Gradient Descent

Thuật toán Gradient Descent dùng để tìm điểm cực trị của hàm số

Đối với nhiều hàm số, việc khảo sát bằng cách tính đạo hàm rất khó khăn hoặc bất khả thi,

Gradient Descent cung cấp một phương thức tổng quát cho những trường hợp này.

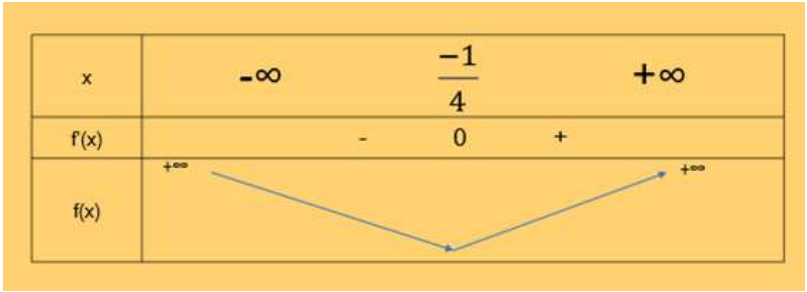
1 Gradient Descent với hàm một biến

Xét hàm số $y = F(x)$ và ta cần tìm điểm cực tiểu của hàm số

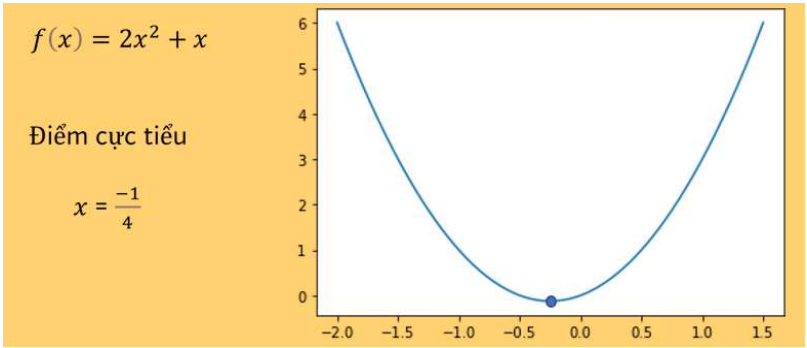
- Ta tính đạo hàm và có nhận xét sau:
  - Điểm Local Minimum(cực tiểu) có đạp hàm $f'(x) = 0$
  - Giá trị đạo hàm phía bên trái điểm cực tiểu là không dương, phía bên phải là không âm
  - Giá trị đạo hàm phía bên trái điểm cực tiểu là không âm, phía bên phải là không dương

Cho bài toán $f(x) = 2x^2 + x$, ta tính được đạo hàm là $f'(x) = 4x + 1$

và giải $f'(x) = 0$ ta được $x = \frac{-1}{4}$, ta có bảng biến thiên sau

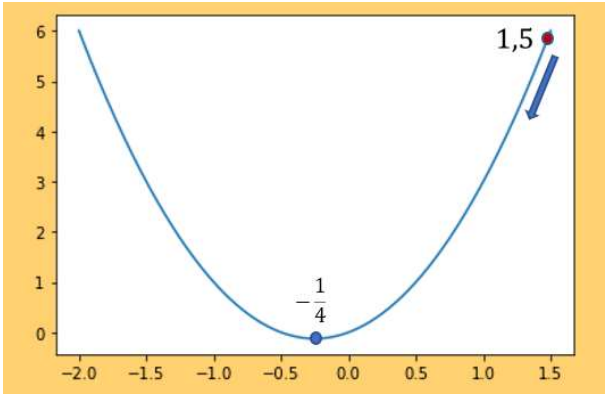


Vậy ta biết được đimer cực tiểu là $x = \frac{-1}{4}$ và hàm số giống như 1 parabol



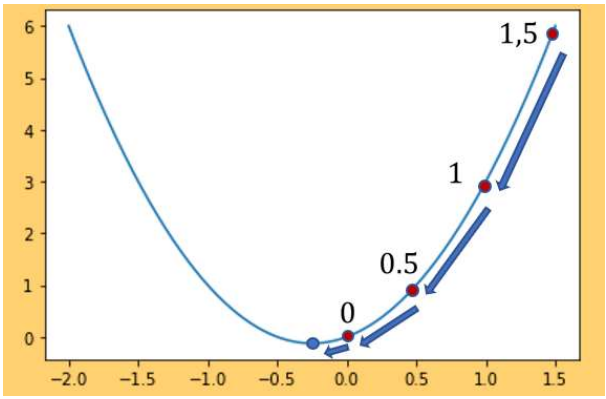
1.1 Cách Gradient Descent Hoạt Động

Xuất phát từ 1 điểm x bất kì, ví dụ trong hình là $x = 1.5$. Ta cần điều chỉnh x để nó có xu hướng tiến về điểm mà hàm số đạt cực tiểu còn gọi là bước nhảy.

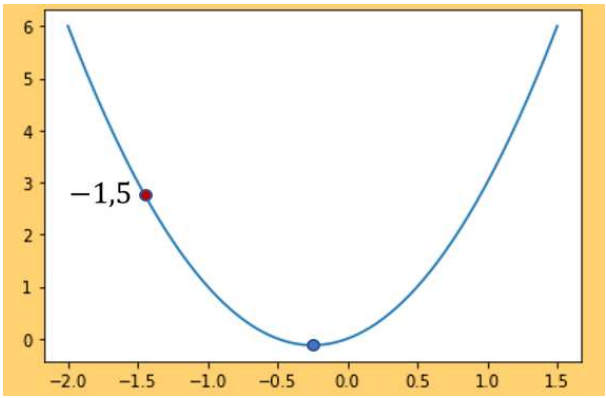


Giả sử ta cho bước nhảy là 0.5, thì sau khi chuyển ta đc $x = 1.5 - 0.5 = 1$

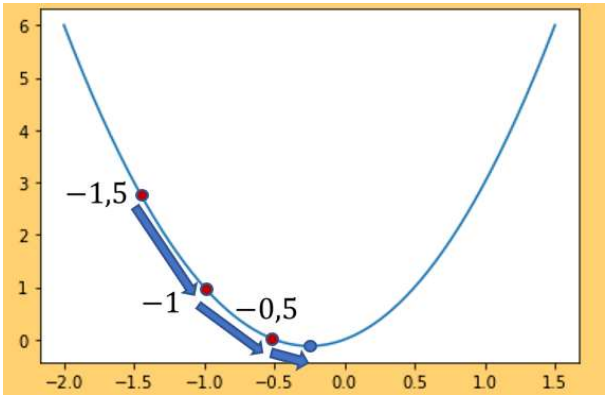
Ta cứ tiếp tục các bước để cho x tiến dần về điểm cực tiểu và lúc đó đạo hàm cũng dần bằng 0



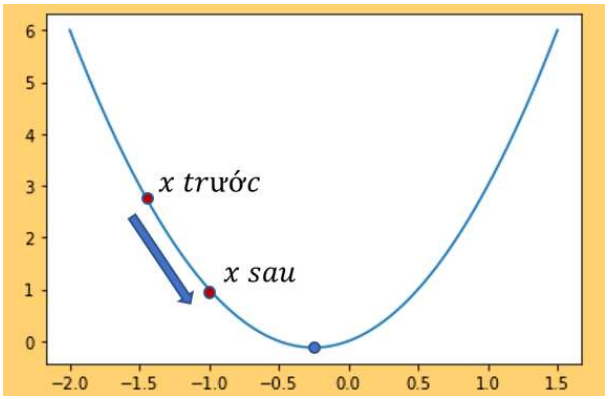
Tương tự ta cũng có thể lấy điểm xuất phát ban đầu là $x = -1.5$



và sau các bước lặp

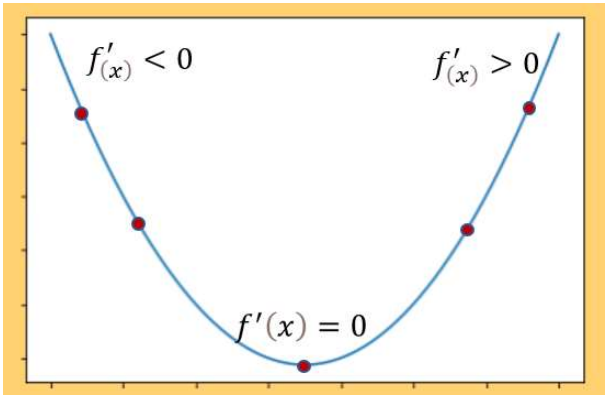


Từ các hình minh hoạ trên ta được tổng kết sau



- Cập nhật $X_s = X_t - \Delta$, với $\Delta = \alpha \cdot f'(X_t)$
  - α: hằng số học tập
  - $f'(X_t)$: đạo hàm tại điểm X_t
- Lặp lại cho đến khi đạo hàm tại điểm X tiến gần tới 0

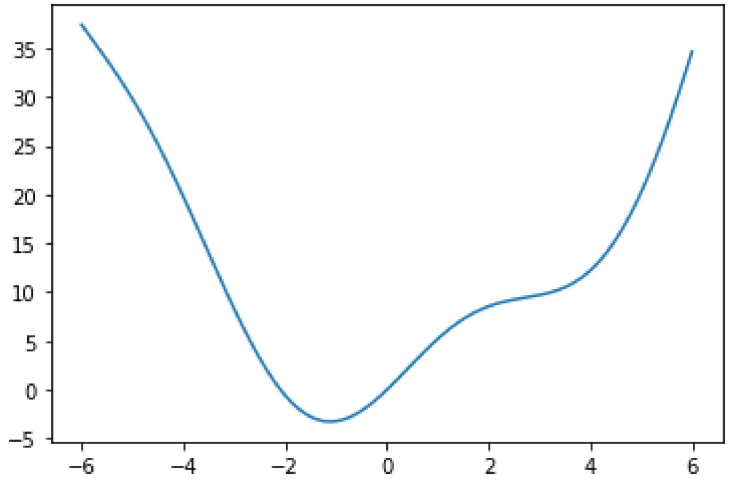
Tại sao lại là trừ Δ?



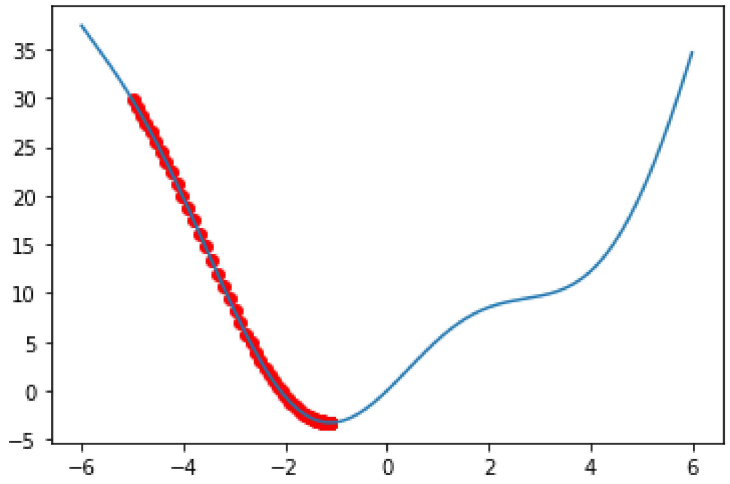
- Vì khi $f'(x) > 0$ cần trừ x đi 1 đại lượng Δ thì điểm x sẽ dịch dần về điểm có $f'(x) = 0$
- Khi $f'(x) < 0$ cần cộng x thêm đại lượng Δ thì điểm x sẽ dịch dần về điểm có $f'(x) = 0$
- Đại lượng Δ này luôn ngược dấu với đạo hàm

1.2 Ảnh hưởng của Learning Rate

Ta xét hàm $f(x) = x^2 + 5sin(x)$

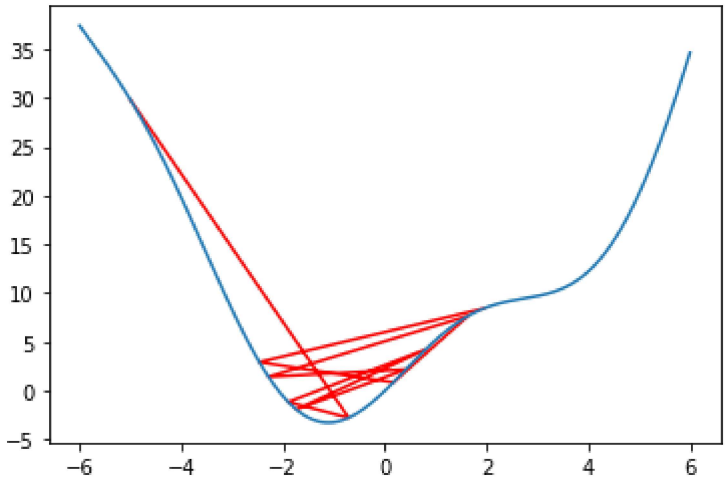


Với điểm ban đầu $x_0 = -5, \Delta = 0.01$



NOTE: các điểm đỏ là các điểm x sau khi thực hiện trừ Δ

Với điểm ban đầu $x_0 = -5, \Delta = 0.5$



Ta quan sát thấy hai điều:

- Với learning rate nhỏ 0.01 tốc độ hội tụ rất chậm. Trong ví dụ này tôi chọn tối đa 100 vòng lặp nên thuật toán dừng lại trước khi tới đích, mặc dù đã rất gần. Trong thực tế, khi việc tính toán trở nên phức tạp, learning rate quá thấp sẽ ảnh hưởng tới tốc độ của thuật toán rất nhiều, thậm chí không bao giờ tới được đích.
- Với learning rate lớn 0.5 thuật toán tiến rất nhanh tới gần đích sau vài vòng lặp. Tuy nhiên, thuật toán không hội tụ được vì bước nhảy quá lớn, khiến nó cứ quẩn quanh ở đích.

$\rightarrow$ Việc lựa chọn learning rate rất quan trọng trong các bài toán thực tế. Việc lựa chọn giá trị này phụ thuộc nhiều vào từng bài toán và phải làm một vài thí nghiệm để chọn ra giá trị tốt nhất. Ngoài ra, tùy vào một số bài toán, GD có thể làm việc hiệu quả hơn bằng cách chọn ra learning rate phù hợp hoặc chọn learning rate khác nhau ở mỗi vòng lặp.

Code

In [1]:
```
# Thêm thư viện
import numpy as np
import matplotlib.pyplot as plt
import math
```

In [2]:
```
# Viết hàm f(x) và f'(x)
def F(x):
    return x*x + 5 * np.sin(x)
```

```
def f(x):
    return 2 * x + 5 * np.cos(x)
```

In [3]:

```
# Viết hàm Gradient Descent
def GradientDescent(learningRate,x_o,loop,lim):
    """
    đầu vào:
        - learningRate: Hằng số học
        - x_o         : điểm x ban đầu
        - loop        : số vòng lặp
        - lim         : giới hạn của đạo hàm
    đầu ra :
        - list_x      : danh sách các điểm x
            - list_x[-1] là toạ độ điểm cực tiểu
    """
    list_x = [x_o]
    for i in range(loop):
        list_x.append(list_x[-1] - learningRate * f(list_x[-1]))
        if abs(f(list_x[-1])) <= lim:
            break
    return list_x
```

In [4]:

```
# Khởi tạo dữ liệu
X = np.arange(-6,6,0.1)
Y = F(X)
```

In [5]:

```
# Tìm x (x khởi đầu từ 5 và -5)
right = GradientDescent(0.01,5,1000,1e-3)
left  = GradientDescent(0.01,-5,1000,1e-3)
```

In [6]:

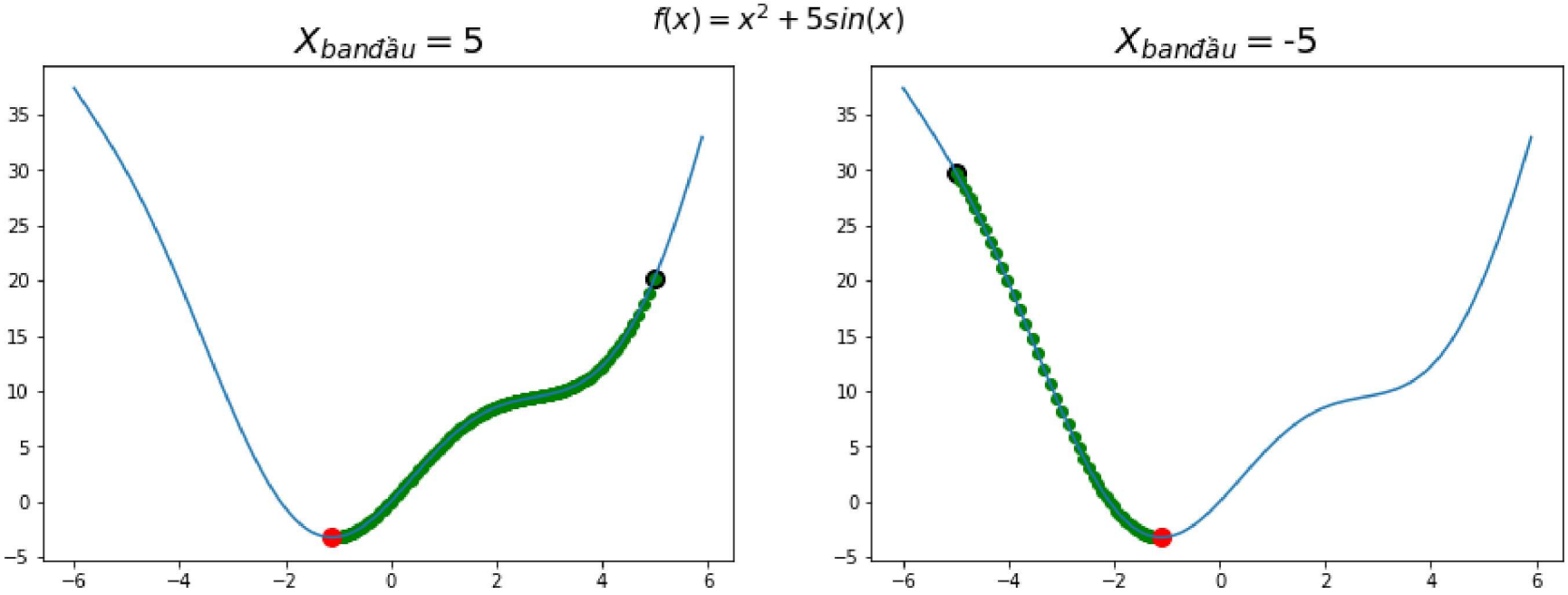
```
# vẽ biểu đồ
fig, ax = plt.subplots(nrows = 1,ncols = 2,figsize=(15,5))

fig.suptitle('$f(x) = x^2 + 5sin(x)$', fontsize=16)


ax[0].plot(X,Y)
ax[0].set_title("$X_{ban đầu} = 5$",fontsize=20)
ax[0].scatter(right[0],F(right[0]),c='black',s=100)
for i in right:
```

```
    ax[0].scatter(i,F(i),color='green')
ax[0].scatter(right[-1],F(right[-1]),color='red',s=100)


ax[1].plot(X,Y)
ax[1].set_title("$X_{ban đầu} = $-5",fontsize=20)
ax[1].scatter(left[0],F(left[0]),c='black',s=100);
for i in left:
    ax[1].scatter(i,F(i),color='green')
ax[1].scatter(left[-1],F(left[-1]),color='red',s=100);
```



2 Gradient Descent Với Hàm 2 Biến

Cần tìm điểm cực tiểu của hàm $f(\Theta)$, trong đó Θ (theta) là 1 vector được dùng để ký hiệu tập hợp các tham số của một mô hình cần tối ưu

Đạo hàm của hàm số đó tại một điểm Θ bất kì ta được $\nabla_\Theta f(\Theta)$ (hình tam giác ngược đọc là nabla).

Tương tự như hàm 1 biến, thuật toán Gradient Descent cho hàm nhiều biến cũng bắt đầu bằng một điểm dự đoán Θ_0

sau đó, ở vòng lặp thứ t quy tắc cập nhật là:

- $\Theta_{t+1} = \Theta_t - \alpha \nabla_\Theta f(\Theta_t)$

hoặc có thể viết như sau $\Theta = \Theta - \alpha \nabla_\Theta f(\Theta)$

Note: luôn luôn đi ngược hướng với đạo hàm(đã được đề cập ở GD 1 biến).

2.1 Thực hiện

Xét hàm $f(x, y) = x^2 + y^2$

Thực hiện đạo hàm riêng với x và y

ta được:

$f'(x) = 2x$

$f'(y) = 2y$

In [7]:
```
def F(x,y):
    return x**2 + y**2
def f(x,y):
    return np.array([ 2*x ,2*y ])
```

In [8]:
```
def norm(matrix):
    N = matrix.shape[0]
    sum = 0
    for i in range(N):
        sum += matrix[i]**2
    return math.sqrt(sum)
```

In [9]:
```
def GradientDescent(theta,learningRate,mine):
    x,y = theta
    gradient = f(x,y)
    Norm = norm(gradient)

    his = [[x,y]]

    while Norm > mine:
        newTheta = -gradient
        x += learningRate * newTheta[0]
```

```
        y += learningRate * newTheta[1]
        his  = np.vstack((his,[x,y]))
        gradient = f(x,y)
        Norm = norm(gradient)
    return his
```

In [10]:
```
answer = GradientDescent((10,10),0.1,1e-3)
```

In [11]:
```
answer[:,][-1] ## Kết quả tìm được
```

Out[11]:
```
array([0.00034845, 0.00034845])
```

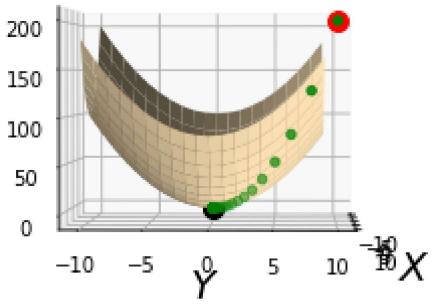
In [12]:
```
X = np.arange(-10,10,1)
```

In [13]:
```
Y = np.arange(-10,10,1)
```

In [14]:
```
X,Y = np.meshgrid(X,Y)
```

In [15]:
```
Z = F(X,Y)
```

In [16]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(answer[:,0], answer[:,1], F(answer[:,0],answer[:,1]), color='green')
ax.scatter3D(answer[:,0][0], answer[:,1][0], F(answer[:,0][0],answer[:,1][0]), color='red',s=100)
ax.scatter3D(answer[:,0][-1], answer[:,1][-1], F(answer[:,0][-1],answer[:,1][-1]), color='black',s = 100)
ax.plot_surface(X, Y, Z,color='moccasin')
ax.set_xlabel('$X$', fontsize=20)
ax.set_ylabel('$Y$',fontsize=20);
ax.view_init(1,2);
```


200
150
100
50
0
−10
−5
0
5
10
Y
X