NYCU Pattern Recognition, Homework 1

312551814, **Malik DAHMANI**

Part. 1, Coding (60%):

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

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

```
2024-04-02 11:05:12.148 | INFO     | __main__:main:173 - LR_CF.weights=array([2.8491883 , 1.0188675 , 0.48562739, 0.1937254 ]), LR_CF.intercept=-33.8223
```

(40%) Linear Regression Model - Gradient Descent Solution

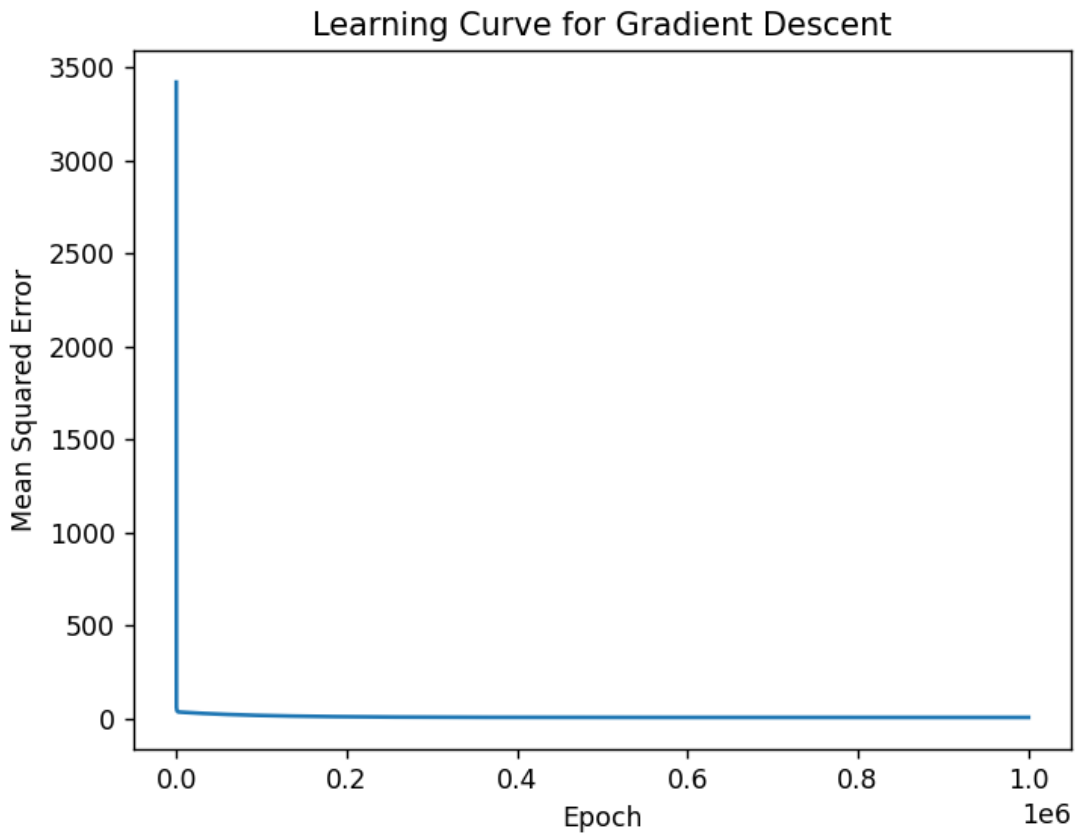
2. (0%) Show the learning rate and epoch (and batch size if you implement mini-batch gradient descent) you choose.

```
LR_GD.fit(train_x, train_y, learning_rate=1e-4, epochs=1000000)
```

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

```
LR_GD.weights=array([2.84575631, 1.01779039, 0.47489042, 0.19126502]), LR_GD.intercept=-33.6441
```

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.

```
Prediction difference between Gradient descent and Closed form: 47.5831
```

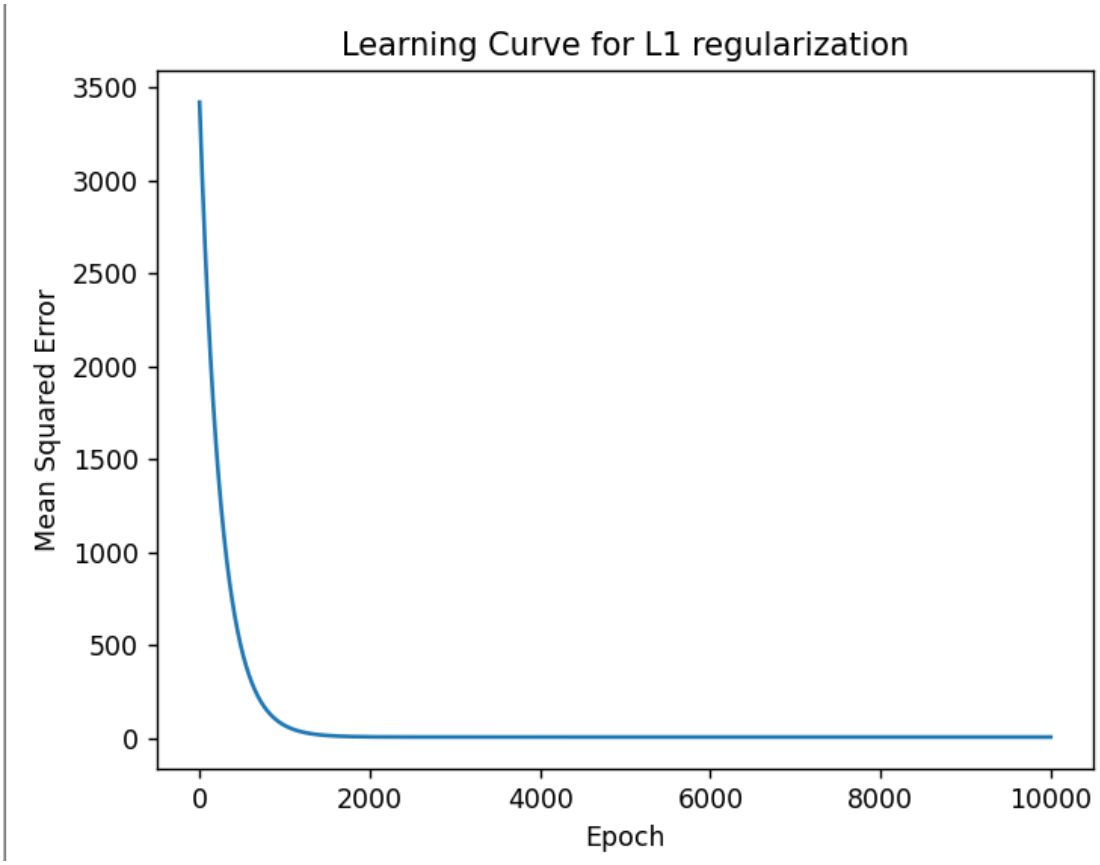
```
- mse_cf=4.1997, mse_gd=4.1978. Difference between Gradient descent and Closed form:  -0.047%
```

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

Weights:

```
LR_L1.weights=array([ 7.39930478, 17.64629104,  0.8163391 ,  0.55151594]), LR_L1.intercept=55.2175
```

Learning Curve:



(10%) Code Check and Verification

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

```
============================= test session starts =============================
platform win32 -- Python 3.8.3, pytest-7.1.3, pluggy-1.0.0
rootdir: h:\Desktop\NYCU\Pattern recognition\release\release
plugins: hypothesis-6.56.0, cov-3.0.0, sugar-0.9.5
collected 2 items

312551814_HW1\test_main.py ..                                            [100%]

============================== 2 passed in 2.26s ==============================
Finished running tests!
```

Part. 2, Questions (40%):

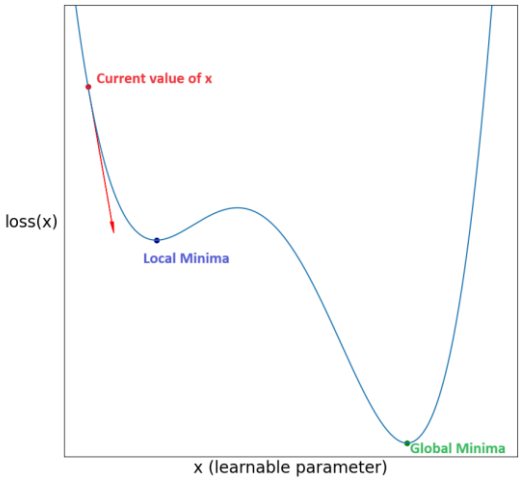
1. (10%) Please describe the Vanishing Gradient Problem in detail, and provide **at least two solutions** to overcome this problem.

The vanishing gradient problem is a recurring problem when implementing gradient descent. This problem occurs when the gradient becomes very small. As the gradient becomes very small, the change in weights becomes minimal, preventing the training from converging to the optimal solution. As a result, learning becomes slower and slower. In a neural network, this means that the first layers of the network learn more slowly than the last.

To overcome this problem, a first solution might be to initialise the weights to prevent them from becoming too small, thus facilitating training convergence.

Another method would be to normalise the input variables, by subtracting the mean and dividing by the standard deviation of each variable. By normalising the input variables before training, the gradients will stabilise, thus avoiding the vanishing gradient problem.

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.



The problem of gradient descent stopping at a local minimum is a recurring problem when implementing gradient descent. To overcome this problem, we can first use random restarts. The technique of random consist of restarting the optimization algorithm with different initial values. We then choose the most optimal value from all the solutions. However, this is not an optimized solution as it is time consuming and uncertain.

Another method is to use the stochastic gradient algorithm. This algorithm randomly samples a subset of the data points at each iteration and then uses the gradient of these data points to update the model parameters. The algorithm won't get stuck because it is constantly exploring new parts of the function.

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.

The basic assumptions of linear regression between the features and target are:

- Linearity of the model
- The variance of the residuals is constant (homoscedasticity)
- Residuals are independent
- The residuals follow a normal distribution with zero mean and variance.
- Non-collinearity of explanatory variables

There are many techniques that can help linear regression go beyond these assumptions.

Firstly, we can perform polynomial regressions of attributes to observe non-linear relationships in the model.

Another example is Ridge regression. Ridge regression was introduced by Hoerl and Kennard (1970) to solve the invertibility problem of the ($X^T X$) matrix. Ridge regression is a linear regression with a quadratic constraint on the coefficients. In this way, ridge regression shrinks the coefficients towards zero, reducing their variance. This is useful when the variables are highly correlated (multi-linear), which often distorts the numerical resolution. In addition, ridge regression can also help with the assumption of constant variance of the residuals (homoscedasticity) by stabilising the variance of the coefficients, potentially stabilising the variance of the residuals.