NYCU Introduction to Machine Learning, Homework 1

[112550151], [周佳瑩]

Part. 1, Coding (60%):

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

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

```
2024-03-14 23:13:57.944 | INFO     | __main__:main:77 - LR_CF.weights=
```

```
2025-10-06 14:18:26.695 | INFO     | __main__:main:125 - LR_CF.weights=array(
[2.85501274, 1.01785863, 0.47202168, 0.19608925]), LR_CF.intercept=-33.6956
```

(40%) Linear Regression Model - Gradient Descent Solution

2. (10%)
   - Show the hyperparameters of your setting (e.g., learning rate, number of epochs, batch size, etc.).
   - Show the weights and intercepts of your linear model.

```
LR.gradient_descent_fit(train_x, train_y, lr=0.01, epochs=100)
```

```
2024-03-14 23:54:18.052 | INFO     | __main__:main:84 - LR_GD.weights=
```

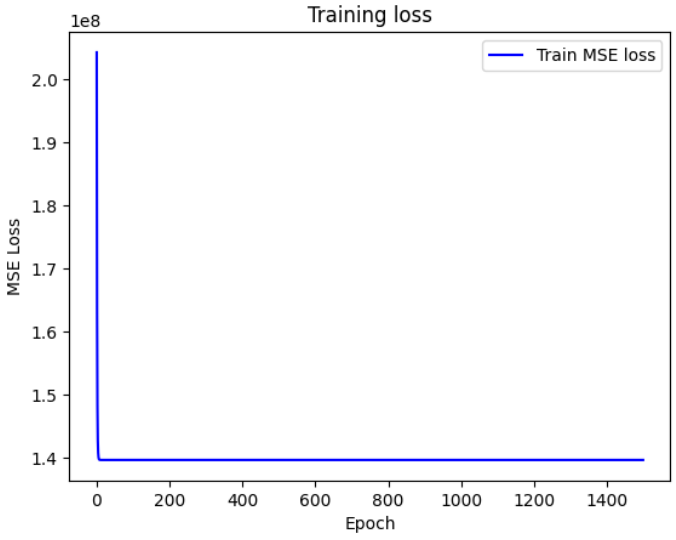
Hyperparameters:

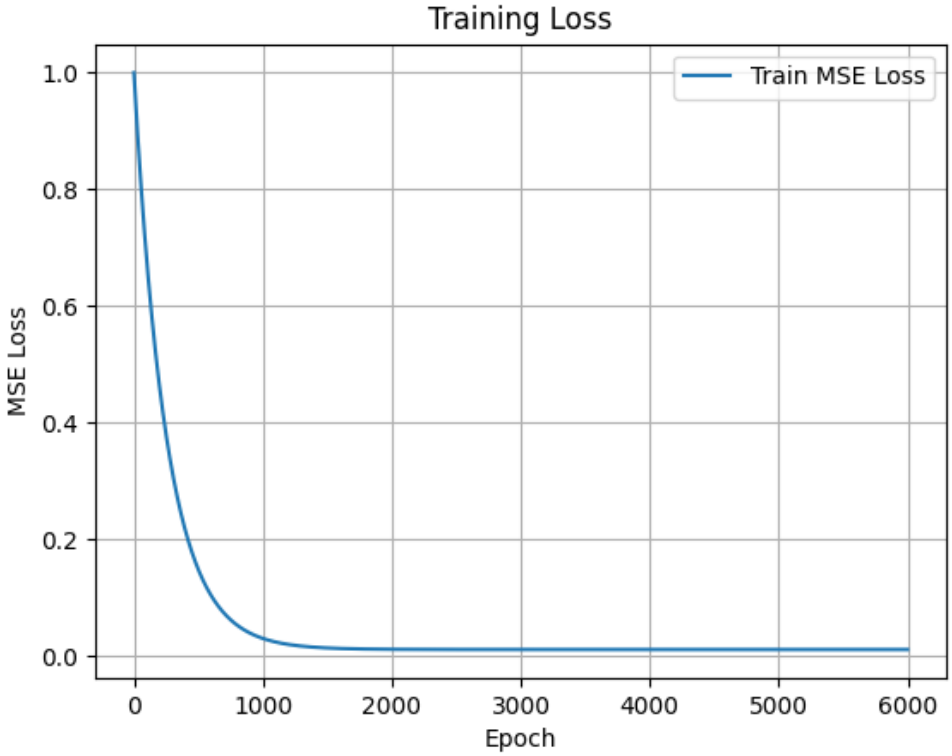
```
epoches = 6000
losses, lr_history = LR_GD.fit(train_x, train_y, epoches, learning_rate=1e-3)
```

Weights and intercepts:

```
2025-10-06 17:44:42.047 | INFO     | __main__:main:148 - LR_GD.weights=array(
[2.85498446, 1.01785158, 0.47202619, 0.19609752]), LR_GD.intercept=-33.6950
```

3. (10%) Plot the learning curve. (x-axis=epoch, y-axis=training loss)





4. (20%) Show your MSE.cf, MSE.gd, and error rate between your closed-form solution and the gradient descent solution.

```
2024-03-14 23:54:18.055 | INFO     | __main__:main:93 - Prediction difference:
2024-03-14 23:54:18.055 | INFO     | __main__:main:98 - mse_cf=, mse_gd=. Difference: 0.027%
```

```
2025-10-06 17:44:54.021 | INFO     | __main__:main:170 - Prediction difference: 0.0001
2025-10-06 17:44:54.021 | INFO     | __main__:main:176 - mse_cf=4.2903, mse_gd=4.2903. Difference: 0.000%
```

(10%) Code Check and Verification

5. (10%) Lint the code and show the PyTest results.

```
seanyu@bulbasaur:~/lectures/nycu/ml-and-pattern-recognition/hw1 *$ poetry run flake8 solution.py
seanyu@bulbasaur:~/lectures/nycu/ml-and-pattern-recognition/hw1 *$
```

```
============================= test session starts ==============================
platform linux -- Python 3.9.5, pytest-8.0.2, pluggy-1.4.0
rootdir: /
collected 2 items

test_main.py 2024-03-16 11:52:21.189 | INFO     | test_main:test_regression_cf:27 - model.weights=array([[3.]]), model.intercept=array(
[4.])
.2024-03-16 11:52:21.190 | INFO     | main:fit:57 - EPOCH 0, loss=3147.416663702691
2024-03-16 11:52:21.644 | INFO     | main:fit:57 - EPOCH 10000, loss=0.29281584845965486
2024-03-16 11:52:22.094 | INFO     | main:fit:57 - EPOCH 20000, loss=0.0053609642405778 5
2024-03-16 11:52:22.544 | INFO     | main:fit:57 - EPOCH 30000, loss=9.815021195041223e-05
2024-03-16 11:52:22.998 | INFO     | main:fit:57 - EPOCH 40000, loss=1.7969648133316264e-06
2024-03-16 11:52:23.450 | INFO     | main:fit:57 - EPOCH 50000, loss=3.2899394472691304e-08
2024-03-16 11:52:23.905 | INFO     | main:fit:57 - EPOCH 60000, loss=6.023324157052075e-10
2024-03-16 11:52:24.363 | INFO     | test_main:test_regression_gd:39 - model.weights=array([3.]), model.intercept=3.9999966785390386
.

============================== 2 passed in 4.37s ===============================
```

Before:

```
PS C:\Users\midor\Documents\大學\大三上\ML\HW1> flake8 main.py
main.py:99:1: W293 blank line contains whitespace
main.py:117:1: E302 expected 2 blank lines, found 1
main.py:141:1: W293 blank line contains whitespace
main.py:146:47: E251 unexpected spaces around keyword / parameter equals
main.py:146:49: E251 unexpected spaces around keyword / parameter equals
main.py:163:5: E303 too many blank lines (2)
2     E251 unexpected spaces around keyword / parameter equals
1     E302 expected 2 blank lines, found 1
1     E303 too many blank lines (2)
2     W293 blank line contains whitespace
```

After:

```
PS C:\Users\midor\Documents\大學\大三上\ML\HW1> flake8 main.py
PS C:\Users\midor\Documents\大學\大三上\ML\HW1>
```

```
PS C:\Users\midor\Documents\大學\大三上\ML\HW1> pytest ./test_main.py -s
============================= test session starts =============================
platform win32 -- Python 3.11.5, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\midor\Documents\大學\大三上\ML\HW1
plugins: anyio-4.8.0
collected 2 items

test_main.py (100, 1)
2025-10-07 00:34:17.744 | INFO     | test_main:test_regression_cf:30 - model.weights=array([[3.]]), model.intercept=array([4.])
.(100, 1)
2025-10-07 00:34:17.746 | INFO     | main:fit:109 - EPOCH 0, loss=1.0000, lr=0.0001
2025-10-07 00:34:17.942 | INFO     | main:fit:109 - EPOCH 10000, loss=0.0183, lr=0.0001
2025-10-07 00:34:18.161 | INFO     | main:fit:109 - EPOCH 20000, loss=0.0003, lr=0.0001
2025-10-07 00:34:18.412 | INFO     | main:fit:109 - EPOCH 30000, loss=0.0000, lr=0.0001
2025-10-07 00:34:18.588 | INFO     | main:fit:109 - EPOCH 40000, loss=0.0000, lr=0.0001
2025-10-07 00:34:18.845 | INFO     | main:fit:109 - EPOCH 50000, loss=0.0000, lr=0.0001
2025-10-07 00:34:19.053 | INFO     | main:fit:109 - EPOCH 60000, loss=0.0000, lr=0.0001
2025-10-07 00:34:19.231 | INFO     | test_main:test_regression_gd:44 - model.weights=array([2.99999751]), model.intercept=4.000000000000036
.

============================== 2 passed in 2.81s ==============================
```

Part. 2, Questions (40%):

1. (10%) Linear models $y = w^\top x + b$ have limited fitting power.
   a. In one sentence, explain why a single linear model is limited.
   b. Give one concrete task that a single linear model cannot solve, and state why no single hyperplane/affine function solves it.

   A single linear model assumes that the relationship between the variables is linear, it might not work well when the relationship is non-linear, and it can be significantly affected by outliers.
   Since the parameter needs to enter linearly, linear models fail at trigonometric functions. Linear models only learn affine mappings, which cannot represent periodic or cyclic relationships like sine and cosine, as these functions are nonlinear and bounded, while affine functions are unbounded and monotonic. Some examples are sleep cycles or hormonal levels over a day, or the height of ocean tides or waves over time.

2. (15%) Why do we add a regularization term in linear regression? What are the differences between L2 regularization (Ridge) and L1 regularization (Lasso)? Please explain in detail.

We add regularization to prevent overfitting as it adds a penalty for complexity by adding constraints to the model to reduce the risk of memorizing noise, it also improves generalization by encouraging simpler models that perform better on new data. L1 regularization or Lasso (Least Absolute Shrinkage and Selection Operator) adds the absolute value of magnitude of the coefficients as a penalty, this shrinks some coefficients to zero, which eliminates them, and helps in selecting the important features and ignoring the insignificant ones. On the other hand, L2 regularization or Ridge regression adds the squared magnitude of the coefficients as a penalty, it handles multicollinearity by shrinking the coefficients of correlated features instead of eliminating them.

3. (15%)

- What is overfitting? Under what conditions can a model overfit? (List two) How can overfitting be alleviated? (List two)

Overfitting is when the model learns too much from the training data, or when a model fits too closely to the training data. It can occur when 1. there isn't enough training data, or 2. when the model is too complex. We can alleviate overfitting by 1. collecting more training data and improving the quality of the training data, or 2. decreasing the complexity of the model.