

MẠNG NƠON NHÂN TẠO và GIẢI THUẬT DI TRUYỀN

Neural Network & Genetic Algorithm



Biên soạn: ThS. Phạm Đình Tài
pdtai@ntt.edu.vn
0985.73.39.39

CHƯƠNG 4

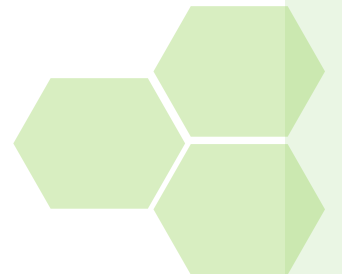
Tối ưu hóa hàm mục tiêu



- ✓ **Giới thiệu hai giải thuật học**
- ✓ **Thuật toán bước giảm cực đại (Gradient Descent Learning)**
- ✓ **Thuật toán L-M (Levenberg Marquardt)**
- ✓ **Thực hành với một trong số các thuật toán trên**

■ Loss function

Về mặt bản chất, **loss function** (hàm mất mát) là hàm cho phép xác định mức độ sai khác của kết quả dự đoán so với giá thực thực cần dự đoán. Nó là một phương pháp đo lường chất lượng của mô hình dự đoán trên tập dữ liệu quan sát. Nếu mô hình dự đoán sai nhiều thì giá trị của **loss function** sẽ càng lớn và ngược lại nếu nó dự đoán càng đúng thì giá trị của **loss function** sẽ càng thấp.



NGUYÊN LÝ CHUNG

■ Ví dụ1:

Với mỗi dự đoán mà mô hình đưa ra, chúng ta đưa ra một giá trị mất mát do dự đoán sai là trị tuyệt đối của hiệu giữa giá trị dự đoán và giá trị thực trong bộ dữ liệu quan sát.

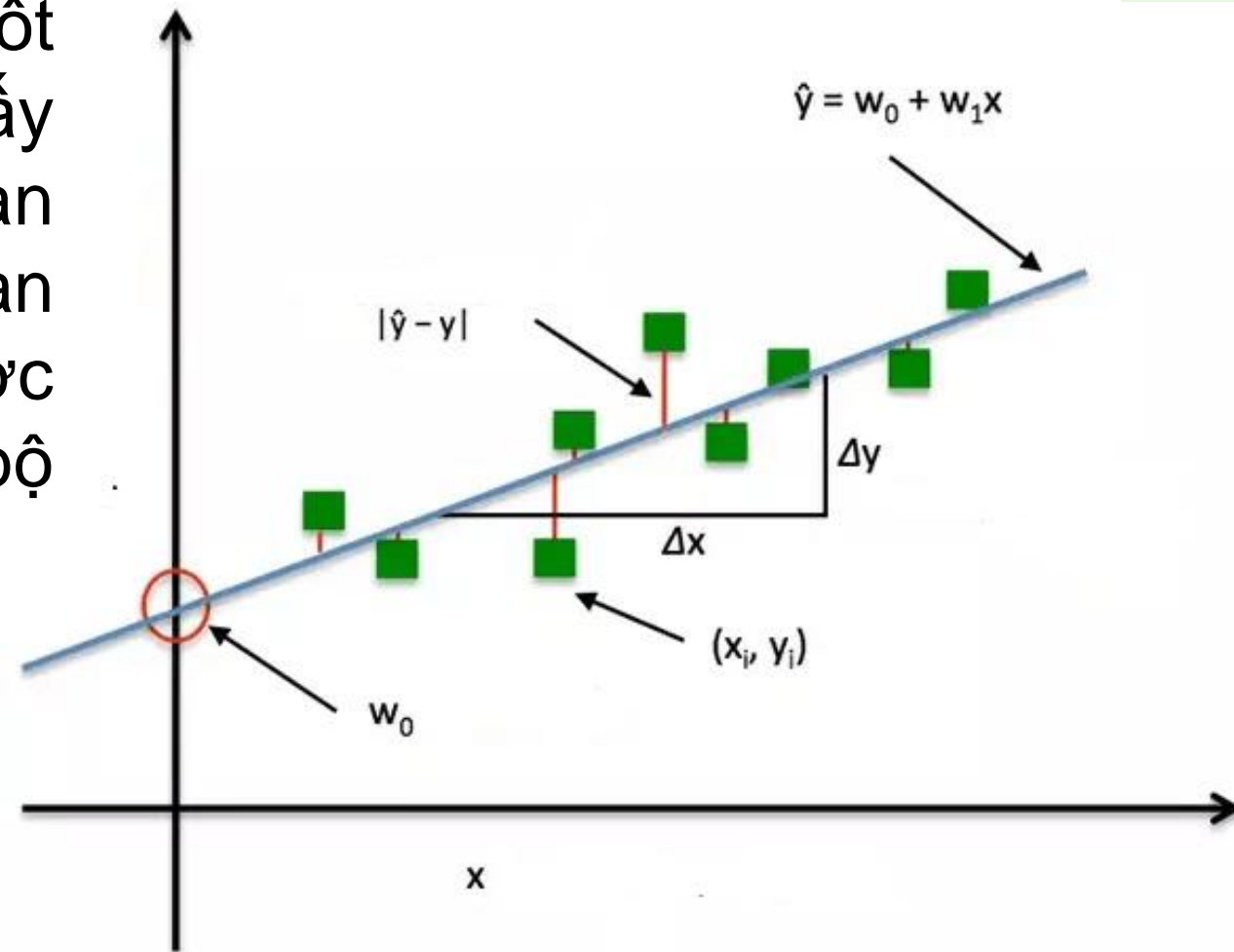
Hay $\text{loss} = |y_{\text{predicted}} - y|$, trong đó $y_{\text{predicted}}$ là giá trị dự đoán của mô hình và y là giá trị nhãn đúng trong bộ dữ liệu quan sát. Lấy một số điểm dự đoán ta tính được giá trị mất mát tại một số điểm trong bộ dữ liệu quan sát như sau:

Giá trị thật y	Giá trị dự báo $y_{\text{predicted}}$	Giá trị mất mát do dự đoán sai
60	60	0

NGUYÊN LÝ CHUNG

Như vậy để đánh giá mức độ tốt của mô hình dự đoán, chúng ta lấy trung bình giá trị mất mát của toàn bộ các điểm trên bộ dữ liệu quan sát. Điều này giúp ta xác định được một **loss function** trên toàn bộ bộ dữ liệu quan sát:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N |f_w(x^{(i)}) - y^{(i)}|$$

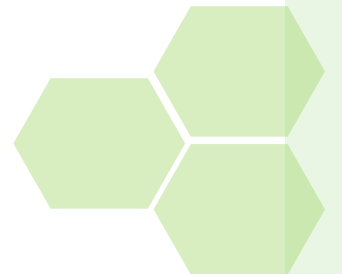


NGUYÊN LÝ CHUNG

■ Ví dụ2:

Thay vì hàm trị tuyệt đối ở trên, bây giờ chúng ta có muốn phạt nặng nhiều lần hơn cho các dự đoán sai lớn, và để có thể tính được đạo hàm dễ dàng hơn, khi đó chúng ta có thể chọn hàm trung bình bình phương:

$$\mathcal{L}(m_0, m_1) = \frac{1}{2m} \sum_{i=1}^m (f(x_i, m_0, m_1) - y)^2$$



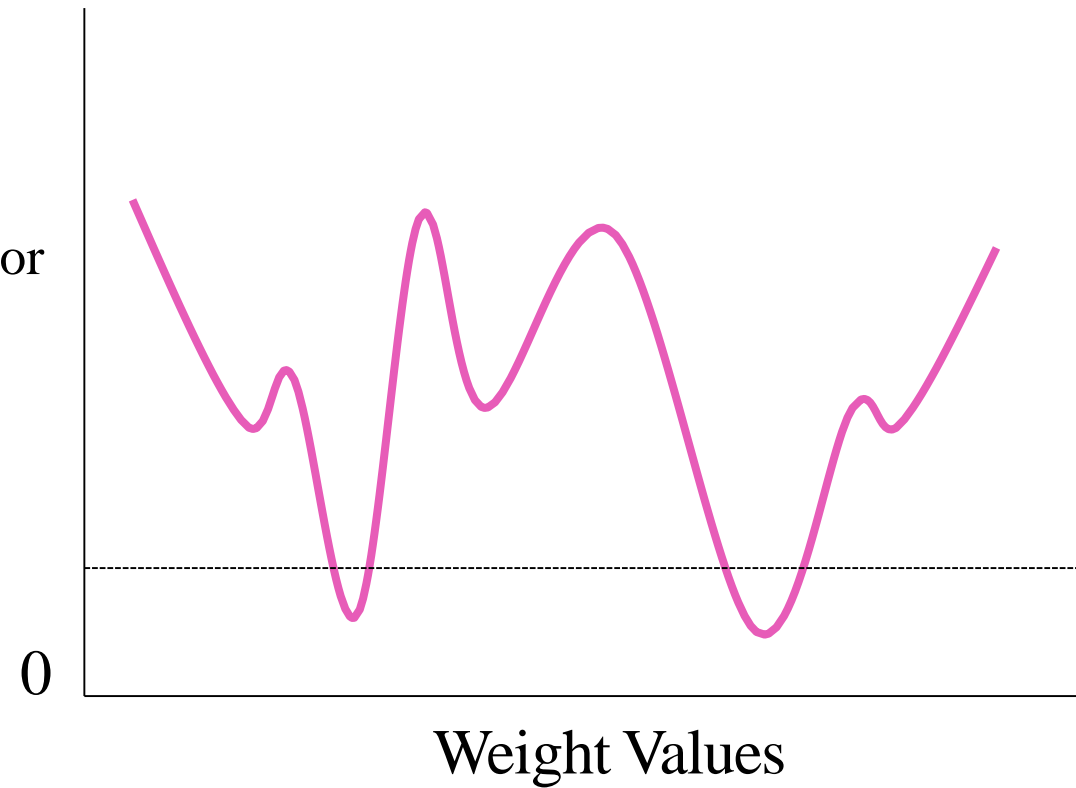
NGUYÊN LÝ CHUNG

- Giả sử tập số liệu học có p mẫu: $\{\mathbf{x}_i, d_i\}$ với $i = 1, \dots, p$
- Với mỗi vector đầu vào \mathbf{x}_i , giá trị đích cần đạt $f(\mathbf{x}_i) \sim d_i$ với mọi giá trị của i
- Nguyên lý của học: tối thiểu hóa sai số giữa các đầu ra của mạng $f(\mathbf{x}_i)$ và giá trị đích cần đạt được
- Thông thường người ta gọi là tối thiểu hóa **hàm mục tiêu (objective function)**

$$E = \frac{1}{2} \sum_{i=1}^p (f(\mathbf{x}_i) - d_i)^2 \rightarrow \min$$

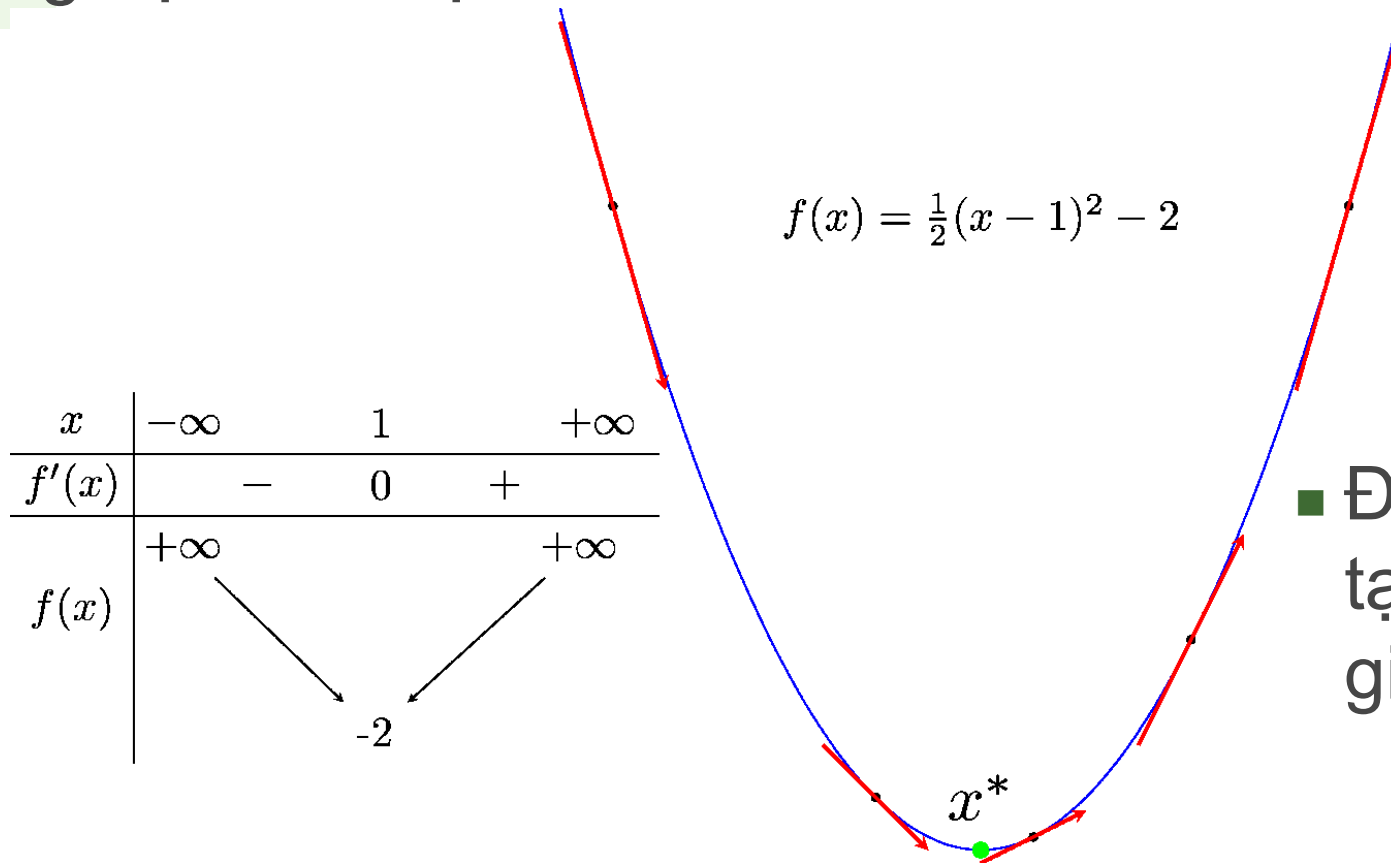
■ Tối ưu hóa hàm mục tiêu (objective function)

SSE:
Sum Squared Error
 $\sum (t_i - z_i)^2$



NGUYÊN LÝ CHUNG

- Tiếp tuyến với đồ thị hàm số tại một điểm có hệ số góc bằng đạo hàm tại điểm đó

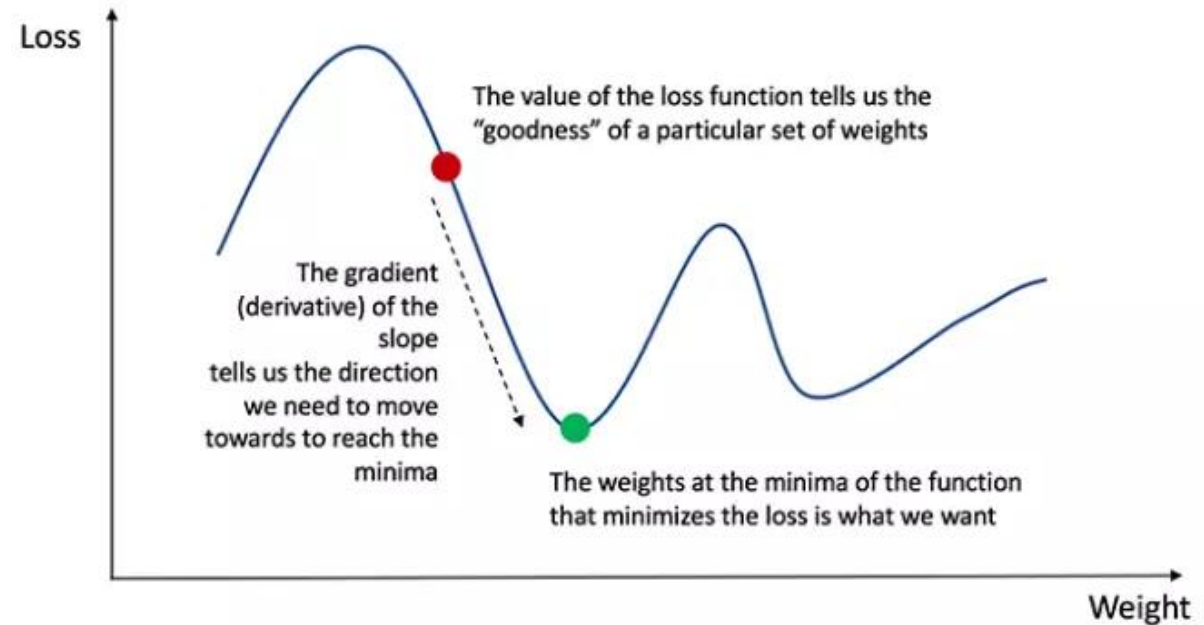


- Điểm cực tiểu là điểm mà tại đó đạo hàm của $f(x)$ có giá trị bằng 0

NGUYÊN LÝ CHUNG

■ Trong đại đa số các trường hợp, tìm x_{\min} sao cho $f'(x) = 0$ là không khả thi với f là một hàm bất kỳ bởi các nguyên nhân:

- Sự phức tạp của dạng của đạo hàm
- Dữ liệu có số chiều lớn (high dimension)
- Kích thước của tập dữ liệu lớn (large dataset)

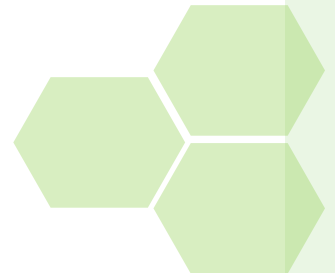


Gradient descent algorithm

HƯỚNG TIẾP CẬN

■ Nội dung của phương pháp như sau:

- Xuất phát từ một vị trí khởi đầu $x(0)$ ta đi tìm $x^{(1)}$ sao cho $f(x^{(1)}) < f(x^{(0)})$
- Tiếp tục tìm $x^{(2)}$ sao cho $f(x^{(2)}) < f(x^{(1)})$
- Liên tục như vậy ta sẽ thu được kết quả là một chuỗi $\{x^{(t)}\}$ sao cho $\{f(x^{(t)})\}$ là chuỗi giảm dần và sẽ đạt tới một cực tiểu nào đó
- Khi đó:
$$x^{(t)} \xrightarrow{t \rightarrow \infty} x_{\min}$$



THUẬT TOÁN HỌC

- Công thức này là hệ quả của phép triển khai Taylor:

$$f\left(x^{(t)} + \Delta\right) \approx f\left(x^{(t)}\right) + \Delta \cdot f'\left(x^{(t)}\right) + O\left(\Delta^2\right)$$

- Nếu chọn: $\Delta = -\eta \cdot f'\left(x^{(t)}\right)$ với Δ đủ nhỏ thì:

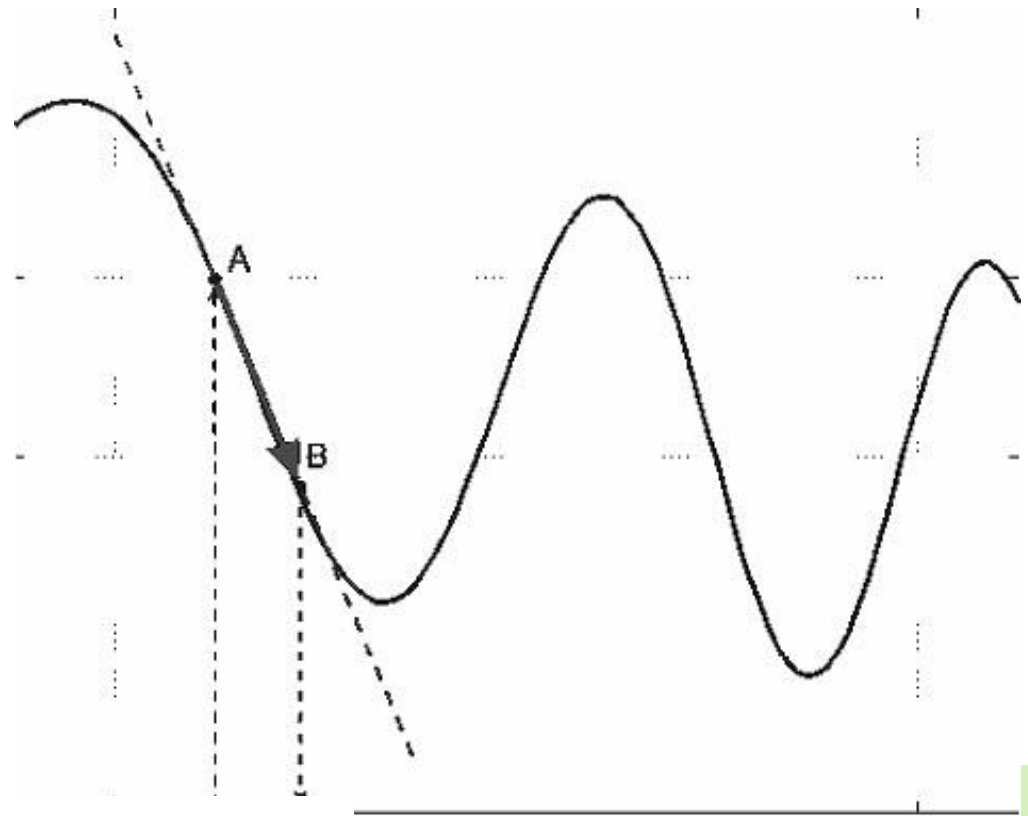
$$f\left(x^{(t)} + \Delta\right) \approx f\left(x^{(t)}\right) - \eta \cdot \left(f'\left(x^{(t)}\right)\right)^2 + O\left(\Delta^2\right) < f\left(x^{(t)}\right)$$

- Nếu chọn: $x^{(t+1)} = x^{(t)} + \Delta = x^{(t)} - \eta \frac{\partial f}{\partial x} \Big|_{x=x^{(t)}}$ thì giá trị tại $X^{(t+1)}$ sẽ giảm

https://vi.wikipedia.org/wiki/%C4%90%E1%BB%8Bnh_l%C3%BD_Taylor

THUẬT TOÁN HỌC

- Trong đó **Deta** là tốc độ học của mạng

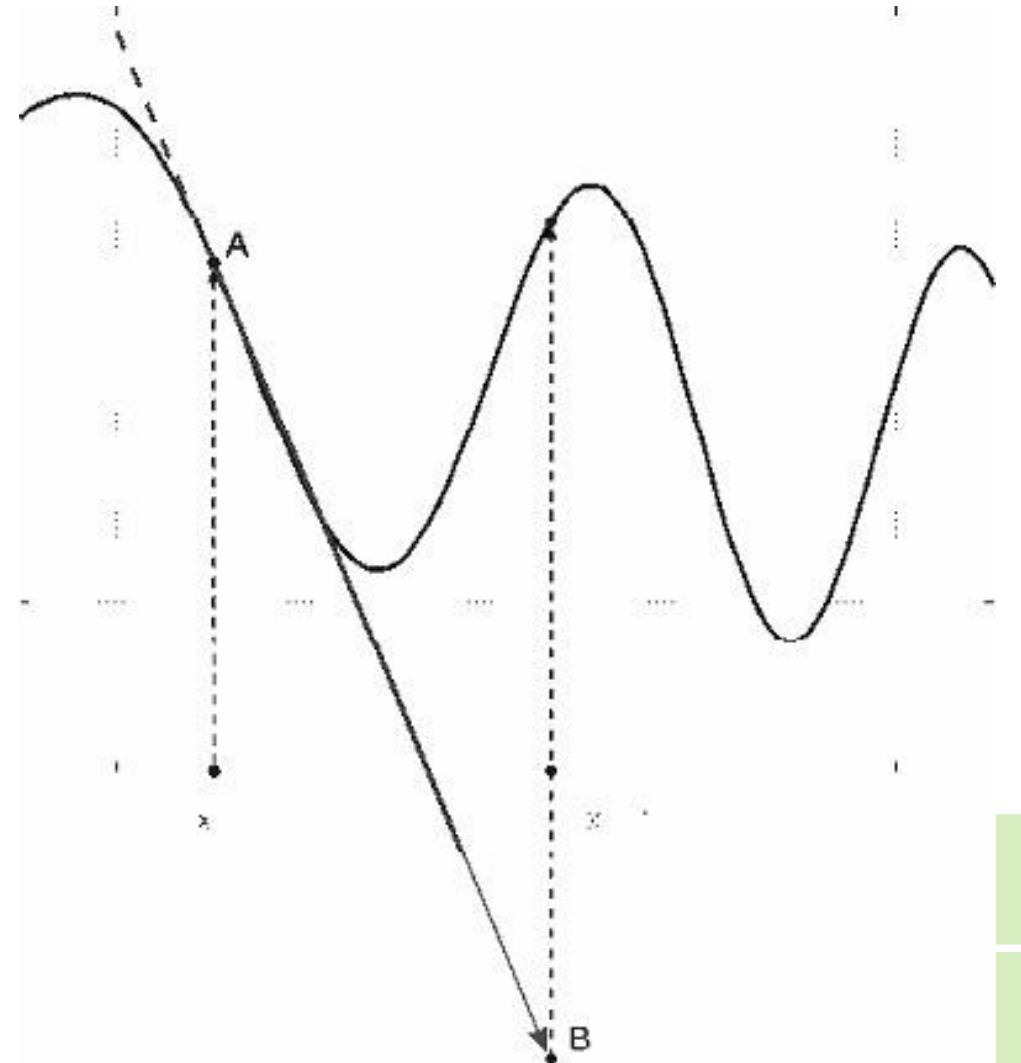


$$x^{(t+1)} = x^{(t)} - \eta \left. \frac{\partial f}{\partial x} \right|_{x=x^{(t)}}$$

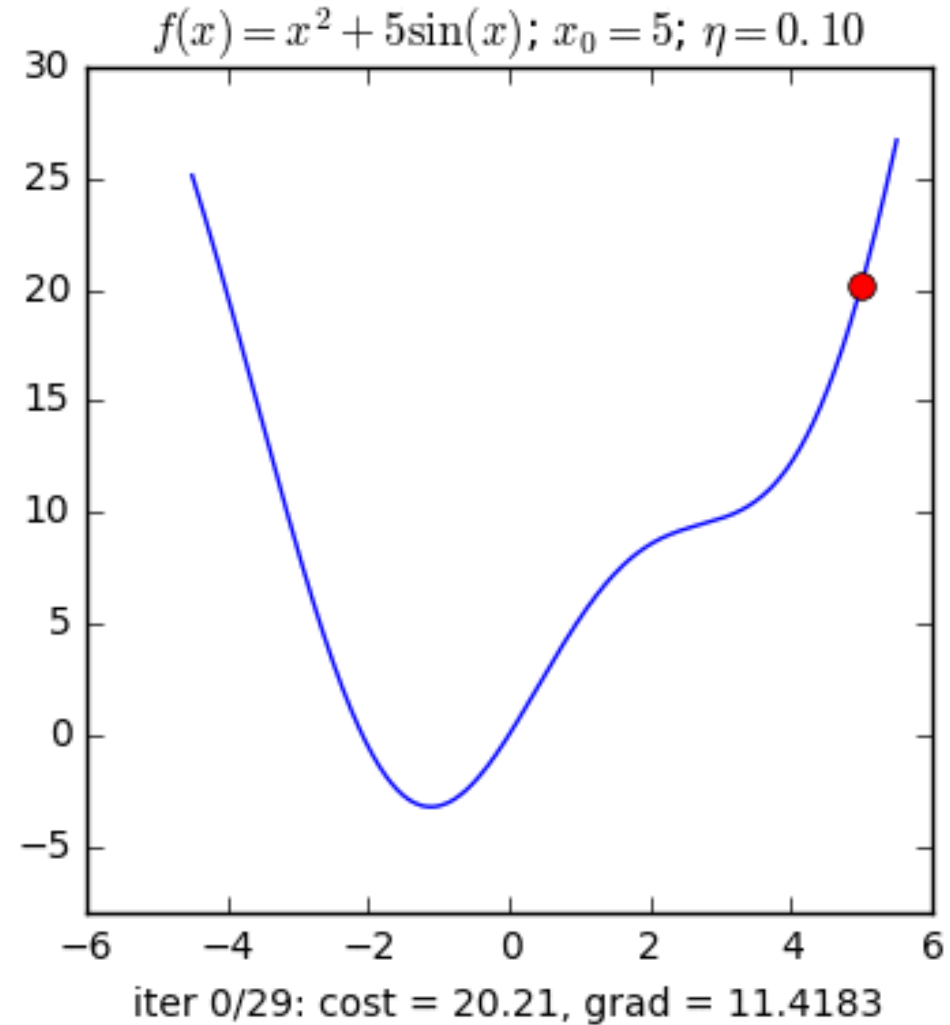
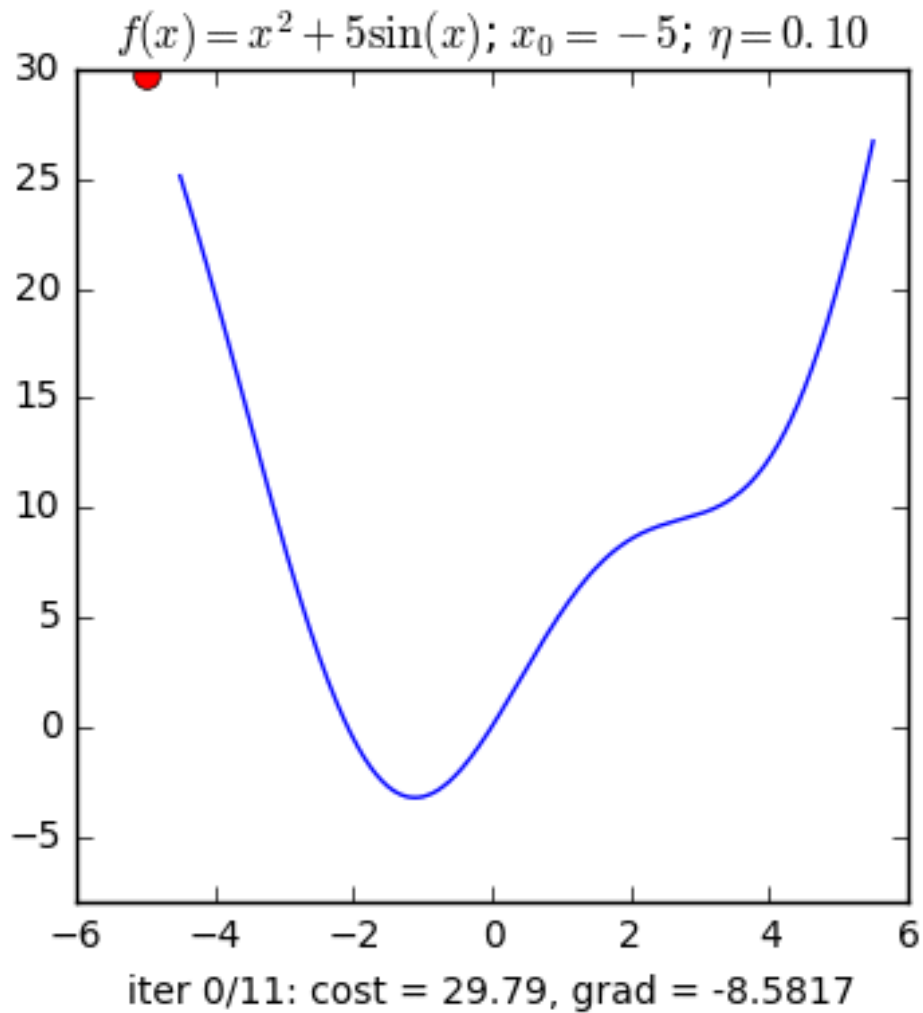
THUẬT TOÁN HỌC

$$x^{(t+1)} = x^{(t)} - \eta \left. \frac{\partial f}{\partial x} \right|_{x=x^{(t)}}$$

- Nếu **Deta** lớn quá, có thể bỏ qua cực trị và không tốt cho quá trình cập nhật mạng.



HƯỚNG TIẾP CẬN



GRADIENT DESCENT

- Khi nhắc đến sự thay đổi đổi thì ta thường nhắc tới **Gradient**.

Gradient của hàm cho biết hàm tăng mạnh như thế nào.

- **Ví dụ:**

- Với hàm 1 chiều: $f(x) = x^2$ thì **Gradient** của nó được ký hiệu và tính như sau:

$$Grad(x) = \frac{\partial f(x)}{\partial (x)} = 2x$$

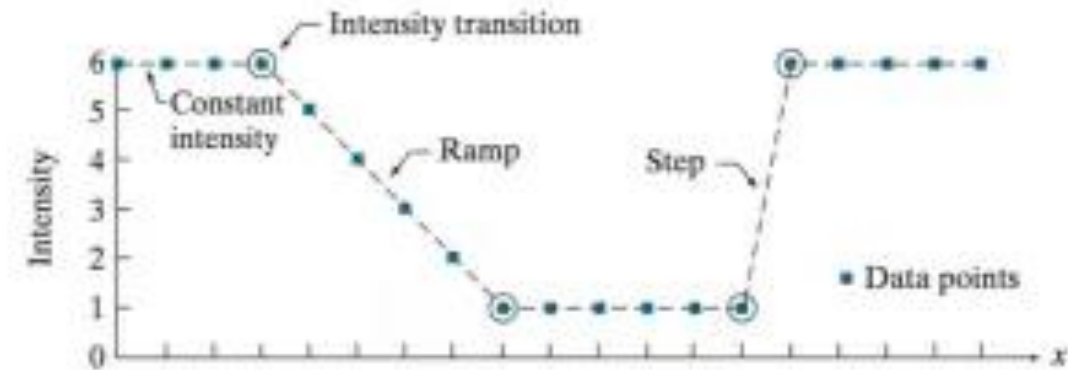
- $Grad(2) = 4$ chỉ ra hướng tăng của hàm là bên phải
- $Grad(-1) = -2$ chỉ ra hướng tăng của hàm nằm ở bên trái.

GRADIENT DESCENT

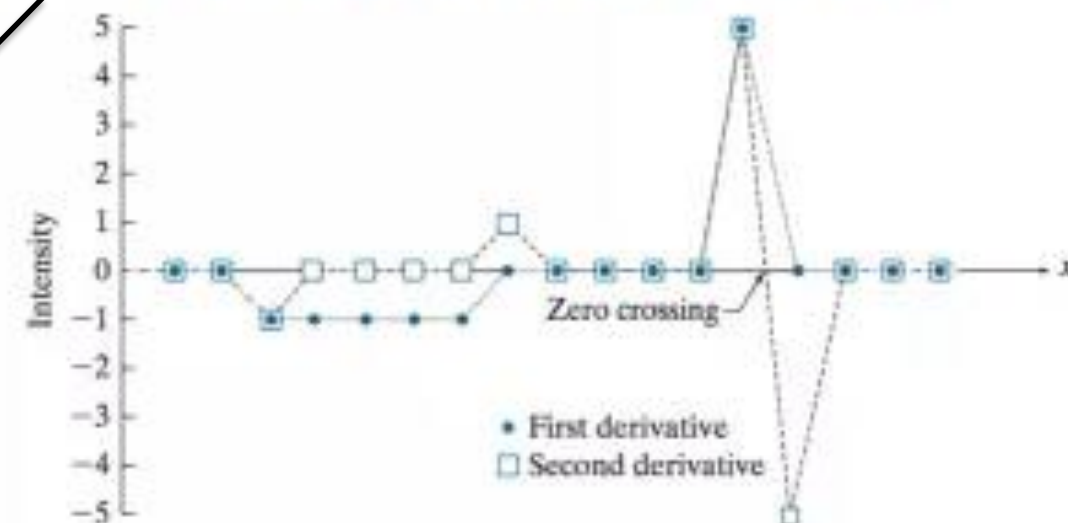
- **Gradient** không chỉ tính bằng đạo hàm bậc 1, ta cũng thể tính bằng đạo hàm bậc. Với độ biến đổi của mức sáng bên trên ta tính được đạo hàm bậc 1, bậc 2 và công thức tương ứng như sau:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$



Values of scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	1	6	6	6	6	6	x
1st derivative	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	0	5	0	0	0	0	0	
2nd derivative	0	0	-1	0	0	0	0	0	1	0	0	0	0	0	5	-5	0	0	0	0	



GRADIENT DESCENT LEARNING

- Mục đích: giảm thiểu lỗi của mạng mỗi khi hệ số mạng thay đổi
- Hàm mục tiêu: $E = \sum (t_i - z_i)^2$
- Tìm một giải thuật thay đổi trọng số sao cho đạo hàm của hàm mục tiêu theo trọng số là âm

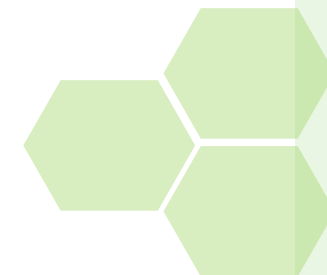
$$\frac{\partial E}{\partial w_{ij}} < 0$$

- Chú ý do phương pháp này sử dụng đạo hàm nên **hàm ngưỡng** không phù hợp để sử dụng phương pháp này

GRADIENT DESCENT LEARNING

Gradient descent là thuật toán tìm giá trị nhỏ nhất của hàm số $f(x)$ dựa trên đạo hàm. Thuật toán:

- **Bước 1:** Khởi tạo giá trị $x=x_0$ tùy ý
- **Bước 2:** Gán $x = x - \text{learning_rate} * f'(x)$
(learning_rate là hằng số không âm ví dụ learning_rate = 0.001)
- **Bước 3:** Tính lại $f(x)$:
 - Nếu $f(x)$ đủ nhỏ thì dừng lại.
 - Ngược lại tiếp tục bước 2

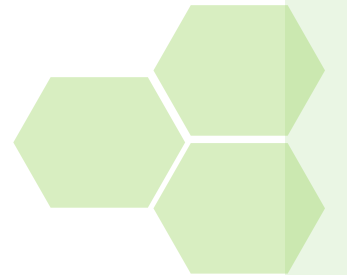


GRADIENT DESCENT CHO 1 NEURON

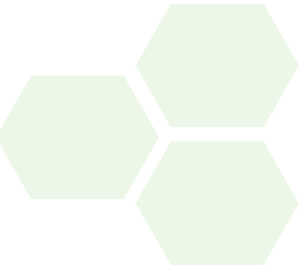
- Bộ số liệu: p mẫu $\{\mathbf{x}_i, d_i\}, i = 1, \dots, p,$
- Đầu ra tương ứng với từng đầu vào

$$y_i = f\left(\sum_{j=0}^N W_j x_{ij}\right)$$

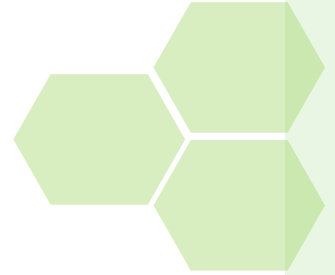
- Sai số trên bộ dữ liệu học: $E = \frac{1}{2} \sum_{i=1}^p (y_i - d_i)^2$
- Ta cần tìm các trọng số: $W_j \ (j = 0, 1, \dots, N)$
- Với các giá trị khởi tạo: $W_j^{(t+1)} = W_j^{(t)} - \eta \frac{\partial E}{\partial W_j} \Big|_{[\mathbf{w}] = [\mathbf{w}]^{(t)}}$



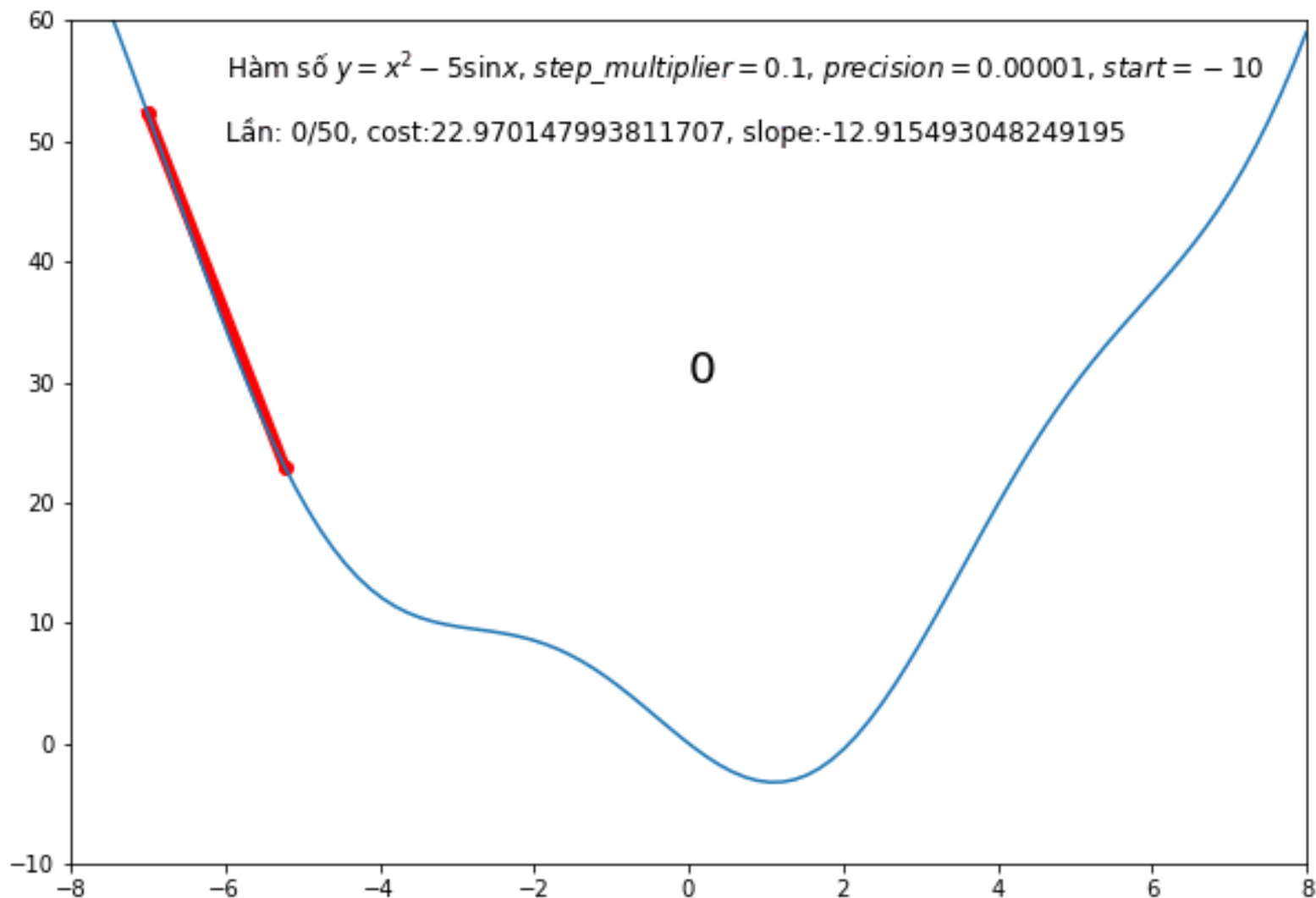
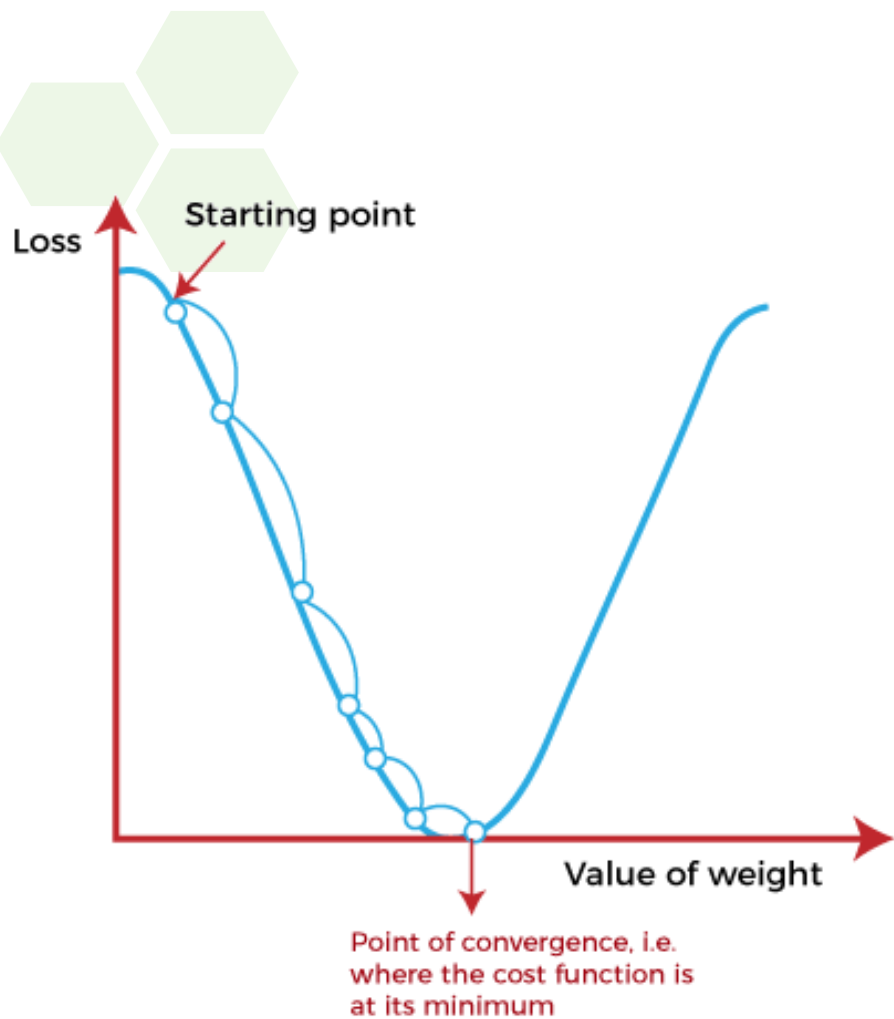
GRADIENT DESCENT CHO 1 NEURON


$$\frac{\partial E}{\partial W_{\alpha}} = \sum_{i=1}^p (y_i - d_i) \frac{\partial y_i}{\partial W_{\alpha}}$$

$$= \sum_{i=1}^p (y_i - d_i) f' \left(\sum_{j=0}^N W_j x_{ij} \right) \frac{\partial \left(\sum_{j=0}^N W_j x_{ij} \right)}{\partial W_{\alpha}}$$

$$= \sum_{i=1}^p (y_i - d_i) f' \left(\sum_{j=0}^N W_j x_{ij} \right) x_{i\alpha}$$


ĐỒ THỊ HÀM SAI SỐ THEO BƯỚC LẶP



THUẬT TOÁN L-M

- Thuật toán Levenberg-Marquardt (L-M) dựa trên triển khai bậc 2 của khai triển Taylor

$$E(\mathbf{W} + \mathbf{p}) = E(\mathbf{W}) + [\mathbf{g}(\mathbf{W})]^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{H}(\mathbf{W}) \mathbf{p} + O(\mathbf{p}^3)$$

- Với \mathbf{p} : là khoảng lân cận trong khai triển
- $\mathbf{g}(\mathbf{W})$: là vector gradient của E theo W

$$\mathbf{g}(\mathbf{W}) = \nabla \mathbf{E} = \left[\frac{\partial E}{\partial W_1}, \frac{\partial E}{\partial W_2}, \dots, \frac{\partial E}{\partial W_n} \right]^T$$

- $\mathbf{H}(\mathbf{W})$ là ma trận đạo hàm bậc 2 $\mathbf{H}(\mathbf{W}) = [H_{ij}] = \begin{bmatrix} \frac{\partial^2 E}{\partial W_1 \partial W_1} & \dots & \frac{\partial^2 E}{\partial W_n \partial W_1} \\ \vdots & & \vdots \\ \frac{\partial^2 E}{\partial W_1 \partial W_n} & \dots & \frac{\partial^2 E}{\partial W_n \partial W_n} \end{bmatrix}$

THUẬT TOÁN L-M

- Tại điểm cần tìm $g(W) = 0$ và $H(W)$ xác định dương

- Giả sử: $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} + \mathbf{p}$
$$\frac{\partial E(\mathbf{W}^{(t+1)})}{\partial \mathbf{p}} = \frac{\partial E(\mathbf{W}^{(t)} + \mathbf{p})}{\partial \mathbf{p}} \approx 0$$

$$\mathbf{g}(\mathbf{W}^{(t)}) + \mathbf{H}(\mathbf{W}^{(t)})\mathbf{p}^{(t)} = 0$$

- Khi đó:

$$\mathbf{p}^{(t)} = -[\mathbf{H}(\mathbf{W}^{(t)})]^{-1} \mathbf{g}(\mathbf{W}^{(t)})$$

- Để tránh bước dịch chuyển quá lớn:

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta [\mathbf{H}(\mathbf{W}^{(t)})]^{-1} \mathbf{g}(\mathbf{W}^{(t)})$$

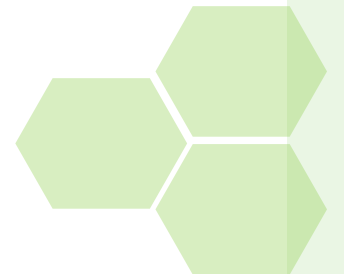
THUẬT TOÁN L-M

- Về cơ bản: nếu ta sử dụng triển khai Taylor ở bậc cao hơn thì sẽ có tốc độ hội tụ nhanh hơn
- Tuy nhiên, các công thức sẽ phức tạp và cồng kềnh
- Vì vậy trong thực tế, ta sẽ chỉ sử dụng đến các triển khai bậc 2



ĐỐI VỚI THUẬT TOÁN L-M

- Với 3 mẫu học, sau 100 bước lặp
- Sai số khi sử dụng Gradient Descent: $E = 0.211$
- Sai số khi sử dụng L-M: $E = 0.0182$
- L-M hội tụ nhanh hơn



■ Học với hệ số thích nghi

- ❖ Deta: ảnh hưởng đến quá trình học
- ❖ Cách lựa chọn đơn giản nhất: lấy giá trị cố định
- ❖ Tuy nhiên cách này thường cho giải thuật rơi vào cực trị địa phương hoặc cần số bước lặp lớn khi Deta nhỏ hoặc bị giao động xung quanh cực trị nếu như Deta quá lớn

■ Khắc phục: sử dụng phương pháp hệ số thích nghi.

- Nếu $E^{(k)} < E^{(k-1)}$ thì tăng hệ số Deta: $\eta^{(k+1)} = \eta^{(k)} \cdot \varepsilon_{inc}$
- Nếu $E^{(k)} > E^{(k-1)}$ thì giảm hệ số Deta: $\eta^{(k+1)} = \eta^{(k)} \cdot \varepsilon_{dec}$

- Học với quán tính

$$\Delta W^{(t)} = -\eta \nabla E^{(t)} + \alpha^{(t)} \Delta W^{(t-1)}$$

- $\alpha^{(t)}$ là hệ số quán tính có giá trị thuộc $[0, 1]$

XẤP XỈ THUẬT TOÁN L-M

- Thuật toán L-M khá phức tạp do phải tính nghịch đảo của ma trận $H(W)$
- [Hagan94] đề xuất thay thế $H(W)$ bởi $G(W)$

$$E(W) = \frac{1}{2} \sum_{i=1}^M e_i^2(W) \quad e_i(W) = y_i(W) - d_i$$

$$\mathbf{e}(W) = \begin{bmatrix} e_1(W) \\ e_2(W) \\ \dots \\ e_M(W) \end{bmatrix}, \quad \mathbf{J}(W) = \begin{bmatrix} \frac{\partial e_1}{\partial W_1} & \frac{\partial e_1}{\partial W_2} & \dots & \frac{\partial e_1}{\partial W_n} \\ \frac{\partial e_2}{\partial W_1} & \frac{\partial e_2}{\partial W_2} & \dots & \frac{\partial e_2}{\partial W_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_n}{\partial W_1} & \frac{\partial e_n}{\partial W_2} & \dots & \frac{\partial e_n}{\partial W_n} \end{bmatrix}$$

$$\mathbf{g}(W) = [\mathbf{J}(W)]^T \mathbf{e}(W)$$

$$\mathbf{G}(W) = [\mathbf{J}(W)]^T \mathbf{J}(W) + \mathbf{R}(W)$$

Thank you !



<https://www.youtube.com/watch?v=sDv4f4s2SB8>