hw6 linear model (II) 2018012518

1.Given a Gaussian linear regression model, Maximum likelihood estimation of w under Gaussian noise assumption is equivalent to the least square loss. Please prove it.

we assume that $y_n|w,x_n\ N(x^Tx_n,\sigma^2)$

for point (x_n,y_n)

$p(y_n|w,x_n)=\frac{1}{\sqrt{2\pi\sigma}}exp\{-\frac{1}{2\sigma^2}(y_n-w^Tx_n)^2\}$

log-likelihood for linear regression on the whole dataset D_n:

$logp(D_n;w)=\sum_{n=1}^{N}logp(y_n|w,x_n)$

$=\frac{N}{2}log12\pi\sigma^2-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n-w^Tx_n)^2$

maximum likelihood estimation of w is

$argmax(w)\frac{N}{2}log12\pi\sigma^2-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n-w^Tx_n)^2$

$=argmax(w)\ \ -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n-w^Tx_n)^2$

$=argmin\sum_{n=1}^{N}(y_n-w^Tx_n)^2$

So maximum likelihood estimation of w under Gaussian noise assumption is equivalent to the least square loss.

2.Given a Laplacian linear regression model, Maximum likelihood estimation of w under Laplacian noise assumption is equivalent to absolute loss minimizer. Please prove it.

if we assume that $y_n|w,x_n\ Laplace(x^Tx_n,b)$,

for point (x_n,y_n),

$$p(y_n|w,x_n)=\frac{1}{2b}exp\{-\frac{|y_n-w^Tx_n|}{b}\}$$

the maxinum likelihood estimation for w:

$$argmax(w)\sum_{n=1}^{N} logp(y_n|w, x_n) = argmin(w)\sum_{n=1}^{N} |y_n - w^T x_n|$$

so laplacian noise assumption leads to absolute loss minimizer.

3.Given a linear regression model, please write down the Tikhonov Form and Ivanov Form of Ridge Regression, and these two forms of Lasso Regression as well.

for ridge regression:

Tikhonov Form:
the ridge regression solution for regularization parameter $\lambda \geq 0$ is

$$\hat{w} = \ argmin(w \in R^d)\frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda||w||_2^2$$

Ivanov Form:
the ridge regression solution for complexity parameter$r \geq 0is$

$$\hat{w} = argmin(\parallel w \parallel_2^2 \leq r)\frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2$$

Lasso Regression
Tikhonov Form:
the lasso regression solution for regularization parameter $\lambda \geq 0is$

$$\hat{w} = argmin(w \in R^d)\frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda \parallel w \parallel_1$$

$where \parallel w \parallel_1 = |w_(1)| + \ + |w_(d)| is the square of l_2 - norm$

Ivanov Form:
the lasso regression solution for complexity parameter $r \geq 0is$

$$\hat{w} = argmin(\parallel w \parallel_2^2 \leq r)\frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2$$

4.By adding a Ridge Regression in the linear regression model of Question

4 in hw5-linear-model, can we get a lower generalization error? If yes, use cross validation to attain the best regularization parameter λ, whose possible values are [1.e-06, 1.e-05, 1.e-04, 1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03, 1.e+04, 1.e+05, 1.e+06]. If no, please explain why. See the tutorial of linear model in sklearn

hw5 code and answer:

```
kf = KFold(n_splits = 2)
for train_index,test_index in kf.split(X):
  print("TRAIN:",train_index,"TEST:",test_index)
  X_train,X_test=X[train_index],X[test_index]
  y_train,y_test=y[train_index],y[test_index]
model.fit(X_train,y_train)
array = [(model.predict(X_test)-y_test)*(model.predict(X_test)-y_test)]
sum = 0
for i in array[0]:
    sum = (sum + i)
accuracy = sum / len(X_test)
##accuracy衡量了“每个”预测值和实际值之间，对于“每单位”实际值平均的误差的平方
##
```

```
TRAIN: [50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
 98 99] TEST: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49]
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49] TEST: [50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
 98 99]

0.6996050279547127
```

Figure 1: hw5 code and answer

use ridge regression,we get ridge coefficients as a function of the regularization

```python
alphas = [1.e-06,1.e-05,1.e-04,1.e-03,1.e-02,1.e-01,1.e+00,1.e+01,
          1.e+02,1.e+03,1.e+04,1.e+05,1.e+06]
clf = linear_model.Ridge(fit_intercept=False)
coefs=[]
for a in alphas:
    clf.set_params(alpha=a)
    clf.fit(X,y)
    coefs.append(clf.coef_)

ax = plt.gca()

ax.plot(alphas, coefs)
ax.set_xscale('log')
ax.set_xlim(ax.get_xlim()[::-1])
plt.grid()
plt.xlabel('alpha')
plt.ylabel('weights')
plt.title('Ridge coefficients as a function of the regularization')
plt.axis('tight')
plt.show()
```

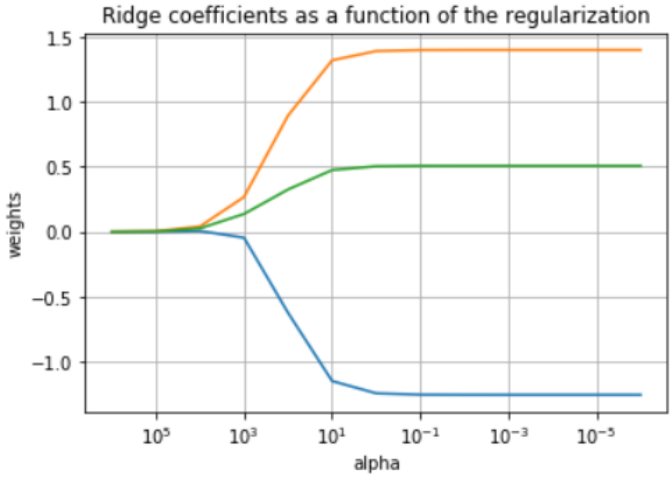

Figure 2: ridge

use cross validation to compute the error when λ is different and draw the graphs here is the ouput, the pictures are similar for all σ : now compute the error and output them we can see that when λ = 1.e-06, the error is minimum and smaller than the result of the linear regression

5.By adding a Lasso Regression in the linear regression model of Question 4 in hw5-linear-model, can we get a lower generalization error? If yes, use cross validation to attain the best regularization parameter σ, whose possible values are [1.e-06, 1.e-05, 1.e-04, 1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01,

```python
for a in alphas:
    y = df['y'].values
    reg = linear_model.Ridge(alpha=a)
    reg.fit(X,y)
    ##print(reg.coef_)
    ##print(reg.intercept_)
    plt.plot(X,reg.predict(X))
    plt.show()
    plt.plot(X,y)
    plt.show()
    array = [(reg.predict(X_test)-y_test)*
             (reg.predict(X_test)-y_test)]
    ##calculate the error
    ##array
    sum = 0
    for i in array[0]:
        sum = (sum + i)
        ##i
    accuracy = sum / len(X_test)
    ##error divided by the number of data_test
    print(accuracy)
```

Figure 3

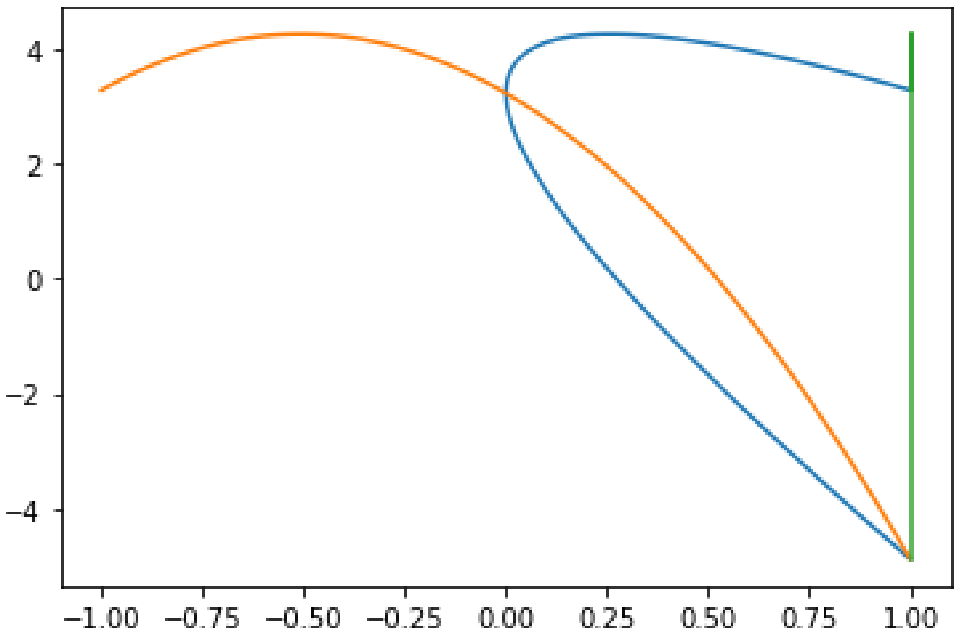

Figure 4

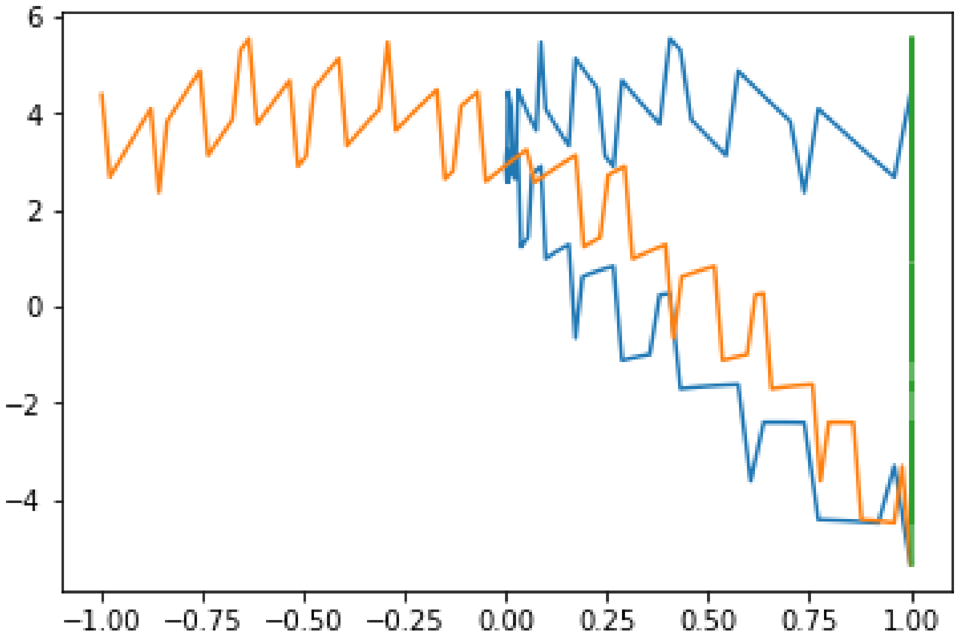

Figure 5

```
0.45016486934142014
0.4501648831479087
0.45016502149024157
0.45016643265378326
0.4501833136186504
0.4506244314801106
0.478231309562976
1.4846740201868605
6.781997345002201
9.90998973662976
10.367168825396252
10.414958362191257
10.419758921149324
```

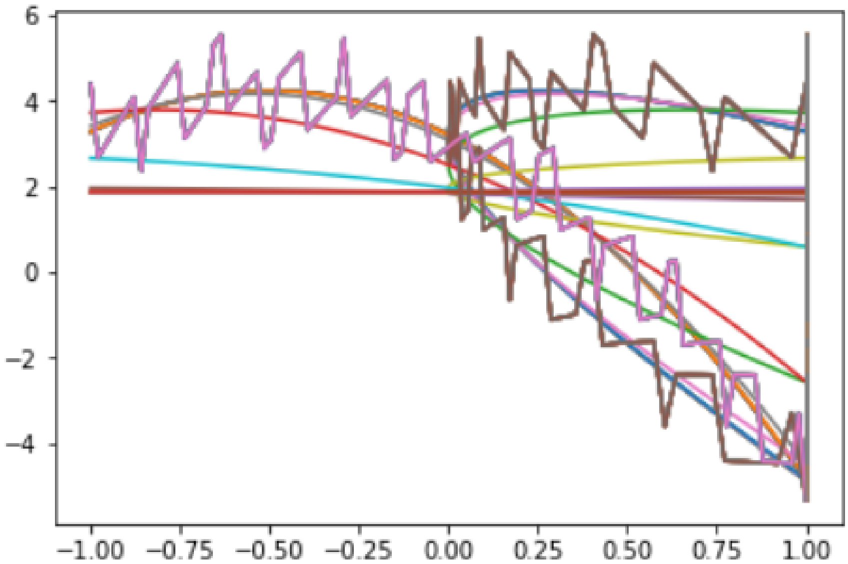

Figure 6

1.e+02, 1.e+03, 1.e+04, 1.e+05, 1.e+06]. If no, please explain why.
here shows the lasso coefficients as a function of the regularization given different λ show the graphs and compute the accuracy the graphs of different λ given are also similar: here is the compute result: the same as ridge regression,when λ = 1.e-06, the error is minimum and smaller than the result of the linear regression

```
]: clf = linear_model.Lasso(fit_intercept=False)
   coefs=[]
   for a in alphas:
       clf.set_params(alpha=a)
       clf.fit(X,y)
       coefs.append(clf.coef_)

   ax = plt.gca()

   ax.plot(alphas, coefs)
   ax.set_xscale('log')
   ax.set_xlim(ax.get_xlim()[::-1])
   plt.grid()
   plt.xlabel('alpha')
   plt.ylabel('weights')
   plt.title('Lasso coefficients as a function of the regulari
   plt.axis('tight')
   plt.show()
```

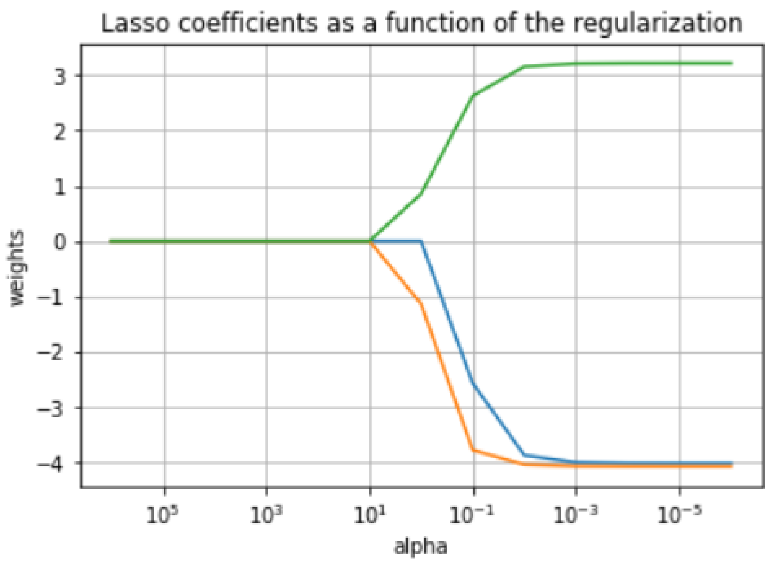

Figure 7

```
for a in alphas:
    y = df['y'].values
    reg = linear_model.Lasso(alpha=a)
    reg.fit(X,y)
    ##print(reg.coef_)
    ##print(reg.intercept_)
    plt.plot(X,reg.predict(X))
    plt.show()
    plt.plot(X,y)
    plt.show()
    array = [(reg.predict(X_test)-y_test)*
             (reg.predict(X_test)-y_test)]
    ##calculate the error
    ##array
    sum = 0
    for i in array[0]:
        sum = (sum + i)
        ##i
    accuracy = sum / len(X_test)
    ##error divided by the number of data_test
    print(accuracy)
```

Figure 8

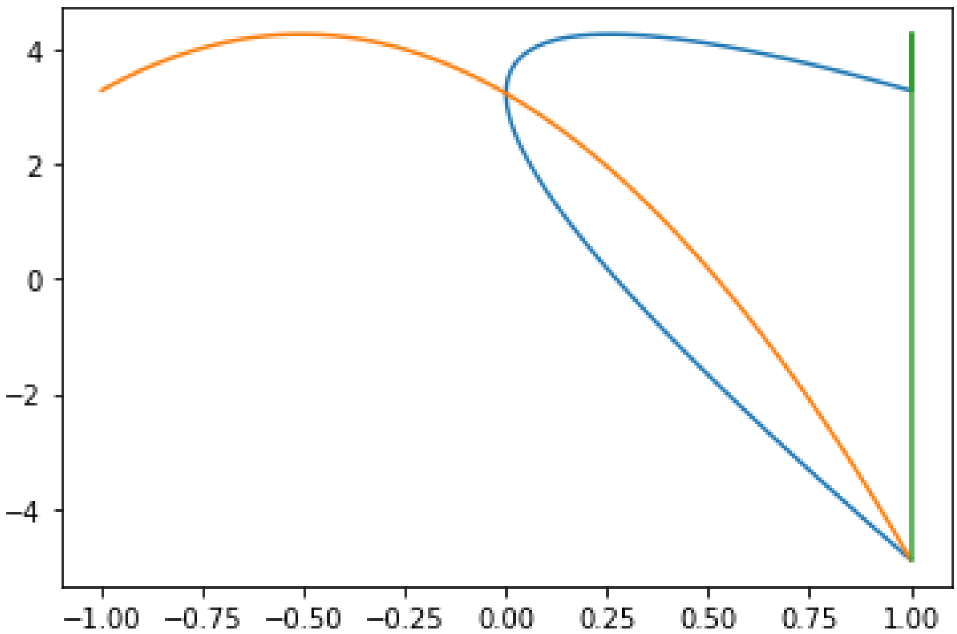

Figure 9

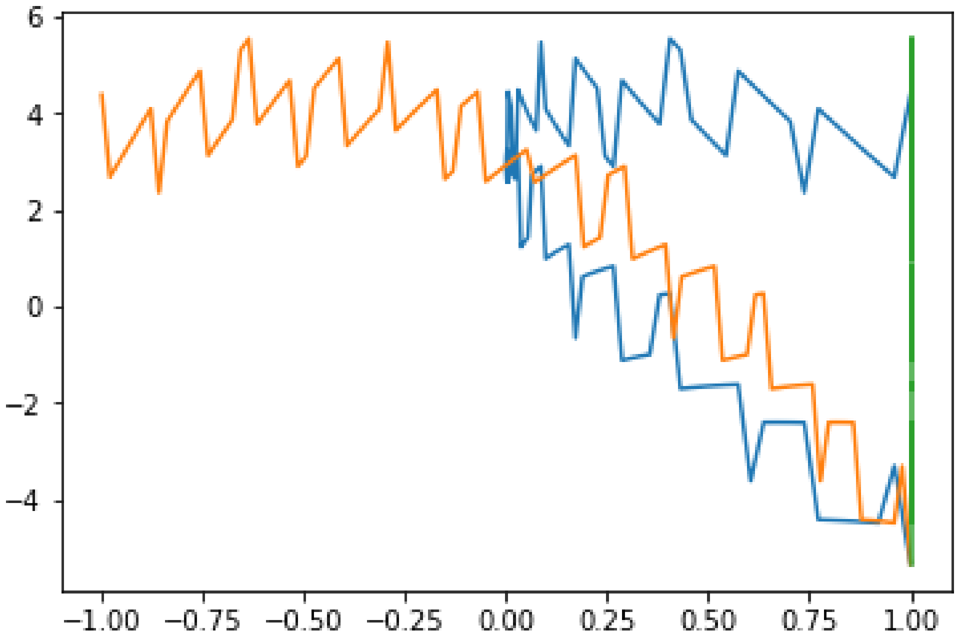

Figure 10

```
0.4501649060559865
0.45016525201982566
0.45016888283471507
0.4502223086336315
0.45246833162484074
0.6461050617414362
6.927399111050329
10.420292560112511
10.420292560112511
10.420292560112511
10.420292560112511
10.420292560112511
10.420292560112511
```

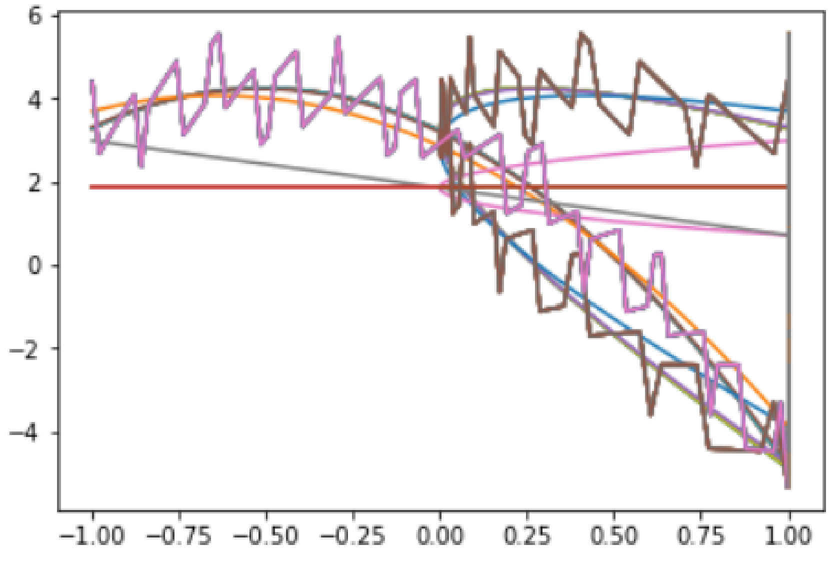

Figure 11