NYCU Pattern Recognition, Homework 1

311553010, 陳姿羽

Part. 1, Coding (70%):

1. (0%) Show the learning rate and epoch you choose
   learning rate: _ 0.001__
   epoch: _ 250000__

```
batch_size = x_train.shape[0]

# TODO
# Tune the parameters
# Refer to slide page 9
lr = 0.001
epochs = 250000

linear_reg = LinearRegression()
linear_reg.fit(x_train, y_train, lr=lr, epochs=epochs, batch_size=batch_size)
```

```
Don't cheat.
```

2. (5%) Show the weights and intercepts of your linear model.
   weights: _ 380.13935218__
   intercepts: _1382.38621394__

```
print("Intercepts: ", linear_reg.weights[-1])
print("Weights: ", linear_reg.weights[:-1])
```

```
Intercepts:  [1382.38621394]
Weights:  [[380.13935218]]
```

3. (5%) What's your final training loss (MSE)?
   training loss (MSE): _ 69781032.74212956__

```
print('training loss: ', linear_reg.evaluate(x_train, y_train))
```

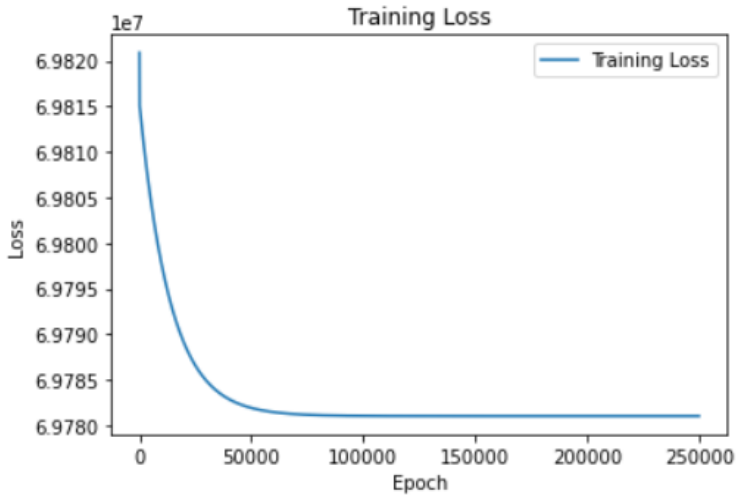
```
training loss:  69781032.74212956
```

4. (5%) What's the MSE of your validation prediction and validation ground truth?
   validation loss (MSE): _ 68460131.75953527__

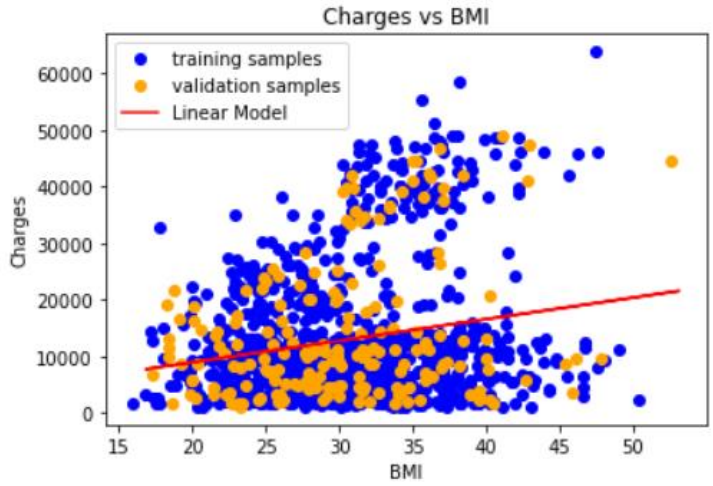
```
print('validation loss: ', linear_reg.evaluate(x_val, y_val))
```

```
validation loss:  68460131.75953527
```

5. (5%) Plot the training curve. (x-axis=epoch, y-axis=loss)



6. (5%) Plot the line you find with the training and validation data.



7. (0%) Show the learning rate and epoch you choose.
learning rate: _ 0.0007__
epoch: _ 650000__

```
batch_size = x_train.shape[0]

# TODO
# Tune the parameters
# Refer to slide page 10
lr = 0.0007
epochs = 650000

linear_reg = LinearRegression()
linear_reg.fit(x_train, y_train, lr=lr, epochs=epochs, batch_size=batch_size)
```

```
Don't cheat.
```

8. (10%) Show the weights and intercepts of your linear model.
   weights: _ [[259.85072835]
   [-383.5471172]
   [333.3318584]
   [442.55699937]
   [24032.21979141]
   [-416.01494403]]__
   intercepts: _ [-11857.0282986]__

```
print("Intercepts: ", linear_reg.weights[-1])
print("Weights: ", linear_reg.weights[:-1])
```

```
Intercepts:  [-11857.0282986]
Weights:  [[  259.85072835]
 [ -383.5471172 ]
 [  333.3318584 ]
 [  442.55699937]
 [24032.21979141]
 [ -416.01494403]]
```

9. (5%) What's your final training loss?
   training loss (MSE): _ 17348585.126771744__

```
print('training loss: ', linear_reg.evaluate(x_train, y_train))
```

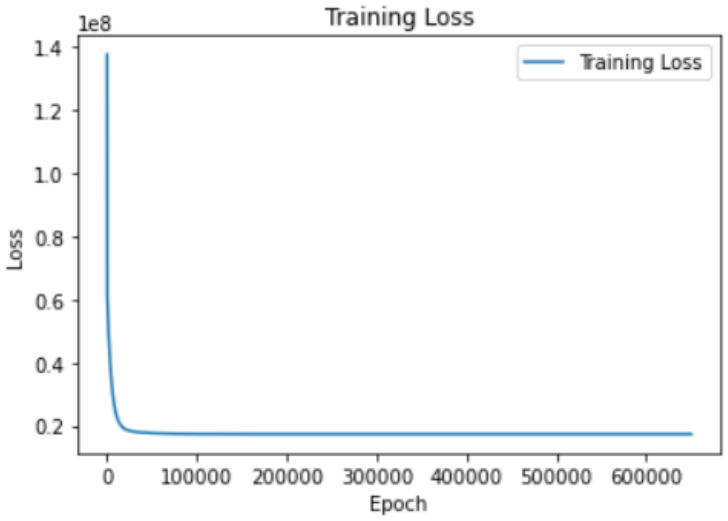
```
training loss:  17348585.126771744
```

10. (5%) What's the MSE of your validation prediction and validation ground truth?
    validation loss (MSE): _ 20979281.826353323__

```
print('validation loss: ', linear_reg.evaluate(x_val, y_val))
```

```
validation loss:  20979281.826353323
```

11. (5%) Plot the training curve. (x-axis=epoch, y-axis=loss)



12. (20%) Train your own model and fill the testing CSV file as your final predictions.

learning rate: _0.0007__
epoch: _ 650000 __
batch_size: _938 (x_train.shape[0])__
Used features: _age, BMI, children, smoker, region__

In the beginning, I change the batch size, but even with a large size, the loss did not decrease.
So I started dropping one of the features in each training, and I found that the loss value when dropping " sex" feature is smaller than dropping other features.
I also trained the model with each single feature. Although the training loss and validation loss of the "smoker" feature are smaller than the "BMI" feature, the test MSE is very large.

Changing Batch Size	
Intercepts: [-11540.42493193] Weights: [[236.10080822] [-451.54937859] [312.24072076] [420.1796225] [24133.75007038] [-377.16008108]] training loss: 18211149.6209228 validation loss: 22649930.886980962 test MSE: 755844.7783730414	batch_size = 5
Using Multiple features	
Intercepts: [-3386.05658591] Weights: [[-796.28236288] [400.34006299] [543.10187093] [24032.6654536] [-491.02680251]] training loss: 24090353.415048793 validation loss: 26005084.901599493 test MSE: 6348210.995338958	Used features: sex, BMI, children, smoker, region
Intercepts: [-12048.36316374] Weights: [[260.60682788] [332.64107964] [438.35131252] [23998.9619874] [-415.0599551]] training loss: 17366813.45282496 validation loss: 20905056.738847155 test MSE: 18387.87397378454	Used features: age, BMI, children, smoker, region

Intercepts: [-2623.55100389] Weights: [[274.91244025] [-298.75853988] [477.39790463] [24009.01560562] [-158.73530214]] training loss: 19292484.791827664 validation loss: 23566495.242793098 test MSE: 1937779.968213982	Used features: age, sex, children, smoker, region
Intercepts: [-11499.96461412] Weights: [[261.06985265] [-355.70794064] [335.21136918] [24014.09686791] [-417.13365624]] training loss: 17487805.060798217 validation loss: 20886128.788246308 test MSE: 138362.44720553013	Used features: age, sex, BMI, smoker, region
Intercepts: [-7300.61578581] Weights: [[259.88308989] [935.60688226] [325.83112724] [333.95695541] [-518.57572441]] training loss: 62651163.206047155 validation loss: 66173761.142910026 test MSE: 50268095.96592337	Used features: age, sex, BMI, children, region
Intercepts: [-12185.87979373] Weights: [[260.57635914] [-378.50475257] [322.25988598] [443.44964443] [24045.87440746]] training loss: 17452991.80886793 validation loss: 21226585.52378672 test MSE: 100348.39304963441	Used features: age, sex, BMI, children, smoker
Using Single feature	
Intercepts: [2243.81167627] Weights: [[273.89057981]] training loss: 64784100.682424344 validation loss: 70639001.44945896 test MSE: 52378782.55245814	Used features: age
Intercepts: [12816.95032141] Weights: [[623.48248159]] training loss: 72345103.80626337 validation loss: 72564869.59674115 test MSE: 57303658.169101566	Used features: sex

Intercepts: [12607.56579355] Weights: [[488.95050298]] training loss: 72223340.54117437 validation loss: 72417523.67664002 test MSE: 57400506.51422108	Used features: children
Intercepts: [8435.77121276] Weights: [[23927.08027315]] training loss: 27256672.11805316 validation loss: 29383833.044153757 test MSE: 9264873.392777003	Used features: smoker
Intercepts: [13576.32419011] Weights: [[-292.36814679]] training loss: 72341034.87291823 validation loss: 72987638.8120458 test MSE: 57850267.869166344	Used features: region

Part. 2, Questions (30%):

(7%) 1. What's the difference between Gradient Descent, Mini-Batch Gradient Descent, and Stochastic Gradient Descent?

The difference between SGD and GD: when updating parameters, GD uses all the training data to calculate the loss function and update the weight, while SGD randomly selects a batch from the training data. When the training data is huge, SGD can update parameters faster.

Mini-batch gradient descent combines the concepts of SGD and GD. Divide the training data into small batches of fixed size, and calculate the average gradient for each batch.

(7%) 2. Will different values of learning rate affect the convergence of optimization? Please explain in detail.

The learning rate can be used to adjust the weights.

If the learning rate is too small, the weights change too slowly and the time to converge is too long. If the learning rate is too large, the weights change too fast, and the optimal solution may not be found.

(8%) 3. Suppose you are given a dataset with two variables, X and Y, and you want to perform linear regression to determine the relationship between these variables. You plot the data and notice that there is a strong nonlinear relationship between X and Y. Can you still use linear regression to analyze this data? Why or why not? Please explain in detail.

I wouldn't use a linear regression model analyze the data. Because linear regression is looking for a linear relationship between multiple independent variables and dependent variables.

When X and Y have a strong nonlinear relationship. Even with linear regression, the predicted results may not be accurate.

(8%) 4. In the coding part of this homework, we can notice that when we use more features in the data, we can usually achieve a lower training loss. Consider two sets of features, A and B, where B is a subset of A. (1) Prove that we can achieve a non-greater training loss when we use the features of set A rather than the features of set B. (2) In what situation will the two training losses be equal?

(1) Assume that using the set A of features has lower training loss than using the set B of features. And we train a model using set A of features and achieve a lower training loss than using set B of features.

However, because set B of features is a subset of set A of features, we can use set B of features to train the model. Apply the model to the data with set A of features and achieve a greater training loss than using the set B of features. This contradicts our hypothesis that using set B leads to lower training loss.

Therefore, we can conclude that we can achieve a non-greater training loss when we use the features of set A rather than the features of set B.

(2) When the set of features (A – B) and dependent variables have no linear relationship, the training loss of A and B features will be equal.