NYCU Pattern Recognition, Homework 1

311553032, 年晧鳴

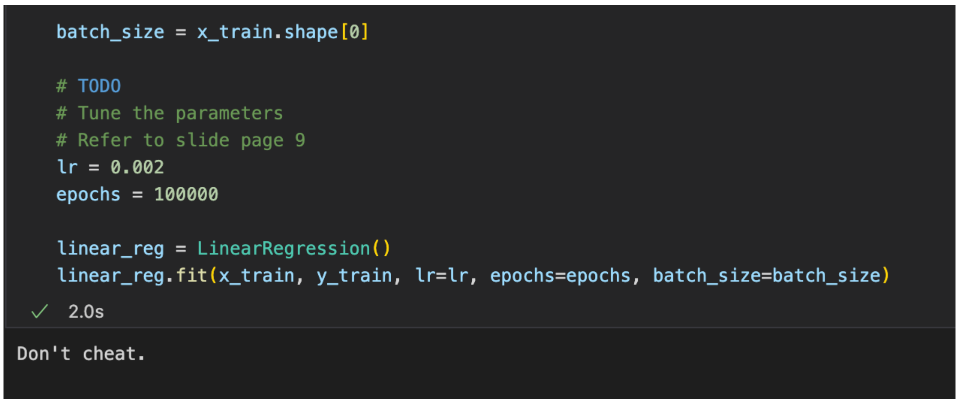
Part. 1, Coding (70%):

You should type the answer and also screenshot at the same time. Otherwise, no points will be given. The screenshot and the figures we provided below are just examples. **The results below are not guaranteed to be correct.** Please convert it to a pdf file before submission. You should use English to answer the questions. After reading this paragraph, you can delete it.

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

   learning rate: _0.002__

   epoch: __100000_

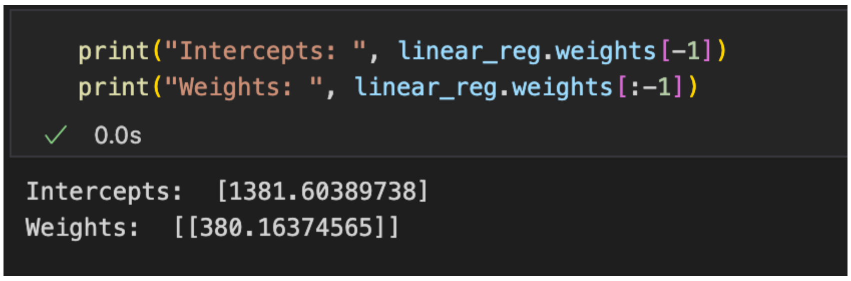


```
batch_size = x_train.shape[0]

# TODO
# Tune the parameters
# Refer to slide page 9
lr = 0.002
epochs = 100000

linear_reg = LinearRegression()
linear_reg.fit(x_train, y_train, lr=lr, epochs=epochs, batch_size=batch_size)
✓ 2.0s

Don't cheat.
```

2. (5%) Show the weights and intercepts of your linear model.

   weights: 1381.6

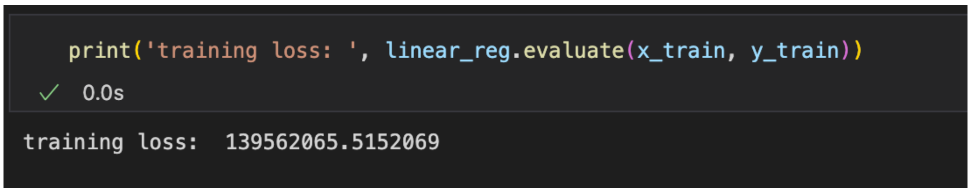
   intercepts: 380.1



```
print("Intercepts: ", linear_reg.weights[-1])
print("Weights: ", linear_reg.weights[:-1])
✓ 0.0s

Intercepts:  [1381.60389738]
Weights:  [[380.16374565]]
```

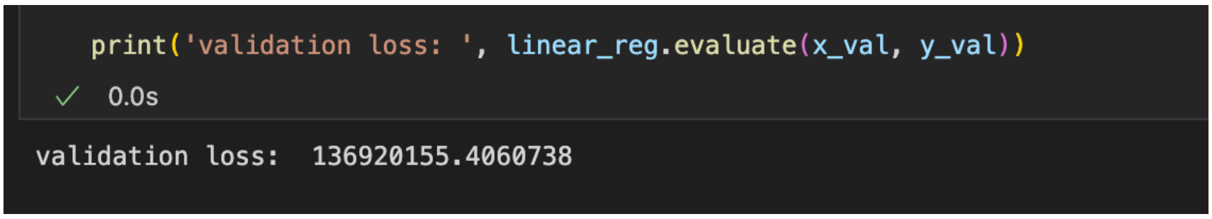
3. (5%) What' s your final training loss (MSE)?

   training loss (MSE): 139562065.5152069



```
print('training loss: ', linear_reg.evaluate(x_train, y_train))
✓ 0.0s

training loss:  139562065.5152069
```

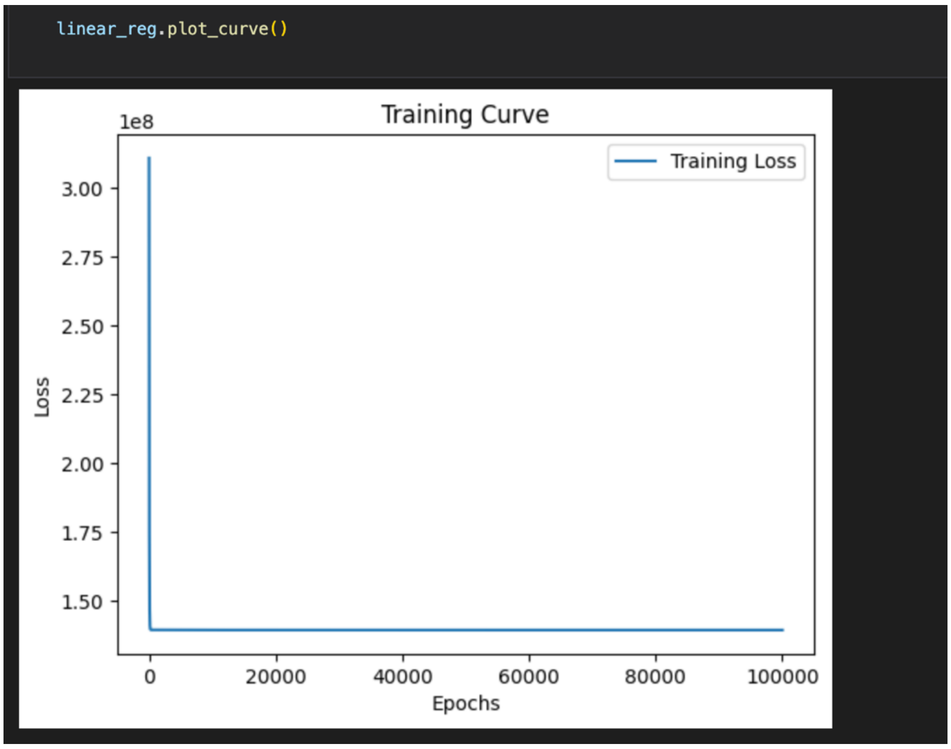
4. (5%) What’s the MSE of your validation prediction and validation ground truth?
   validation loss (MSE): 136920155.4060738



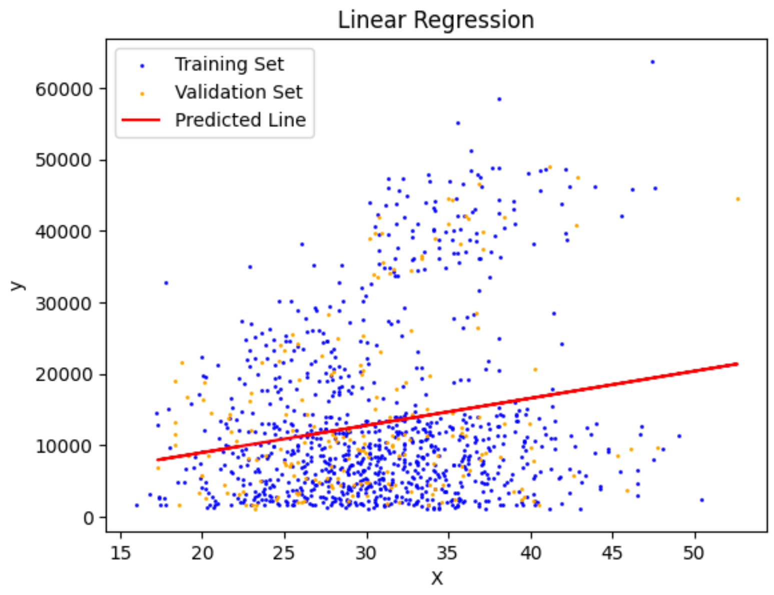
```
print('validation loss: ', linear_reg.evaluate(x_val, y_val))
✓ 0.0s
validation loss:  136920155.4060738
```

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

```
linear_reg.plot_curve()
```

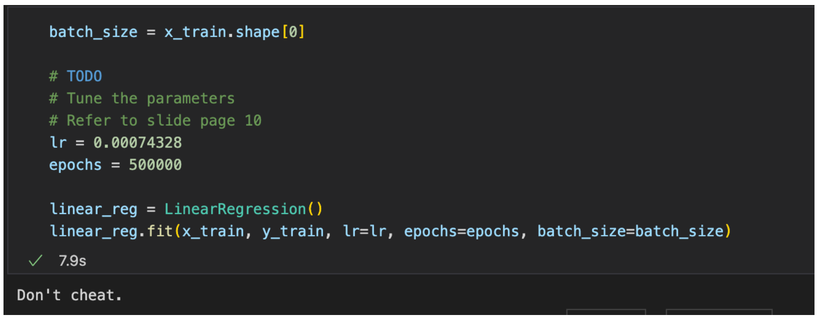


6. (5%) Plot the line you find with the training and validation data.



7. (0%) Show the learning rate and epoch you choose.
   learning rate: 0.00074328

epoch: 500000



```
batch_size = x_train.shape[0]

# TODO
# Tune the parameters
# Refer to slide page 10
lr = 0.00074328
epochs = 500000

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

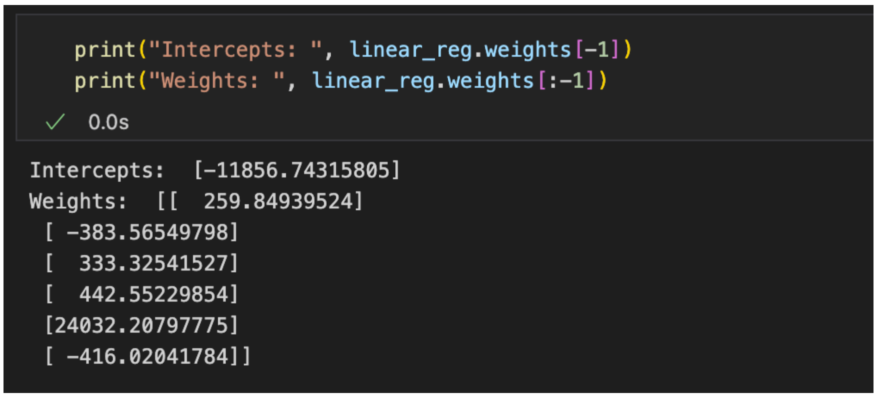
✓ 7.9s

```
Don't cheat.
```

8. (10%) Show the weights and intercepts of your linear model.

weights: [-11856.74315805]

intercepts: [[259.84939524] [-383.56549798] [333.32541527] [442.55229854] [24032.20797775] [-416.02041784]]



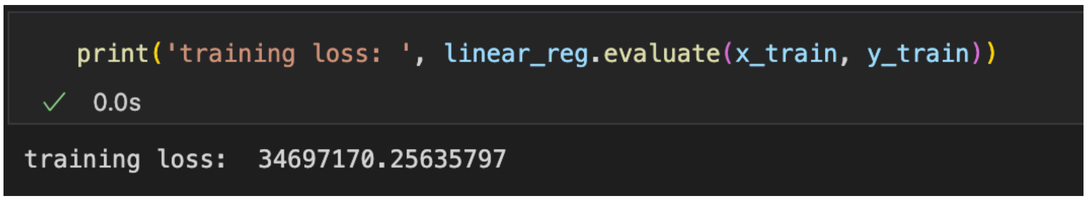
```
print("Intercepts: ", linear_reg.weights[-1])
print("Weights: ", linear_reg.weights[:-1])
```

✓ 0.0s

```
Intercepts:  [-11856.74315805]
Weights:  [[  259.84939524]
 [ -383.56549798]
 [  333.32541527]
 [  442.55229854]
 [24032.20797775]
 [ -416.02041784]]
```

9. (5%) What’s your final training loss?

training loss (MSE): 34697170.25635797



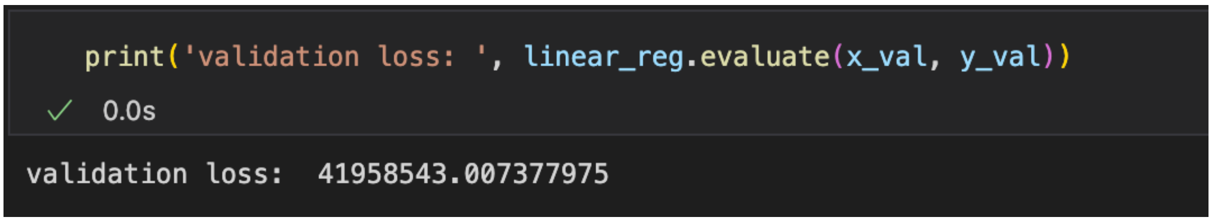
```
print('training loss: ', linear_reg.evaluate(x_train, y_train))
```

✓ 0.0s

```
training loss:  34697170.25635797
```

10. (5%) What’s the MSE of your validation prediction and validation ground truth?

validation loss (MSE): 41958543.007377975

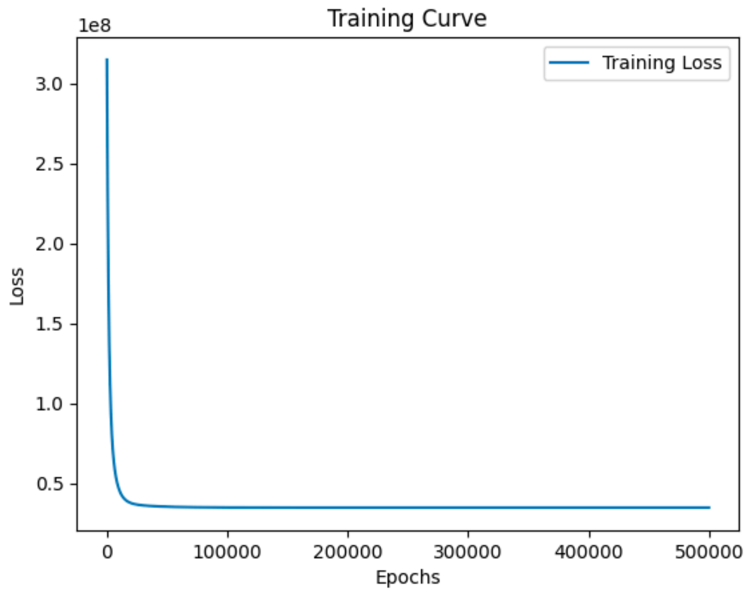


```
print('validation loss: ', linear_reg.evaluate(x_val, y_val))
```

✓ 0.0s

```
validation loss:  41958543.007377975
```

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



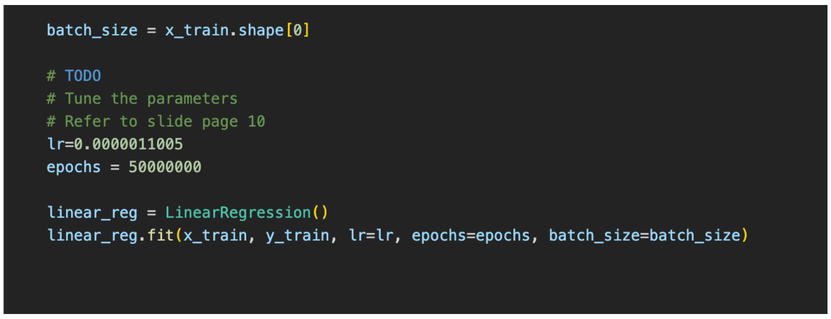
12. (20%) Train your own model and fill the testing CSV file as your final predictions.

learning rate: 0.0000011005

epoch: 50000000

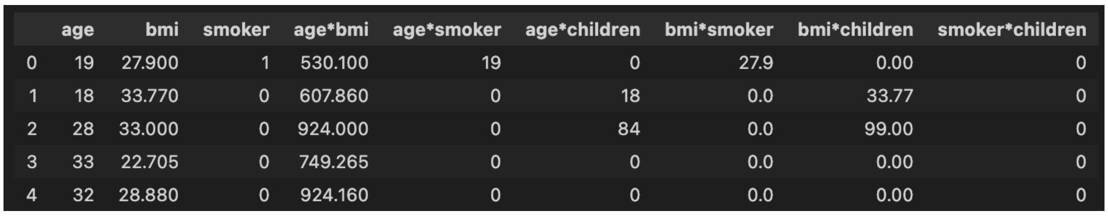
batch_size: X_train.shape[0]

because dataset size is lower ， and then linear regression not have local minimun， so use entire data.



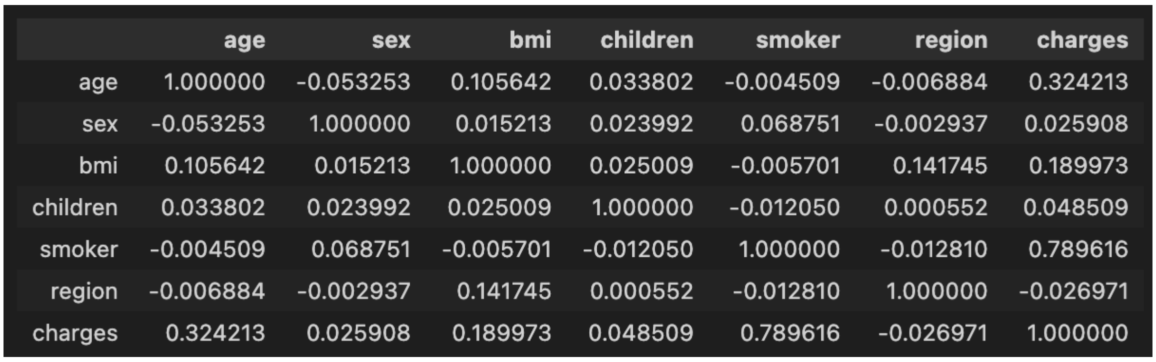
```
batch_size = x_train.shape[0]

# TODO
# Tune the parameters
# Refer to slide page 10
lr=0.0000011005
epochs = 50000000

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

Used features:



	age	bmi	smoker	age*bmi	age*smoker	age*children	bmi*smoker	bmi*children	smoker*children
0	19	27.900	1	530.100	19	0	27.9	0.00	0
1	18	33.770	0	607.860	0	18	0.0	33.77	0
2	28	33.000	0	924.000	0	84	0.0	99.00	0
3	33	22.705	0	749.265	0	0	0.0	0.00	0
4	32	28.880	0	924.160	0	0	0.0	0.00	0

In the table below, you can see that the relevance of each feature to charges is different, so I chose age bmi smoker children as a feature and used Polynomialfeatures to expand the fe ature and enhance it, and finally I found that children did not have a substantial help for tra ining Therefore, he was removed.



	age	sex	bmi	children	smoker	region	charges
age	1.000000	-0.053253	0.105642	0.033802	-0.004509	-0.006884	0.324213
sex	-0.053253	1.000000	0.015213	0.023992	0.068751	-0.002937	0.025908
bmi	0.105642	0.015213	1.000000	0.025009	-0.005701	0.141745	0.189973
children	0.033802	0.023992	0.025009	1.000000	-0.012050	0.000552	0.048509
smoker	-0.004509	0.068751	-0.005701	-0.012050	1.000000	-0.012810	0.789616
region	-0.006884	-0.002937	0.141745	0.000552	-0.012810	1.000000	-0.026971
charges	0.324213	0.025908	0.189973	0.048509	0.789616	-0.026971	1.000000

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

Part. 2, Questions (30%):

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

Gradient Descent uses the entire dataset in each iteration, Mini-Batch Gradient Descent uses small batches, and Stochastic Gradient Descent uses single data points.

Batch gradient descent : size = n

It does this using the entire training dataset in each iteration, which can be computationally expensive for la rge datasets.

Mini-batch gradient descent : size = 1 < batch size < n

It splits the training dataset into small batches or subsets of the data and computes the gradients and parame ter updates for each batch.

stochastic gradient descent : size = 1

updates the model parameters using a single randomly selected training instance or data point fro m the training dataset in each iteration. This approach is much faster than Gradient Descent, but it can result in more noise in the optimization process due to the high variance in the estimates of th e gradients.

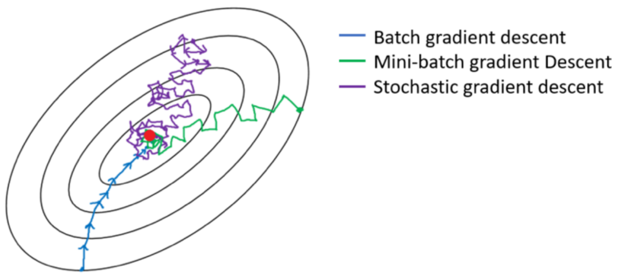
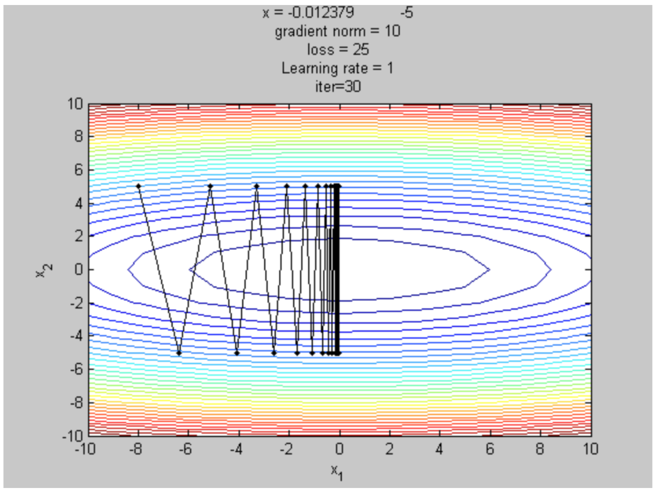


Figure by : https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

The choice of algorithm depends on the dataset size, computational resources available, and the desired spe ed and accuracy of the optimization process.

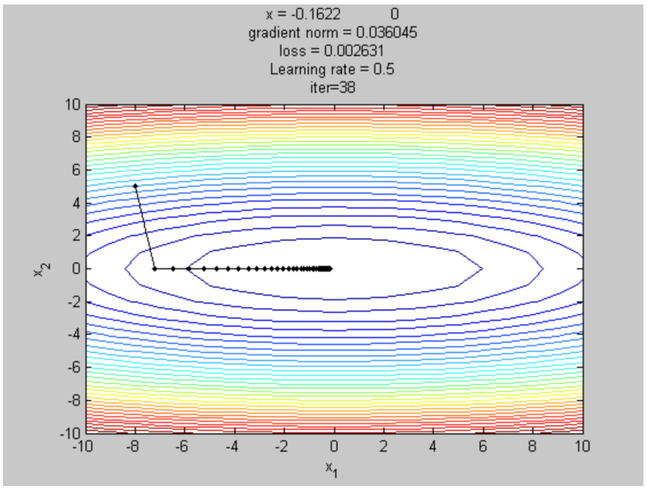
(7%) 2. Will different values of learning rate affect the convergence of optimization? Please explai n in detail.

A higher learning rate leads to faster convergence, but it also increases the risk of overshooting the minimum and causing the algorithm to fail to converge.



learning rate = 1

A lower learning rate, result in slower convergence, but it reduces the risk of overshooting the min imum and ensures more stable convergence.

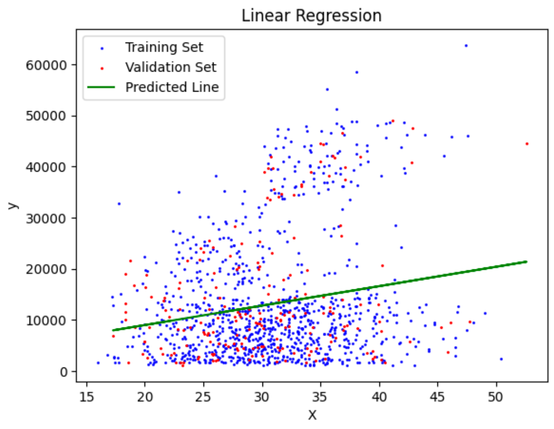


learning rate = 0.5

(8%) 3. Suppose you are given a dataset with two variables, X and Y, and you want to perform lin ear regression to determine the relationship between these variables. You plot the data and notice t hat 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 appropriate to use linear regression to analyze data.

like this photo :



Linear regression is not suitable for capturing non-linear relationships between variables. If you w ant to use linear regression to analyze the data, you can use Polynomial regression or use other no n-linear methods to analyze.

(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 featu res of set A rather than the features of set B. (2) In what situation will the two training losses be eq ual?

(1) : Because B is a subset of A, as long as the weight of the additional features of A-B are fixed to 0, whe n you put them into training, you can get the same weight parameters as B. Therefore, when you add a dditional features into training, you can have more parameters to predict the result, so the loss will be much lower than B.

In other words, using more features from set A can only improve the performance of the linear regressi on model, and it can never lead to a higher training loss compared to using fewer features from set B.

(2) : Including the additional features in set A will not improve the performance of the linear regression m odel, and it will not lead to a lower training loss compared to using only the features from set B. There fore, the two training losses will be equal when the additional features in set A do not contribute to the performance of the model.