hw6 linear model (II)

1. Given a Gaussian linear regression model, Maximum likelihood estimation of ***w*** under Gaussian noise assumption is equivalent to ***least square loss minimization***.

$$\min_{\mathrm{w}} \sum_{n=1}^{N} (y_n - w^T x_n)^2$$

Please prove it.

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

$$= \frac{\mathrm{N}}{2} \log \frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - w^T x_n)^2$$

Maximum likelihood estimation of w is:

$$\max_{w} \frac{\mathrm{N}}{2} \log \frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - w^T x_n)^2$$

$$= \max_{\arg w} -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - w^T x_n)^2$$

$$= \min_{\mathrm{w}} \sum_{n=1}^{N} (y_n - w^T x_n)^2$$

To here, we have proved it!

2. Given a Laplacian linear regression model, Maximum likelihood estimation of ***w*** under Laplacian noise assumption is equivalent to ***absolute loss (L1 loss) minimizer.*** Please prove it.

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

The maximum likelihood estimation for w:

$$\max_{w} \sum_{n=1}^{N} \log p(y_n | w, x_n)$$

$$= \max_{\arg w} \sum_{n=1}^{N} \log \frac{1}{2b} \exp\left\{-\frac{|y_n - w^T x_n|}{b}\right\}$$

$$= \max_{arg\ w} \sum_{n=1}^{N} \log \exp\left\{-\frac{|y_n - w^T x_n|}{b}\right\}$$

$$= \max_{arg\ w} \sum_{n=1}^{N} -\frac{|y_n - w^T x_n|}{b}$$

$$= \min_{arg\ w} \sum_{n=1}^{N} \frac{|y_n - w^T x_n|}{b}$$

$$= \min_{arg\ w} \sum_{n=1}^{N} |y_n - w^T x_n|$$

To here, we have proved it!

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.

Ridge regression:

Tikhonov Form:

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

L2-norm

Ivanov Form:

$$\hat{w} = \arg \min_{||w||_2^2 \le r} \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

Lasso Regression:

Tikhonov Form:

$$\hat{w} = \arg \min_{w \in R^d} \frac{1}{n} \sum (w^T x_i - y_i)^2 + \lambda ||w||_1$$

L1-norm

Ivanov Form:

$$\hat{w} = \arg \min_{||w||_2^2 \le r} \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

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: https://scikit-learn.org/stable/modules/linear_model.html if you need some help.

(1) Importing the packages and read in data:

```
import numpy as np
from sklearn import linear_model
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.utils import shuffle

df = pd.read_csv('dataset.csv')
x1 = np.array(df['x1']).T
x2 = np.array(df['x2']).T
x3 = np.array(df['x3']).T
X = np.array([df['x1'], df['x2'], df['x3']]).T
y = df['y'].values

Alphas = np.logspace(-6, 6, 13)
```

(2) Setting the hyper-parameter of Ridge model from 1e-6 to 1e6, and use 'for' loop to find the corresponding mse value
(we use the mse values on the shuffled dataset to test the generalization ability of the Ridge model)

```
Alphas = np.logspace(-6, 6, 13)

Xs, y = shuffle(X, y, random_state=0)
for alpha in Alphas:
    reg = linear_model.Ridge(alpha=alpha)
    reg.fit(X, y)
    mse = cross_val_score(reg, Xs, y, cv=5, scoring='neg_mean_squared_error')
    res = [-each for each in mse]
    res = np.array(res)
    print(res.mean())

print('*' * 40)
```

(3) Result of MSEs:

```
1.6827707088946489
1.6827706911125329
1.6827705132968869
1.6827687356925076
1.6827510148452416
1.6825793152680693
1.6814026129907955
1.7144809327680597
3.1222088369008896
6.843267150878235
7.903975444985295
8.02788561049255
8.040482196862499
***************************************
```

(4) Here we can find out that **when** $\boldsymbol{\lambda = 1e0}$**,** the mse value **reaches its valley**. Besides, when $\boldsymbol{\lambda \leq 1e1}$, the mses on the cross-validation data is **relatively small**.

(5) Analysis:
The reason for our introducing λ is to prevent the w parameter from growing too big(**normalization**); and in the experiment, the function of hyper-parameter λ is clear:

The bigger λ means we want to make the ||w|| smaller (**constraining ||w||** is to optimize the generalization ability of the model)

However, in the experiment, we can also show that, bigger λ doesn't necessarily means the improved performance on validation dataset(too big will decrease the generalization ability).

```
[-0.02471858 -0.04884407 -0.16961218]
[-0.02471858 -0.04884407 -0.16961217]
[-0.02471858 -0.04884403 -0.16961205]
[-0.02471855 -0.04884366 -0.1696109 ]
[-0.02471821 -0.04884    -0.16959941]
[-0.02471482 -0.04880337 -0.1694845 ]
[-0.02468025 -0.04844008 -0.16834409]
[-0.02427392 -0.045081   -0.1577408 ]
[-0.01897214 -0.02655405 -0.09699781]
[-0.00498355 -0.00515883 -0.0201061 ]
[-0.00058569 -0.00056876 -0.00225421]
[-5.95974325e-05 -5.74622861e-05 -2.28190147e-04]
[-5.97020419e-06 -5.75215931e-06 -2.28470880e-05]
*****************************************
```

Answer:

Yes

1e0

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 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: https://scikit-learn.org/stable/modules/linear_model.html if you need some help.

(1) Importing the packages and read in data:

(2) Setting the hyper-parameter of Lasso model from 1e-6 to 1e6(**logspace**), and use loop to find the corresponding **mse (minimum squared error)** value
(we use the mse values on the shuffled dataset to test the generalization ability of the Lasso model)

```
Alphas = np.logspace(-6, 6, 13)
for alpha in Alphas:
    reg = linear_model.Lasso(alpha=alpha)
    reg.fit(X, y)
    mse = cross_val_score(reg, Xs, y, cv=5, scoring='neg_mean_squared_error')
    res = [-each for each in mse]
    res = np.array(res)
    print(res.mean())
```

(3) Result of MSEs:

```
****************************************
1.682770952909705
1.6827711190297596
1.6827698169308278
1.682758639651242
1.6828504043666623
1.7039465028664469
2.901760424051594
8.041884159707724
8.041884159707724
8.041884159707724
8.041884159707724
8.041884159707724
8.041884159707724
```

(4) Here we can find out that when $\lambda = 1\mathrm{e}-3$, the mse value reaches its valley.
Besides, when $\lambda \leq 1\mathrm{e}-1$, the mses on the cross-validation data is relatively small.

(5) Analysis:
The reason for our introducing λ is to prevent the learnable-parameter $\mathbf{w}$ from

growing too big(**normalization)**; and in the experiment, the function of hyper-parameter λ is clear:

The bigger λ means we want to **make the ||w|| smaller** (constraining ||w|| is to optimize the generalization ability of the model)

However, in the experiment, we can also prove that, bigger λ doesn't necessarily means the improved performance on validation dataset(too big will decrease the generalization ability)

Answer:

Yes

1e-3

Summary:

The goal of introducing the normalization hyper-parameter λ is to improve the generalization ability of the model.

But if λ grows too big, the generalization error rate may even increase. So we need to test and find the optimal hyper-parameter λ.