NYCU Pattern Recognition, Homework 1

312553024, 江尚軒

Part. 1, Coding (60%):

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

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

```
2024-03-28 21:13:53.034 | INFO     | __main__:main:96 - LR_CF.weights=array([2.8491883 , 1.0188675 , 0.48562739, 0.1937254 ]), LR_CF.intercept=-33.8223
```

(40%) Linear Regression Model - Gradient Descent Solution

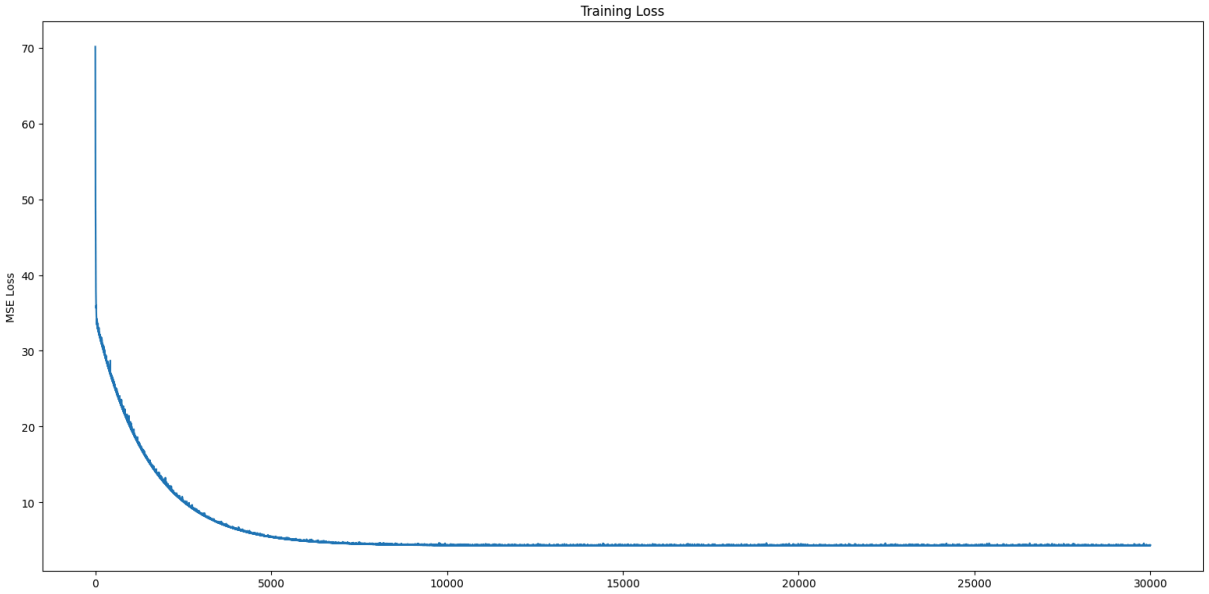
2. (0%) Show the learning rate and epoch (and batch size if you implement mini-batch gradient descent) you choose.

```
# Without L1 regularization
losses = LR_GD.fit(train_x, train_y, learning_rate=1e-4, epochs=30000, batch_size=64)
```

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

```
2024-03-28 21:14:47.883 | INFO     | __main__:main:105 - LR_GD.weights=array([2.84924262, 1.01843923, 0.48556183, 0.19383542]), LR_GD.intercept=-33.8205
```

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



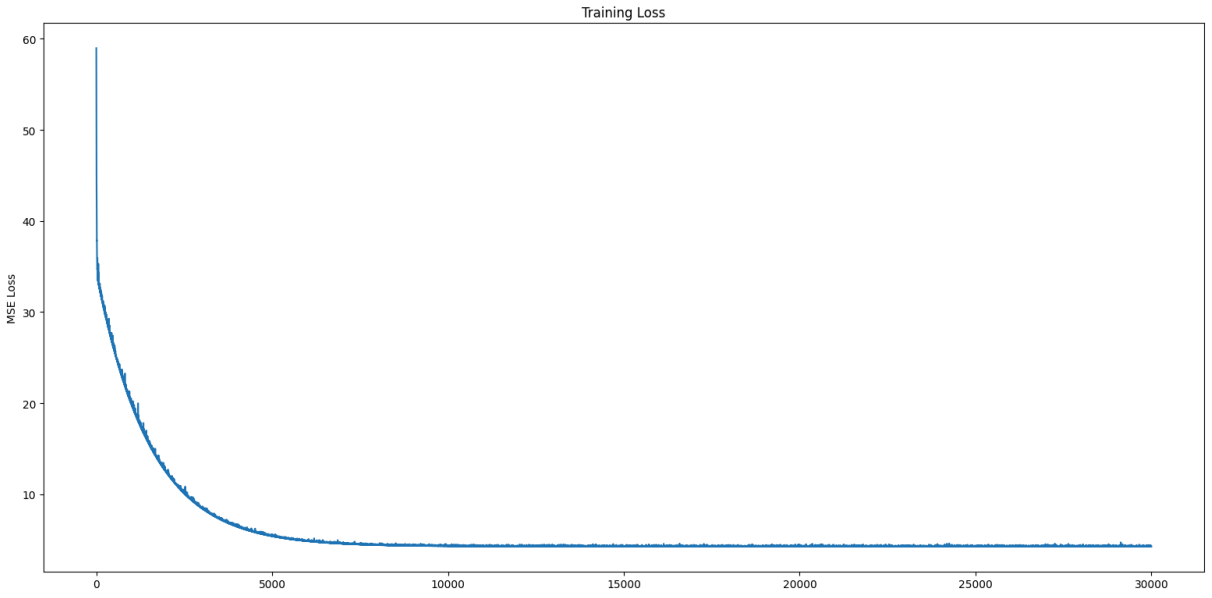
5. (20%) Show your error rate between your closed-form solution and the gradient descent solution.

```
2024-03-28 21:14:47.889 | INFO     | __main__:main:129 - Prediction difference: 55.2920
2024-03-28 21:14:47.890 | INFO     | __main__:main:134 - mse_cf=4.1997, mse_gd=4.2018. Difference: 0.049%
```

6. (Bonus 5 points) Implement the L1 regularization into the gradient descent method and show the weights, intercept, and learning curve.

```
losses = LR_GD.fit(train_x, train_y, learning_rate=1e-4, epochs=30000, batch_size=64, alpha=0.01)
```

```
2024-03-28 21:29:57.107 | INFO     | __main__:main:105 - LR_GD.weights=array([2.84915647, 1.01936857, 0.48558701, 0.19382921]), LR_GD.intercept=-33.8205
```




```
2024-03-28 21:29:57.115 | INFO     | __main__:main:129 - Prediction difference: 73.2607
2024-03-28 21:29:57.116 | INFO     | __main__:main:134 - mse_cf=4.1997, mse_gd=4.1994. Difference: -0.009%
```

(10%) Code Check and Verification

7. (10%) Lint the code and show the PyTest results.

   Lint:

```
HW1/code$ flake8 main.py
HW1/code$
```

   PyTest:

```
(1122-pattern-recognition) (base) adsl-1-2@adsl-1-2:~/Andy/NYCU/碩一/碩一下/圖形識別/1122-pattern-recognition/Homework/HW1/code$ pytest ./test_main.py -s
============================= test session starts ==============================
platform linux -- Python 3.12.2, pytest-8.1.1, pluggy-1.4.0
rootdir: /home/adsl-1-2/Andy/NYCU/碩一/碩一下/圖形識別/1122-pattern-recognition/Homework/HW1/code
collected 2 items

test_main.py 2024-03-28 22:10:53.873 | INFO     | test_main:test_regression_cf:27 - model.weights=array([[3.]]), model.intercept=array([4.])
.2024-03-28 22:11:28.068 | INFO     | test_main:test_regression_gd:40 - model.weights=array([[3.]]), model.intercept=3.9999999999992095
.

============================== 2 passed in 34.72s ==============================
```

Part. 2, Questions (40%):

1. (10%) Please describe the Vanishing Gradient Problem in detail, and provide **at least two solutions** to overcome this problem.

   In deep neural networks, training can run into a roadblock called the vanishing gradient problem. This happens when the error signals used to adjust the network's connections (weights) fade away as they travel backward through the layers. With weak signals, the network struggles to update its weights effectively, hindering its ability to learn.

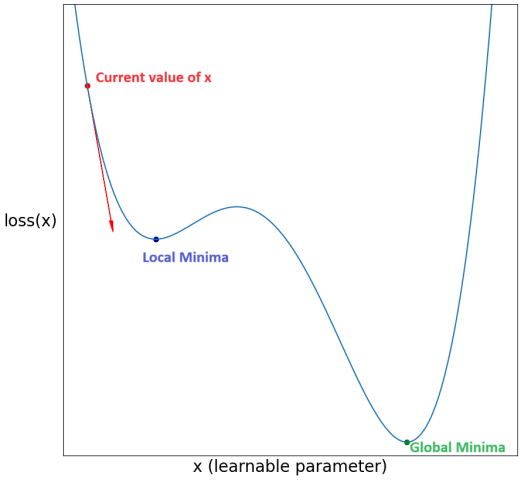
   Solutions:

   (1) Using ReLU Activation Function

   (2) Batch Normalization

   (3) Residual Networks (ResNets)

2. (15%) Gradient descent often suffers from the issue of getting stuck at local minima (refer to the figure provided). Please provide **at least two methods** to overcome this problem and discuss how these methods work.



(1) **Stochastic Gradient Descent (SGD):** This clever trick uses mini-batches of data instead of the whole dataset. This injects randomness, making the learning process less prone to getting stuck in shallow dips (local minima) and often leading to faster convergence.

(2) **Momentum:** Imagine a ball rolling downhill. Momentum in this context acts like that extra push. It considers past updates, giving the algorithm more inertia to overcome flat areas or small bumps (local minima) and find a better minimum.

3. (15%) What are the basic assumptions of Linear regression between the features and the target? How can techniques help Linear Regression extend beyond these assumptions? Please at least answer one technique.

Linear regression assumes a straight line explains the data **(linearity)** and errors behave nicely **(independence, constant variance, normal distribution)**.

Transformations (e.g., log): This bends the data to fit a straight line, making the analysis work better. Imagine stretching a curved line to become straight.