NYCU Introduction to Machine Learning, Homework 1

110550130, 劉秉驊

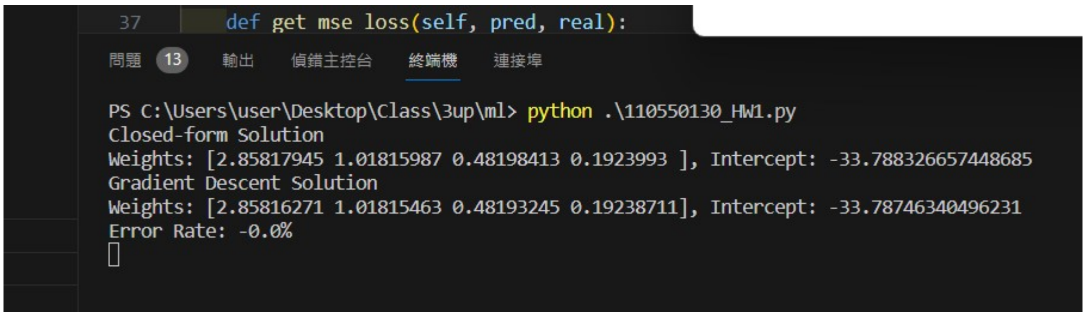
The screenshot and the figures we provided below are just examples. **The results below are not guaranteed to be correct.** Please make sure your answers are clear and readable, or no points will be given. Please also remember to convert it to a pdf file before submission. You should use English to answer the questions. After reading this paragraph, you can delete this paragraph.

Part. 1, Coding (50%):

(10%) Linear Regression Model - Closed-form Solution

1. (10%) Show the weights and intercepts of your linear model.

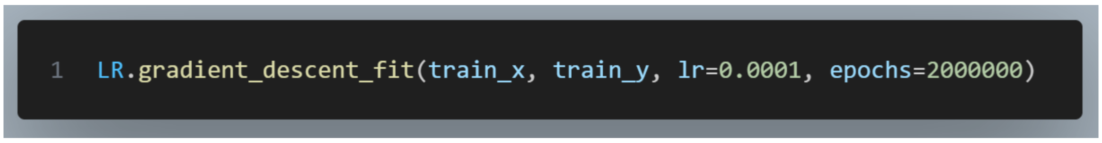
   The weights and intercept of closed form solution is at the 3rd line.



```
PS C:\Users\user\Desktop\Class\3up\ml> python .\110550130_HW1.py
Closed-form Solution
Weights: [2.85817945 1.01815987 0.48198413 0.1923993 ], Intercept: -33.788326657448685
Gradient Descent Solution
Weights: [2.85816271 1.01815463 0.48193245 0.19238711], Intercept: -33.78746340496231
Error Rate: -0.0%
```

(40%) Linear Regression Model - Gradient Descent Solution

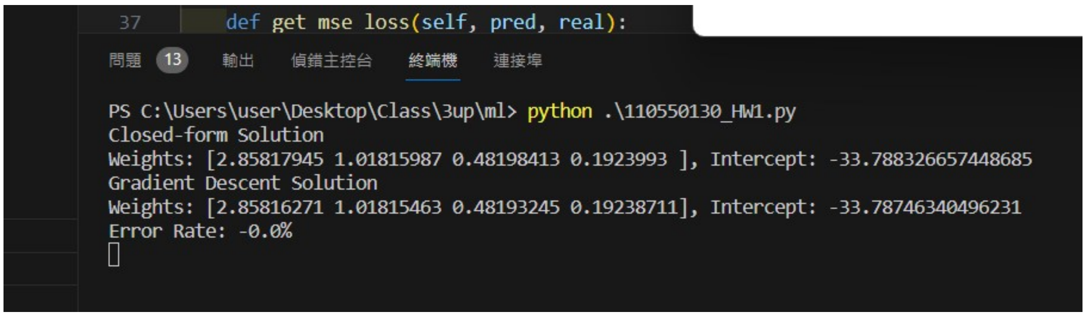
2. (0%) Show the learning rate and epoch (and batch size if you implement mini-batch gradient descent) you choose.



```
LR.gradient_descent_fit(train_x, train_y, lr=0.0001, epochs=2000000)
```

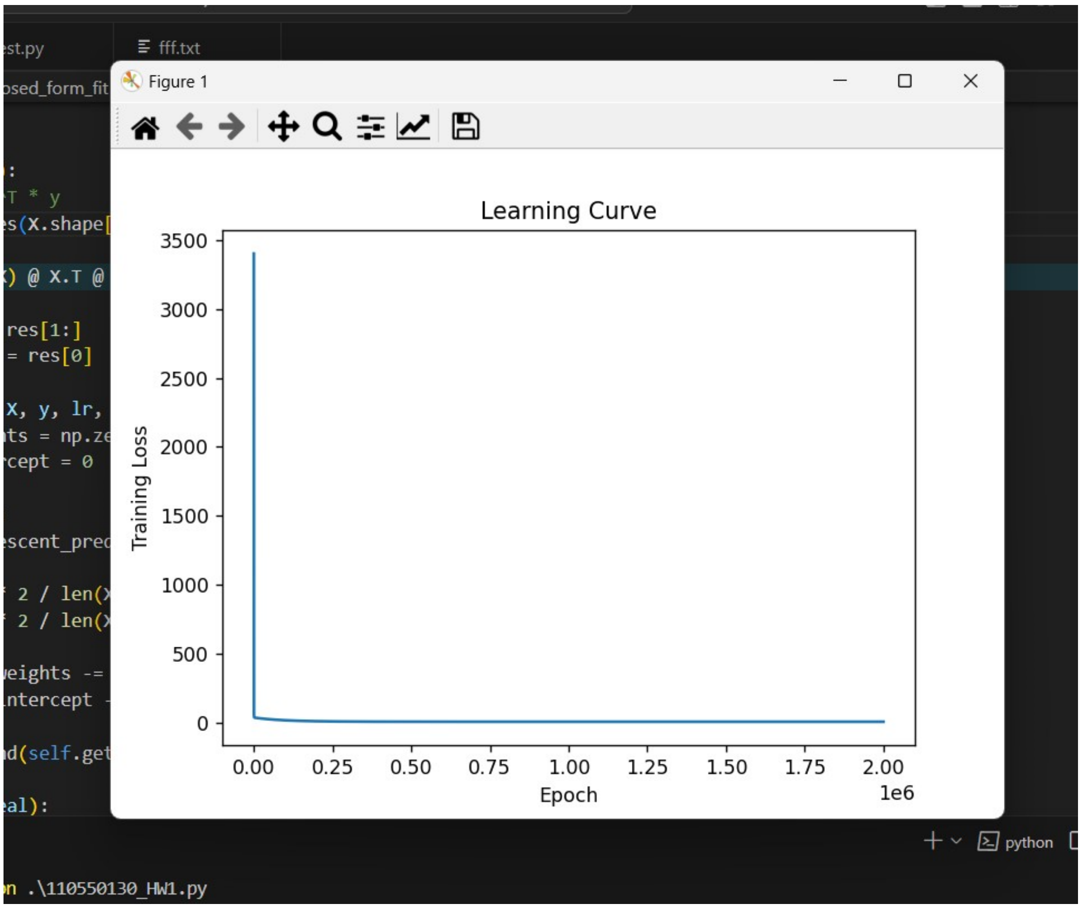
3. (10%) Show the weights and intercepts of your linear model.

   The weights and intercept of gradient descent solution is at the 5th line.



```
PS C:\Users\user\Desktop\Class\3up\ml> python .\110550130_HW1.py
Closed-form Solution
Weights: [2.85817945 1.01815987 0.48198413 0.1923993 ], Intercept: -33.788326657448685
Gradient Descent Solution
Weights: [2.85816271 1.01815463 0.48193245 0.19238711], Intercept: -33.78746340496231
Error Rate: -0.0%
```

4. (10%) Plot the learning curve. (x-axis=epoch, y-axis=training loss)

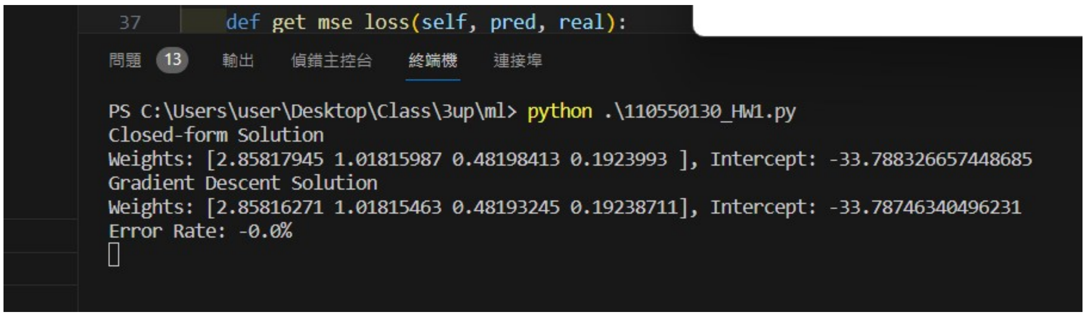


```
):
T * y
s(X.shape[
) @ X.T @

res[1:]
= res[0]

X, y, lr,
ts = np.ze
cept = 0

scent_pred

 2 / len(X
 2 / len(X

eights -=
ntercept 

d(self.get

al):
```

```
n .\110550130_HW1.py
```

5. (20%) Show your error rate between your closed-form solution and the gradient descent solution.

   The error rate is at the last line.



```
    def get mse loss(self, pred, real):
```

```
PS C:\Users\user\Desktop\Class\3up\ml> python .\110550130_HW1.py
Closed-form Solution
Weights: [2.85817945 1.01815987 0.48198413 0.1923993 ], Intercept: -33.788326657448685
Gradient Descent Solution
Weights: [2.85816271 1.01815463 0.48193245 0.19238711], Intercept: -33.78746340496231
Error Rate: -0.0%
```

Part. 2, Questions (50%):

1. (10%) How does the value of learning rate impact the training process in gradient descent? Please explain in detail.
   Write or type your answer here.

   The learning rate decides the number of the weights we would like to evaluate. If we have a tremendously small learning rate, it might take a lot of time to converge. In the other hand, if we have a big learning rate, it might be hard to converge, that is, diverge.

2. (10%) There are some cases where gradient descent may fail to converge. Please provide at least two scenarios and explain in detail.
   Write or type your answer here.

   Divergence happens when we choose a too big learning rate that for the first thing, the gradient might be tremendously large, that is close to infinity, and for the second thing, we might find it hard to get the minimum cost, as known as the best solution, that the gradient swings beside.

3. (15%) Is mean square error (MSE) the optimal selection when modeling a simple linear regression model? Describe why MSE is effective for resolving most linear regression problems and list scenarios where MSE may be inappropriate for data modeling, proposing alternative loss functions suitable for linear regression modeling in those cases.
   Write or type your answer here.

   For most simple linear regression model, I think MSE is a good choice to evaluate since first, it is easy to implement and compute that sees every error the same weight and second, it is easy to analyze the best solution at the global minimum. However, MSE is not good at handling problems like non-gaussian errors that doesn't have normal distribution but we can use mean absolute error to evaluate. Second, MSE gets inappropriate weights when heteroscedasticity happens so that we can use weighted least squares depending on different variance of errors.

4. (15%) In the lecture, we learned that there is a regularization method for linear regression models to boost the model's performance. (p18 in linear_regression.pdf)

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

   4.1. (5%) Will the use of the regularization term always enhance the model's performance? Choose one of the following options: "Yes, it will always improve," "No, it will always worsen," or "Not necessarily always better or worse."
   4.2. We know that λ is a parameter that should be carefully tuned. Discuss the following situations: (both in 100 words)
      4.2.1. (5%) Discuss how the model's performance may be affected when λ is set too small. For example, λ=10^(-100) or λ=0

4.2.2. (5%) Discuss how the model's performance may be affected when λ is set too large. For example, λ=1000000 or λ=10^100

Write or type your answer here.

Q4.1:
Not necessarily always better or worse. Regularization might destroy the relationship among the data causing underfit.

Q4.2.1:
If λ is too small, we cannot obtain an obvious effect of regularization since λEw(w) is close to zero, which still leads to overfit if the origin model has the problem of overfit.

Q4.2.2:
If λ is too big, the error function will be controlled by λ since Ed(w) is largely smaller than λEw(w) that produces a poor performance of learning for the model, which regularizes overhead and causes underfit.