CHƯƠNG 2
MẠNG NEURAL CƠ BẢN (P1)

Khoa Khoa học và Kỹ thuật thông tin
Bộ môn Khoa học dữ liệu

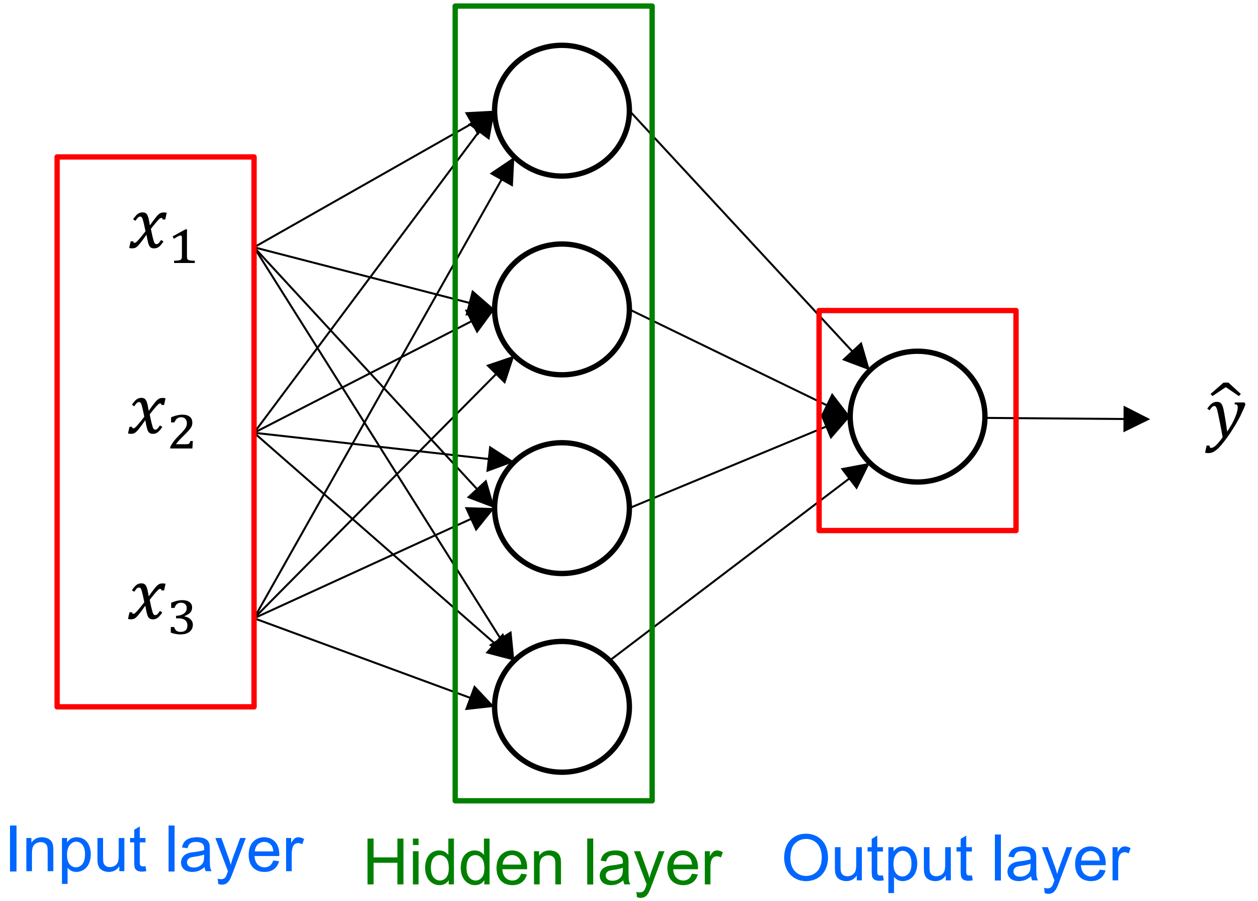
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, KHU PHỐ 6, PHƯỜNG LINH TRUNG, QUẬN THỦ ĐỨC, TP. HỒ CHÍ MINH
[T] 028 3725 2002 101 | [F] 028 3725 2148 | [W] www.uit.edu.vn | [E] info@uit.edu.vn

NỘI DUNG

1. Tổng quan.
2. Tính toán và biểu diễn trên mạng neural.
3. Hàm kích hoạt.
4. Gradient Descent trong mạng neural.
5. Khởi tạo trọng số.

Tổng quan

Tổng quan



Các lớp trong một mạng neural nhân tạo

– **Input layer**: Nhận dữ liệu đầu vào.

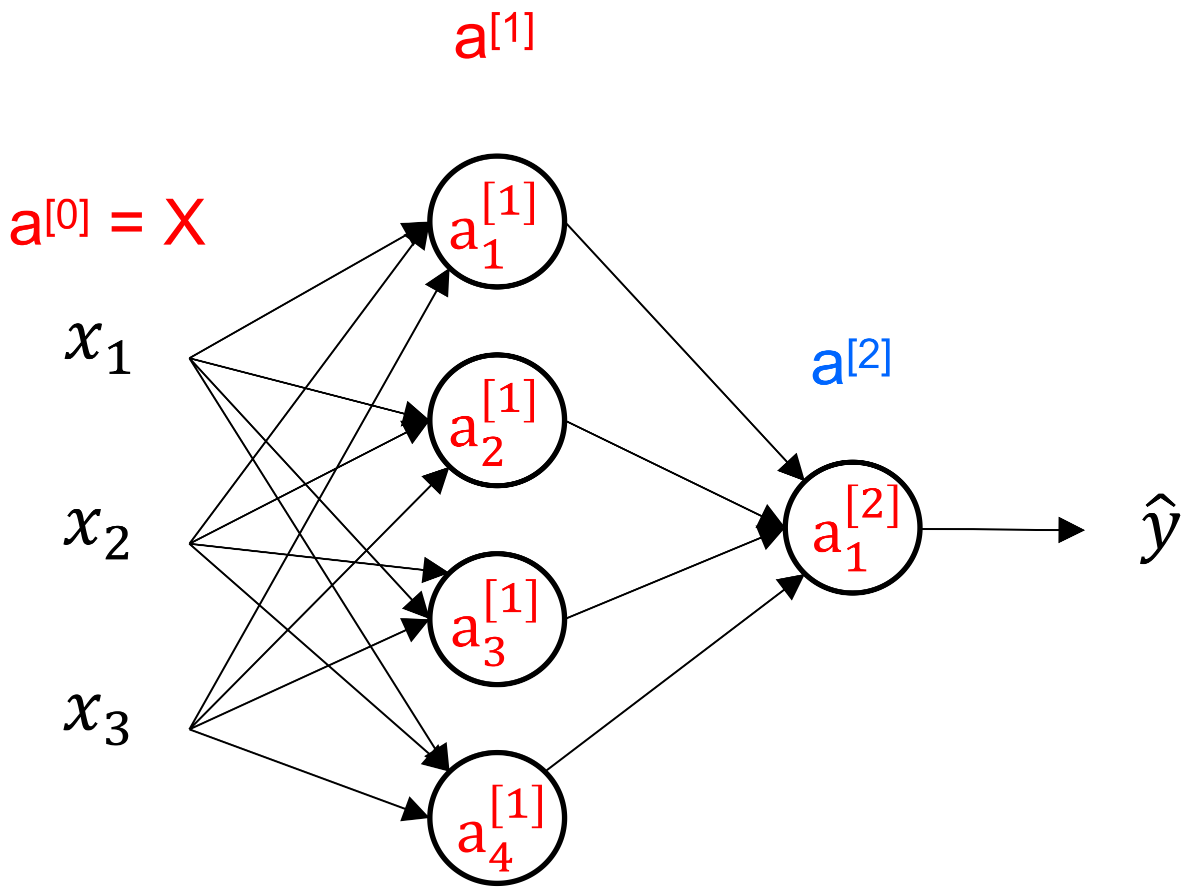
– **Hidden layer**: Kết nối giữa các layer với nhau, gồm input layer, các lớp hidden layers khác và output layer.

– **Output layer**: Đưa ra kết quả từ dữ liệu đầu vào. Dữ liệu kết quả có thể là:

  + Label – dạng categorical đối với bài toán phân lớp (classification).

  + Value – dạng numeric đối với bài toán hồi quy (regression) hay xếp hạng (ranking).

Các ký hiệu



Tập dữ liệu huấn luyện:

X: các vector đặc trưng

y: các nhãn ứng với từng vector đặc trưng trong X

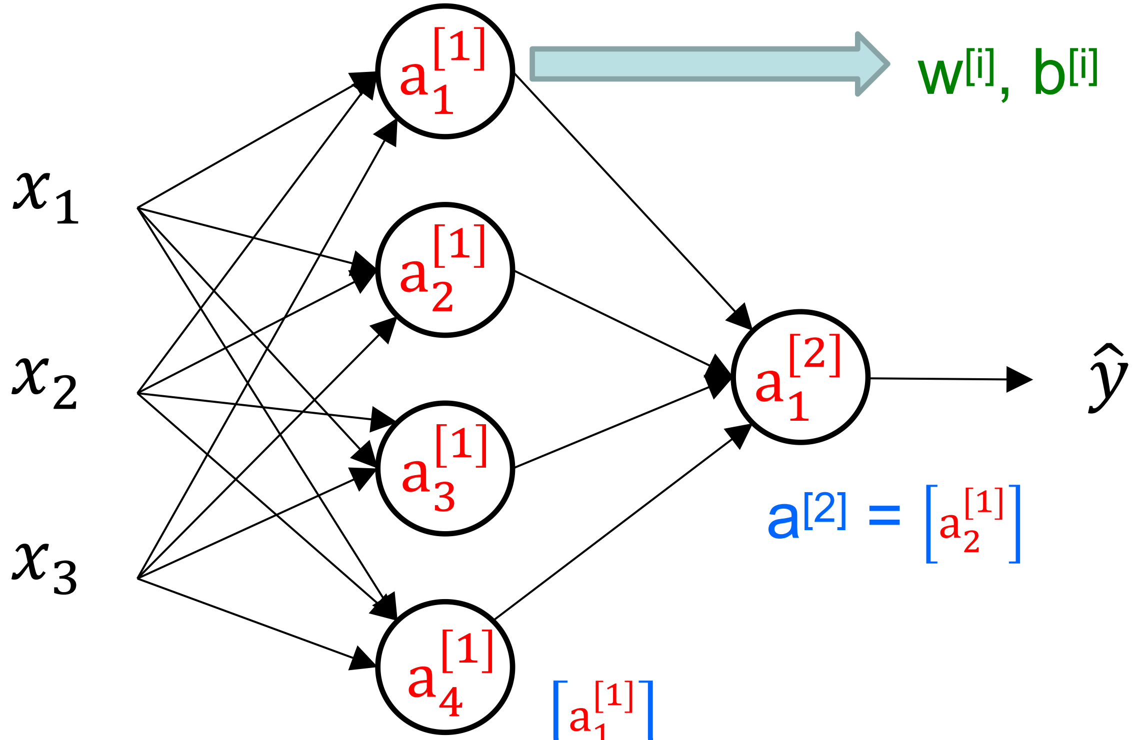
Mạng neural:

a: hàm kích hoạt (activation function).

$a^{[i]}$: giá trị hàm kích hoạt tại lớp thứ i trong mạng neural. Giá trị của $a^{[i]}$ là một vector chứa bộ tham số kích hoạt.

$a_1^{[1]} \in a^{[i]}$ là một bộ tham số kích hoạt gồm: w và b.

Các ký hiệu

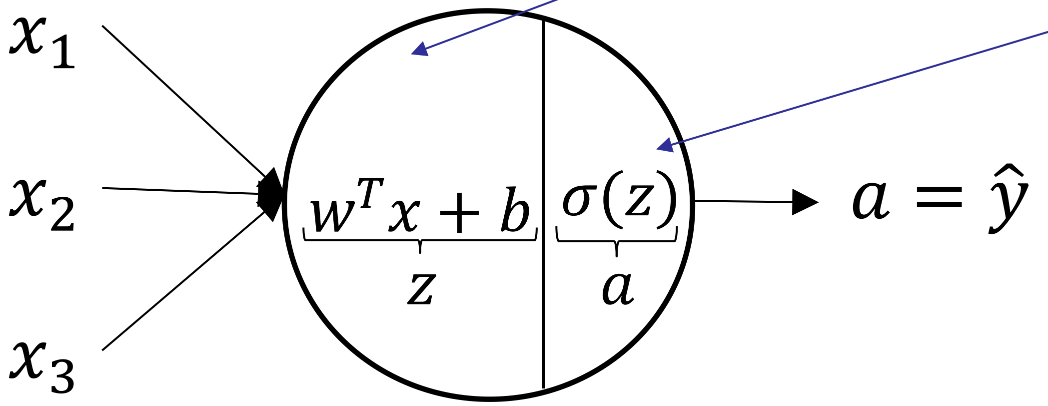


$$a^{[0]} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} \qquad a^{[2]} = \begin{bmatrix} a_2^{[1]} \end{bmatrix}$$

Tính toán và biểu diễn trên mạng neural

Trường hợp 1 điểm dữ liệu

Mạng neural



Hàm học:

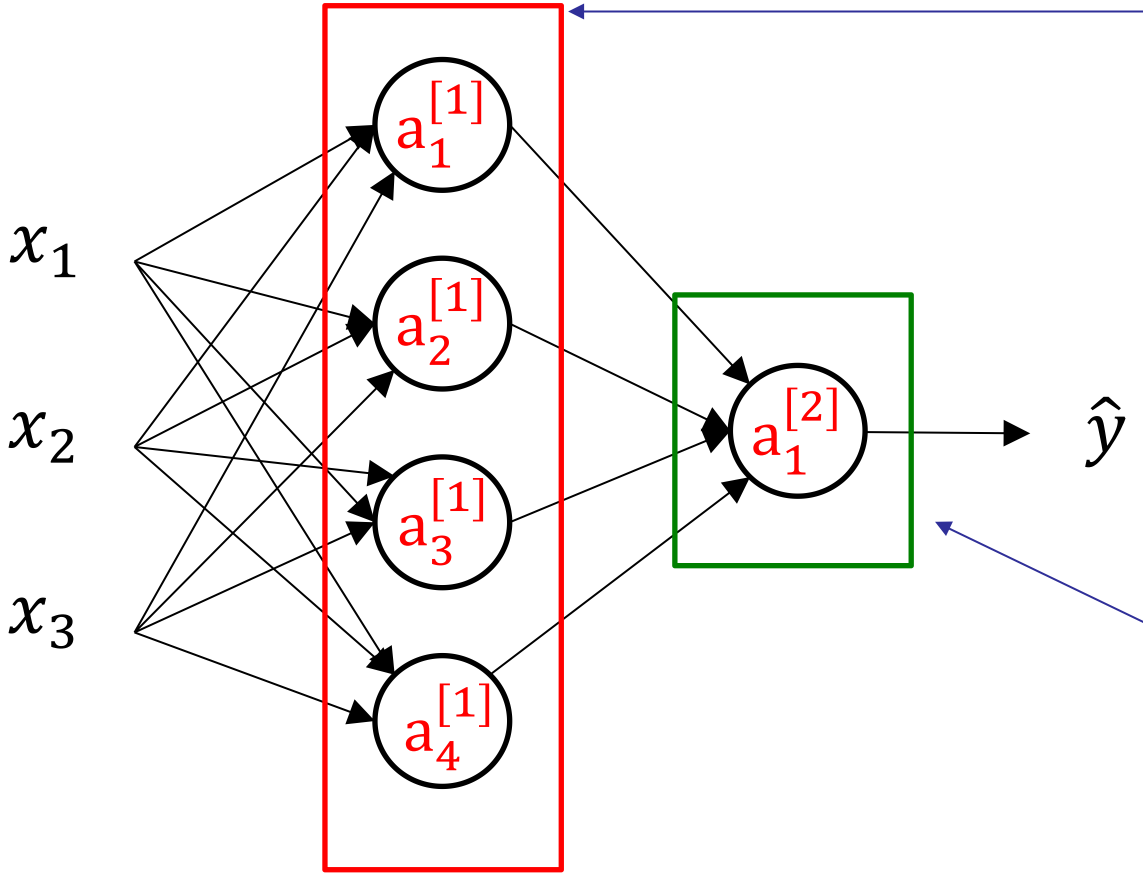
$$z = w^T x + b.$$

Hàm kích hoạt:

$$a = \sigma(z) = \frac{1}{1+e^{-z}}$$

Chức năng hàm kích hoạt: Cho phép thông tin nào được truyền qua, thông tin nào bị giữ lại.

Mạng neural



$a^{[0]}$

Lớp thứ 1:

$$z_1^{[1]} = w_1^{[1].T} x + b_1^{[1]} \mid a_1^{[1]} = \sigma(z_1^{[1]})$$
$$z_2^{[1]} = w_2^{[1].T} x + b_2^{[1]} \mid a_2^{[1]} = \sigma(z_2^{[1]})$$
$$z_3^{[1]} = w_3^{[1].T} x + b_3^{[1]} \mid a_3^{[1]} = \sigma(z_3^{[1]})$$
$$z_4^{[1]} = w_4^{[1].T} x + b_4^{[1]} \mid a_4^{[1]} = \sigma(z_4^{[1]})$$

Lớp thứ 2:

$$z_1^{[2]} = w_1^{[2].T} a^{[1]} + b_1^{[2]} \mid a_1^{[2]} = \sigma(z_1^{[2]})$$

Biểu diễn cho lớp 1

UIT VNUHCM University of Information Technology

Biểu thức ban đầu

$$\boxed{z_1^{[1]}} = w_1^{[1].T}x + b_1^{[1]} \mid a_1^{[1]} = \sigma(z_1^{[1]})$$

$$\boxed{z_2^{[1]}} = w_2^{[1].T}x + b_2^{[1]} \mid a_2^{[1]} = \sigma(z_2^{[1]})$$

$$\boxed{z_3^{[1]}} = w_3^{[1].T}x + b_3^{[1]} \mid a_3^{[1]} = \sigma(z_3^{[1]})$$

$$\boxed{z_4^{[1]}} = w_4^{[1].T}x + b_4^{[1]} \mid a_4^{[1]} = \sigma(z_4^{[1]})$$

Vector hoá

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} w_1^{[1]}x + b_1^{[1]} \\ w_2^{[1]}x + b_2^{[1]} \\ w_3^{[1]}x + b_3^{[1]} \\ w_4^{[1]}x + b_4^{[1]} \end{bmatrix} \xrightarrow{\sigma(z)} \begin{bmatrix} \sigma(w_1^{[1]}x + b_1^{[1]}) \\ \sigma(w_2^{[1]}x + b_2^{[1]}) \\ \sigma(w_3^{[1]}x + b_3^{[1]}) \\ \sigma(w_4^{[1]}x + b_4^{[1]}) \end{bmatrix}$$

(4x3) (3x1) (4x1) (4x1) (4x1)

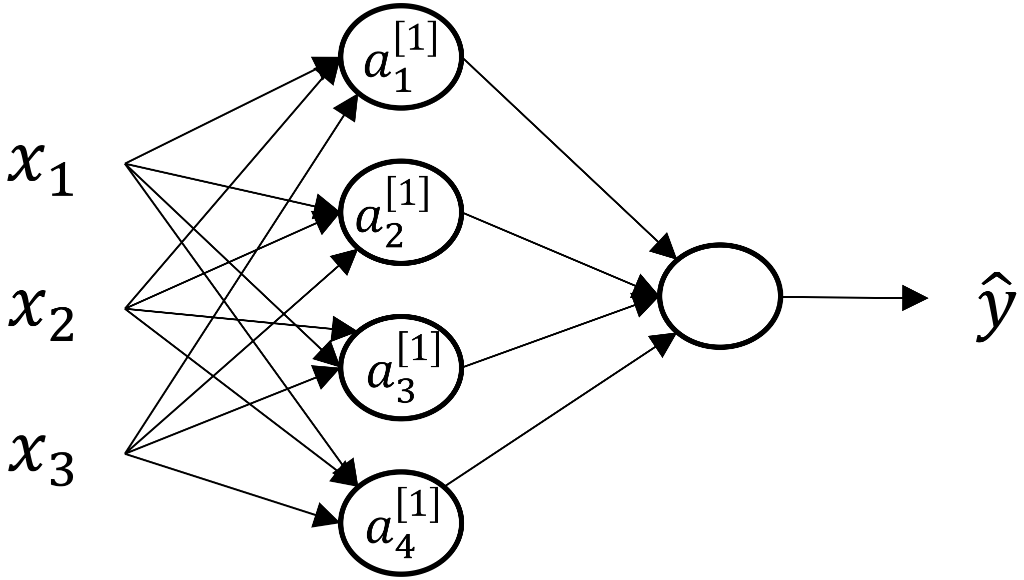
Biểu diễn cho lớp 2

Biểu thức ban đầu $z_1^{[2]} = w_1^{[2].T} a^{[1]} + b_1^{[2]} \mid a_1^{[2]} = \sigma(z_1^{[2]})$

Vector hoá $[w_1] * \begin{bmatrix} \sigma(w_1^{[2]} a_1^{[1]} + b_1^{[1]}) \\ \sigma(w_2^{[2]} a_2^{[1]} + b_2^{[1]}) \\ \sigma(w_3^{[2]} a_3^{[2]} + b_3^{[1]}) \\ \sigma(w_4^{[2]} a_4^{[2]} + b_4^{[1]}) \end{bmatrix} + [b_1] = \left[w_1^{[2]} x + b_1^{[2]} \right] \xrightarrow{\sigma(z)} \left[\sigma(w_1^{[2]} x + b_1^{[2]}) \right]$

(1x4) (4x1) (1x1) (1x1) (4x1)

Tổng kết



Given input x:

Truyền xuôi
(feed forward)

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

(4x1) (4x3) (3x1) (4x1)

$$a^{[1]} = \sigma(z^{[1]})$$

(4x1) (4x1)

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

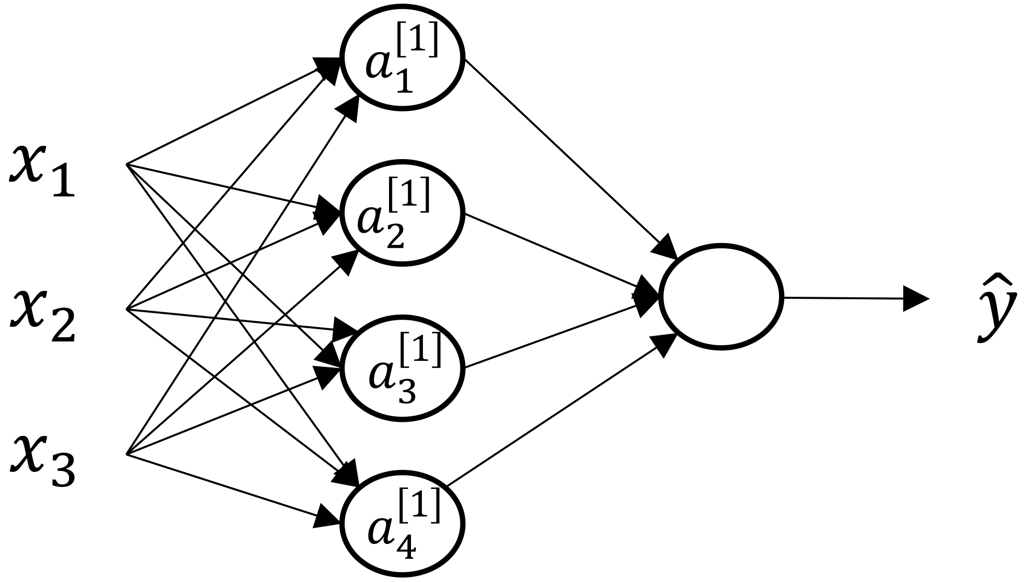
(1x1) (1x4) (4x1) (1x1)

$$a^{[2]} = \sigma(z^{[2]})$$

(1x1) (1x1)

Trường hợp m điểm dữ liệu

Tổng kết



Given m input x:

For i = 1 to m:

$$z^{[1](i)} = W^{[1](i)}x^{(i)} + b^{[1]}$$

$$a^{[1](i)} = \sigma(z^{[1](i)})$$

$$z^{[2](i)} = W^{[2](i)}a^{[1](i)} + b^{[2](i)}$$

$$a^{[2](i)} = \sigma\left(z^{[2](i)}\right) \rightarrow \hat{y}^{(i)}$$

Biểu diễn các giá trị X, z và a

Given m input x:

For i = 1 to m:

$$z^{[1](i)} = W^{[1](i)} x^{(i)} + b^{[1]}$$

(4xm) (4x3) (3xm) (4x1)

$$a^{[1](i)} = \sigma(z^{[1](i)})$$

(4xm) (4xm)

$$z^{[2](i)} = W^{[2](i)} a^{[1](i)} + b^{[2](i)}$$

(1xm) (1x4) (4xm) (1x1)

$$a^{[2](i)} = \sigma(z^{[2](i)}) \rightarrow \hat{y}^{(i)}$$

(1xm) (1xm)

$$X = \left[\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} \begin{bmatrix} x_1^{(3)} \\ x_2^{(3)} \\ x_3^{(3)} \end{bmatrix} \cdots \begin{bmatrix} x_1^{(m)} \\ x_2^{(m)} \\ x_3^{(m)} \end{bmatrix}\right]$$

$$z = \left[\begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \\ z_4^{(1)} \end{bmatrix} \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \end{bmatrix} \begin{bmatrix} z_1^{(3)} \\ z_2^{(3)} \\ z_3^{(3)} \\ z_4^{(3)} \end{bmatrix} \cdots \begin{bmatrix} z_1^{(m)} \\ z_2^{(m)} \\ z_3^{(m)} \\ z_4^{(m)} \end{bmatrix}\right]$$

$$a = \left[\begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \\ a_4^{(1)} \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(3)} \\ a_2^{(3)} \\ a_3^{(3)} \\ a_4^{(3)} \end{bmatrix} \cdots \begin{bmatrix} a_1^{(m)} \\ a_2^{(m)} \\ a_3^{(m)} \\ a_4^{(m)} \end{bmatrix}\right]$$

Hàm kích hoạt

Hàm kích hoạt

Given input x:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = \sigma(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = \sigma(z^{[2]})$$

σ là hàm kích hoạt trong ví dụ ở trên.

Hàm σ là hàm sigmioid, có dạng:

$$\sigma = \frac{1}{1 + e^{-z}}$$

Ngoài hàm sigmoid ra, ta có thể sử dụng các hàm kích hoạt khác như: tanh, relu, maxout, ELU, ...

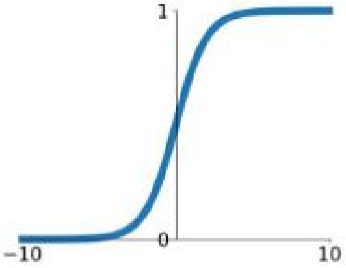
Chức năng của hàm kích hoạt là điều khiển thông tin truyền qua các lớp của mạng neural:

Cho hoặc không cho thông tin đi qua

Một số hàm kích hoạt

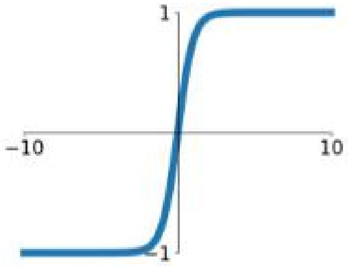
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



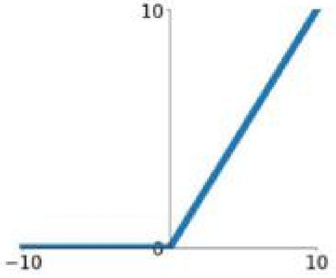
tanh

$\tanh(x)$



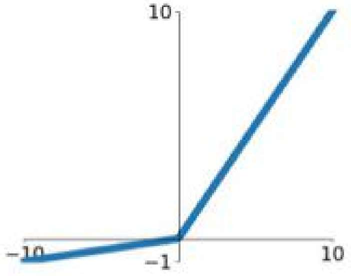
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

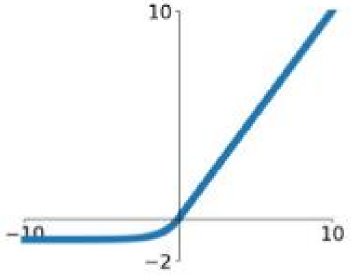


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

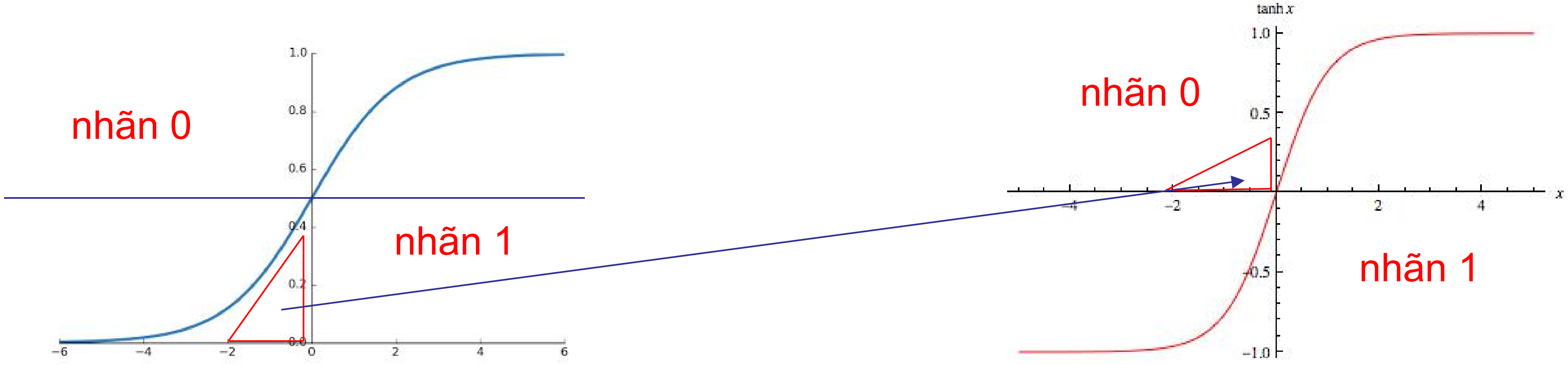
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Sigmoid với tanh

Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$

Tanh: $g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$



– Mean = 0.5.

– Vùng tam giác màu đỏ sẽ ở nhãn 1.

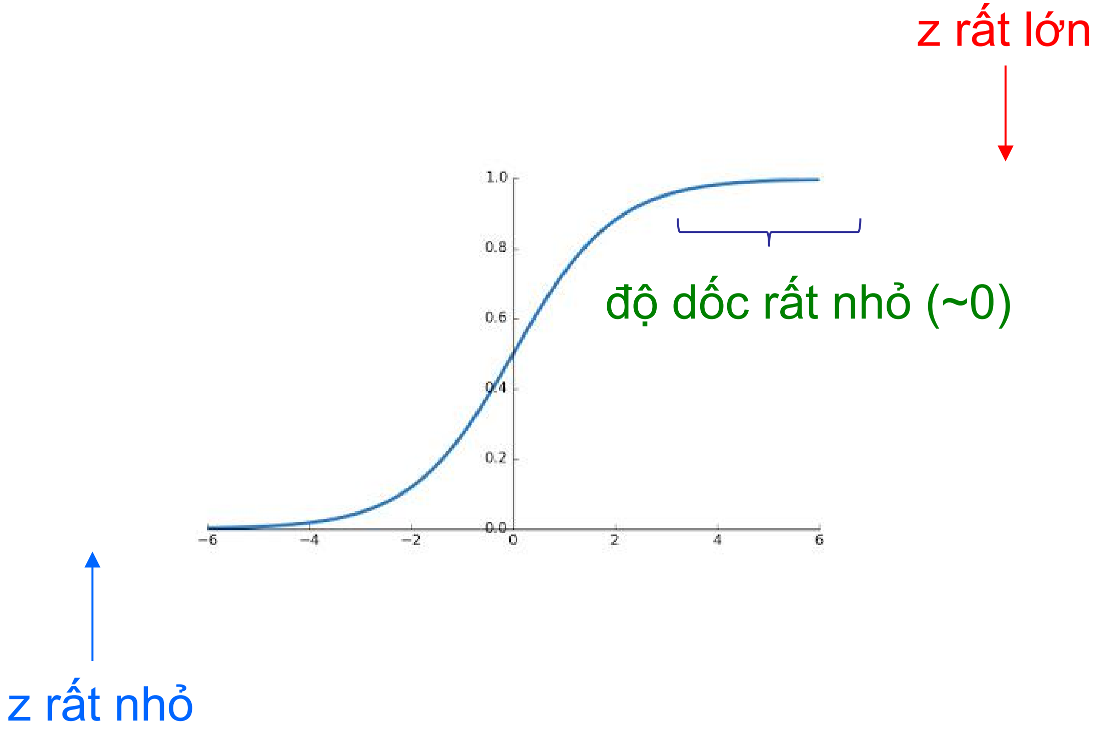
– Mean = 0.

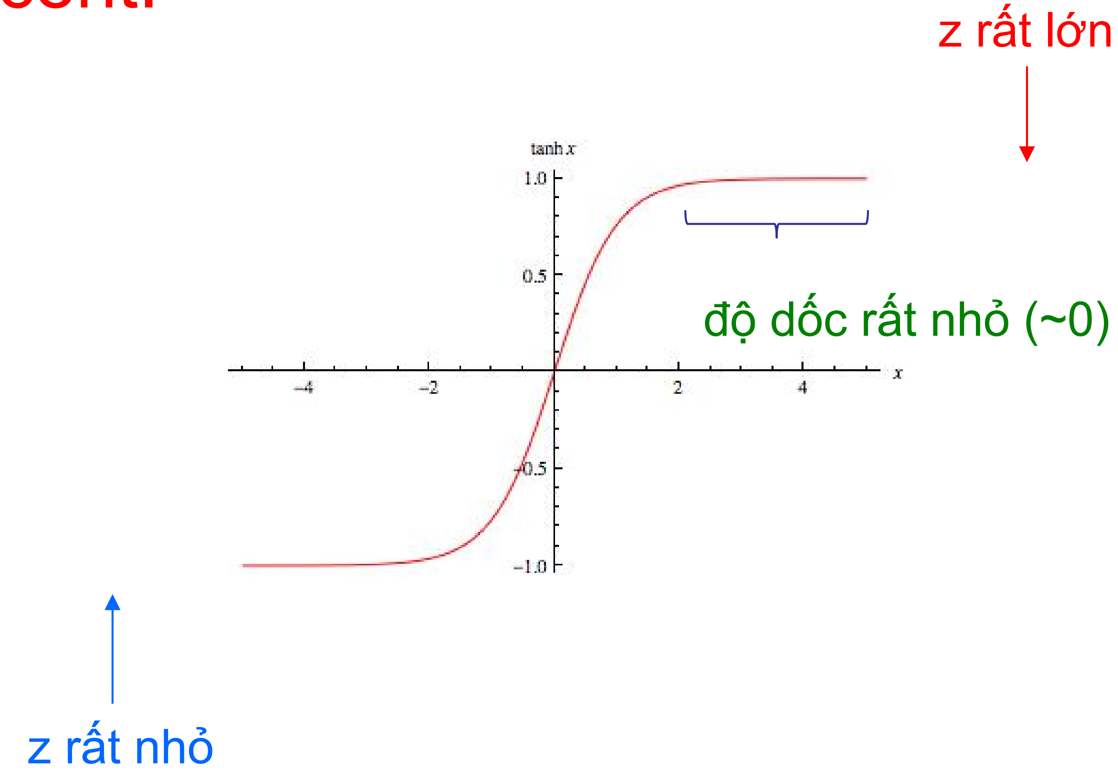
– Vùng tam giác màu đỏ sẽ ở nhãn 0.

Chọn hàm kích hoạt như thế nào phải dựa vào đặc điểm của bộ dữ liệu huấn luyện

Nhược điểm của Sigmoid và Tanh

– Nếu z quá lớn, hoặc z quá nhỏ, đạo hàm (độ dốc) sẽ rất nhỏ ➔ làm chậm quá trình gradient descent.





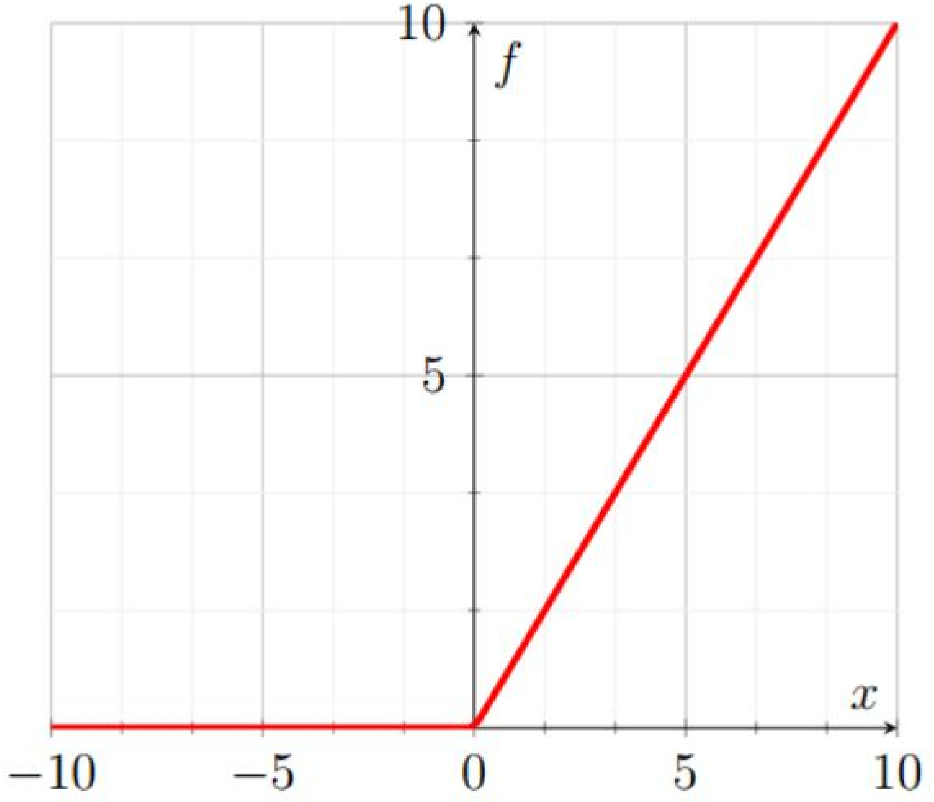
ReLU

Công thức: r(z) = max(0,z)

Ý nghĩa:

- Đạo hàm sẽ luôn là 1, miễn là giá trị z dương.
- Đạo hàm sẽ luôn là 0 khi z âm.

➔ Giải quyết được trường hợp z vô cùng lớn, hoặc là vô cùng bé.



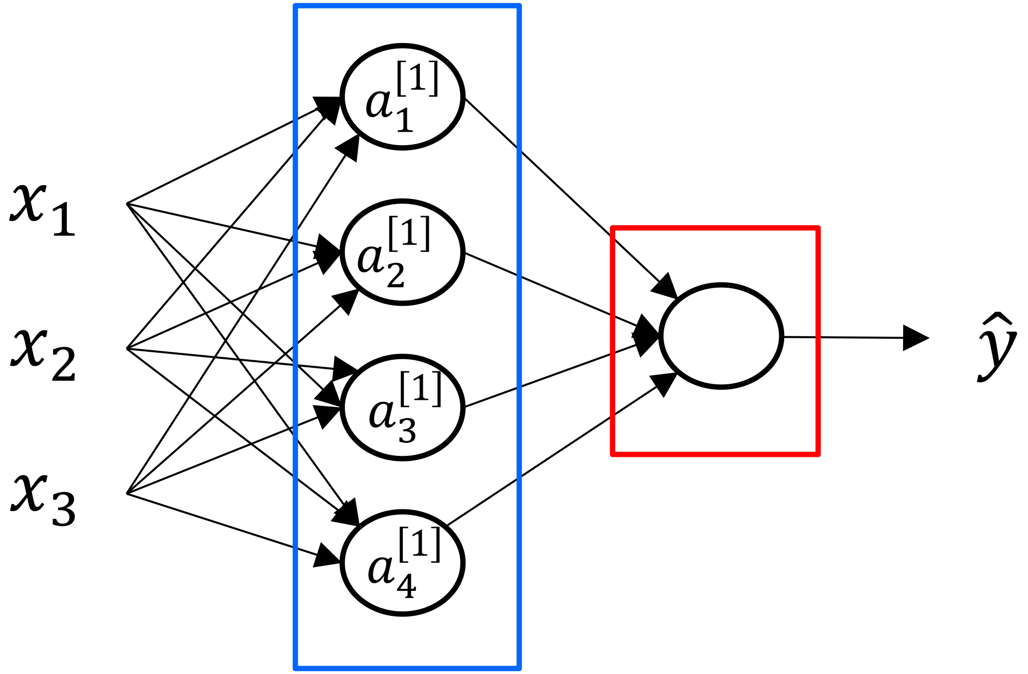
z = 0.0000000000000....

UIT VNUHCM University of Information Technology

Một số lưu ý

- Giữa các lớp khác nhau trong mạng neural thì sẽ có các hàm kích hoạt khác nhau.
- Chọn hàm kích hoạt như thế nào là tuỳ vào đặc điểm của dữ liệu huấn luyện.

$\sigma(a^{[2]})$ là hàm sigmoid

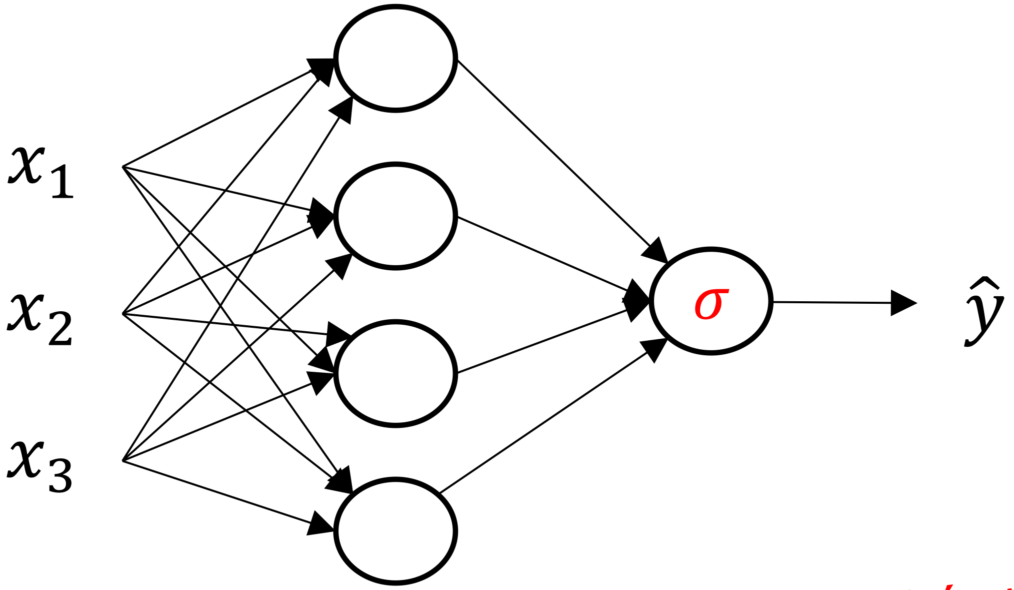


$g(a^{[1]})$ là hàm tanh

Tại sao lại cần Activation fuction phi tuyến (non-linear)

Tuyến tính vs Phi tuyến

Mạng neural



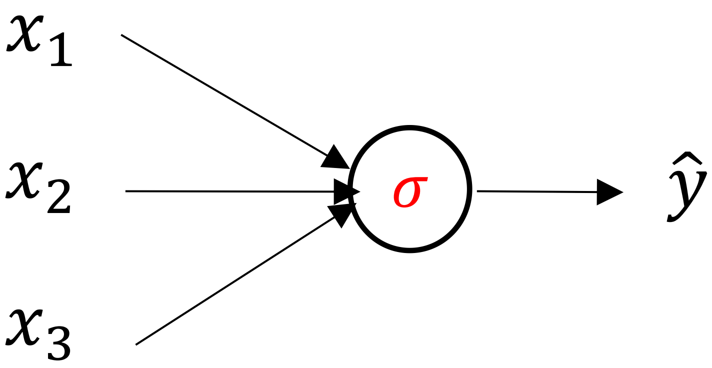
$z^{[1]} = W^{[1]} * X + b^{[1]}$

a là hàm tuyến tính

➔ $a^{[1]} = z$ ➔ $\hat{y} = \sigma(z)$

Có khác nhau gì giữa mạng neural và logistic regression ?

Logistic regression

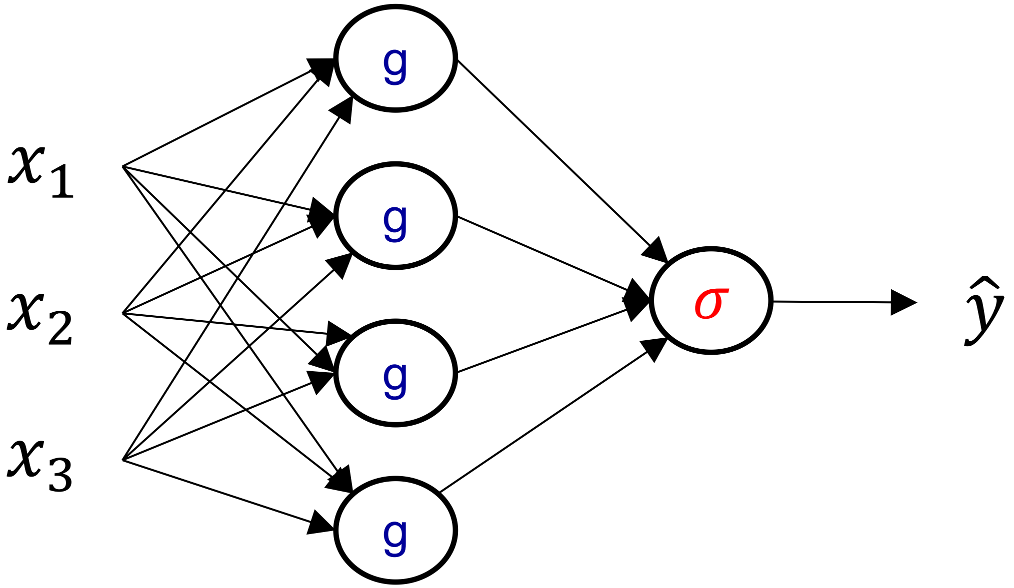


$z^{[1]} = W^{[1]} * X + b^{[1]}$

$\hat{y} = \sigma(z)$

Tuyến tính vs Phi tuyến

Mạng neural

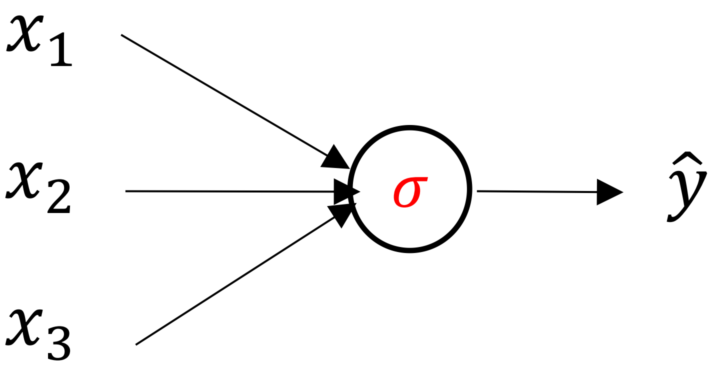


$z^{[1]} = W^{[1]} * X + b^{[1]}$

Sử dụng hàm kích hoạt tanh:

➔ $a^{[1]} = g(z)$ ➔ $\hat{y} = g(\sigma(z))$

Logistic regression



$z^{[1]} = W^{[1]} * X + b^{[1]}$

$\hat{y} = \sigma(z)$

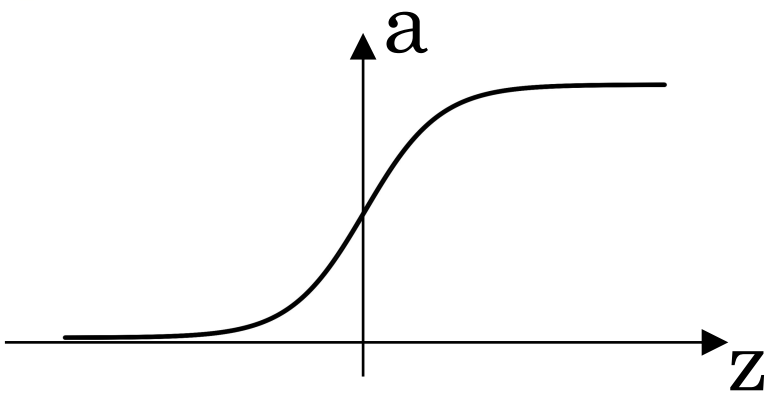
Kết luận

- Nếu sử dụng hàm tuyến tính, thì chức năng của các hidden layer gần như là vô dụng.
  - Không khác so với các mô hình máy học truyền thống là bao nhiêu (trong ví dụ trên là Logistic Regression).
- Ngoại lệ: Chỉ có một nơi sử dụng được hàm tuyến tính: bài toán hồi quy tuyến tính

$$g(x) = z$$

Đạo hàm của một số hàm kích hoạt

Sigmoid



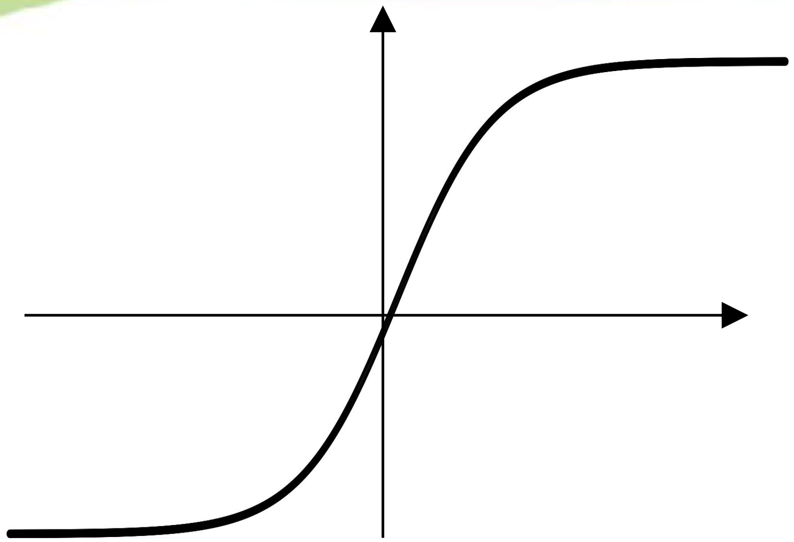
$$g(z) = \frac{1}{1+e^{-z}}$$

$$g'(z) = \frac{\mathrm{d}(g(z))}{\mathrm{dx}} = \frac{1}{1+e^{-z}} * \left(1 - \frac{1}{1+e^{-z}}\right)$$

= g(z) * (1-g(z))

z	g(z)	g'(z)
10	1	1*(1-1) = 0
-10	0	0*(1-0) = 0
0	0.5	0.5*(1-0.5) = 0.25

Tanh



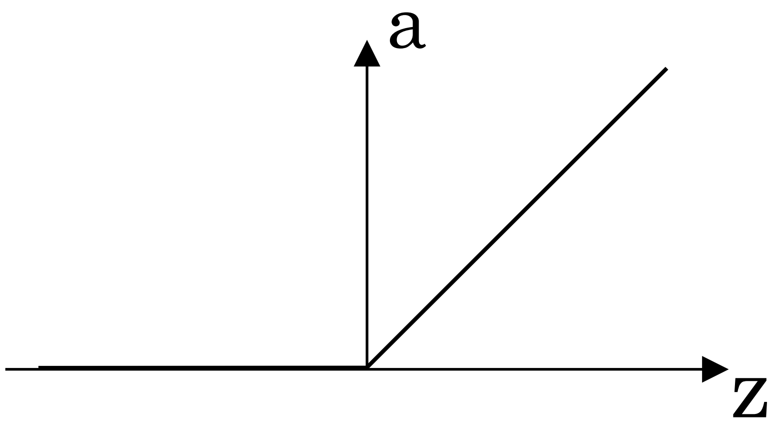
$$g(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = \frac{d(g(z))}{dx} = 1 - \tanh(z)^{2}$$

$$= 1 - (g(z))^2$$

z	g(z)	g'(z)
10	1	1 - 1 = 0
-10	-1	$1 - (-1)^2 = 0$
0	0	1 – 0 = 1

ReLU



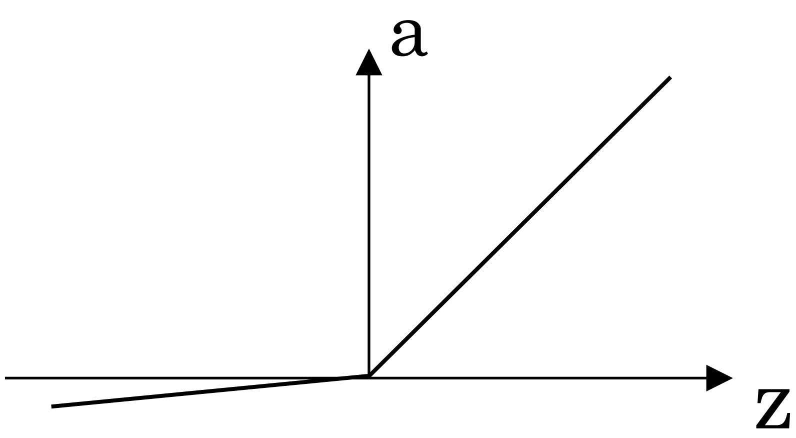
ReLU

$$g(z) = max(0, z)$$

$$g'(z) = \begin{cases} 0, \ if\ z < 0 \\ 1, \ if\ z \geq 0 \\ unf,\ z = 0 \end{cases}$$

chỉ đúng trong toán học

Leaky ReLU



Leaky ReLU

$$g(z) = max(0.01 * z, z)$$

$$g'(z) = \begin{cases} 0.01, & if\ z < 0 \\ 1, & if\ z \geq 0 \end{cases}$$

Gradient Descent trong mạng neural

Quá trình tính gradient

$$x,\ W^{[1]},\ b^{[1]} \rightarrow \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \rightarrow \boxed{a^{[1]} = \sigma(z^{[1]})} \rightarrow \boxed{z^{[2]} = W^{[2]}x + b^{[2]}} \rightarrow \boxed{a^{[2]} = \sigma(z^{[2]})} \rightarrow \boxed{\mathcal{L}(a^{[2]}, y)}$$

$$\frac{dL}{da^{[1]}} \qquad W^{[2]},\ b^{[2]}$$

$$\frac{dL}{dz^{[1]}} = w^{[2]}.T * dz^{[2]} \otimes g'^{[1]} * dz^{[1]}$$

$$\frac{dL}{dz^{[2]}} = a^{[2]} - Y$$

$$\frac{dL}{da^{[2]}}$$

$$\frac{dL}{dw^{[1]}} = dz^{[1]}x^T$$

$$\frac{dL}{dw^{[2]}} = dz^{[2]}a^{[1].T}$$

$$\frac{dL}{db^{[1]}} = dz^{[1]}$$

$$\frac{dL}{db^{[2]}} = dz^{[2]}$$

Tổng hợp

Feed forward

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = \sigma(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = \sigma(z^{[2]})$$

Back propagation

$dz^{[2]} = A^{[2]} - Y.$

$dw^{[2]} = (1/m) * dz^{[2]} * A^{[1].T}$

$db^{[2]} = 1/m * np.sum(dz^{[2]}, axis=1, keepdims = True)$

(1 - np.power(A1, 2)

$dz^{[1]} = w^{[2].T} * dz^{[2]} \otimes [g'^{[1]}(z^{[1]})]$

$dw^{[1]} = (1/m) * dz^{[1]} * X^{T}$

$db^{[2]} = (1/m) * np.sum(dz^{[1]}, axis=1, keepdims = True)$

Tính GD

Các tham số (parameter):

$W^{[1]}$,	$b^{[1]}$,	$W^{[2]}$,	$b^{[2]}$
$(n^{[1]}, n^{[0]})$	$(n^{[1]}, 1)$	$(n^{[2]}, n^{[1]})$	$(n^{[2]}, 1)$

Chiều:

$nx = n^{[0]}, n^{[1]}, n^{[2]}$.

Hàm chi phí (Cost function):

$$J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}, y)$$

Mục tiêu:

Tìm các giá trị của các tham số $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$ sao cho J đạt tối ưu.

```
Repeat : {
        Tính ŷ(i) (i = 1...m)
        dw[1] = dJ / dw[1] ;dw[2] = dJ/dw[2]
        db[1] = dJ / db[1] ;db[2] = dJ/db[2]
        W[1] = W[1] - α(dJ / dw[1])
        W[2] = W[2] - α(dJ / dw[2])
        b[1] = b[1] - α(dJ / db[1])
        b[2] = b[2] - α(dJ / db[2])
}
//Lặp cho tới khi hội tụ
```

Khởi tạo trọng số

Thuật toán gradient descent

– Bước 1: Khởi tạo trọng số W và b.

+ W = W0.

+ b = b0.

```
W  := W0
b  := b0
```

Khởi tạo trọng số W và b như thế nào ?

– Bước 2: Lặp

Ứng với mỗi bước lặp, cập nhật trọng số cho W và b.

$$W = W - \alpha * \frac{dJ(w,b)}{dw}.$$

$$b = b - \alpha * \frac{dJ(w,b)}{db}.$$

α được gọi là learning rate.

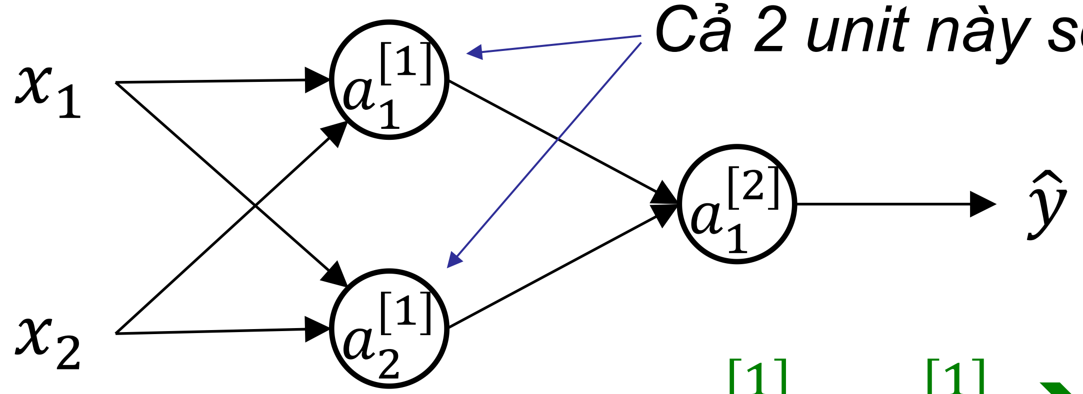
Repeat {

$$W := W - \alpha * \frac{dJ(w,b)}{dw}$$

$$b := b - \alpha * \frac{dJ(w,b)}{db}$$

}

Khởi tạo tham số bằng 0



Cả 2 unit này sẽ giống nhau

$a_1^{[1]} = a_2^{[1]}$ ➔ $\mathrm{dz}_1^{[1]} = \mathrm{dz}_2^{[1]}$ ⇨ $\mathrm{dw} = \begin{bmatrix} \mathrm{v} & \mathrm{v} \\ \mathrm{v} & \mathrm{v} \end{bmatrix}$

$\mathrm{w}^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

$\mathrm{b}^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$$a_1^{[1]} = \sigma\left(\mathrm{w}_1^{[1]} * \mathrm{x} + \mathrm{b}_1^{[1]}\right) = 0$$

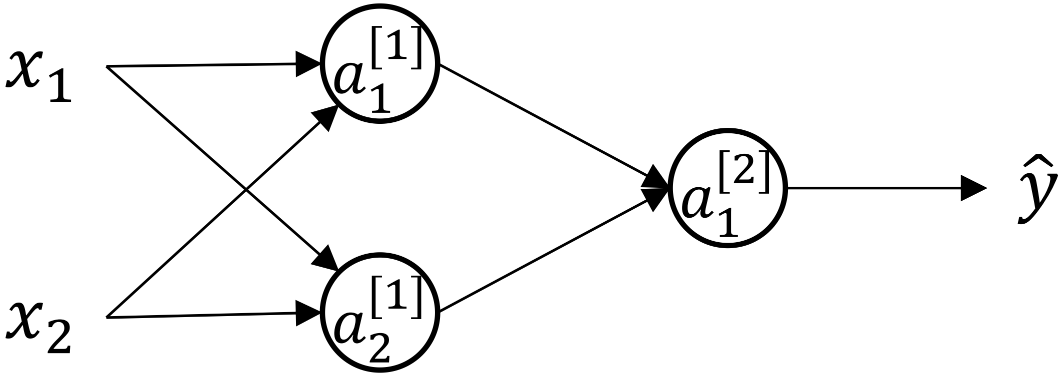
$$a_2^{[1]} = \sigma\left(\mathrm{w}_2^{[1]} * \mathrm{x} + \mathrm{b}_2^{[1]}\right) = 0$$

dw giống nhau ở 2 unit:

➔ khi thay vào gradient descent, trọng số W không có gì thay đổi sau mỗi lần chạy ở các unit.

➔ gradient descent không còn ý nghĩa.

Khởi tạo tham số khác 0



Khởi tạo W gồm các số ngẫu nhiên theo phân phối Gauss. W có chiều là (2,2)

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$w^{[i]}$ = np.random.randn((2,2)) * $\boxed{0.01}$

$b^{[i]}$ = np.zero((2,1))

Tính tương tự với $W^{[2]}$, $b^{[2]}$

w rất lớn → z có thể rất lớn hoặc rất nhỏ → làm chậm quá trình gradient descent.

Câu hỏi

Với Logistic regression, khởi tao W = 0 được hay không ?

Trả lời:

Được, vì Logistic regression chỉ có 1 unit duy nhất (không có hidden layer)

TÀI LIỆU THAM KHẢO

1. Khoá học *Neural Network and Deep learning*, deeplearning.ai.
2. Ian Goodfellow, Yoshua Bengio, Aaron Courvile, *Deep learning*, MIT Press, 2016.
3. Andrew Ng., *Machine Learning Yearning*. Link: https://www.deeplearning.ai/machine-learning-yearning/
4. Vũ Hữu Tiệp, *Machine Learning cơ bản*, NXB Khoa học và Kỹ thuật, 2018.