ML2024: Homework Assignment 1

Yukun Tian

58122315

Key Laboratory of New Generation Artificial Intelligence Technology

&Its Interdisciplinary Applications, Ministry of Education, China

March 2024

Codes included are all available at Github:

https : *//github.com/TTiannaiTT/ML2024Spring*

1 Problem 1

Solution:

1.

The matrix form of the optimization problem should be as follows:

$$\min_{\mathbf{w},\mathbf{b}} l(\mathbf{w},\mathbf{b}) = \frac{1}{2}\|(\mathbf{X}\mathbf{w}+\mathbf{b}-\mathbf{y})\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \tag{1}$$

$$s.t. \quad \lambda \geqslant 0 \tag{2}$$

$$\mathbf{w} \in \mathbf{R}^{d\times 1} \tag{3}$$

$$\mathbf{b} \in \mathbf{R}^{m\times 1} \tag{4}$$

$$\mathbf{X} \in \mathbf{R}^{m\times d} \tag{5}$$

$$\mathbf{y} \in \mathbf{R}^{m\times 1} \tag{6}$$

Definition:

$\mathbf{X}$: is the input matrix of m samples and d features.

$\mathbf{y}$: is the ground truth label of the input $\mathbf{X}$.

$\mathbf{w}$: is the parameter of the ridge regression model.

$\mathbf{b}$: is the parameter of the ridge regression model.

2.

Yes, the optimal parameter is absolutely unique for $\lambda > 0$.

Proof:

The function $l(\mathbf{w}, \mathbf{b})$ is obviously a convex function for the parameter $\mathbf{w}$ and $\mathbf{b}$, as the summation, norm and the quadratic function are both convex. So the optimal value for the parameters is unique. And we can even solve the closed form of the optimal value for the function $l(\mathbf{w}, \mathbf{b})$ by calculating the derivative.

3.

For this question, we can rewrite our loss function $l(\mathbf{w}, \mathbf{b})$ as follows:

$$l(\mathbf{w}, \mathbf{b}) = \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathbf{T}}(\mathbf{X}\mathbf{w} - \mathbf{y}) + 0.05\mathbf{w}^{\mathbf{T}}\mathbf{w} \tag{7}$$

This time we make the $\mathbf{W}$ equals to $(\mathbf{w}, b)$. And the new $\mathbf{X}$ denotes the $(\mathbf{X}, \mathbf{1})$

So for the new loss function, the derivative of it is:

$$\frac{\partial l(\mathbf{w}, \mathbf{b})}{\partial \mathbf{w}} = \mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathbf{T}}\mathbf{y} + 0.1\mathbf{w} \tag{8}$$

In order to get the optimal value, let the two derivatives equal to 0, and then we can get:

$$\mathbf{w}^* = (0.1\mathbf{E} + \mathbf{X}^{\mathbf{T}}\mathbf{X})^{-\mathbf{1}}\mathbf{X}^{\mathbf{T}}\mathbf{y} \tag{9}$$

And now, with the help of Matlab, we can use the dataset given to calculate the result:

$$\mathbf{W}^* = [0.5437; -0.6036; -0.1114; -0.0790] \tag{10}$$

$$\mathbf{w}^* = [0.5437; -0.6036; -0.1114] \tag{11}$$

$$b^* = -0.079 \tag{12}$$

The Matlab code is as follows:

```
% Calculate the optimal parameter for the ridge regression
X = [2 1 3 1; 5 3 6 1;4 2 5 1;3 5 2 1;1 7 2 1;6 1 4 1]
y = [0;0;0;-3;-3;3]
```

```
W = inv(0.1*ones(4)+transpose(X)*X)*X.'*y
W
```

And we can also use Matlab to check our answer:

```
% Checking your answer
a = [4,2,5] % A sample from the input dataset
W  =[0.5437;
-0.6036;
-0.1114]
result = a*W-0.079
```

4.

Proof:

To maximize the conditional distribution with the prior knowledge that the $\theta_j \sim N(0, \tau^2)$ and $\epsilon \sim N(0, \sigma^2)$, so we have:

$$y_i \sim N(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}_i, \sigma^2) \tag{13}$$

So by Bayesian estimation, we can rewrite the question as follows:

$$max\ p(\boldsymbol{\theta}|\mathbf{y}) \tag{14}$$

$$\Leftrightarrow max\ p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathbf{y}) \tag{15}$$

$$\Leftrightarrow max \prod_{i=1}^{m} p(\mathbf{y}_i|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{16}$$

$$\Leftrightarrow max \sum_{i=1}^{m} In\ p(\mathbf{y}_i|\boldsymbol{\theta}) + In\ p(\boldsymbol{\theta}) \tag{17}$$

Now, we can take the two Gaussian distribution into account and we can easily get that:

$$p(\mathbf{y}_i|\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma} exp[-\frac{1}{2}(\frac{\mathbf{y}_i - \boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}_i}{\sigma})^2] \tag{18}$$

$$p(\boldsymbol{\theta}_i) = \frac{1}{\sqrt{2\pi}\tau} exp[-\frac{1}{2}(\frac{\boldsymbol{\theta}_i}{\tau})^2] \tag{19}$$

Therefore, when we use 18 and 19 into 17, and we ignore the constant value. Then we will have Eq.(1.1):

$$\min \frac{1}{2}\sum_{i=1}^{m}(\mathbf{y}_i - \boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}_i)^2 + \frac{1}{\tau^2}\|\boldsymbol{\theta}\|_2^2 \tag{20}$$

Now the $\frac{\lambda}{2}$=$\frac{1}{\tau^2}$, and now the estimate of θ^* is equivalent to solving the Eq.(1.1) with b=0. □

□

2 Problem 2

Solution:

1.

Proof:

For the sigmoid function, to prove that it is non-convex. We can take that: $z_1 = 1$, $z_2 = 3$ and $\lambda = 0.5$. Then we have:

$$\lambda\sigma(z_1) + (1-\lambda)\sigma(z_2) \approx 0.841 \tag{21}$$

$$\sigma(\lambda z_1 + (1-\lambda)z_2) \approx 0.880 \tag{22}$$

So we have:

$$\lambda\sigma(z_1) + (1-\lambda)\sigma(z_2) < \sigma(\lambda z_1 + (1-\lambda)z_2) \tag{23}$$

So the sigmoid function is a non-convex function for z. And z is a linear function of β, so the composition of the both is a non-convex function for β.

To prove the Eq.(2.2) is convex for parameter β, we can just prove the second partcial derivative (Hessian matrix) of it is postive:

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{m}(-\hat{x}_i y_i + \frac{e^{\beta^{\mathbf{T}}\hat{x}_i}\hat{x}_i}{1+e^{\beta^{\mathbf{T}}\hat{x}_i}}) \tag{24}$$

By using the same method, we can calculate the Hessian matrix of $\frac{\partial^2 l(\beta)}{\partial \beta^2}$, and it's a positive definite matrix. So the overall function $l(\beta)$ is a convex function for β. □

2.

The Eq.(2.1) should rewrite as:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{i=1}^{n} e^{z_i}} \tag{25}$$

$$s.t. \qquad \mathbf{z} = \mathbf{W}\mathbf{x}_i + \mathbf{b} = \beta\mathbf{x}_i \tag{26}$$

In this situation, we define some principle to simplify our representation:

First, the y_i should be given as a vector in one-hot manner, which means for x_i that belongs

to the class j, its label y_i should be as follows:

$$\begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \tag{27}$$

Only the jth element in the vector is no-zero.

So the likelihood term is:

$$p(y_i|\beta) = y_i^{\mathbf{T}}[\sigma(\beta \mathbf{x_i})] \tag{28}$$

So the log-likelihood function is:

$$l(\beta) = ln(\prod_{i=1}^{m} y_i^{\mathbf{T}} p(y_i|\beta)) \tag{29}$$

$$= \sum_{i=1}^{m} In(y_i^{\mathbf{T}} p(y_i|\beta)) \tag{30}$$

$$= \sum_{i=1}^{m} In(y_i^{\mathbf{T}} \sigma(\beta \mathbf{x}_i)) \tag{31}$$

□

3 Problem 3

Solution:

1.

In order to calculate the AUC, we can use the format:

$$AUC = \frac{\sum (p_i, n_j)_{p_i > n_j}}{P * N} \tag{32}$$

P is the number of the positive samples, and N is negative samples. p_j and n_j represent the score or posibility the classifier predicts.

So now, with the help of 32, we can use the following code to calculate the AUC:

```
import numpy as np
from sklearn.metrics import roc_auc_score
```

```
# use python sklearn to calculate AUC
def get_auc(y_labels, y_scores):
        auc = roc_auc_score(y_labels, y_scores)
        print('AUC calculated by sklearn tool is {}'.format(auc))
        return auc

y_labels = np.array([0, 0, 1, 1, 0,1,0,1,1,0])
y1_score = np.array([0.38,0.28,0.67,0.38,0.11,0.43,0.88,0.54,0.29,0.75])
y2_score = np.array([0.19,0.89,0.47,0.89,0.95,0.49,0.23,0.66,0.15,0.66])
get_auc(y_labels, y1_score)
get_auc(y_labels, y2_score)
```

After executing the code, we get the AUC for C_1 and C_2 are:

$$AUC_{C_1} = 0.54 \tag{33}$$

$$AUC_{C_2} = 0.40 \tag{34}$$

□

2.

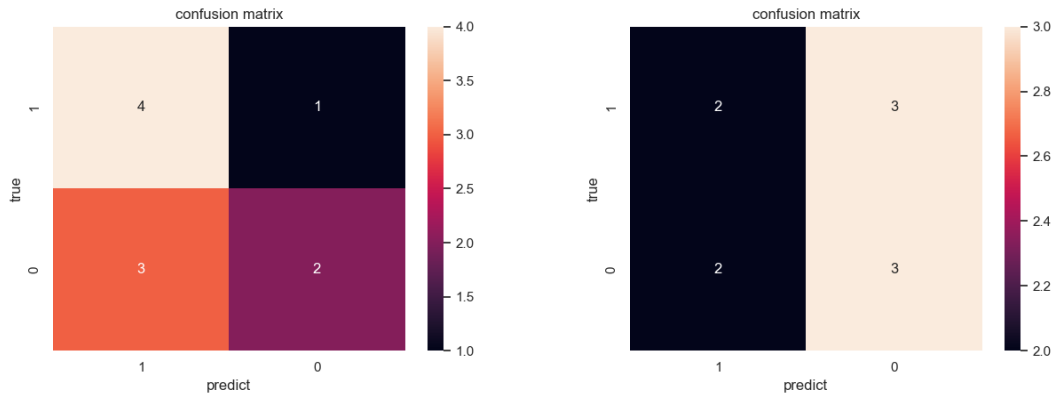
We can calculate the confusion matrix of the two classifier by the following code:

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

def preProcess(y_score,threshold):
        y_pred = [0 for i in range(len(y_score))]
        # print(y_pred)
        for i in range(len(y_score)):
        if y_score[i] >= threshold:
        y_pred[i] = 1
        print(y_pred)
        return y_pred

# draw the confusion matrix
```

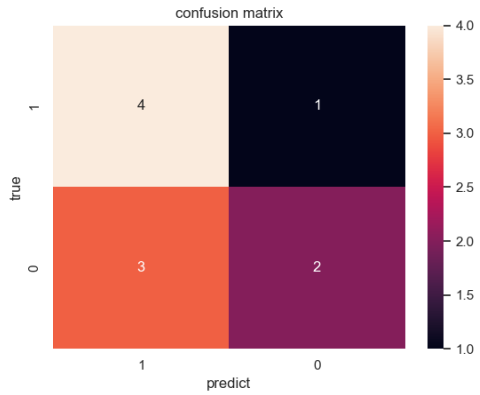
```
sns.set()
f,ax = plt.subplots()
y_true = [0,0,1,1,0,1,0,1,1,0]
y1_score = [0.38,0.28,0.67,0.38,0.11,0.43,0.88,0.54,0.29,0.75]
y2_score = [0.19,0.89,0.47,0.89,0.95,0.49,0.23,0.66,0.15,0.66]
print(y_true)

y1_pred = preProcess(y1_score,0.5)
y2_pred = preProcess(y2_score,0.5)
#
print("Confusion␣Matrix:␣")

C2 = confusion_matrix(y_true,y1_pred,labels=[1,0])
#  C2
print(C2)
sns.heatmap(C2,annot=True,xticklabels=[1,0], yticklabels=[1,0]) # draw
ax.set_title('confusion␣matrix') #title
ax.set_xlabel('predict') #x
ax.set_ylabel('true') #y
plt.show()
plt.savefig("heatmap.png")
```

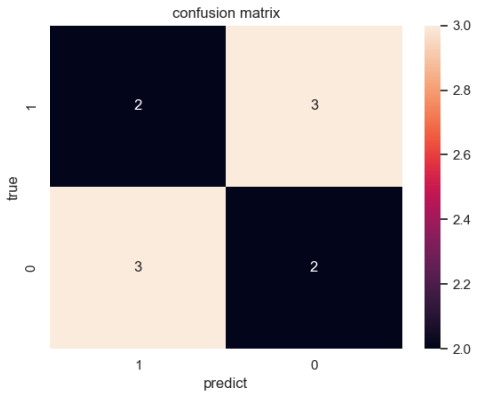
And the results are as follows:

For classifier 1 (0.3 First, and then 0.5):



For classifier 2 (0.3 First, and then 0.5):





3.

The formula of F1-score is:

$$F1 = \frac{2 * P * R}{P + R} = \frac{2 * TP}{N + TP - TN} \tag{35}$$

So it's very easy for us to calculate the F1-score for the both classifier by using the confusion matrix and the formula above:

The F1-score for classifier 1 is:

threshold = 0.3: 0.67 threshold =0.5: 0.44

The F1-score for classifier 2 is:

threshold = 0.3: 0.67 threshold =0.5: 0.40