NYCU Pattern Recognition, Homework 1

311551170, 林琨堯

Part. 1, Coding (70%):

1. (0%) Show the learning rate and epoch you choose
   learning rate: 0.0007
   epoch: 200000

```
batch_size = x_train.shape[0]

# TODO
# Tune the parameters
# Refer to slide page 9
lr = 0.0007
epochs = 200000

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

```
Don't cheat.
```

0. (5%) Show the weights and intercepts of your linear model.
   weights: 1382.48659588
   intercepts: 380.13622217

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

```
Intercepts:  [1382.48659588]
Weights:  [[380.13622217]]
```

0. (5%) What's your final training loss (MSE)?
training loss (MSE): 139562065.48352107

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

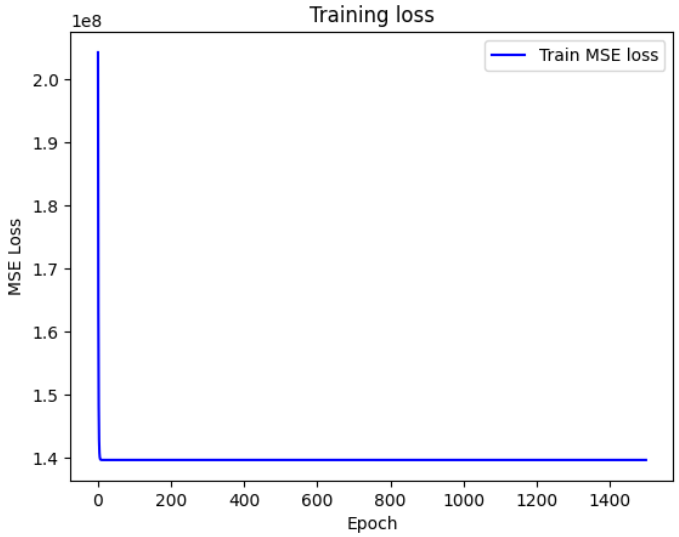
```
training loss:  139562065.48352107
```

0. (5%) What's the MSE of your validation prediction and validation ground truth?
validation loss (MSE): 136920277.39539412

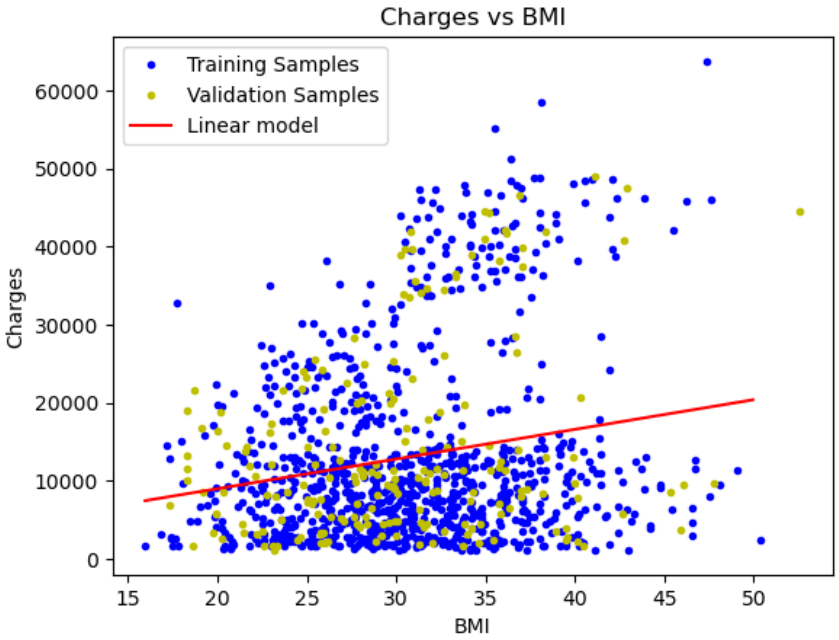
```
print('validation loss: ', linear_reg.evaluate(x_val, y_val))
```

```
validation loss:  136920277.39539412
```

0. (5%) Plot the training curve. (x-axis=epoch, y-axis=loss)



0. (5%) Plot the line you find with the training and validation data.



0. (0%) Show the learning rate and epoch you choose.

learning rate: 0.00037

epoch: 585000

```
batch_size = x_train.shape[0]

# TODO
# Tune the parameters
# Refer to slide page 10
lr = 0.00037
epochs = 585000

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

```
Don't cheat.
```

0. (10%) Show the weights and intercepts of your linear model.

weights: 259.85060879,

-383.5487657,

333.33128054,
442.55657777,
24032.21873189,
-416.01543495,

intercepts: -11857300272547

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

```
Intercepts:  [-11857.00272547]
Weights:  [[  259.85060879]
 [ -383.5487657 ]
 [  333.33128054]
 [  442.55657777]
 [24032.21873189]
 [ -416.01543495]]
```

0. (5%) What's your final training loss?

training loss (MSE): 34697170.25360506

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

```
training loss:  34697170.25360506
```

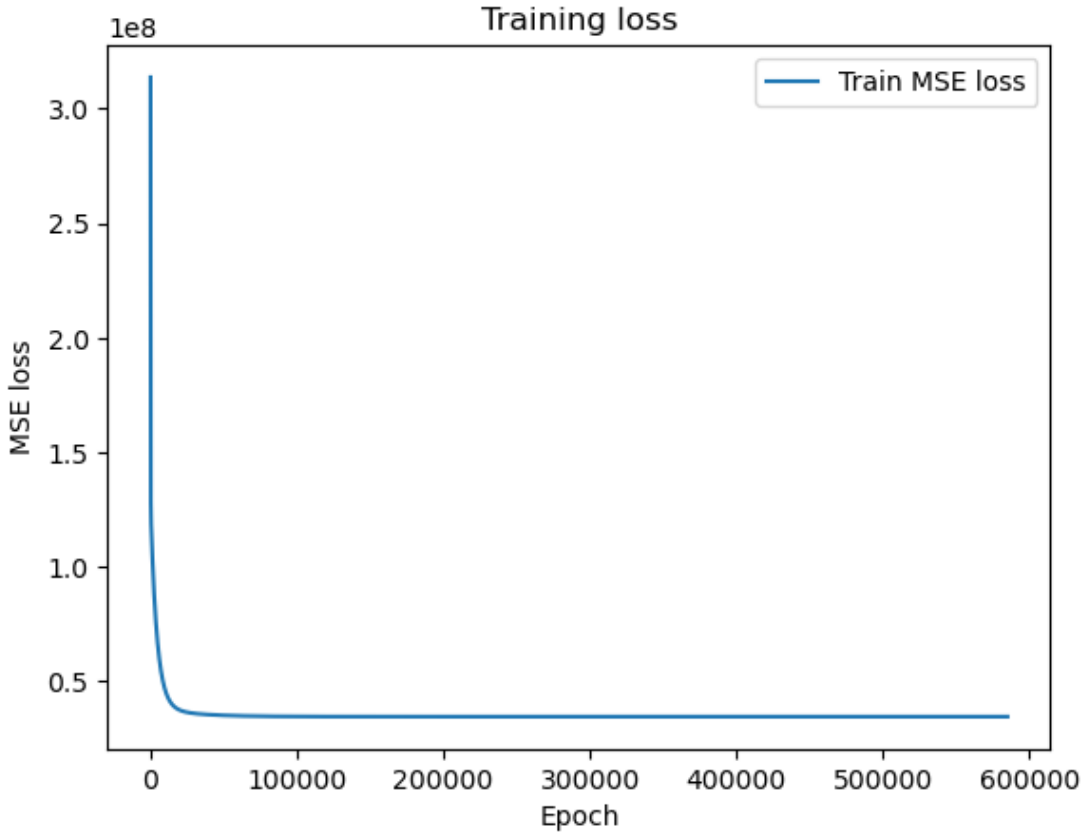
0. (5%) What's the MSE of your validation prediction and validation ground truth?

validation loss (MSE): 41958561.80088313

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

```
validation loss:  41958561.80088313
```

0. (5%) Plot the training curve. (x-axis=epoch, y-axis=loss)



0. (20%) Train your own model and fill the testing CSV file as your final predictions.

learning rate: _0.00007__

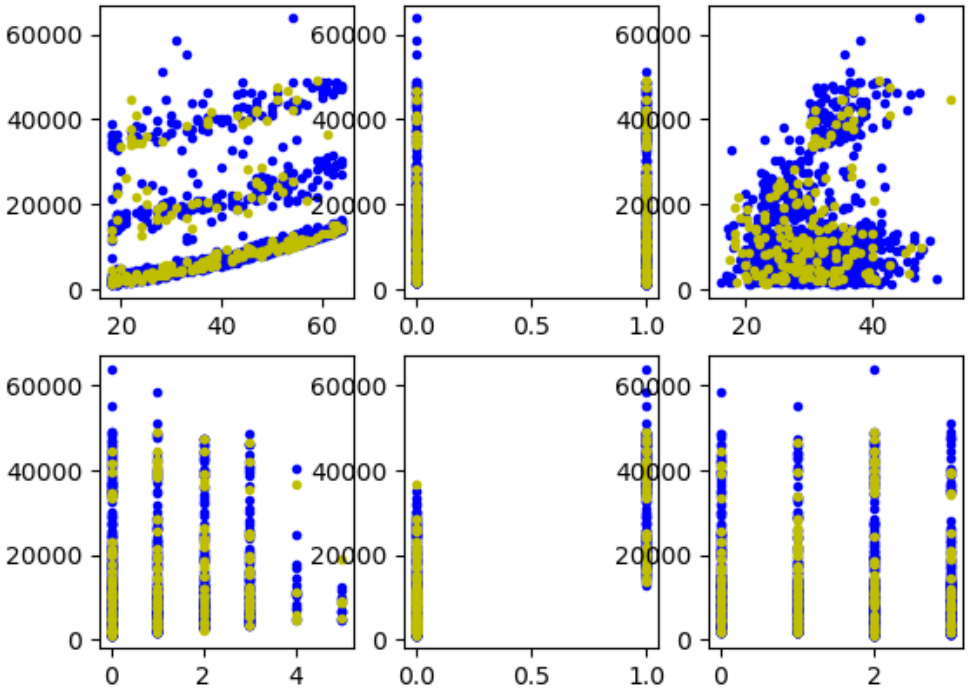
epoch: _20000__

batch_size: __50_

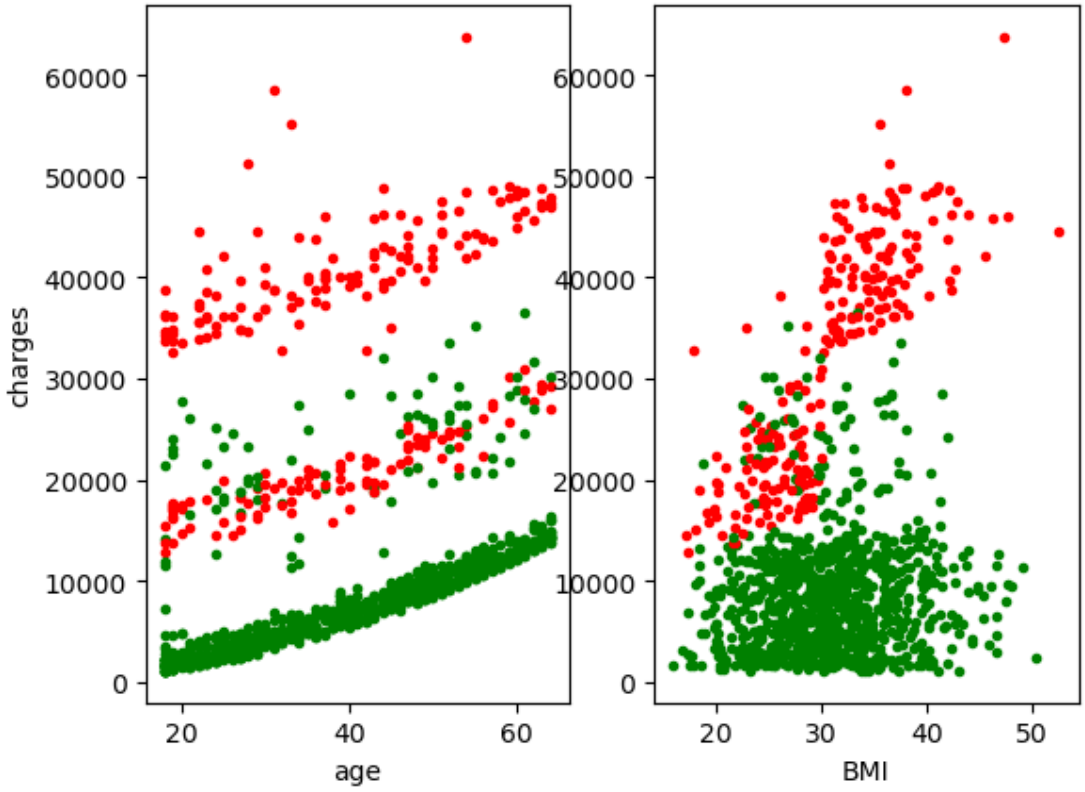
Used features: age、sex、smoker_bmi、children、region

What data analysis have you done? Why choose the above setting? Other strategies? (please explain in detail; otherwise, no points will be given.)

Analysis: Use a dot diagram to represent the relationship between each feature and charge. Age, BMI are positive correlation with charge, and smokers cost more than people who don't smoke.



Then, classify smoke(red)r or not(green). Smoking affect the relationship between BMI and charge more clearly.



So I merge smoker and BMI to a new feature, smoker_bmi.

Setting: Via experiment, batch_size=80 rise the loss, and batch_size=20 train slowly, so I set batch_size=50. Then, because of smoker_bmi, models converge easily, I choose epochs as small as possible. Then I modify lr to make the loss smaller.

Part. 2, Questions (30%):

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

The main difference between the gradient descent, stochastic gradient descent and mini-batch gradient descent is the batch size.

The gradient descent takes the whole dataset as training data, it will have the best accuracy of the other two methods, but when the data size or the iterations are too huge, it may take much time to calculate the result.

The batch size of the stochastic gradient descent is just one. It can improve the problem of gradient descent, but when the learning rate is too large, the weight may be fluctuating, it will never reach the minima but keep dancing around it.

The mini-batch gradient descent is the compromise of these two implementations, for each iteration, it uses a batch of fixed number of training data which is less than the actual data dataset, this way can keep the advantages of the gradient descent and the stochastic gradient descent.

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

Yes. The learning rate controls how quickly the model converges to optimization. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.
A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

(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.

It is not suitable for using linear regression to analyze a nonlinear relationship. The representation of linear regression is a straight line, and nonlinear relationship is a curve, so linear regression will not fit the nonlinear relationship well.

(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 features of A can be classified into three parts by the relationship with label, positive, negative and zero correlation. We can achieve a low training loss by giving the positive correlation features higher weights than the others. Therefore, no matter which part feature B belongs to, the training loss with B will not be lower

than the training loss with A.
(2) When B belongs to the positive correlation parts.