M04W01 - Exercises

(Linear Regression)

Ngày 9 tháng 10 năm 2024

Giới thiệu về tập data: Cho trước dữ liệu về quảng cáo có 200 samples (rows) được lưu trong file advertising.csv, gồm 4 thông tin TV, Radio, Newspaper, và Sales (hình 1).

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9
8.7	48.9	75	7.2
57.5	32.8	23.5	11.8
120.2	19.6	11.6	13.2
8.6	2.1	1	4.8
199.8	2.6	21.2	15.6

Hình 1: Một vài sample data từ dữ liệu quảng cáo advertising.csv

Bài tập 1 (kỹ thuật đọc và xử lý dữ liệu từ file .csv): Cho trước file dữ liệu advertising.csv, hãy hoàn thành function **prepare_data(file_name_dataset)** trả về dữ liệu đã được tổ chức (X cho input và y cho output).

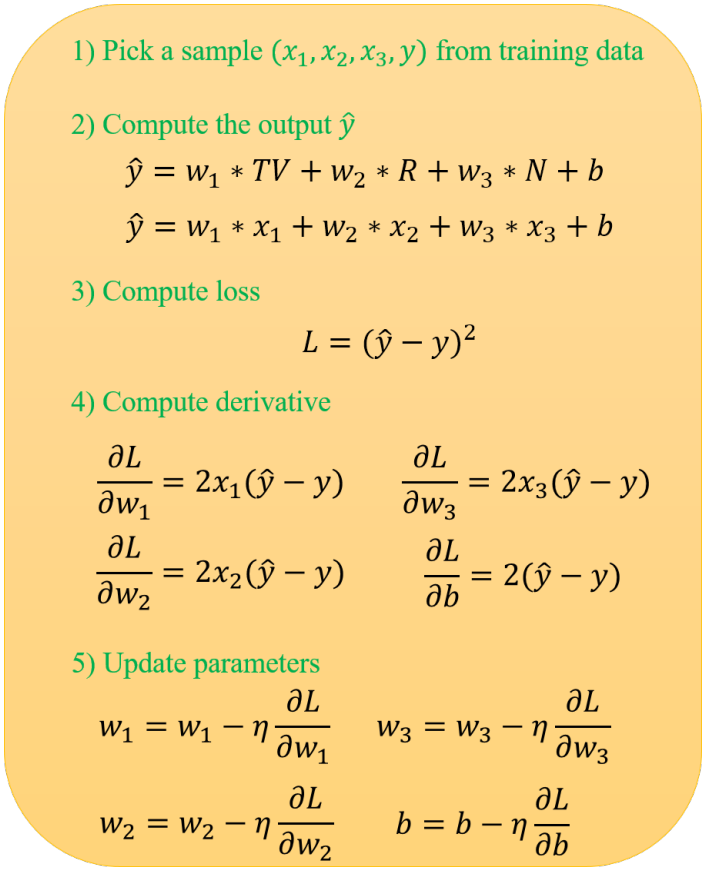
```

# dataset
import numpy as np
import matplotlib.pyplot as plt
import random

def get_column(data, index):

    #your code here ***************************

    return result

def prepare_data(file_name_dataset):
  data = np.genfromtxt(file_name_dataset, delimiter=',', skip_header=1).tolist()
  N = len(data)

  # get tv (index=0)
  tv_data = get_column(data, 0)

  # get radio (index=1)
  radio_data = get_column(data, 1)

  # get newspaper (index=2)
  newspaper_data = get_column(data, 2)
```

```
  # get sales (index=3)
  sales_data = get_column(data, 3)

  # building X input  and y output for training
  X = [tv_data, radio_data, newspaper_data]
  y = sales_data
  return X,y
```

```
Multiple choices:
Question 1:
X,y = prepare_data('advertising.csv')
list = [sum(X[0][:5]), sum(X[1][:5]), sum(X[2][:5]), sum(y[:5])]
print(list)
Select one of the following answers:

a) [624.1, 175.1, 300.5, 78.9]

b) [625, 175.1, 75.0, 7.2]

c) [626, 175.1, 75.0, 7.2]

d) [627.8, 175.1, 75.0, 7.2]
```

Bài tập 2 (kỹ thuật huấn luyện data dùng one sample - linear regression): Sử dụng kết quả dữ liệu đầu vào X, và dữ liệu đầu ra y từ bài 1, để phát triển chương trình dự đoán thông tin sales (y) từ X bằng cách dùng giải thuật linear regression with one sample-training với loss được tính bằng công thức Mean Squared Error $L = (\hat{y} - y)^2$. Sơ đồ hoạt động của giải thuật được mô tả ở hình 2. Nhiệm vụ của bạn là hoàn thành function **implement_linear_regression(X_data, y_data, epoch_max, lr)** và trả về 4 tham số w1, w2, w3, b và lịch sử tính loss như bên dưới.

```
def implement_linear_regression(X_data, y_data, epoch_max = 50,lr = 1e-5):
  losses = []

  w1, w2, w3, b = initialize_params()

  N = len(y_data)
  for epoch in range(epoch_max):
      for i in range(N):
          # get a sample
          x1 = X_data[0][i]
          x2 = X_data[1][i]
          x3 = X_data[2][i]

          y  = y_data[i]

          # compute output
          y_hat = predict(x1, x2, x3, w1, w2, w3, b)

          # compute loss
          loss = compute_loss_mse(y, y_hat)

          # compute gradient w1, w2, w3, b
          dl_dw1 = compute_gradient_wi(x1, y, y_hat)
          dl_dw2 = compute_gradient_wi(x2, y, y_hat)
          dl_dw3 = compute_gradient_wi(x3, y, y_hat)
          dl_db  = compute_gradient_b(y, y_hat)

          # update parameters
```



Hình 2: Các bước để thực hiện train linear regression model

```
            w1 = update_weight_wi(w1, dl_dw1, lr)
            w2 = update_weight_wi(w2, dl_dw2, lr)
            w3 = update_weight_wi(w3, dl_dw3, lr)
            b  = update_weight_b(b, dl_db, lr)

            # logging
            losses.append(loss)
    return (w1,w2,w3,b, losses)
```

Để hoàn thành function implement_linear_regression(X_data, y_data, epoch_max, lr) bạn cần phải hoàn thành cái sub-function sau đây:

2.1 Hoàn thành function **initialize_params()** để khởi tạo ngẫu nhiên giá trị ban đầu cho w1, w2, w3 theo gaussian random.gauss(mu=0.0, sigma=0.01) và b = 0. Ở bước này các bạn có thể dùng hàm sau để khởi tạo bốn tham số trên.

```
def initialize_params():
    w1 = random.gauss(mu=0.0, sigma=0.01)
    w2 = random.gauss(mu=0.0, sigma=0.01)
    w3 = random.gauss(mu=0.0, sigma=0.01)
    b  = 0
    return w1, w2, w3, b
```

Vì để thống nhất việc đánh giá kết quả cho toàn bài tập trong module này, các bạn được yêu cầu khởi tạo cố định wi, cũng như b như sau (trong thực tế bạn nên sử dụng hàm random phía trên nhé):

```
    def initialize_params():
        w1, w2, w3, b = (0.016992259082509283, 0.0070783670518262355,
```

```
                        -0.002307860847821344, 0)
    return w1, w2, w3, b
```

2.2 Hoàn thành function **predict(x1, x2, x3, w1, w2, w3, b)** để trả về kết quả dự đoán y tương ứng

```
def predict(x1, x2, x3, w1, w2, w3, b):

    # your code here ***************************

    return result
```

```
Multiple choices:
Question 2:
y = predict(x1=1, x2=1, x3=1, w1=0, w2=0.5, w3=0, b=0.5)
print(y)
Select one of the following answers:

a) 1.0

b) 2.0

c) 3.0

d) 4.0
```

2.3 Hoàn thành function **compute_loss(y_hat, y)** để tính loss giữa kết quả dự đoán y_hat và giá trị thực y, sử dụng Mean Squared Error

```
def compute_loss(y_hat, y):

    # your code here ***************************

    return loss
```

```
Multiple choices:
Question 3:
l = compute_loss(y_hat=1, y=0.5)
print(l)
Select one of the following answers:

a) 0.25

b) 0.26

c) 0.27

d) 0.28
```

2.4 Hoàn thành function **compute_gradient_wi(xi, y, y_hat))** để tính đạo hàm của hàm loss $L = (\hat{y} - y)^2$ theo wi và function **compute_gradient_b(y, y_hat)** để tính đạo hàm của hàm loss $L = (\hat{y} - y)^2$ theo b.

```
# compute gradient
def compute_gradient_wi(xi, y, y_hat):

    # your code here ***************************

    return dl_dwi

def compute_gradient_b(y, y_hat):

```

```
    # your code here ****************************

    return dl_db
```

```
Multiple choices:
Question 4:
# MSE loss
g_wi = compute_gradient_wi(xi=1.0, y=1.0, y_hat=0.5)
print(g_wi)
Select one of the following answers:

a) -1.0

b) -2.0

c) 1.0

d) 2.0
```

```
Multiple choices:
Question 5:
g_b = compute_gradient_b(y=2.0, y_hat=0.5)
print(g_b)
Select one of the following answers:

a) -1.0

b) -3.0

c) 1.0

d) 2.0
```

2.5 Hoàn thành function **update_weight_wi(wi, dl_dwi, lr)** để cập nhật wi sau khi tính đạo làm hàm loss L theo wi, và function **update_weight_b(b, dl_db, lr)** để update bias (b) sau khi tính đạo làm hàm loss L theo b.

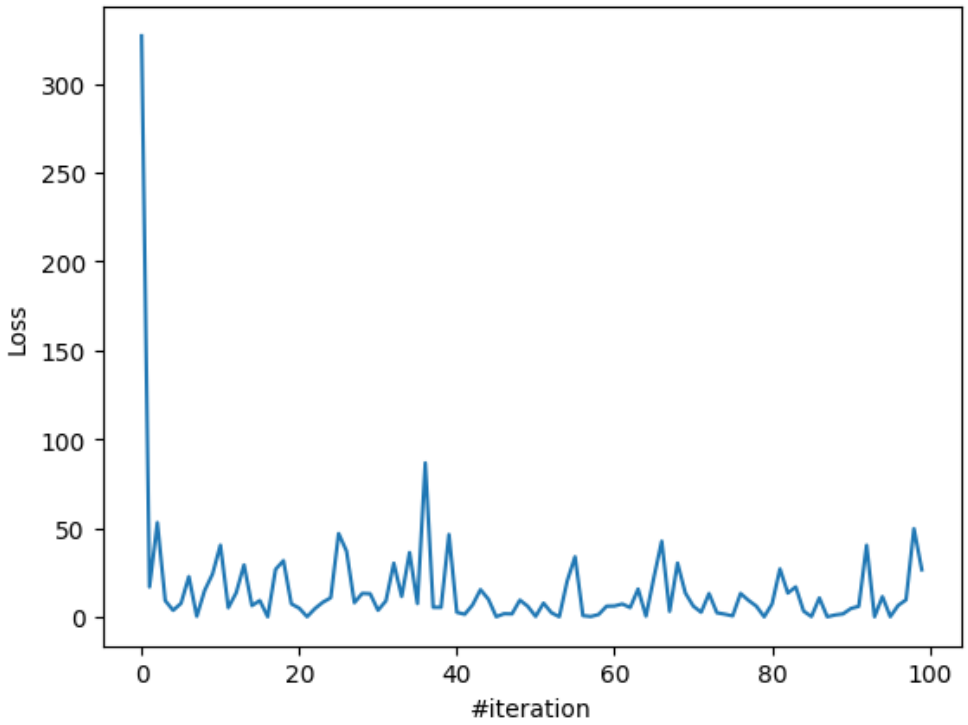
```

# update weights
def update_weight_wi(wi, dl_dwi, lr):

  # your code here ****************************

    return wi

def update_weight_b(b, dl_db, lr):

  # your code here ****************************

    return b
```

```
Multiple choices:
Question 6:
after_wi = update_weight_wi(wi=1.0, dl_dwi=-0.5, lr = 1e-5)
print(after_wi)
Select one of the following answers:

a) 1.000005

b) -3.0
```

```
c) 1.0

d) 2.0
```

```
Multiple choices:
Question 7:
after_b = update_weight_b(b=0.5, dl_db=-1.0, lr = 1e-5)
print(after_b)
print(after_wi)
Select one of the following answers:

a) 0.50001

b) -3.0

c) 1.0

d) 2.0
```

2.6 Thực hiện huấn luyện data bằng cách gọi hàm **implement_linear_regression(X, y)** và vẽ đồ thị kết quả cho 100 giá trị loss đầu tiên (loss cho 100 lần cập nhật đầu tiên) như bên hình 3.



Hình 3: Kết quả loss $L = (\hat{y} - y)^2$ sau 100 iteration cập nhật.

```
(w1,w2,w3,b, losses) = implement_linear_regression(X,y)
plt.plot(losses[:100])
plt.xlabel("#iteration")
plt.ylabel("Loss")
plt.show()
```

```
Multiple choices:
Question 8:
X,y = prepare_data('advertising.csv')
```

```
(w1,w2,w3,b, losses) = implement_linear_regression(X,y)
print(w1, w2, w3)
Select one of the following answers:

a) w1 = 0.074, w2 = 0.15, w3 = 0.17

b) w1 = 1.074, w2 = 1.15, w3 = 1.17

c) w1 = 2.074, w2 = 0.15, w3 = 2.17

d) w1 = 3.074, w2 = 0.15, w3 = 3.17
```

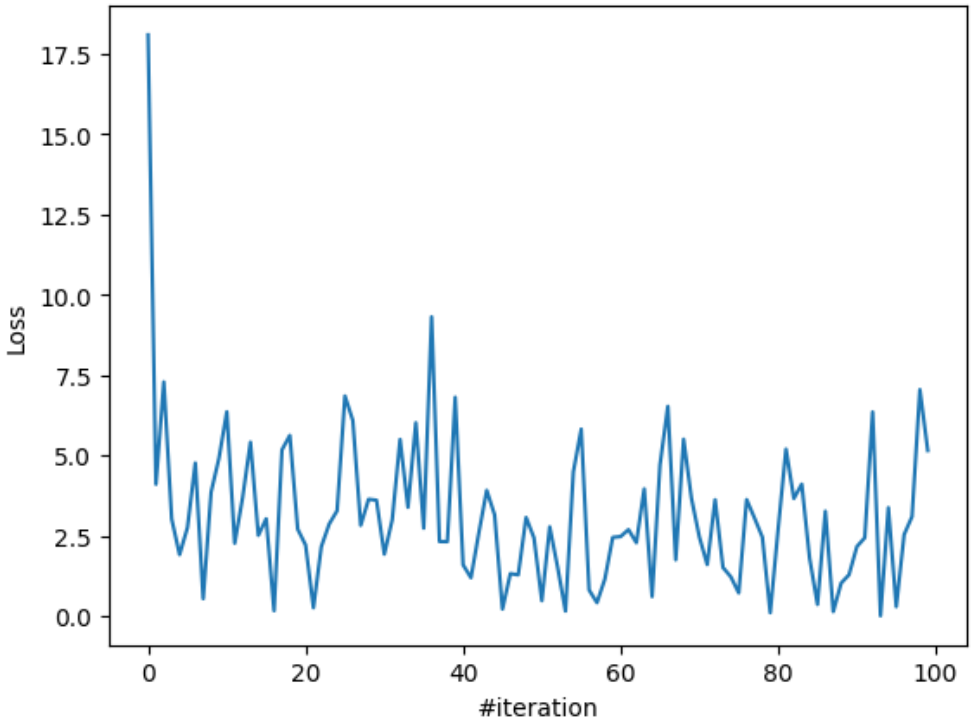
```
Multiple choices:
Question 9:

# given new data
tv = 19.2
radio = 35.9
newspaper = 51.3

X,y = prepare_data('advertising.csv')
(w1,w2,w3,b, losses) = implement_linear_regression(X, y, epoch_max=50, lr=1e-5)
sales = predict(tv, radio, newspaper, w1, w2, w3, b)
print(f'predicted sales is {sales}')

Select one of the following answers:

a) predicted sales is 6.18

b) predicted sales is 8.18

c) predicted sales is 7.18

d) predicted sales is 9.18
```

2.7 Thực hiện huấn luyện data bằng cách thay thế hàm loss L = $(\hat{y} - y)^2$ bằng hàm loss MAE L = $|\hat{y} - y|$ và vẽ đồ thị kết quả hàm loss trong 100 interation đầu tiên như bên hình 4.

```
Multiple choices:
Question 10:
l = compute_loss_mae(y_hat=1, y=0.5)
print(l)
Select one of the following answers:

a) 0.5

b) 1.4

c) 1.0

d) 2.0
```

Bài tập 3 (kỹ thuật huấn luyện data dùng batch N samples - linear regression): Cải tiến giải thuật ở bài tập 2, bằng cách huấn luyện giải thuật linear regression sử dụng N samples-training. Công việc của bạn ở bài tập này là bạn cần implement lại function **implement_linear_regression_nsamples** sử dụng N sample-training với MSE loss function $L(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y} - y)^2$ và MAE loss function (**optional**) $L(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N}|\hat{y} - y|$

```
1 def implement_linear_regression_nsamples(X_data, y_data, epoch_max=50, lr=1e-5):
```



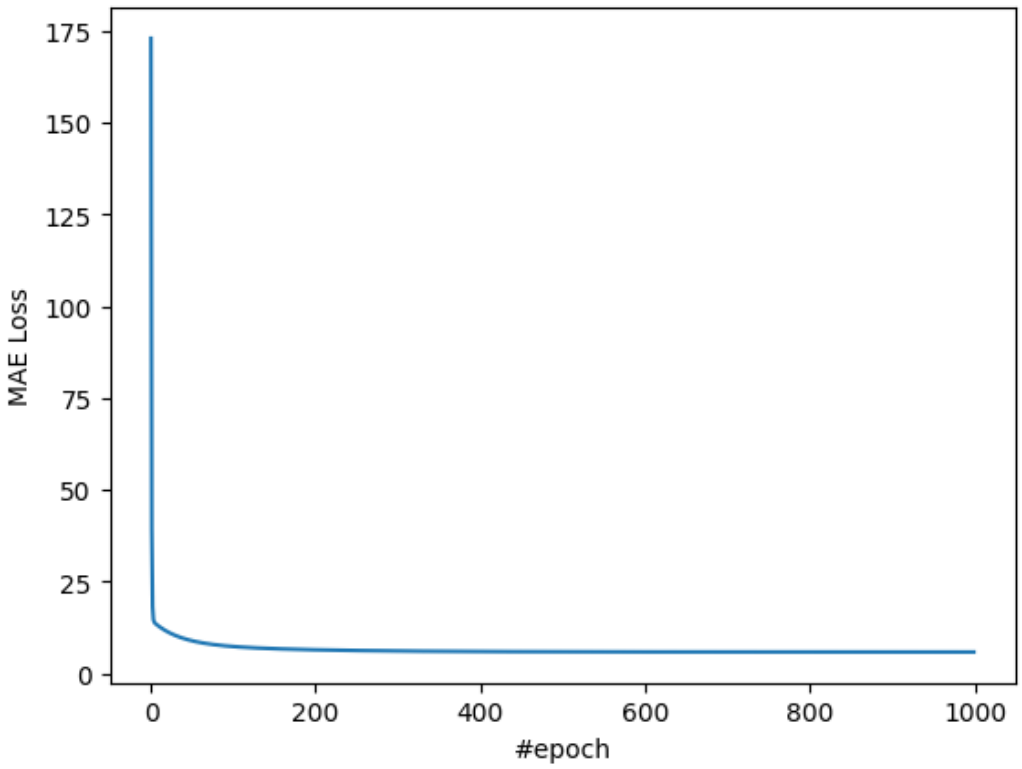
Hình 4: Kết quả loss MAE sau 100 iterations

```
losses = []

w1, w2, w3, b = initialize_params()
N = len(y_data)

for epoch in range(epoch_max):

    loss_total = 0.0
    dw1_total = 0.0
    dw2_total = 0.0
    dw3_total = 0.0
    db_total  = 0.0

    for i in range(N):
        # get a sample
        x1 = X_data[0][i]
        x2 = X_data[1][i]
        x3 = X_data[2][i]

        y  = y_data[i]

        # compute output
        y_hat = predict(x1, x2, x3, w1, w2, w3, b)

        # compute loss
        loss = compute_loss_mse(y, y_hat)

        # accumulate loss
        # your code here **************

        # compute gradient w1, w2, w3, b
```

```
            dl_dw1 = compute_gradient_wi(x1, y, y_hat)
            dl_dw2 = compute_gradient_wi(x2, y, y_hat)
            dl_dw3 = compute_gradient_wi(x3, y, y_hat)
            dl_db  = compute_gradient_b(y, y_hat)

            # accumulate gradient w1, w2, w3, b

            # your code here **************


        # (after processing N samples) - update parameters

        # your code here **************


        # logging
        losses.append(loss_total/N)
    return (w1,w2,w3,b, losses)
```

Thực thi đoạn code bên dưới để hiển thị kết quả huấn luyện (hình 5) sử dụng kỹ thuật N-sample training với loss function MSE và epoches = 1000.

```
(w1,w2,w3,b, losses) = implement_linear_regression_nsamples(X, y,
                                                             epoch_max=1000,
                                                             lr=1e-5)
print(losses)
plt.plot(losses)
plt.xlabel("#epoch")
plt.ylabel("MSE Loss")
plt.show()
```

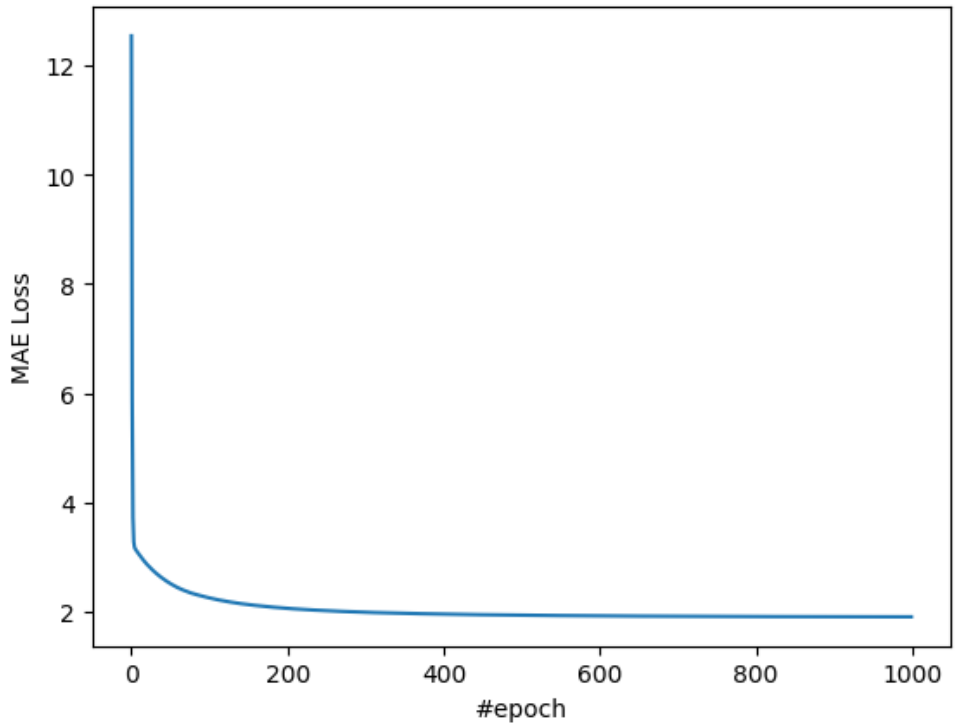


Hình 5: Kết quả MSE loss sau 1000 epoches sử dung N-sample training

```
Multiple choices:
Question 11:
X,y = prepare_data('advertising.csv')
#using MSE loss
(w1,w2,w3,b, losses) = implement_linear_regression_nsamples(X, y,
                                                             epoch_max=1000,
                                                             lr=1e-5)
print(w1,w2,w3)

Select one of the following answers:

a) w1 = 4.0786, w2 = 0.009, w3 = 3.387e-06

b) w1 = 1.0786, w2 = 0.009, w3 = 3.387e-06

c) w1 = 2.0786, w2 = 0.009, w3 = 3.387e-06

d) w1 = 0.0671, w2 = 0.1575, w3 = 0.0296
```

Thực thi đoạn code bên dưới để hiển thị kết quả huấn luyện (hình 6) sử dụng kỹ thuật N-sample training với loss function MAE và epoches = 1000.

```
(w1,w2,w3,b, losses) = implement_linear_regression_nsamples(X, y,
                                                             epoch_max=1000,
                                                             lr=1e-5)
print(losses)
plt.plot(losses)
plt.xlabel("#epoch")
plt.ylabel("MSE Loss")
plt.show()
```



Hình 6: Kết quả MAE loss sau 1000 epochs sử dung N-sample training

Bài tập 4 Như chúng ta đã biết, mục đích của linear regression là tìm hàm xấp xỉ y = ax1 + bx2 + cx3 + bx0. Trong đó x1 là TV, x2 là Radio, x3 là Newspapers, và x0 = 1. Đầu tiên, bạn cần tổ chức lại dữ liệu đầu vào ở bài tập 1 theo dạng danh sách các feature (x0, x1, x2, x3). Ví dụ theo hình 1, dữ liệu đầu vào dòng thứ 1 và 2 ta có thể tổ chức lại như sau:
X[0] = [1, x1, x2, x3] = [1, 230.1, 37.8, 69.2]
X[1] = [1, x1, x2, x3] = [1, 44.5, 39.3, 45.1]
....
X[199] = [1, x1, x2, x3] = [1, 232.1, 8.6, 8,7]

Để implement ý tưởng trên vào chương trình, bạn có thể sử dụng function bên dưới:

```
def prepare_data(file_name_dataset):
  data = np.genfromtxt(file_name_dataset, delimiter=',', skip_header=1).tolist()

  # get tv (index=0)
  tv_data = get_column(data, 0)

  # get radio (index=1)
  radio_data = get_column(data, 1)

  # get newspaper (index=2)
  newspaper_data = get_column(data, 2)

  # get sales (index=3)
  sales_data = get_column(data, 3)

  # building X input  and y output for training
  # Create list of features for input
  X = [[1, x1, x2, x3] for x1, x2, x3 in zip(tv_data, radio_data, newspaper_data)]
  y = sales_data
  return X,y
```

Song song đó, bạn có thể sử dụng hàm **initialize_params** để khởi tạo danh sách weights ban đầu chứa các giá trị bias, w1, w2, w3 như sau:

```
def initialize_params():
    bias = 0
    w1 = random.gauss(mu=0.0, sigma=0.01)
    w2 = random.gauss(mu=0.0, sigma=0.01)
    w3 = random.gauss(mu=0.0, sigma=0.01)

    # comment this line for real application
    return [0, -0.01268850433497871, 0.004752496982185252, 0.0073796171538643845]
    # return [bias, w1, w2, w3]
```

Nhiệm vụ của bạn ở bài tập này là cần hoàn thành lại các function sau đây:

```
#Predict output by using y = x0*b + x1*w1 + x2*w2 + x3*w3

def predict(X_features, weights):

    #your code here ......

    return result

def compute_loss(y_hat, y):
    return (y_hat - y)**2

# compute gradient
def compute_gradient_w(X_features, y, y_hat):

```

```
    #your code here ......

    return dl_dweights

# update weights
def update_weight(weights, dl_dweights, lr):

    #your code here ......

    return weights
```

Sau khi hiện thực xong các function trên, bạn tiến hành chạy function bên dưới để tiến hành huấn luyện dữ liệu và trực quan hoá kết quả:

```
def implement_linear_regression(X_feature, y_ouput, epoch_max=50,lr=1e-5):

  losses = []
  weights = initialize_params()
  N = len(y_ouput)
  for epoch in range(epoch_max):
      print("epoch", epoch)
      for i in range(N):
          # get a sample - row i
          features_i = X_feature[i]
          y = sales_data[i]

          # compute output
          y_hat = predict(features_i, weights)

          # compute loss
          loss = compute_loss(y, y_hat)

          # compute gradient w1, w2, w3, b
          dl_dweights = compute_gradient_w(features_i, y, y_hat)

          # update parameters
          weights = update_weight(weights, dl_dweights, lr)

          # logging
          losses.append(loss)
  return weights, losses

X,y = prepare_data('advertising.csv')
W,L = implement_linear_regression(X,y)
plt.plot(L[0:100])
plt.xlabel("#iteration")
plt.ylabel("Loss")
plt.show()
```

Nếu chương trình bạn chính xác, bạn sẽ có được kết quả giống như hình 3 nhé.

```
Multiple choices:
Question 12:
X,y = prepare_data('advertising.csv')
W,L = implement_linear_regression(X, y, epoch_max=50, lr=1e-5)
# Print loss value at iteration 9999
print(L[9999])
Select one of the following answers:

a) 31.33

b) 33.33
```

```
c) 34.33

d) 35.33
```

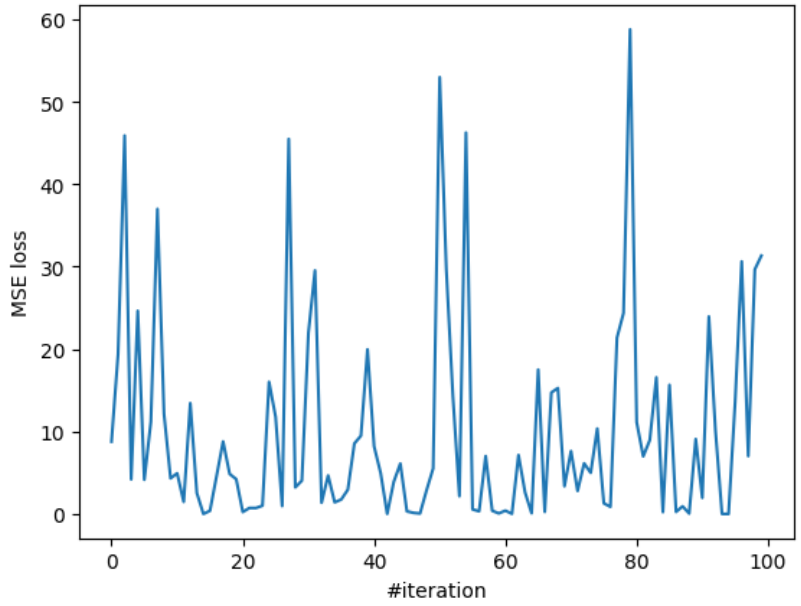
Bài tập 5 Tìm hiểu kỹ thuật Feature Scaling thông qua min and max(**optional):** Ở bài tập này các bạn cần phải chuẩn hoá dữ liệu đầu vào X trước thuật nhằm tăng tốc độ hội tụ của giải thuật linear regression. Yêu cầu của bài tập là các bạn cần chuẩn hoá data theo MinMax scale $X_{new} = \frac{X_{old} - X_{min}}{X_{max} - X_{min}}$ trước khi đưa vào huấn luyện data. Nhiệm vụ của bạn ở bài tập này là cần implement hàm **min_max_scaling()** để chuẩn hoá dữ liệu input X như bên dưới:

```

def prepare_data(file_name_dataset):
  data = np.genfromtxt(file_name_dataset, delimiter=',', skip_header=1).tolist()

  # get tv (index=0)
  tv_data = get_column(data, 0)

  # get radio (index=1)
  radio_data = get_column(data, 1)

  # get newspaper (index=2)
  newspaper_data = get_column(data, 2)

  # get sales (index=3)
  sales_data = get_column(data, 3)

  # scale data (only for features)
  # remenber to scale input features in inference, therefore, we need to save max, min values
  (tv_data, radio_data, newspaper_data), (max_data_1, max_data_2, max_data_3, min_data_1, min_data_2, min_data_3) = min_max_scaling(tv_data,radio_data, newspaper_data)

  # building X input  and y output for training
  # Create a list of features for input
  X = [[1, x1, x2, x3] for x1, x2, x3 in zip(tv_data, radio_data, newspaper_data)]
  y = sales_data
  return X,y
```

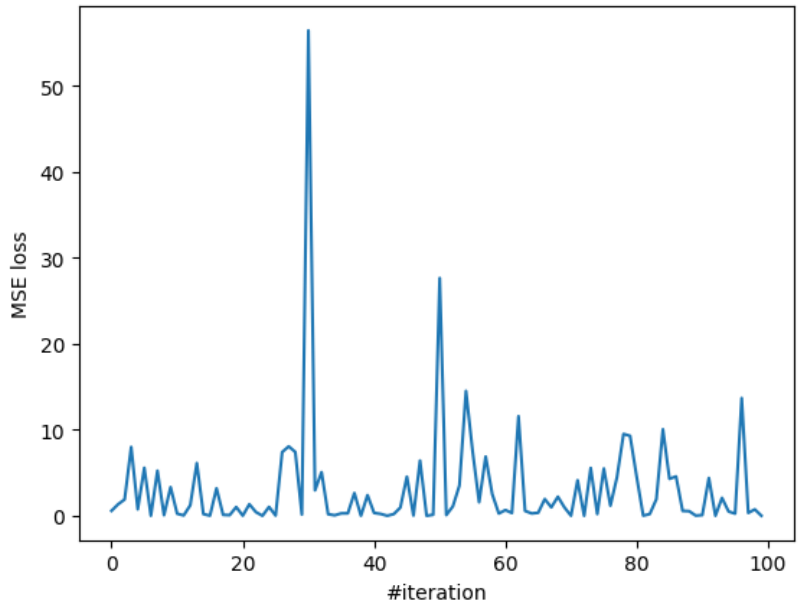
```

def min_max_scaling(data1, data2, data3):
    max_data_1 = max(data1)
    max_data_2 = max(data2)
    max_data_3 = max(data3)

    min_data_1 = min(data1)
    min_data_2 = min(data2)
    min_data_3 = min(data3)


    # Implement your code to normalize data1, data2, data3 using min and max value

    return (data1, data2, data3), (max_data_1, max_data_2, max_data_3, min_data_1, min_data_2, min_data_3)
```

Hình 7 Hiển thị kết quả hàm Loss ở 100 iteration cuối trước khi thực hiện feature scaling. Bạn có thể quan sát và thấy rằng hàm loss không ổn định và rất khó tiến hội tụ. Trong khi đó hình 8 hiển thị kết

quả hàm Loss ở 100 iteration cuối sau khi thực hiện feature scaling. Bạn có thể quan sát và thấy rằng hàm loss ổn định hơn nhiều.



Hình 7: Kết quả MSE Loss ở 100 iteration cuối trước khi thực hiện fearure-scaling.



Hình 8: Kết quả MSE Loss ở 100 iteration cuối trước sau thực hiện fearure-scaling.

```
X,y = prepare_data('advertising.csv')
W,L = implement_linear_regression(X,y)
plt.plot(L[-100:])
plt.xlabel("#iteration")
plt.ylabel("MSE loss")
plt.show()
```