda\theta\Big) \\ &= \sum_i^N g(\theta^T x_i)(1-g(\theta^T x_i))x_i x_i^T + \lambda \end{aligned} \tag{5}$$

Let $g(\theta^T x_i) = g_\theta(x_i)$. Substituting [5] in [3], we get,

$$\boxed{\theta(t+1) = \theta(t) - \eta\Big[\sum_i^N g_\theta(x_i)(1-g_\theta(x_i))x_i x_i^T + \lambda\Big]^{-1}\Big(\sum_i^N (g_\theta(x_i) - y_i)x_i + \lambda\theta\Big)}$$