Chương 5
Gradient Descent

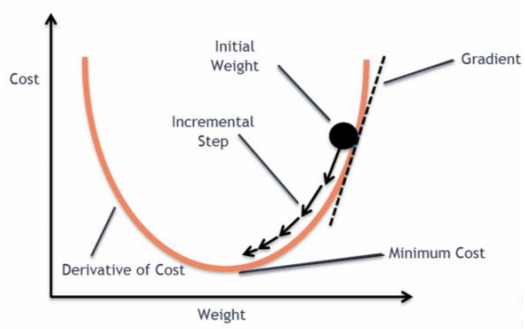
TS. Phạm Tuấn

ptuan@ute.udn.vn

Giới thiệu

Gradient Descent là một kỹ thuật cơ bản, rất quan trọng và cũng như là phần lõi của của các thuật toán máy học.

Tối ưu (Optimization) là một phần quan trọng trong Học máy cũng như Trí tuệ nhân tạo. Tối ưu thường được hiểu đơn giản là tìm điểm cực trị của một hàm số. Làm thế nào để tìm cực trị của một hàm số ?. Và **gradient descent** một kỹ thuật mạnh mẽ giúp chúng ta trong bài toán tối ưu.



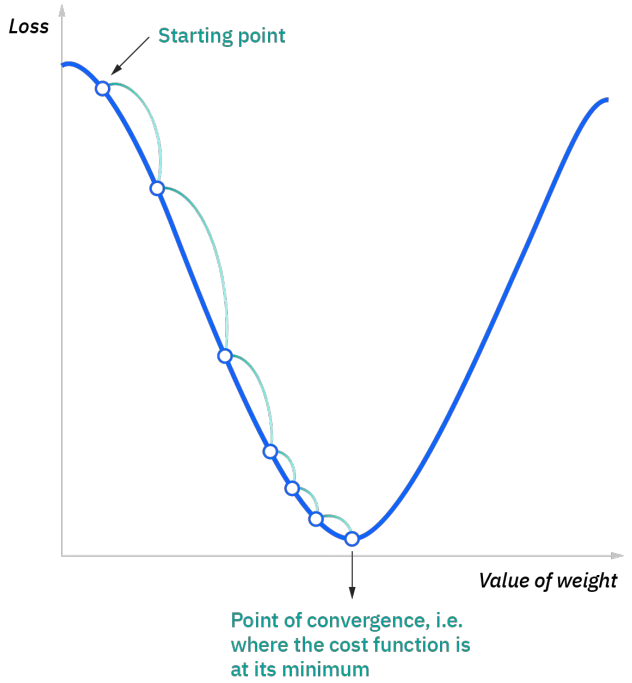
Ý tưởng

Tưởng tượng chúng ta đang ở vị trí nào đó của một thung lũng và chúng ta muốn tìm điểm thấp nhất của thung lũng đó.

Tại điểm đang đứng, **chúng ta tìm điểm lân cận thấp hơn và chúng ta bước xuống**.

Ta lặp đi lặp lại quá trình đó nhiều lần thì chúng ta sẽ đến điểm thấp nhất của thung lũng đó.

Gradient descent hoạt động dựa trên ý tưởng như vậy.

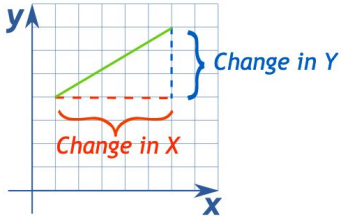


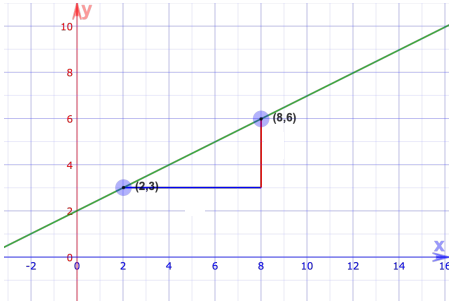
Gradient là gì ?

Gradient (còn được gọi là Slope) của một đường thẳng thì chính là dốc của của đường thẳng đó.

Gradient được tính toán bởi sự thay đổi của Y chia cho sự thay đổi của X.

$$\textbf{Gradient} = \frac{\text{Change in Y}}{\text{Change in X}}$$

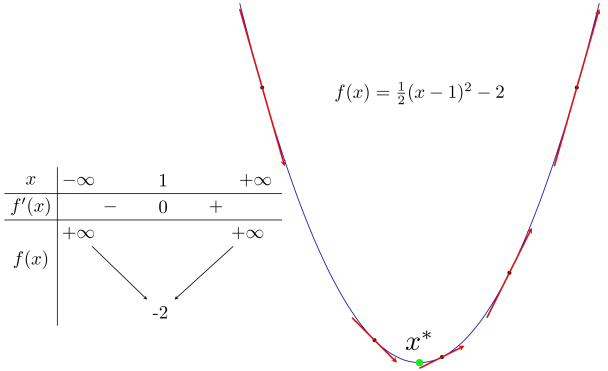




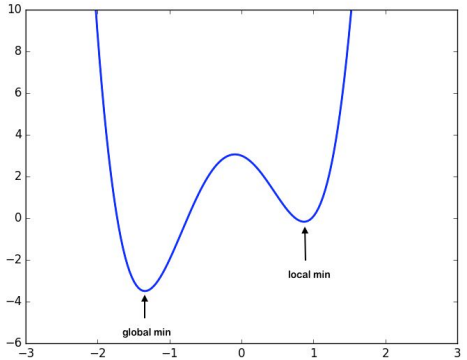
Gradient là gì ?

Giá trị đạo hàm thường có cực khác nhau: **âm** và **dương**.

Hàm mô hình thường có 1 hay nhiều điểm cực trị



Bảng đạo hàm và hàm số



Global vs local minima

Hàm mất mát và hàm mô hình

Trong trí tuệ nhân tạo có 2 hàm số mà chúng ta phải thường xuyên làm việc đó là:

1. Hàm mô hình
2. Hàm mất mát (loss function)

Chúng ta lấy ví dụ với bài toán simple linear regression.

$$y = w_0 + xw_1$$

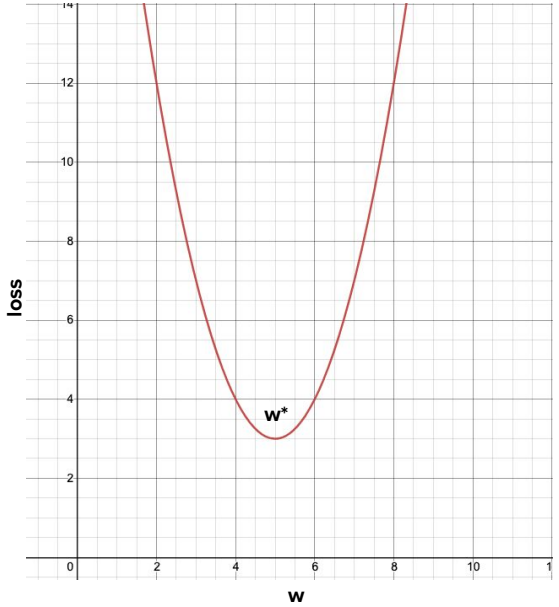
$$L = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

Điểm cực trị

Gọi **w*** là điểm cực tiểu, cũng chính là nơi đạo hàm của hàm mất mát bằng không.

Đạo hàm của các điểm bên trái **w*** là không dương và bên phải **w*** là không âm.

Để đi đến cực tiểu thì hướng di chuyển ngược dấu với hướng của đạo hàm ?

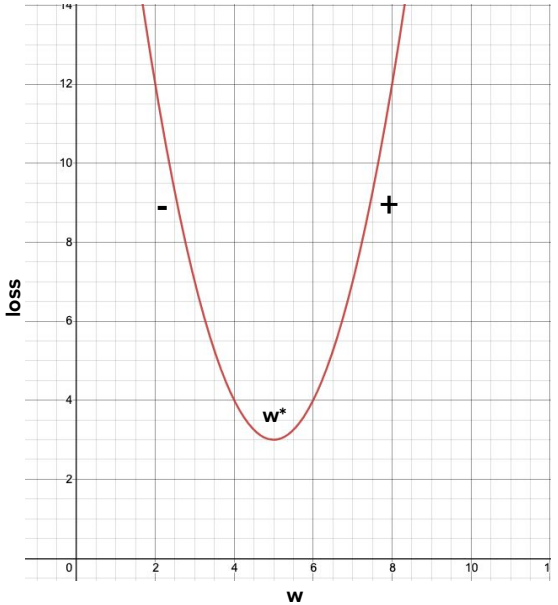


Ví dụ cho hàm 1 biến

Ta gọi $\Delta = w_{t+1} - w_t$ là lượng dịch chuyển, và ta luôn muốn w_{t+1} càng tiến tới gần w* => Δ ngược dấu với dấu của đạo hàm $L(w_t)'$.

Do đó, chúng ta đặt: $\Delta = -\alpha L(w_t)' = w_{t+1} - w_t$

$$w_{t+1} = w_t - \alpha L(w_t)'$$



Công thức tổng quát

Bên dưới là công thức tổng quát cho Gradient descent với hàm một biến

$$w_{t+1} = w_t - \alpha \frac{\partial L(w_t)}{\partial w}$$

Trong đó **alpha** được gọi là **learning rate**, w_t là vị trí cũ và w_{t+1} là vị trí mới.

Pseudo code

- Thiết lập các giá trị: alpha, w_init, iteration, tolerance
- Tính đạo hàm của hàm mất mát *L*’
- For step in range(iteration):
  - w_new = w_old - alpha*loss(w_old)’
  - If (w_new-w_old)<tolerance:
    - break
- Output = w_new

Lưu ý

- Điểm khởi tạo
- Learning rate

Ví dụ trên Python

Xét hàm số như sau:

$$f(w) = w^2 + 5\sin(w)$$

Chúng ta tính đạo hàm của hàm số trên ta được:

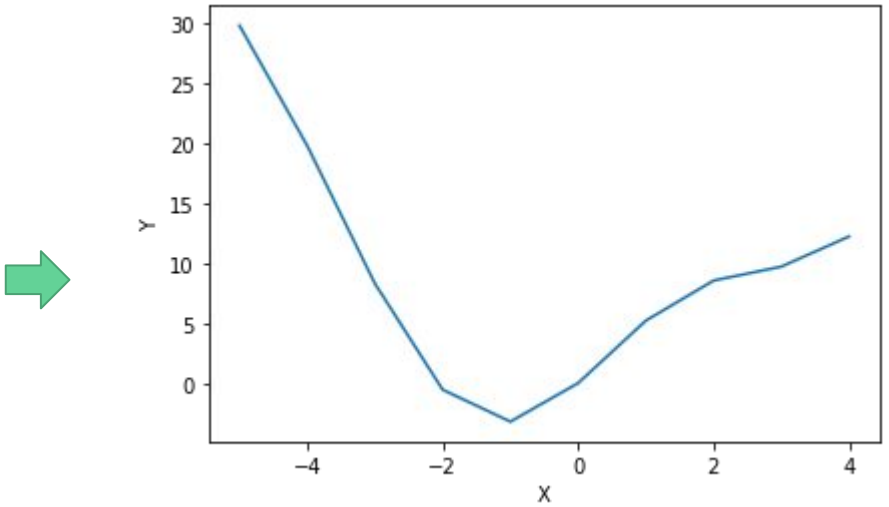
$$f(w)' = 2w + 5\cos(w)$$

Cuối cùng, chúng ta tìm giá trị mới của w bằng hàm cập nhật như sau:

$$w_{t+1} = w_t - \alpha\left(2w + 5\cos(w_t)\right)$$

Ví dụ trên Python

```
import math
import numpy as np
import matplotlib.pyplot as plt
def grad(w):
    return 2*w+ 5*np.cos(w)

def loss(w):
    return w**2 + 5*np.sin(w)

x_axis = np.arange(-5,5)
y_axis = loss(x_axis)
plt.plot(x_axis, y_axis)
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```



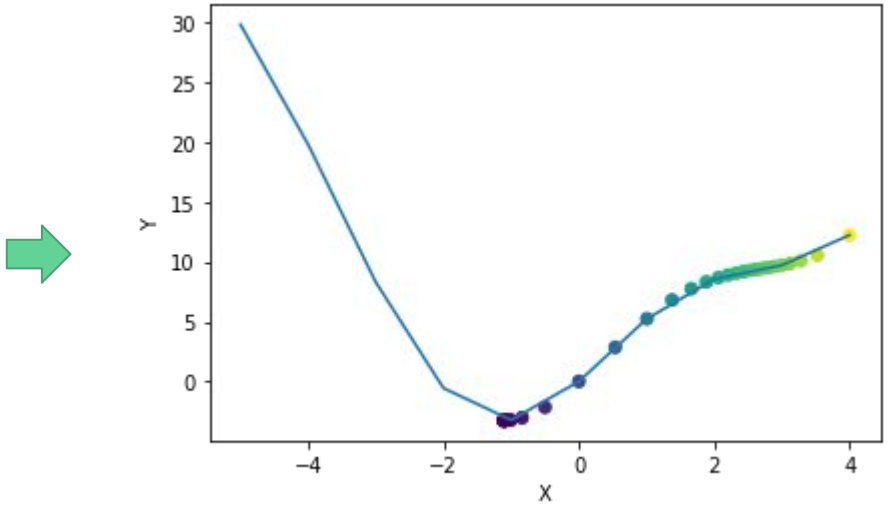
Ví dụ trên Python

```
def gradient_descent(alpha, w_init, iteration, tolerance):
    w_list = [w_init]
    for it in range(iteration):
#         print('loss = {:.3f} at iteration {:d}'.format(loss(w_list[-1]), it))
        w_new = w_list[-1] - alpha*grad(w_list[-1])
        if abs(grad(w_new)) < tolerance:
            break
        w_list.append(w_new)
    return w_list, it

alpha = 0.1
w_init = 4
tolerance = 0.001
iteration = 100
w_list, iteration_number = gradient_descent(alpha, w_init,
iteration,tolerance)

print('Minima at w = {:.3f}, loss = {:.3f}, obtained after {:d} iterations'\
      .format(w_list[-1], loss(w_list[-1]), iteration_number))
```

Minima at w = -1.110, loss = -3.246, obtained after 28 iterations

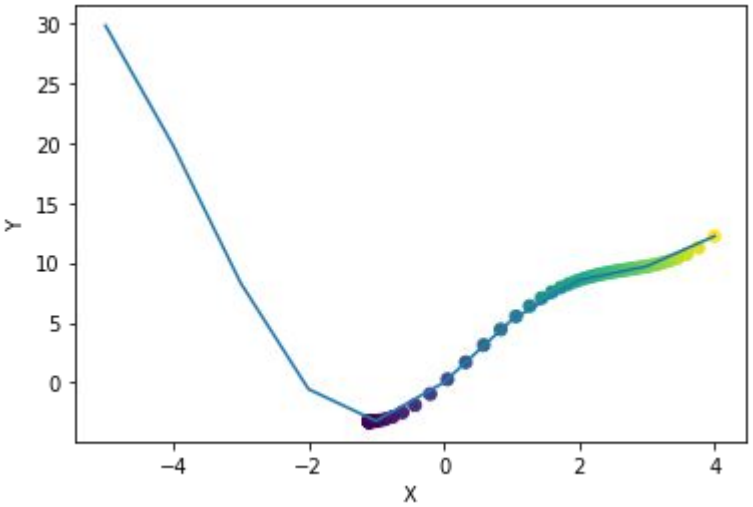
Ví dụ trên Python

```
loss_list = loss(np.array(w_list))
plt.scatter(w_list, loss_list, c = w_list)
plt.plot(x_axis, y_axis)
plt.xlabel('X')
plt.ylabel('Y')
plt.figure(figsize=(5,5), dpi=300)
plt.show()
```



Ví dụ trên Python

```
alpha = 0.05
w_list, iteration_number = gradient_descent(alpha,
w_init, iteration,tolerance)

loss_list = loss(np.array(w_list))
plt.scatter(w_list, loss_list, c = w_list)
plt.plot(x_axis, y_axis)
plt.xlabel('X')
plt.ylabel('Y')
plt.figure(figsize=(5,5), dpi=300)
plt.show()
```



Bài tập nhóm số 5

1. Download file dữ liệu tại đây. Dùng Pandas đọc file. (1 điểm)
2. Dùng sklearn để xây dựng hàm tuyến tính, cũng như print ra giá trị của các biến số. (1 điểm)
3. Vẽ các điểm và đường thẳng tuyến tính đã giải bằng sklearn.(1 điểm)
4. Viết hàm đạo hàm. (1 điểm)
5. Xây dựng hàm gradient descent để tìm lời giải cho hàm tuyến tính, và print ra giá trị. (4 điểm)
6. Vẽ các điểm và đường thẳng tuyến tính đã giải bằng gradient descent.(1 điểm)
7. Vẽ các điểm cũng như các đường thẳng mà trong quá trình tính toán của hàm gradient descent. (1 điểm)

Thank you !

Bài tập nhóm số 5

1. Download file dữ liệu tại đây. Dùng Pandas đọc file. (1 điểm)
2. Xây dựng hàm tuyến tính bằng các giải chính xác và print ra giá trị của các giá trị của mô hình. (2 điểm)
3. Viết def linregloss(y,yhat), tính toán giá trị mean square error của tổng tất cả sample: (2 điểm)
4. Viết hàm def gradientdescent(y, yhat, epsilon, w, int). Để tính các parameters (w) tối ưu. (2 điểm)
5. Viết hàm def linregpredict(X, parameters), để tính ra giá giá trị dự đoán. (1 điểm)
6. Viết hàm main để tính parameters (w), và gọi hàm linregpredict để tính ra giá trị dự đoán (1 điểm)
7. Vẽ các điểm và đường thẳng tuyến tính đã giải bằng hàm main phía trên.(1 điểm)