NYCU Introduction to Machine Learning, Homework 1

111550035, 蔡昀錚

Part. 1, Coding (60%):

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

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

```
Part 1-1. Linear regression model-Close
2024-10-04 14:31:34.958 | INFO     | __main__:main:112 - LR_CF.weights=[[2.8491883  1.0188675  0.48562739 0.1937254 ]],
LR_CF.intercept=[-33.8222531]
```

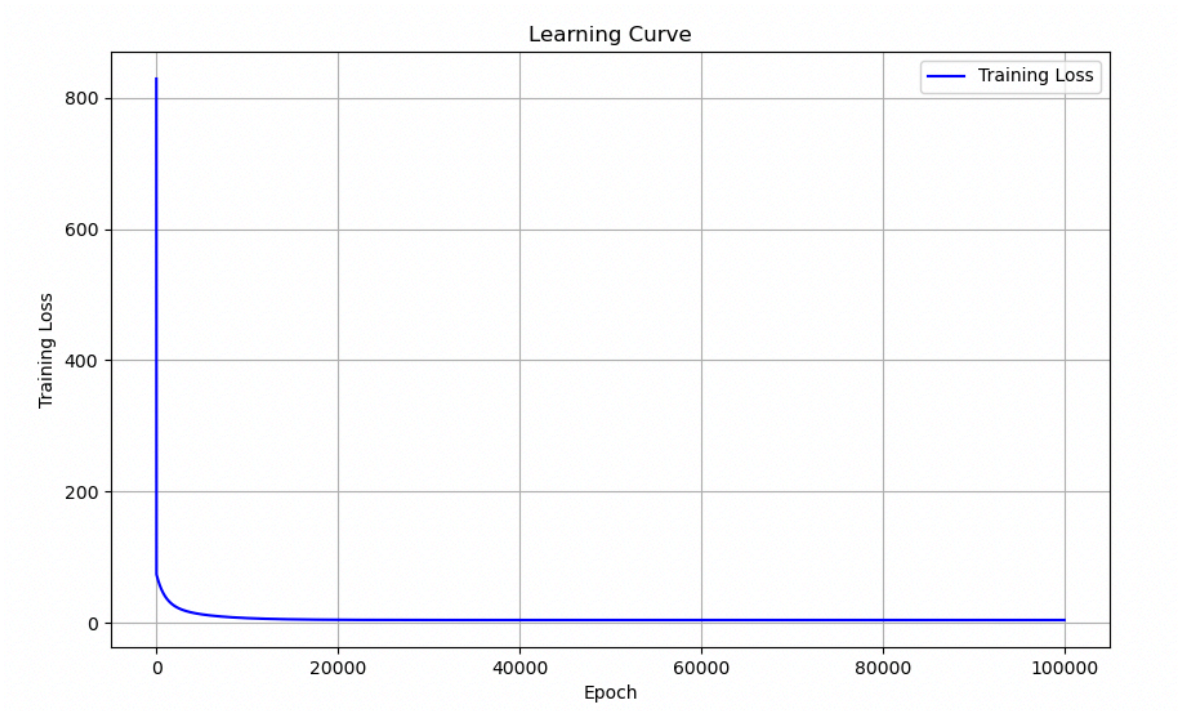
(40%) Linear Regression Model - Gradient Descent Solution

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.

```
losses = LR_GD.fit(x_train, y_train, learning_rate=1e-4, epochs=100000)
```

```
2024-10-07 20:51:26.875 | INFO     | __main__:main:146 - LR_GD.weights: [[2.84917361 1.01886357 0.48556619 0.1937153 ]], LR_GD.intercept: -33.821442655175424
```

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.

```
2024-10-07 20:52:07.847 | INFO     | __main__:main:154 - Prediction difference: 0.0000
2024-10-07 20:52:07.847 | INFO     | __main__:main:155 - mse_cf=4.1997, mse_gd=4.1997. Difference: 0.0004%
```

(10%) Code Check and Verification

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

```
❯ pytest ./test_main.py -s
============================================================ test session starts ============================================================
platform darwin -- Python 3.9.20, pytest-8.3.3, pluggy-1.5.0
rootdir: /Users/tsai_m/Desktop/ML/111550035_HW1
collected 2 items

test_main.py 2024-10-07 20:55:34.183 | INFO     | test_main:test_regression_cf:27 - model.weights=array([[3.]]), model.intercept=array([4.])
.2024-10-07 20:55:34.184 | INFO     | main:fit:90 - EPOCH 0, loss=13345.120000068295
2024-10-07 20:55:34.357 | INFO     | main:fit:90 - EPOCH 10000, loss=2.33033455706813e-27
2024-10-07 20:55:34.527 | INFO     | main:fit:90 - EPOCH 20000, loss=2.33033455706813e-27
2024-10-07 20:55:34.701 | INFO     | main:fit:90 - EPOCH 30000, loss=2.33033455706813e-27
2024-10-07 20:55:34.875 | INFO     | main:fit:90 - EPOCH 40000, loss=2.33033455706813e-27
2024-10-07 20:55:35.043 | INFO     | main:fit:90 - EPOCH 50000, loss=2.33033455706813e-27
2024-10-07 20:55:35.214 | INFO     | main:fit:90 - EPOCH 60000, loss=2.33033455706813e-27
2024-10-07 20:55:35.393 | INFO     | test_main:test_regression_gd:39 - model.weights=array([[3.]]), model.intercept=np.float64(3.999999999999958)
.

============================================================ 2 passed in 2.55s =============================================================
```

Part. 2, Questions (40%):

1. (10%) How does the presence of outliers affect the performance of a linear regression model? How should outliers be handled? List at least two methods.

My answer:

First, outliers can distort the slope and intercept of the regression line. Since linear regression is sensitive to outliers because it minimizes the SSE, which may lead to poor performance and reduce the accuracy of predictions.

Second, outliers can be handles by two methods:

- Transformation: Transform the data using log, box-cox transformation, in order to compress the range of data points, which can avoid extreme values.
- Remove outlier from the dataset. (Need enough data and valid reason)

2. (15%) How do different values of learning rate (too large, too small…) affect the convergence of optimization? Please explain in detail.

My answer:

Learning rate controls the size of the steps the model takes towards the minimum of the loss function. In detail, learning rate define how much the weights and intercept change during each epoch. In other words, learning rate is the coefficient of the slope that will be deducted from weights and intercept of previous epoch.

- Large Learning Rate: the model cannot find the optimal solution, it may oscillate or even diverge (loss function increases).
- Small Learning Rate: May lead to slow convergence, it can be time-consuming or it might get stuck in local minimum.

```
#   update weights and intercept
self.weights -= learning_rate * dw
self.intercept -= learning_rate * db
```

3. (15%)
   - What is the prior, likelihood, and posterior in Bayesian linear regression. [Explain the concept in detail rather than writing out the mathematical formula.]
   - What is the difference between Maximum Likelihood Estimation (MLE) and Maximum A Posteriori Estimation (MAP)? (Analyze the assumptions and the results.)

My answer:

(1)

- Prior: Represents our prediction of the parameters before seeing any data. It can be based on previous knowledge or assumptions.
- Likelihood: Represents how likely the observed data is with specific values of the model parameters. It quantifies the probability of the data. If the parameters are such that the model fits the data well, the likelihood will be high. Otherwise, If the parameters lead to a poor fit of the data, the likelihood will be low.
- Posterior: A type of conditional probability that results from updating the prior probability with likelihood and observed data. If the likelihood provides strong evidence, it will heavily influence the posterior, and the prior will have less impact. Otherwise, If the data is noisy or provides little information, the prior will have more influence on the posterior. $posterior \propto likelihood \times prior$

(2)

MLE: estimates the parameters of a statistical model by finding the values that maximize the likelihood function. It focuses solely on the observed data, ignoring any prior beliefs about the parameters.

MAP: incorporates prior beliefs about the parameters. MAP finds the parameter values that maximize the posterior distribution, which is the combination of the likelihood and the prior.

In conclusion, MAP incorporates prior information along with the data, leading to estimates that reflect both the data and the prior. MLE is purely data-driven and disregards prior beliefs, while MAP incorporates prior information, making it useful when the data is sparse or noisy.