Machine Learning Homework 02

陳柏翔 4109061012

Problem 01 (Regression of Ridge & Lasso)：

Code：

(詳見附檔 problem_01.py)

Result：

```
>> Least Square Linear Regression :
Optimal weights = [2.42763158 0.40131579 0.05263158]

>> Ridge Regression :
Optimal weights (lambda=0) = [2.42763158 0.40131579 0.05263158]
Optimal weights (lambda=0.01) = [2.42763158 0.40131579 0.05263158]
Optimal weights (lambda=0.1) = [2.42763158 0.40131579 0.05263158]
Optimal weights (lambda=1) = [2.46839654 0.39387891 0.0499002 ]
Optimal weights (lambda=10) = [2.75837743 0.33686067 0.03835979]

>> Lasso Regression :
Optimal weights (lambda=0) = [2.42763158 0.40131579 0.05263158]
Optimal weights (lambda=0.01) = [2.43228618 0.40129934 0.05072368]
Optimal weights (lambda=0.1) = [2.47417763 0.40115132 0.03355263]
Optimal weights (lambda=1) = [2.62109375 0.38671875 0.        ]
Optimal weights (lambda=10) = [3.4296875 0.2109375 0.        ]
```

Problem 02 (Ridge Regression)：

Code：

(詳見附檔 problem_02.py)

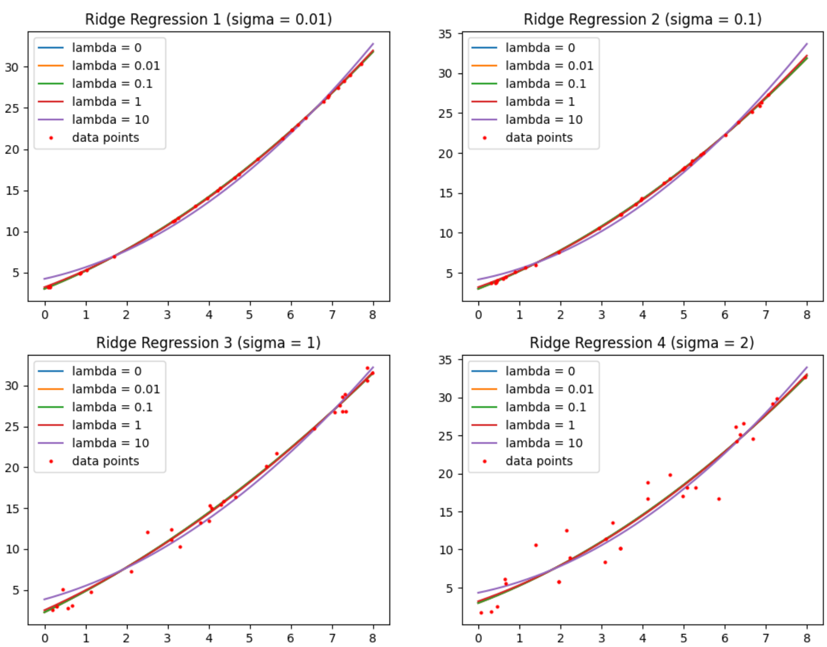
Result：

```
Ridge Regression 1 (sigma = 0.01):
y = 3.011 + 1.995*x + 0.2*x^2  (lambda = 0)
y = 3.013 + 1.994*x + 0.201*x^2  (lambda = 0.01)
y = 3.033 + 1.979*x + 0.202*x^2  (lambda = 0.1)
y = 3.219 + 1.842*x + 0.219*x^2  (lambda = 1)
y = 4.234 + 1.099*x + 0.309*x^2  (lambda = 10)

Ridge Regression 2 (sigma = 0.1):
y = 2.982 + 2.016*x + 0.199*x^2  (lambda = 0)
y = 2.985 + 2.014*x + 0.199*x^2  (lambda = 0.01)
y = 3.006 + 1.995*x + 0.202*x^2  (lambda = 0.1)
y = 3.202 + 1.826*x + 0.225*x^2  (lambda = 1)
y = 4.16 + 1.004*x + 0.335*x^2  (lambda = 10)

Ridge Regression 3 (sigma = 1):
y = 2.242 + 2.447*x + 0.152*x^2  (lambda = 0)
y = 2.245 + 2.445*x + 0.152*x^2  (lambda = 0.01)
y = 2.27 + 2.429*x + 0.154*x^2  (lambda = 0.1)
y = 2.506 + 2.271*x + 0.171*x^2  (lambda = 1)
y = 3.837 + 1.387*x + 0.27*x^2  (lambda = 10)

Ridge Regression 4 (sigma = 2):
y = 2.964 + 2.061*x + 0.209*x^2  (lambda = 0)
y = 2.966 + 2.06*x + 0.209*x^2  (lambda = 0.01)
y = 2.99 + 2.043*x + 0.211*x^2  (lambda = 0.1)
y = 3.202 + 1.892*x + 0.229*x^2  (lambda = 1)
y = 4.323 + 1.099*x + 0.325*x^2  (lambda = 10)
```



Problem 03 (Lasso Regression)：

Code：

(詳見附檔 problem_03.py)

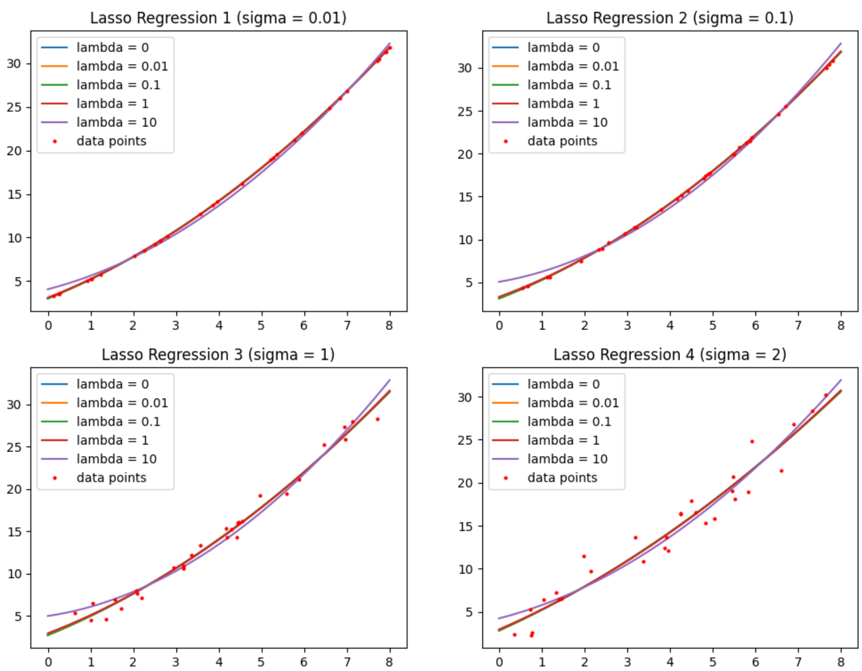
Result：

```
Lasso Regression 1 (sigma = 0.01):
y = 3.002 + 1.999*x + 0.2*x^2  (lambda = 0)
y = 3.003 + 1.998*x + 0.2*x^2  (lambda = 0.01)
y = 3.013 + 1.992*x + 0.201*x^2  (lambda = 0.1)
y = 3.109 + 1.926*x + 0.208*x^2  (lambda = 1)
y = 4.069 + 1.269*x + 0.282*x^2  (lambda = 10)

Lasso Regression 2 (sigma = 0.1):
y = 3.106 + 1.952*x + 0.205*x^2  (lambda = 0)
y = 3.107 + 1.95*x + 0.205*x^2  (lambda = 0.01)
y = 3.125 + 1.94*x + 0.206*x^2  (lambda = 0.1)
y = 3.3 + 1.841*x + 0.217*x^2  (lambda = 1)
y = 5.053 + 0.847*x + 0.328*x^2  (lambda = 10)

Lasso Regression 3 (sigma = 1):
y = 2.741 + 2.072*x + 0.19*x^2  (lambda = 0)
y = 2.744 + 2.071*x + 0.19*x^2  (lambda = 0.01)
y = 2.764 + 2.059*x + 0.191*x^2  (lambda = 0.1)
y = 2.968 + 1.94*x + 0.205*x^2  (lambda = 1)
y = 5.01 + 0.746*x + 0.342*x^2  (lambda = 10)

Lasso Regression 4 (sigma = 2):
y = 2.829 + 2.248*x + 0.153*x^2  (lambda = 0)
y = 2.83 + 2.247*x + 0.153*x^2  (lambda = 0.01)
y = 2.843 + 2.239*x + 0.154*x^2  (lambda = 0.1)
y = 2.971 + 2.151*x + 0.165*x^2  (lambda = 1)
y = 4.249 + 1.273*x + 0.273*x^2  (lambda = 10)
```



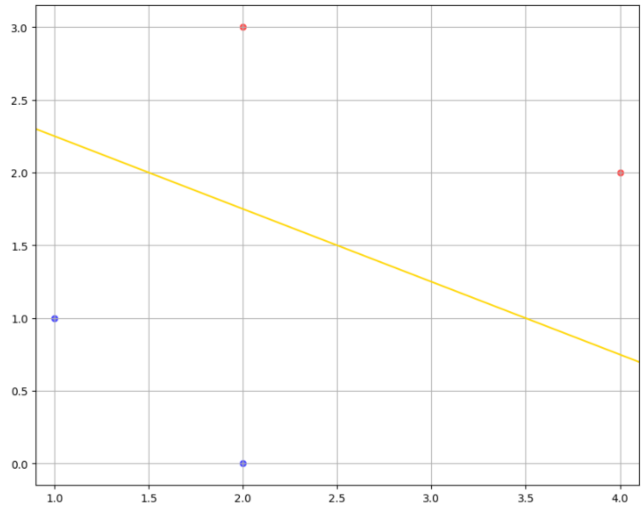
Problem 04 (Support Vector Machine)：

Code：

(詳見附檔 problem_04.py)

Result：

```
>> weights = [0.40041502 0.79979249], bias = -2.200207509496154
```



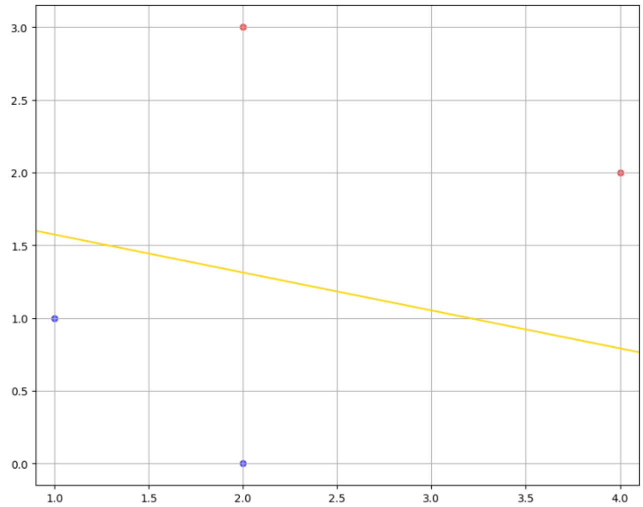
Problem 05 (Logistic Regression)：

Code：

(詳見附檔 problem_05.py)

Result：

```
>> weights = [-0.48893627 -1.8741184 ], bias = 3.438489185347353
```



Problem 06 (Naive Bayes classifier)：

x_1	x_2	x_3	y
1	0	0	0
0	1	0	1
1	0	1	1
0	0	1	1
1	1	0	0
0	0	1	0

From Naive Bayes model：

$p\big((x_1, x_2, x_3) = (1,1,1)\big|\, y = c\big) = p(x_1 = 1|\, y = c)\, p(x_2 = 1|\, y = c)\, p(x_3 = 1|\, y = c)$

Case $\boldsymbol{y = 1}$：

$p\big(y = 1\big|(x_1, x_2, x_3) = (1,1,1)\big)$

$$= \frac{p\big((x_1, x_2, x_3) = (1,1,1)|\, y = 1\big)\, p(y = 1)}{p\big((x_1, x_2, x_3) = (1,1,1)|\, y = 1\big)\, p(y = 1) + p\big((x_1, x_2, x_3) = (1,1,1)|\, y = 0\big)\, p(y = 0)}$$

$$= \frac{\left(\frac{1}{3}\cdot\frac{1}{3}\cdot\frac{2}{3}\right)\cdot\frac{3}{6}}{\left(\frac{1}{3}\cdot\frac{1}{3}\cdot\frac{2}{3}\right)\cdot\frac{3}{6} + \left(\frac{2}{3}\cdot\frac{1}{3}\cdot\frac{1}{3}\right)\cdot\frac{3}{6}} = 0.5$$

Case $\boldsymbol{y = 0}$：

$p\big(y = 0\big|(x_1, x_2, x_3) = (1,1,1)\big)$

$$= \frac{p\big((x_1, x_2, x_3) = (1,1,1)|\, y = 0\big)\, p(y = 0)}{p\big((x_1, x_2, x_3) = (1,1,1)|\, y = 1\big)\, p(y = 1) + p\big((x_1, x_2, x_3) = (1,1,1)|\, y = 0\big)\, p(y = 0)}$$

$$= \frac{\left(\frac{2}{3}\cdot\frac{1}{3}\cdot\frac{1}{3}\right)\cdot\frac{3}{6}}{\left(\frac{1}{3}\cdot\frac{1}{3}\cdot\frac{2}{3}\right)\cdot\frac{3}{6} + \left(\frac{2}{3}\cdot\frac{1}{3}\cdot\frac{1}{3}\right)\cdot\frac{3}{6}} = 0.5$$

Since $p\big(y = 1\big|(x_1, x_2, x_3) = (1,1,1)\big) = p\big(y = 0\big|(x_1, x_2, x_3) = (1,1,1)\big) = 0.5$
$(x_1, x_2, x_3) = (1,1,1)$ will be classified into $y = 0$ or $y = 1$, **both are possible**.

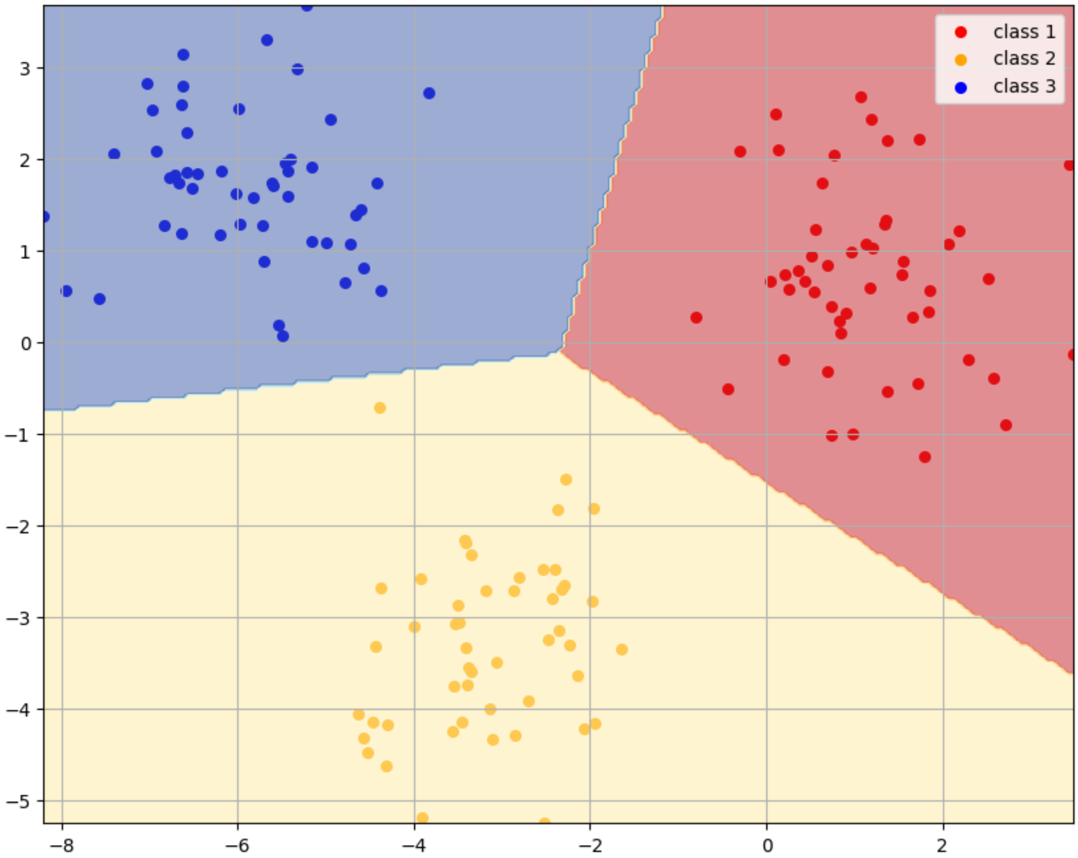
Problem 07 (Logistic Regression for Multi-Class classification)：

Code：

(詳見附檔 problem_07.py)

Result：

```
>> weights =
[[ 1.91956152 -0.67122698 -1.24833454]
[ 1.104031   -3.18791456  2.08388357]]
>> bias = [ 4.6332462  -1.95803809 -2.67520811]
```



Problem 08：

For two-dimensional input data $\boldsymbol{x}_n$ together with $y_n = \{1, 0\}$. (M = 2)

- In the logistic regression, we model

$$p(C_1|\phi(\boldsymbol{x})) = \sigma(w^T\phi(\boldsymbol{x}))$$

where
- $\boldsymbol{w} = [w_0, w_1, \cdots, w_{M-1}]^T$
- $\phi(\boldsymbol{x}) = [\phi_0(\boldsymbol{x}), \phi_1(\boldsymbol{x}), \cdots, \phi_{M-1}(\boldsymbol{x})]^T$
- M adjustable parameters to deal with
- $p(C_2|\boldsymbol{x}) = 1 - p(C_1|\boldsymbol{x})$

- Now we transform the input data $\boldsymbol{x}_n$ to $\phi(\boldsymbol{x}_n)$ for $n = 1,2,\cdots,N$, together with $y_n \in \{0,1\}$, the likelihood function

$$f_Y(\boldsymbol{y}|\boldsymbol{w}) = \prod_{n=1}^{N} p_n^{y_n}(1-p_n)^{1-y_n},$$

where $y = [y_1, y_2, \cdots, y_N]^T$ and $p_n = p(C_1|\phi(\boldsymbol{x}_n)) = \sigma(\boldsymbol{w}^T\phi(\boldsymbol{x}_n)) = \sigma(\boldsymbol{w}^T\phi_{x_n})$

- We define the cross-entropy error function as

$$L(\boldsymbol{w}) = -\ln f_Y(\boldsymbol{y}|\boldsymbol{w}) = -\sum_{n=1}^{N}\{y_n \ln p_n + (1-y_n)\ln(1-p_n)\}$$

- The partial derivative of $L(\boldsymbol{w})$ with respect to w_i is

$$\frac{\partial}{\partial w_i}L(\boldsymbol{w}) = -\frac{\partial}{\partial w_i}\ln f_Y(\boldsymbol{y}|\boldsymbol{w}) = -\frac{\partial}{\partial w_i}\sum_{n=1}^{N}\{y_n \ln p_n + (1-y_n)\ln(1-p_n)\}$$
$$= -\sum_{n=1}^{N}\left\{y_n\left[\frac{\partial}{\partial w_i}\ln p_n\right] + (1-y_n)\left[\frac{\partial}{\partial w_i}\ln(1-p_n)\right]\right\}$$

- We now look at $\frac{\partial}{\partial w_i}\ln p_n$

$$\frac{\partial}{\partial w_i}\ln p_n = \frac{1}{p_n}\frac{\partial}{\partial w_i}p_n = \frac{1}{p_n}\frac{\partial}{\partial w_i}\sigma(\boldsymbol{w}^T\phi_{x_n}) = \frac{1}{p_n}\left[\frac{\partial}{\partial w_i}\boldsymbol{w}^T\phi_{x_n}\right]\left[\frac{\partial}{\partial \boldsymbol{w}^T\phi_{x_n}}\sigma(\boldsymbol{w}^T\phi_{x_n})\right]$$

- We note that $\sigma(a) = \frac{1}{1+\exp\{-a\}}$ and

$$\frac{\partial\sigma(a)}{\partial a} = \frac{\exp\{-a\}}{(1+\exp\{-a\})^2} = \frac{(1-\sigma(a))/\sigma(a)}{(1/\sigma(a))^2} = \sigma(a)(1-\sigma(a))$$

- This gives us

$$\frac{\partial}{\partial w_i}\ln p_n = \frac{1}{p_n}\left[\frac{\partial}{\partial w_i}\boldsymbol{w}^T\phi_{x_n}\right]\left[\frac{\partial}{\partial \boldsymbol{w}^T\phi_{x_n}}\sigma(\boldsymbol{w}^T\phi_{x_n})\right]$$
$$= \frac{1}{p_n}\phi_{x_n,i}\sigma(\boldsymbol{w}^T\phi_{x_n})\left(1-\sigma(\boldsymbol{w}^T\phi_{x_n})\right) = \phi_{x_n,i}\left(1-\sigma(\boldsymbol{w}^T\phi_{x_n})\right)$$

- By the same token,

$$\frac{\partial}{\partial w_i}\ln(1-p_n) = \frac{-1}{1-p_n}\phi_{x_n,i}\sigma(\boldsymbol{w}^T\phi_{x_n})\left(1-\sigma(\boldsymbol{w}^T\phi_{x_n})\right) = -\phi_{x_n,i}\sigma(\boldsymbol{w}^T\phi_{x_n})$$

- Thus,

$$\frac{\partial}{\partial w_i}L(\boldsymbol{w}) = \sum_{n=1}^{N}\left\{y_n\left[\phi_{x_n,i}\left(1-\sigma(\boldsymbol{w}^T\phi_{x_n})\right)\right] + (1-y_n)[-\phi_{x_n,i}\sigma(\boldsymbol{w}^T\phi_{x_n})]\right\}$$
$$= \sum_{n=1}^{N}\{y_n\phi_{x_n,i} - \phi_{x_n,i}\sigma(\boldsymbol{w}^T\phi_{x_n})\}$$
$$= \sum_{n=1}^{N}(y_n - p_n)\phi_{x_n,i}$$

- Finally, we have

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\sum_{n=1}^{N}(y_n - p_n)\phi_{x_n} = \nabla L(\boldsymbol{w})$$

  - the contribution to the gradient from data point is given by the error term $y_n - p_n$, which counts for the prediction error
  - the gradient descent can be adopted to update the weight vector

Links to codes on GitHub：

problem_01.py [link]

problem_02.py [link]

problem_03.py [link]

problem_04.py [link]

problem_05.py [link]

problem_07.py [link]

如不便於下載附檔程式碼
此處附上程式碼連結

by 陳柏翔, EEGuizhi is my GitHub user name