Week 3 assignment

Name: Tianze Zhang
Student number: 19334401
Dataset id: id: 1--2-1

(i)(a) Figure 1.0 shows an overview of the entire dataset, where the first feature x1 on the x-axis, the second feature x2 in the y-axis and the target y on the z-axis. The training data looks like a curve, as Figure 1.0 since x1 in the range -1.0 and 1.0, for all x1 is low, the z axis (y) starts low, but keep increasing and is at its highest point when x1 is 0, and the z axis(y) decreases back as x1 reaches to 1.0. Therefore, ends up in a quadratic curve.

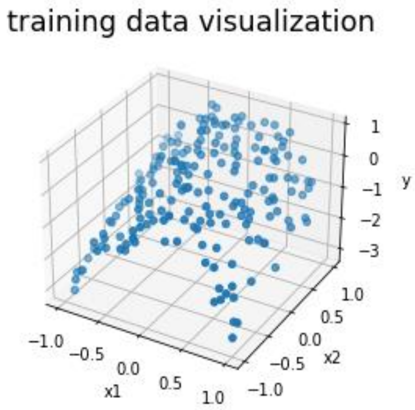

Figure 1.0

(b) As the trained model in the code, when C is low, C=1, all of the regression's coefficient is 0. Therefore the model is written as:
0+0*x0+0*x1+0*x0*x1+0*x1^2+0*x0^3+0*x0^2*x1+0*x0*x1^2+0*x1^3+0*x0^4+0*x0^3*x1+0*x0^2*x1^2+0*x0*x1^3+0*x1^4+0*x0^5+0*x0^4*x1+0*x0^3*x1^2+0*x0^2*x1^3+0*x0*x1^4+0*x1^5

When C=10, the regression coefficients are all 0 as well, except the 3rd and 4th feature which are 0.7 and -0.94 respectively, such as 0+0*x0+0*x1+0.7*x0*x1-0.94*x1^2+0*x0^3+0*x0^2*x1+0*x0*x1^2+0*x1^3+0*x0^4+0*x0^3*x1+0*x0^2*x1^2+0*x0*x1^3+0*x1^4+0*x0^5+0*x0^4*x1+0*x0^3*x1^2+0*x0^2*x1^3+0*x0*x1^4+0*x1^5

When C = 1000 the 2nd,3rd,4th,8th,9th,11th,12th,13th,21th coefficients are -5.9,9.7,-1.71,-6.77,-3.14,-9.08,-6.93,4.13,-6 respectively, such as 0+-5.9*x0+9.7*x1+0.7*x0*x1-1.71*x1^2-6.77*x0^3+0*x0^2*x1+0*x0*x1^2+0*x1^3+-3.14*x0^4+9.08*x0^3*x1+0*x0^2*x1^2+0*x0*x1^3-6.93*x1^4+4.13*x0^5+0*x0^4*x1+0*x0^3*x1^2+0*x0^2*x1^3+0*x0*x1^4-6*x1^5

As you can see above, more and more features are added on due to increasing of the penalty parameter C. This is because for lasso regression J(θ) = 1 m ∑m i=1(hθ(x (i)) − y (i)) 2 + L1, when C is too small, this makes L1 penalty too large, which increases the prediction error, hence losing too many features which ends up in underfitting.

However, while C increases, more and more features will be included since the L1 penalty increases, hence less prediction error.

(c)

Figure 1.1, Figure 1.2 and Figure 1.3 shows lasso regression trained with C=1, C=1/10 and C=1/1000 respectively. As you can see in 1.1, when C is 1, all predictions are the same, resulting in a flat surface which is different from the curve from part (a), this is caused by underfitting. In figure 1.2, when C is 10, the graph results in a quadratic curve shape which is really close to the training data in part 1. When C = 1000, the overall shape is still a curve, but missing some similarity to the part a scatter plot, this is due to overfitting, which included some noise, and resulting in wrong predictions.

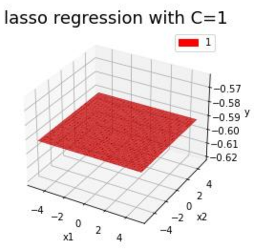

Figure 1.1

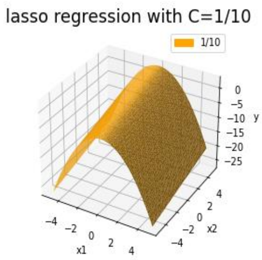

Figure 1.2

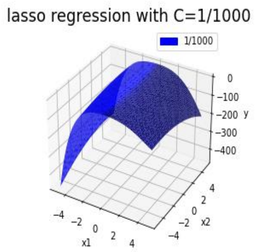

Figure1.3

(d) overfitting occurs when the model includes too many features, eventually including noise as well, which results in some misprediction. Underfitting on the other hand, occurs when the model didn't include enough features, which makes the model useless, also resulting in a low prediction rate.

As discussed before, since the lasso regression is $J(\theta) = 1\ m\ \sum m\ i=1(h\theta(x\ (i)) - y\ (i))\ 2 + L1$, where L1 is $C \sum n\ j=1\ |\theta j\ |$. Increasing C results in more feature but large C causes underfitting and small C cause underfitting. for example, C=1 has no feature, C=10 has a few, but C=1000 has too many, this is proved in the graph, where 1.1 is a flat surface, every prediction is the same, and not useful which is underfitting. While figure 1.2 is closest to the original data, but figure 1.4 misclassifies some part when x1 is between 0-4 which is caused by overfitting.

(e) For ridge regression, trained with C=1, C=1/10 and C=1/1000. For ridge regression with features: '1', 'x0', 'x1', 'x0^2', 'x0 x1', 'x1^2', 'x0^3', 'x0^2 x1', 'x0 x1^2', 'x1^3', 'x0^4', 'x0^3 x1', 'x0^2 x1^2', 'x0 x1^3', 'x1^4', 'x0^5', 'x0^4 x1', 'x0^3 x1^2', 'x0^2 x1^3', 'x0 x1^4', 'x1^5',

The parameters for C=1 are 0, -0.04319639 ,0.90191908, -1.28217778 ,0.01188395,0.06939358,0.07153439, 0.0324441, -0.06096806 ,0.14535317, -0.66336823, -0.10244669, -0.24869514, -0.01518832, -0.02794864, 0.02275903, -0.04584508,0.07559615, -0.02493795, -0.01755684, -0.06481303.

The parameters for C=10 are 0, -0.03515364,0.98514597, -1.59350132,0.06331884,0.08098896,0.13659457, -0.03506544, -0.16457173,0.04747506, -0.41145724, -0.18832001, -0.14645583, 0.00406137, -0.08567604, -0.06356894, -0.08239294, 0.19225232,0.10144249,0.00305155, -0.07479905.

The parameters for C=1000 are [0.00000000e+00, -3.11334282e-02,1.02458236e+00, -1.67711958e+00,7.79869464e-02,7.93540973e-02,1.66305718e-01, -9.25338551e-02, -2.06539915e-01, -5.95035414e-02, -3.32085266e-01, -2.24365145e-01,-1.15426905e-01, 2.27439014e-02, -1.01677688e-01, -1.08899902e-01, -7.75419473e-02 , 2.72414464e-01 1.79270718e-01, -1.86277774e-02,2.85279086e-04.

Ridge regression with C=1, C=10 and C=100 are shown below.

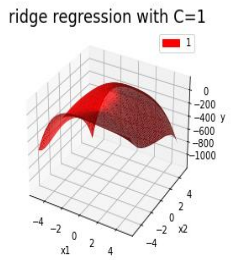

Figure 1.4

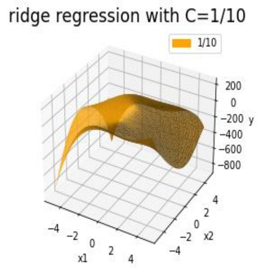

Figure 1.5

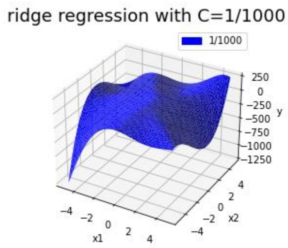

Figure 1.6

In contrast to lasso regression, ridge regression's cost function is $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \theta^T\theta/C$ where $\theta^T\theta$ is L2. L1 regularization penalizes the sum of absolute values of the weights, whereas L2 regularization penalizes the sum of squares of the weights. This means ridge regression has smaller effect when dealing with small C values. This is shown in the parameters, where all lasso regression parameters is 0 when C=1 while ridge regression has some value, also shown in figure 1.4, the graph is closer to a curve than figure 1.1. However, ridge regression might result in more overfitting than lasso, in Figure 1.5 and 1.6 the top of the curve is less smooth than Figure 1.2 and 1.3, hence more overfitting.

(ii)(a) Figure 2.0 shows the MSE of lasso regression as Ci increases. The C value chosen was 0.1, 0.5, 1, 5, 10, 50, 100, which increases by factor of 10 this time. This is a rule of thumb on how to chose C, which ensure a wider range of C is chosen, hence more accurate result.

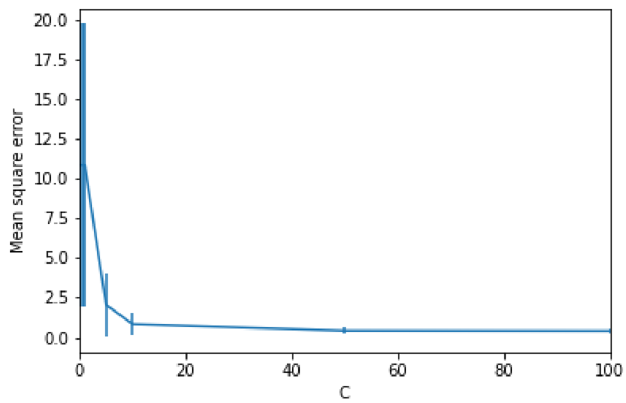

Figure 2.0

(b) As Figure 2.0, you can see that when C is really small, the prediction is really inaccurate (High MSE) due to underfitting. However, when C increases, the model performs better which the mean square error decreases. The model is stable and predicts most accurate for C>10, since the MSE stop changing when C increases, and it's at the lowest.

(c) Figure 2.1 shows C vs MSE of ridge regression with the same C value range.

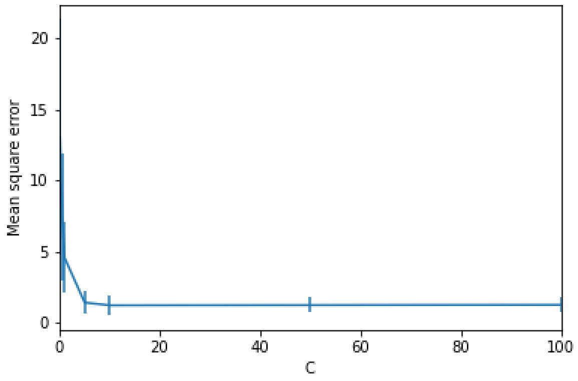

Figure 2.1

As Figure 2.1, you can see that ridge regression follows the same pattern as lasso regression, where the model is performing the worst at the start when C is 0, and when C>10, the MSE is lowest and started to become stable. In compared to lasso in Figure 2.0, it has less MSE when C=0 and C=5 and 10. But when C=50, it has a higher MSE. Which shows the difference between L1 and L2 penalty discussed above.

Appendix

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model import Lasso

from sklearn.linear_model import Ridge

import matplotlib.patches as mpatches

from sklearn.model_selection import KFold

from sklearn.metrics import mean_squared_error

df = pd.read_csv('week3.txt')
```

```
x1 = df.iloc[:,0]

x2 = df.iloc[:,1]

X=np.column_stack((x1,x2))

y = df.iloc[:,2]


fig = plt.figure()

ax = fig.add_subplot(111,projection ='3d')

ax.scatter(X[:,0],X[:,1],y)

ax.set_xlabel('x1')

ax.set_ylabel('x2')

ax.set_zlabel('y')

plt.title("training data visualization",fontsize=18)

plt.savefig('training-data.png', facecolor='w', transparent=False)

plt.show()

#add extra polynomial features

poly_features = PolynomialFeatures(5)

poly_X = poly_features.fit_transform(X)

poly_features.get_feature_names()

#lasso regression with C=1/1000

lasso = Lasso(alpha=1/1000)

lasso.fit(poly_X,y)

#get paramters

print(lasso.intercept_)

print(lasso.coef_)

#lasso regression with C=1/10

lasso1 = Lasso(alpha=1/10)

lasso1.fit(poly_X,y)

print(lasso1.intercept_)

print(lasso1.coef_)

#lasso regression with C=1

lasso2 = Lasso(alpha=1)
```

```
lasso2.fit(poly_X,y)
print(lasso2.intercept_)
print(lasso2.coef_)
Xtest =[]
grid=np.linspace(-5,5)
for i in grid :
  for j in grid:
    Xtest.append([i,j])
Xtest = np.array(Xtest)
poly_features = PolynomialFeatures(5)
poly_X_test = poly_features.fit_transform(Xtest)
# prediction from lasso regression with C=1/1000
y_predict = lasso.predict(poly_X_test)
fig = plt.figure()
ax = fig.add_subplot(111,projection ='3d')
ax.plot_trisurf(Xtest[:,0],Xtest[:,1],y_predict,color='blue')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('y')
plt.title("lasso regression with C=1/1000",fontsize=18)
blue_patch = mpatches.Patch(color='blue', label='1/1000')
plt.legend(handles=[blue_patch])
plt.savefig('lassoReg1000.png', facecolor='w', transparent=False)
plt.show()
#lasso regression with C=1/10
y_predict1 = lasso1.predict(poly_X_test)
fig = plt.figure()
ax = fig.add_subplot(111,projection ='3d')
ax.plot_trisurf(Xtest[:,0],Xtest[:,1],y_predict1,color='orange')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
```

```
ax.set_zlabel('y')


plt.title("lasso regression with C=1/10",fontsize=18)
orange_patch = mpatches.Patch(color='orange', label='1/10')
plt.legend(handles=[orange_patch])
plt.savefig('lassoReg10.png', facecolor='w', transparent=False)
plt.show()
#lasso regression with C=1
y_predict2 = lasso2.predict(poly_X_test)
fig = plt.figure()
ax = fig.add_subplot(111,projection ='3d')
ax.plot_trisurf(Xtest[:,0],Xtest[:,1],y_predict2,color='red')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('y')
plt.title("lasso regression with C=1",fontsize=18)
red_patch = mpatches.Patch(color='red', label='1')
plt.legend(handles=[red_patch])
plt.savefig('lassoReg1.png', facecolor='w', transparent=False)
plt.show()
#ridge regression with C=1/1000
rdg = Ridge(alpha=1/1000)
rdg.fit(poly_X,y)
#get paramters
print(rdg.intercept_)
print(rdg.coef_)
#ridge regression with C=1/10
rdg1 = Ridge(alpha=1/10)
rdg1.fit(poly_X,y)
#get paramters
print(rdg1.intercept_)
```

```
print(rdg1.coef_)
#ridge regression with C=1
rdg2 = Ridge(alpha=1)
rdg2.fit(poly_X,y)
#get paramters
print(rdg2.intercept_)
print(rdg2.coef_)
# prediction from ridge regression with C=1/1000
y_predict_rdg = rdg.predict(poly_X_test)
fig = plt.figure()
ax = fig.add_subplot(111,projection ='3d')
ax.plot_trisurf(Xtest[:,0],Xtest[:,1],y_predict_rdg,color='blue')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('y')
plt.title("ridge regression with C=1/1000",fontsize=18)
blue_patch = mpatches.Patch(color='blue', label='1/1000')
plt.legend(handles=[blue_patch])
plt.savefig('ridgeReg1000.png', facecolor='w', transparent=False)
plt.show()
# prediction from ridge regression with C=1/10
y_predict_rdg1 = rdg1.predict(poly_X_test)
fig = plt.figure()
ax = fig.add_subplot(111,projection ='3d')
ax.plot_trisurf(Xtest[:,0],Xtest[:,1],y_predict_rdg1,color='orange')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('y')
plt.title("ridge regression with C=1/10",fontsize=18)
orange_patch = mpatches.Patch(color='orange', label='1/10')
plt.legend(handles=[orange_patch])
```

```
plt.savefig('ridgeReg10.png', facecolor='w', transparent=False)
plt.show()
# prediction from ridge regression with C=1
y_predict_rdg2 = rdg2.predict(poly_X_test)
fig = plt.figure()
ax = fig.add_subplot(111,projection ='3d')
ax.plot_trisurf(Xtest[:,0],Xtest[:,1],y_predict_rdg2,color='red')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('y')
plt.title("ridge regression with C=1",fontsize=18)
red_patch = mpatches.Patch(color='red', label='1')
plt.legend(handles=[red_patch])
plt.savefig('ridgeReg1.png', facecolor='w', transparent=False)
plt.show()
x = np.arange(0, 1, 0.05).reshape(-1, 1)
y = 10*x + np.random.normal(0.0, 1.0, x.size).reshape(-1, 1)
mean_error = []
std_error = []
C_range = [0.1, 0.5, 1, 5, 10, 50, 100]
for C in C_range:
    model = Lasso(alpha=1/C)
    temp = []
    kf = KFold(n_splits=5)
    for train, test in kf.split(x):
        model.fit(x[train], y[train])
        ypred = model.predict(x[test])
        temp.append(mean_squared_error(y[test], ypred))
    mean_error.append(np.array(temp).mean())
    std_error.append(np.array(temp).std())
plt.errorbar(C_range, mean_error, yerr=std_error)
```

```
plt.xlabel("C")
plt.ylabel("Mean square error")
plt.xlim((0, 100))
plt.savefig('meansquarelasso.png', facecolor='w', transparent=False)
plt.show()
x = np.arange(0, 1, 0.05).reshape(-1, 1)
y = 10*x + np.random.normal(0.0, 1.0, x.size).reshape(-1, 1)
mean_error = []
std_error = []
C_range = [0.1, 0.5, 1, 5, 10, 50, 100]
for C in C_range:
   model = Ridge(alpha=1/C)
   temp = []
   kf = KFold(n_splits=5)
   for train, test in kf.split(x):
      model.fit(x[train], y[train])
      ypred = model.predict(x[test])
      temp.append(mean_squared_error(y[test], ypred))
   mean_error.append(np.array(temp).mean())
   std_error.append(np.array(temp).std())
plt.errorbar(C_range, mean_error, yerr=std_error)
plt.xlabel("C")
plt.ylabel("Mean square error")
plt.xlim((0, 100))
plt.savefig('meansquarerdg.png', facecolor='w', transparent=False)
plt.show()
```