Homework III

Madalena Yang 110206
Madalena Mota 110355

1.

D	y_1	y_2	y_{num}	$\phi(y_1, y_2) = y_1 \times y_2$
x_1	2	2	3,5	4
x_2	1	1	1,0	1
x_3	3	2	3,8	6
x_4	6	3	10,1	18
x_5	8	1	8,5	8

OLS closed form solution

$$w = (X^T X)^{-1} X^T z$$

$$X = \begin{bmatrix} 1 & 4 \\ 1 & 1 \\ 1 & 6 \\ 1 & 18 \\ 1 & 8 \end{bmatrix} \qquad z = \begin{bmatrix} 3,5 \\ 1,0 \\ 3,8 \\ 10,1 \\ 8,5 \end{bmatrix}$$

$w = [1,46136 \quad 0,52955]^T$
(calculated in code)

2. This time, we'll learn a Ridge Regression, so the weights are given by $w = (X^T X + \lambda I)^{-1} X^T z$, with $\lambda = 1$ penalty factor

Ridge
$$w = (X^T X + \lambda I)^{-1} X^T z$$

$w = [0,97319 \quad 0,56921]^T$ → calculated in code

A flexible model is very susceptible to noise, which means it fits the training data well but fails to generalize. Regularization helps prevent overfitting by restricting the model, penalizing large and unstable coefficients. This results in more generalized, well-fitted and robust models.

As expected, after adding Ridge Regularization, the slope barely changed and the intersect decreased from 1,46136 to 0,97319. This is because Ridge penalizes large coefficients to reduce overfitting and to prevent them from influencing predictions too heavily compared to other more important ones.

In conclusion, the results obtained with Regularization were expected = larger coefficients shrunk so they don't influence the predictions as heavily, and the importance of each was calibrated.

3.

OLS model:

$w = [1,46136 \quad 0,52955]^T$

$\hat{z} = f(x) = w_0 + w_1 x_1 + \ldots + w_m x_m$

Mean absolute error

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |\hat{z}_i - z_i|$$

prediction

y_{num}

$\hat{z}_i = w_0 + w_1 x_i$

$\hat{z}_i = 1,46136 + 0,52955 x_i$

D	y_1	y_2	y_{num}	$y_1 \times y_2$	$\hat{z}_i = w_0 + w_1 x_i$	
x_1	2	2	3,5	4	3,57956	TRAIN
x_2	1	1	1,0	1	1,99091	
x_3	3	2	3,8	6	4,63866	
x_4	6	3	10,1	18	10,99326	
x_5	8	1	8,5	8	5,69776	
x_6	0	2	1,0	0	1,46136	TEST
x_7	3	4	6,2	12	7,81596	
x_8	5	1	3,6	5	4,10911	

- Train MAE: $n=5$ samples

$$z = [3{,}5 \quad 1{,}0 \quad 3{,}8 \quad 10{,}1 \quad 8{,}5]^T$$

$$MAE_{OLS\,train} = \frac{1}{5}\left(|3{,}57956-3{,}5| + |1{,}99091-1{,}0| + |4{,}63866-3{,}8| + |10{,}99326-10{,}1| + |5{,}69776-8{,}5|\right) \simeq 1{,}12093$$

- Test MAE: $n=3$ samples

$$z = [1{,}0 \quad 6{,}2 \quad 3{,}6]^T$$

$$MAE_{OLS\,test} = \frac{1}{3}\left(|1{,}46136-1{,}0| + |7{,}81596-6{,}2| + |4{,}10911-3{,}6|\right) \simeq 0{,}86214$$

Ridge model: $w = [0{,}97319 \quad 0{,}56921]^T$ $\qquad \hat{z}_i = 0{,}97319 + 0{,}56921 x_i$

D	$y_1 \times y_2$	$\hat{z}_i$
x_1	4	3,25003
x_2	1	1,54240
x_3	6	4,38845
x_4	18	11,21897
x_5	8	5,52687
x_6	0	0,97319
x_7	12	7,80371
x_8	5	3,81924

• Train MAE : $n = 5$ samples

$$z = [3{,}5 \quad 1{,}0 \quad 3{,}8 \quad 10{,}1 \quad 8{,}5]^T$$

$$MAE_{\text{Ridge train}} = \frac{1}{5}\left(|3{,}25003 - 3{,}5| + |1{,}54240 - 1{,}0| + |4{,}38845 - 3{,}8| + |11{,}21897 - 10{,}1| + |5{,}52687 - 8{,}5|\right)$$

$$\simeq 1{,}09458$$

• Test MAE : $n = 3$ samples

$$z = [1{,}0 \quad 6{,}2 \quad 3{,}6]^T$$

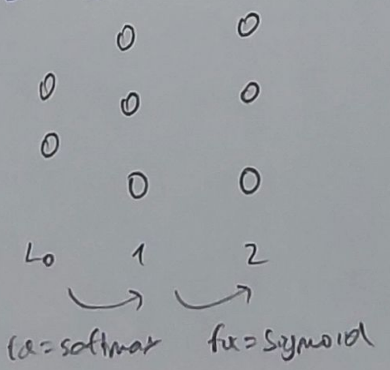
$$MAE_{\text{Ridge test}} = \frac{1}{3}\left(|0{,}97319 - 1{,}0| + |7{,}80371 - 6{,}2| + |3{,}81924 - 3{,}6|\right) \simeq 0{,}61659$$

As expected, both training and testing MAE decreased for the Ridge Regression model compared to OLS. In Ridge, the model achives lower variance, which means it is more stable and less sensitive to noise, at the cost of a little bit more bias. This bias variance tradeoff improves model robustness and stability.
We observed a better generalization of data with Ridge, and even a slightly better train fit. This means Ridge regularized coefficients and removed some instability.

4.

$$W^{[1]} = \begin{bmatrix} 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}2 \\ 0{,}2 & 0{,}1 \end{bmatrix} \qquad b^{[1]} = \begin{bmatrix} 0{,}1 \\ 0 \\ 0{,}1 \end{bmatrix} \qquad W^{[2]} = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad b^{[2]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_1 = [2 \quad 2]^T$$



1. Foward Propagation

$$z^{[i]} = W^{[i]} x^{[i-1]} + b^{[i]}$$

$$x^{[i]} = \underset{\text{activation function}}{fa}(z^{[i]})$$

1. Foward Propagation : $z^{[1]} \longrightarrow x^{[1]} \longrightarrow z^{[2]} \longrightarrow \underset{\text{output}}{x^{[2]}}$

$$z^{[1]} = W^{[1]} x^{[0]} + b^{[1]} = \begin{bmatrix} 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}2 \\ 0{,}2 & 0{,}1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} 0{,}1 \\ 0 \\ 0{,}1 \end{bmatrix} = \begin{bmatrix} 0{,}5 \\ 0{,}6 \\ 0{,}7 \end{bmatrix}$$

$$x^{[1]} = fa(z^{[1]}) = \text{sigmoid}(z^{[1]}) = \sigma(z^{[1]}) = \frac{1}{1 + e^{-z^{[1]}}} = \begin{bmatrix} 0{,}62246 \\ 0{,}64566 \\ 0{,}66819 \end{bmatrix}$$

$$z^{[2]} = w^{[2]} x^{[1]} + b^{[2]} = \begin{bmatrix}1&2&2\\1&2&1\\1&1&1\end{bmatrix}\begin{bmatrix}0,62246\\0,64566\\0,66819\end{bmatrix} + \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}4,25016\\3,58197\\2,93631\end{bmatrix}$$

$$x^{[2]} = fa(z^{[2]}) = softmax(z^{[2]}) = softmax\left(\begin{bmatrix}4,25016\\3,58197\\2,93631\end{bmatrix}\right) \approx \begin{bmatrix}0,56135\\0,28777\\0,15088\end{bmatrix}$$

2. Back Propagation: $\delta^{[2]} \longrightarrow \delta^{[1]}$

Stochastic: one observation

Cross-entropy loss: $-\sum_{i=1}^{N}\sum_{c=1}^{|C|} t_c^{(i)} \log(x^{[i]})$ where t represents the target classes $\{A, B, C\}$

$t_c \in \{0,1\}$ with $\sum_c t_c = 1$

in this case because we're in the last layer, so AF is softmax

$$\delta^{[2]} = \frac{\partial E}{\partial x^{[2]}}\frac{\partial x^{[2]}}{\partial z^{[2]}} = x^{[2]} - t^{[2]} =$$

the target class is A

$$= \begin{bmatrix}0,56135\\0,28777\\0,15088\end{bmatrix} - \begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}-0,43865\\0,28777\\0,15088\end{bmatrix}$$

$$\delta^{[1]} = (w^{[2]})^T \delta^{[2]} \left(\frac{\partial x^{[1]}}{\partial z^{[1]}}\right) = (w^{[2]})^T \delta^{[2]} \circ x^{[1]} \circ (1 - x^{[1]}) =$$

$$= \begin{bmatrix}1&2&2\\1&2&1\\1&1&1\end{bmatrix}^T \begin{bmatrix}-0,43865\\0,28777\\0,15088\end{bmatrix} \circ \begin{bmatrix}0,62246\\0,64566\\0,66819\end{bmatrix} \circ \begin{bmatrix}0,37754\\0,35434\\0,33181\end{bmatrix} = \begin{bmatrix}0\\-0,03452\\-0,09725\end{bmatrix}$$

3. Update weights $(\eta = 0,5)$

$$w^{[2]} = w^{[2]} - \eta \frac{\partial E}{\partial w^{[2]}} = w^{[2]} - \eta\, \delta^{[2]} (x^{[1]})^T =$$

$$= \begin{bmatrix}1&2&2\\1&2&1\\1&1&1\end{bmatrix} - 0,5 \begin{bmatrix}-0,43865\\0,28777\\0,15088\end{bmatrix}\begin{bmatrix}0,62246\\0,64566\\0,66819\end{bmatrix}^T =$$

$$= \begin{bmatrix}1,13652 & 2,14161 & 2,14655\\0,91044 & 1,90710 & 0,90386\\0,95304 & 0,95129 & 0,94959\end{bmatrix}$$

2. Back Propagation

$$\delta^{[l]} = \frac{\partial E}{\partial z^{[l]}}$$

last layer: $\delta^{[l]} = \frac{\partial E}{\partial x^{[l]}}\frac{\partial x^{[l]}}{\partial z^{[l]}}$

other: $\delta^{[l]} = \left(\underbrace{\frac{\partial z^{[l+1]}}{\partial x^{[l]}}}_{w^{[l+1]}}\right)^T \delta^{[l+1]} \left(\underbrace{\frac{\partial x^{[l]}}{\partial z^{[l]}}}_{\text{deriv. of AF}}\right)$

$$\delta^{[l]} = \frac{\partial E}{\partial z^{[l]}} = \frac{\partial E}{\partial x^{[l]}}\frac{\partial x^{[l]}}{\partial z^{[l]}}$$

Sigmoid: (for hidden layer in this case)

$$\frac{\partial x^{[l]}(z^{[l]})}{\partial z^{[l]}} = \sigma(z^{[l]})(1-\sigma(z^{[l]})) = x^{[l]}(1-x^{[l]})$$

3. Update weights

$$w^{[l]} = w^{[l]} - \eta \frac{\partial E}{\partial w^{[l]}}$$

$$\left(b^{[l]} = b^{[l]} - \eta \frac{\partial E}{\partial b^{[l]}}\right)$$

$$\frac{\partial E}{\partial w^{[l]}} = \underbrace{\frac{\partial E}{\partial z^{[l]}}}_{\delta^{[l]}}\frac{\partial z^{[l]}}{\partial w^{[l]}} = \delta^{[l]} \underbrace{\frac{\partial z^{[l]}}{\partial w^{[l]}}}_{x^{[l-1]}}$$

$$\frac{\partial E}{\partial b^{[l]}} = \frac{\partial E}{\partial z^{[l]}}\underbrace{\frac{\partial z^{[l]}}{\partial b^{[l]}}}_{1} = \delta^{[l]}$$

$$b^{[2]} = b^{[2]} - \eta \frac{\partial E}{\partial b^{[2]}} = b^{[2]} - \eta \delta^{[2]} = \begin{bmatrix}1\\1\\1\end{bmatrix} - 0{,}5 \begin{bmatrix}-0{,}43865\\0{,}28777\\0{,}15088\end{bmatrix} = \begin{bmatrix}1{,}21933\\0{,}85612\\0{,}92456\end{bmatrix}$$

$$W^{[1]} = W^{[1]} - \eta \frac{\partial E}{\partial W^{[1]}} = W^{[1]} - \eta \delta^{[1]} (x^{[0]})^T = \qquad x^{[0]} = [2 \quad 2]$$

$$= \begin{bmatrix}0{,}1 & 0{,}1\\0{,}1 & 0{,}2\\0{,}2 & 0{,}1\end{bmatrix} - 0{,}5 \begin{bmatrix}0\\-0{,}03452\\-0{,}09725\end{bmatrix} \begin{bmatrix}2\\2\end{bmatrix}^T =$$

$$= \begin{bmatrix}0{,}1 & 0{,}1\\0{,}13452 & 0{,}23452\\0{,}29725 & 0{,}19725\end{bmatrix}$$

$$b^{[1]} = b^{[1]} - \eta \frac{\partial E}{\partial b^{[1]}} = b^{[1]} - \eta \delta^{[1]} = \begin{bmatrix}0{,}1\\0\\0{,}1\end{bmatrix} - 0{,}5 \begin{bmatrix}0\\-0{,}03452\\-0{,}09725\end{bmatrix} = \begin{bmatrix}0{,}1\\0{,}01726\\0{,}14863\end{bmatrix}$$

Explanation:

A model with no activation fuction simply outputs a linear combination of its inputs, since each layer only performs a linear transformation. As a result, the overall model can only represent linear relationships and fails to capture more complex, nonlinear patterns in the data.

By contrast, a sigmoid activation function introduces nonlinearity, enabling the neural network to learn more complex functions and nonlinear decision boundaries.

Additionally, the sigmoid function squeezes inputs into the (0,1) range, effectively regularizing the output. This allows the model to interpret the output as a probability, which makes classification more intuitive and the results easier to interpret. The compression effect of the sigmoid also smooths the influence of extreme inputs, making the model's output more stable and less sensitive to very large values or even outliers.

Exercise 1

```python
X = np.matrix([
    [1, 4],
    [1, 1],
    [1, 6],
    [1, 18],
    [1, 8]
])

z = np.array([3.5, 1, 3.8, 10.1, 8.5])

# Calculate weights vector
beta_ols = np.linalg.inv(X.T @ X) @ X.T @ z
beta_ols = np.array(beta_ols).flatten()

# Extract intercept and slope
w0, w1 = beta_ols

print(f"Intercept (w0): {w0:.5f}")
print(f"Slope (w1): {w1:.5f}")
```

✓ 0.0s

```
Intercept (w0): 1.46136
Slope (w1): 0.52955
```

Exercise 2

```python
# Calculate Ridge weights vector
penalty = 1

I = np.eye(X.shape[1])
# I[0, 0] = 0  # don't regularize intercept

beta_ridge = np.linalg.inv(X.T @ X + I*penalty) @ X.T @ z
beta_ridge = np.array(beta_ridge).flatten()

# Extract intercept and slope
w0, w1 = beta_ridge

print(f"Intercept (w0): {w0:.5f}")
print(f"Slope (w1): {w1:.5f}")
```

✓ 0.0s

```
Intercept (w0): 0.97319
Slope (w1): 0.56921
```