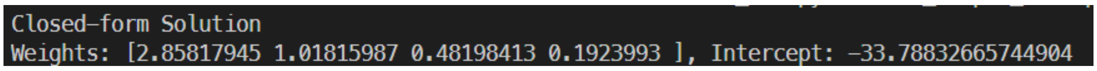
NYCU Introduction to Machine Learning, Homework 1

110705013, 沈昱宏

Part. 1, Coding (50%):

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

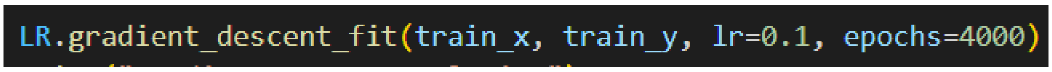
1. (10%) Show the weights and intercepts of your linear model.



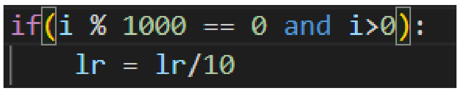
```
Closed-form Solution
Weights: [2.85817945 1.01815987 0.48198413 0.1923993 ], Intercept: -33.78832665744904
```

(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.
   The initial lr is 0.1, and /10 per 1000 epoch.

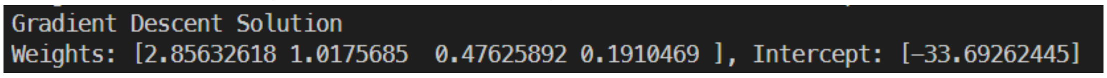


```
LR.gradient_descent_fit(train_x, train_y, lr=0.1, epochs=4000)
```



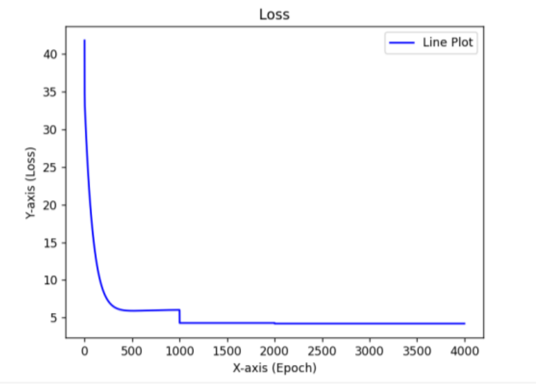
```
if(i % 1000 == 0 and i>0):
    lr = lr/10
```

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

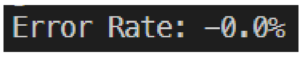


```
Gradient Descent Solution
Weights: [2.85632618 1.0175685  0.47625892 0.1910469 ], Intercept: [-33.69262445]
```

4. (10%) Plot the learning curve. (x-axis=epoch, y-axis=training loss)
   The plot look a little weird only because I divide the lr by 10 every 1000 epoch.

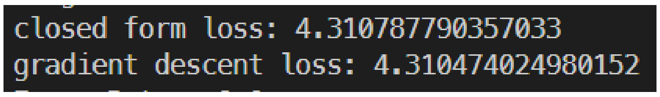


5. (20%) Show your error rate between your closed-form solution and the gradient descent solution.



```
Error Rate: -0.0%
```

It seems that my gradient descent solution perform a little better than closed-form solution on the testing set.



```
closed form loss: 4.310787790357033
gradient descent loss: 4.310474024980152
```

Part. 2, Questions (50%):

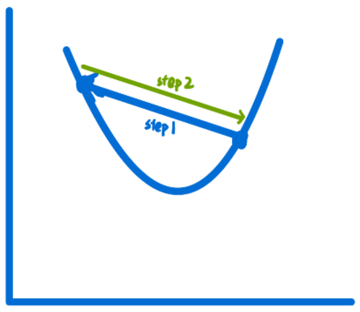
1. (10%) How does the value of learning rate impact the training process in gradient descent? Please explain in detail.

   Ans:
   The learning rate decide how big the step is going to be when updating the model. The larger the lerning rate, the bigger step you take. If the value of learning rate is too small, your weights will converge slowly or only converge to local optimal solutions because the step you took was too small. On the other hand, if the value of learning rate is too large, you might not be able to converge, and the loss will oscillate or even diverge.

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

   Ans:
   (1) When the learning rate is excessively large, the step taken may not reach the local minimum, as illustrated in the graph (the first step go through the minimum point, and land on another point which has bigger loss then the original one, and the second step goes to approximately where you were before taking the first step).

   (2) The loss function your trying to minimize has saddle points (like f(x) = x^3, which has gradient = 0 at point (0,0) but is not the local minimum). If the learning rate is very small and the model is updating toward a lot of saddle points with small steps, it will take a long time for the model to escape from the saddle points, so it will take a long time to converge.



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.

Ans:

MSE is generally a good choice when modeling a simple linear regression model, as it aligns with the maximum likelihood (ML) solution. However, it is not always the best choice. MSE is not so appropriate in the following scenarios: (1) there are a few outliers in your data, the outlier will have huge affect on the loss function and make the model perform poorly. The possible solution to this is use cross-entropy, Huber loss, MAE (mean absolute error), or weighted MSE, which make the effect of outliers smaller. (2) MSE assumes constant variance and a Gaussian distribution of errors, which may not be true in some cases. In such situations, the loss functions mentioned in (1) are more appropriate because they don't rely on these assumptions.

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

Ans:

4.1

“Not necessarily always better or worse”. If a model is complex and overfits, adding a regularization term will improve performance, but if the model don’t have overfitting issues, the regularization term may lead to worse performance.

4.2.1

If the λ is too small, the regularization term you add will only make little affect on the original function, so it is not able to generalize the model by making the model weights smaller. The model’s performance might increase a little if you don’t set λ to 0, but it will not have enough affect to really gerneralize to unseen data.

4.2.2

If the λ is too big, the performance of your model will drop significantly. For example, having a λ = 10^100 will set the weights to be very small (near 0) because the model will always minimizing the Ew(w) and pay little attention on Ed, and the model you are training can not learn anything in the training process, making the model’s performance drop significantly.