Neural Network (continued)

VietAI teaching team

Ôn tập: Kiến trúc

- Khái niệm:
  - Lớp (layer)
  - Nơ-ron (neuron)
  - Trọng số (weight)
  - Đơn vị điều chỉnh (bias)

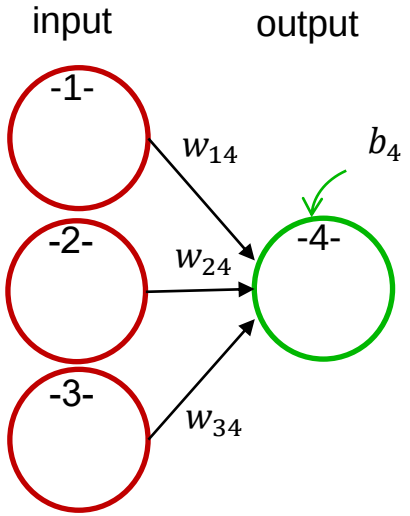
} parameters

- Tính chất:
  - Kết nối đủ (fully connected)
  - Số lượng hidden layer
  - Số lượng neuron mỗi hidden layer

} hyper-parameters

Ôn tập: Lan truyền thuận

- **Ví dụ:** Xét mạng neuron có cấu trúc và parameters sau



$$w_{14} = 0.1$$
$$w_{24} = 0.6 \quad b_4 = 0.8$$
$$w_{34} = 0.5$$

- Giá trị đầu vào:

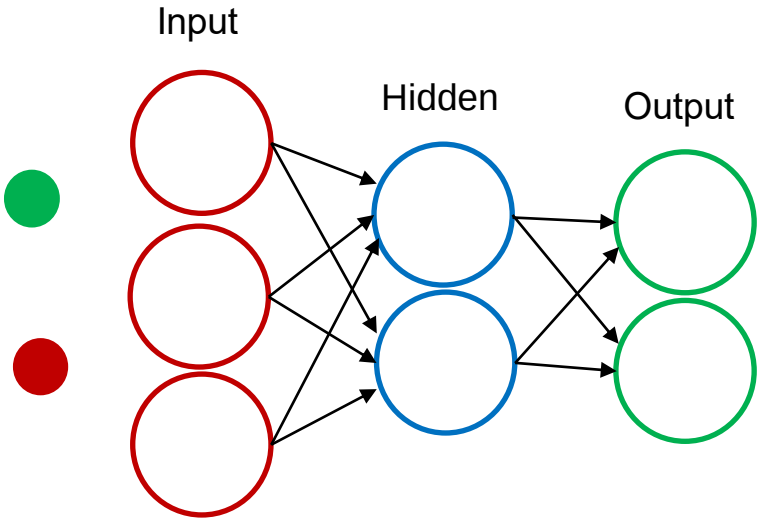
$$x_1 = 5$$
$$x_2 = 3$$
$$x_3 = 2$$

- Quá trình tính output:

$$\hat{y}_4 = x_1 \times w_{14} + x_2 \times w_{24} + x_3 \times w_{34} + b_4$$
$$= 5 \times 0.1 + 3 \times 0.6 + 2 \times 0.5 + 0.8$$
$$= 0.5 + 1.8 + 1 + 0.8$$
$$= 4.1$$

Ôn tập: Hàm kích hoạt

- Xét neural net sau



Activation function

$$\hat{y} = \left(\sum x.w\right) + bias$$

→ $\hat{y} \in (-\infty; +\infty)$

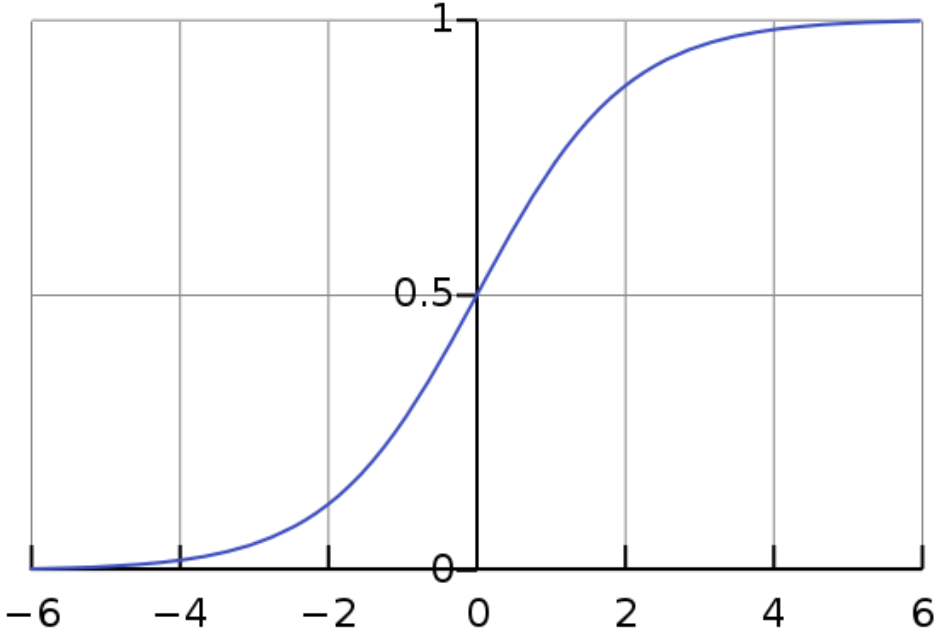
→ Không thể biết neuron nào cần "activate"

Vẫn là mô hình phân lớp tuyến tính dù thêm nhiều hidden layer

Ôn tập: Hàm kích hoạt

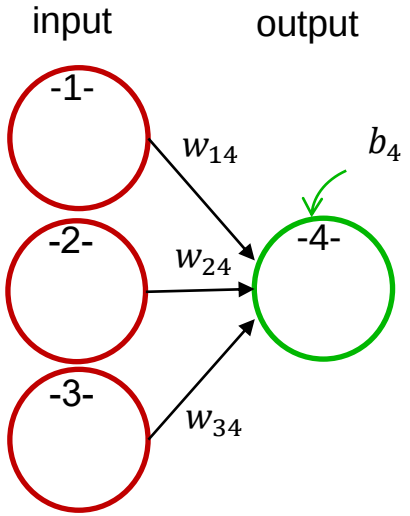
Hàm sigmoid:

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



Ôn tập: Lan truyền thuận

- **Ví dụ:** Xét mạng neuron có cấu trúc và parameters sau



$$w_{14} = 0.1$$
$$w_{24} = 0.6 \quad b_4 = 0.8$$
$$w_{34} = 0.5$$

- Giá trị đầu vào:

$$x_1 = 5$$
$$x_2 = 3$$
$$x_3 = 2$$

- Quá trình tính output:

tạm $\hat{y}_4 = x_1 \times w_{14} + x_2 \times w_{24} + x_3 \times w_{34} + b_4$

$$= 5 \times 0.1 + 3 \times 0.6 + 2 \times 0.5 + 0.8$$
$$= 0.5 + 1.8 + 1 + 0.8$$
$$= 4.1$$

$$\hat{y}_4 = \text{sigmoid}(\textbf{tạm}) = \frac{1}{1+e^{-4.1}} = 0.983$$

Ôn tập: Quá trình học

- Bước 0**:** Tạo cấu trúc mạng, chuẩn bị tập dữ liệu huấn luyện, gồm đầu vào x và nhãn y, tốc độ học α
- Bước 1**:** Khởi tạo ngẫu nhiên tất cả các giá trị weights
- Bước 2**:** Thực hiện Lan truyền thuận đến output layer, được các giá trị h trong hidden neuron và $\hat{y}$ trong output neuron
- **Bước 3:** Lan truyền ngược:

  Đối với output layer

  - **3.1:** Tính độ lỗi: $e_k = y_k - \hat{y}_k$
  - **3.2:** Tính sai số gradient: $\delta_k = \hat{y}_k \times (1 - \hat{y}_k) \times e_k$
  - **3.3:** Tính giá trị cần thay đổi: $\Delta w_{jk} = h_j \times \delta_k$
  - **3.4:** Cập nhật trọng số: $w_{jk} = w_{jk} + \alpha \times \Delta w_{jk}$
  - **3.5:** Cập nhật bias: $b_k = b_k + \alpha \times \delta_k$

  Đối với hidden layer, bỏ **3.1** và thay đổi **3.2**:

  - **3.2:** Tính sai số gradient: $\delta_k = h_k \times (1 - h_k) \times \sum_{i=1}^{n} \delta_i \times w_{ki}$

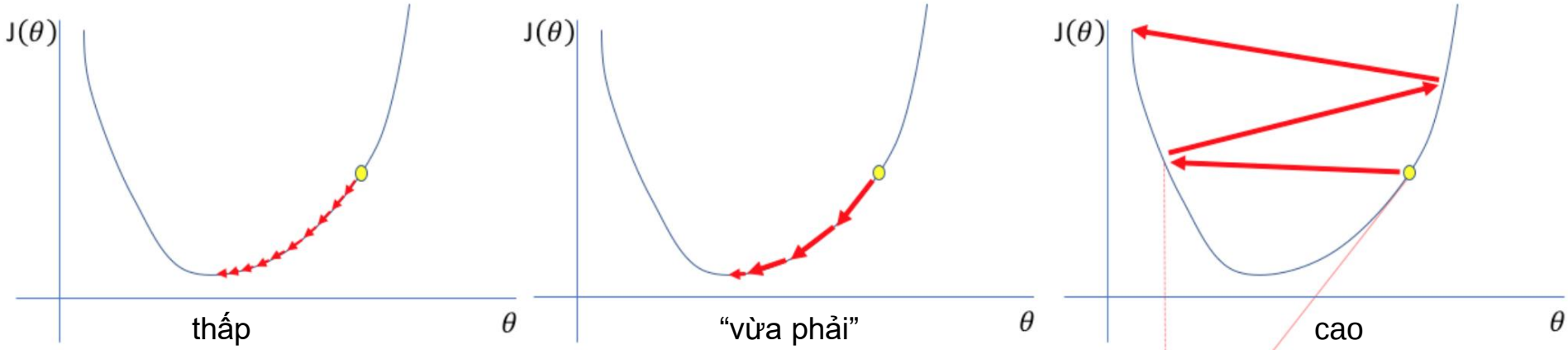
Kết quả

- Với giá trị trọng số và độ điều chỉnh tối ưu, kết quả dự đoán (predict) cho phép toán XOR như sau

Đầu vào		Kết quả mong muốn	Kết quả từ neural network	Sai số
x_1	x_2	y	$\hat{y}$	e
1	1	0	0.0486	-0.0486
0	1	1	0.9543	0.0457
1	0	1	0.9544	0.0456
0	0	0	0.0508	-0.0508

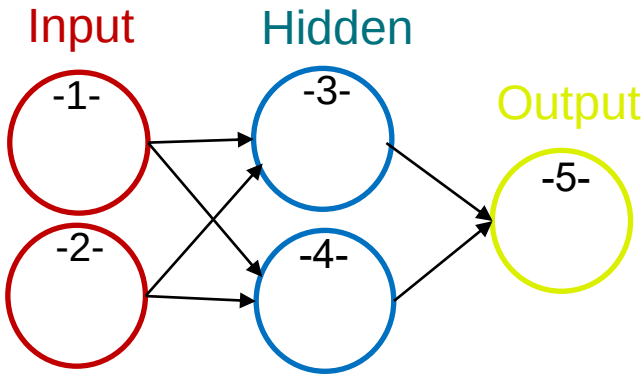
8 Hyper parameters

- Số lượng neuron trong hidden layer:
  - Ít: hội tụ nhanh, phù hợp các bài toán phân lớp đơn giản
  - Nhiều: hội tụ chậm, dễ gây overfitting khi có ít dữ liệu để train. Bù lại model sẽ học được những phân phối phức tạp (nếu đủ dữ liệu)
- Hệ số học/ tốc độ học (learning rate):



8 Tính toán trên ma trận

- Sử dụng phép nhân ma trận (dot product) để thực hiện các phép tính trong neural network



➢ *Tính toán trên scalar* - Input: x_1, x_2

$$h_3 = \textbf{activate}(x_1 \times w_{13} + x_2 \times w_{23} + b_3)$$

$$h_4 = \textbf{activate}(x_1 \times w_{14} + x_2 \times w_{24} + b_4)$$

$$y_5 = \textbf{activate}(h_3 \times w_{35} + h_4 \times w_{45} + b_5)$$

➢ *Tính toán trên ma trận:* Gộp scalar → ma trận - Input: $\mathbf{X} = [x_1 \quad x_2]$

$$\mathbf{H} = \textbf{activate}(\mathbf{X}.\mathbf{W} + \mathbf{b})$$

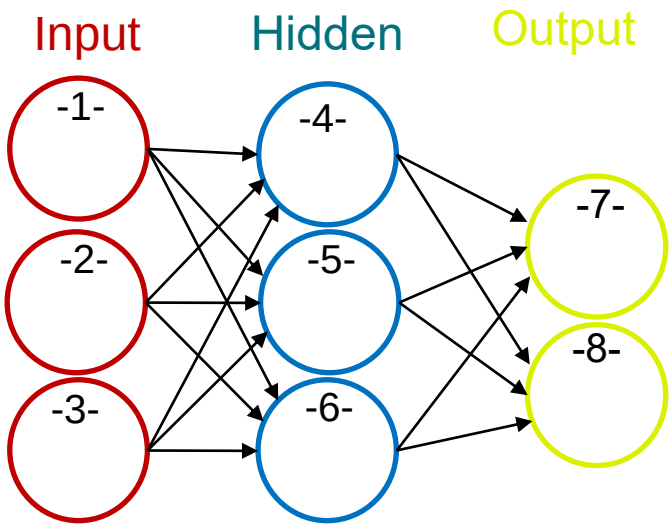
$$= \textbf{activate}([x_1 \quad x_2].\begin{bmatrix} w_{13} & w_{14} \\ w_{23} & w_{24} \end{bmatrix} + [b_3 \quad b_4])$$

$$= \textbf{activate}([x_1 \times w_{13} + x_2 \times w_{23} \quad x_1 \times w_{14} + x_2 \times w_{24}] + [b_3 \quad b_4])$$

$$= \textbf{activate}([h_3 \quad h_4])$$

8 Bài tập về nhà

Bài 1: Cho mạng neuron có cấu trúc sau:



$$w_{14} = 0.2$$
$$w_{15} = 0.1$$
$$w_{16} = -0.3$$
$$w_{24} = 0.4$$
$$w_{25} = 0.7$$
$$w_{26} = 0.3$$
$$w_{34} = -0.5$$
$$w_{35} = 0.1$$
$$w_{36} = 0.9$$

$$w_{47} = 0.4$$
$$w_{48} = 0.6$$
$$w_{57} = -0.2$$
$$w_{58} = -0.1$$
$$w_{67} = 0.7$$
$$w_{68} = 0.8$$

$$b_4 = -0.6$$
$$b_5 = 1.0$$
$$b_6 = -0.2$$
$$b_7 = 0.8$$
$$b_8 = 0.3$$

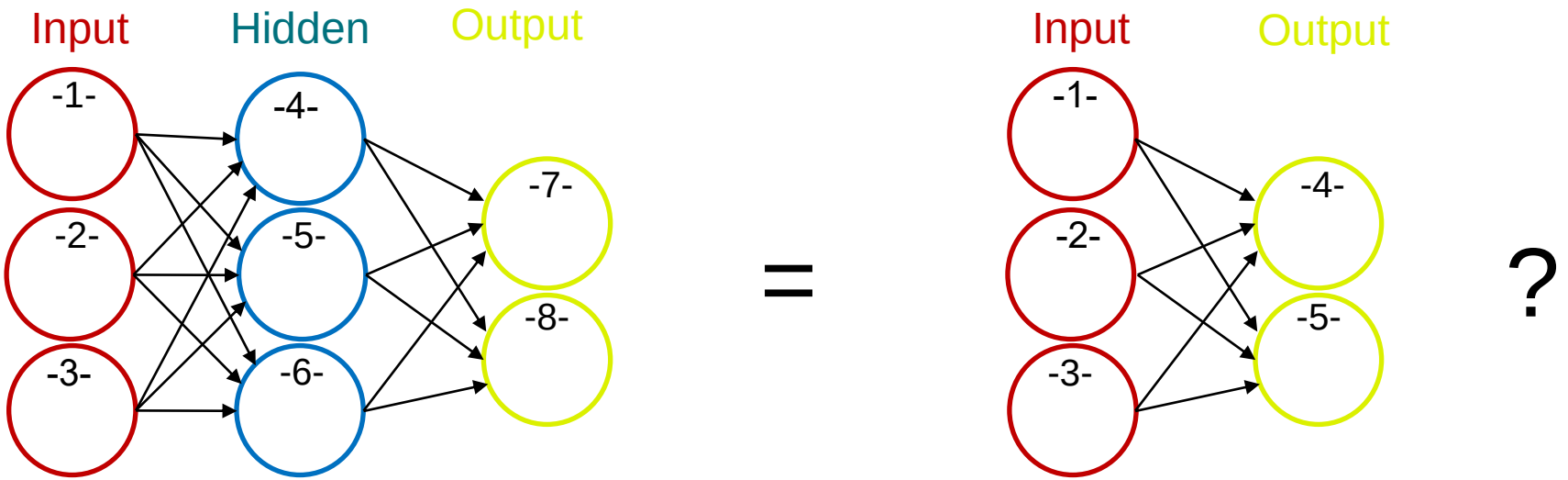
Với tốc độ học $\alpha = 0.1$, hàm kích hoạt là sigmoid và input $\mathbf{X} = [3,1,-2]$.
Hãy tính kết quả lớp output $\mathbf{Y}$ sau khi thực hiện Lan truyền thuận lần 3
Sử dụng phương pháp nhân ma trận thay vì scalar

Forward prop → *back prop* → **forward prop** → *back prop* → **forward prop**

8 Bài tập về nhà

Bài 2 (optional): Hãy chứng minh:

Với parameters tùy ý, nếu không có activation function, thì hai models sau hoàn toàn tương tự nhau



Basic FNN formulas in a nutshell ☺

- Bước 0**:** Tạo cấu trúc mạng, chuẩn bị tập dữ liệu huấn luyện, gồm đầu vào x và nhãn y, tốc độ học α
- Bước 1**:** Khởi tạo ngẫu nhiên tất cả các giá trị weights
- Bước 2**:** Thực hiện Lan truyền thuận đến output layer, được các giá trị h trong hidden neuron và $\hat{y}$ trong output neuron
- **Bước 3:** Lan truyền ngược:

  Đối với output layer

  - **3.1:** Tính độ lỗi: $e_k = y_k - \hat{y}_k$
  - **3.2:** Tính sai số gradient: $\delta_k = \hat{y}_k \times (1 - \hat{y}_k) \times e_k$
  - **3.3:** Tính giá trị cần thay đổi: $\Delta w_{jk} = h_j \times \delta_k$
  - **3.4:** Cập nhật trọng số: $w_{jk} = w_{jk} + \alpha \times \Delta w_{jk}$
  - **3.5:** Cập nhật bias: $b_k = b_k + \alpha \times \delta_k$

  Đối với hidden layer, bỏ **3.1** và thay đổi **3.2**:

  - **3.2:** Tính sai số gradient: $\delta_k = h_k \times (1 - h_k) \times \sum_{i=1}^{n} \delta_i \times w_{ki}$

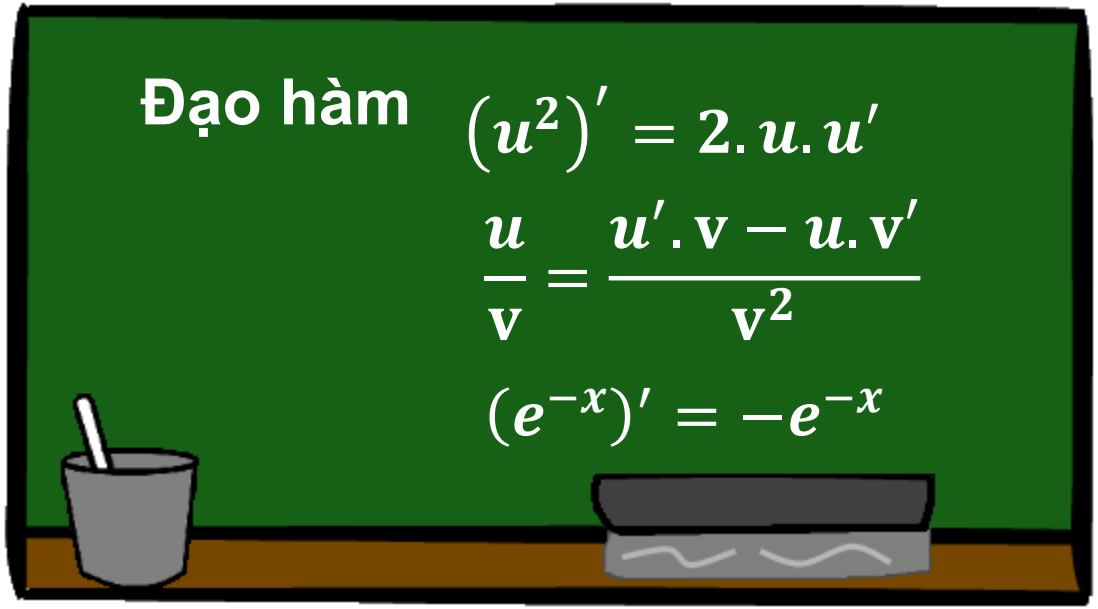
Nội dung

1. Chứng minh công thức lan truyền ngược
2. Chức năng của activation function
3. Một số activation function
4. Một số hàm cost thông dụng
5. Biến thể của gradient descent
6. Momentum
7. Regularization trong mạng neuron

Nội dung

1. Chứng minh công thức lan truyền ngược
2. Chức năng của activation function
3. Một số activation function
4. Một số hàm cost thông dụng
5. Biến thể của gradient descent
6. Momentum
7. Regularization trong mạng neuron

1 Ôn tập đạo hàm



1 Bài tập đạo hàm

Bài tập 1: Đạo hàm hàm số sau (x là biến, y là tham số):

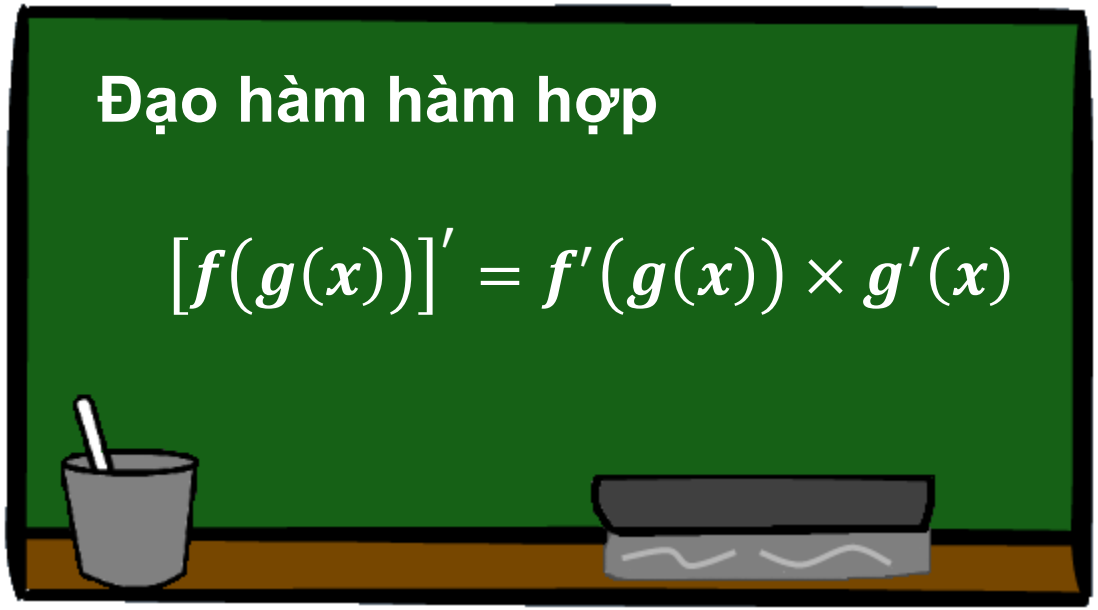
$$c(x) = \frac{1}{2}(y - x)^2$$

1 Bài tập đạo hàm

Bài tập 2: Đạo hàm hàm số sau

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

1 Ôn tập đạo hàm



1 Ôn tập đạo hàm

Ví dụ: Đạo hàm hàm số $f\big(g(x)\big)$:

Biết rằng: $f(x) = 2x^2 - 3$ và $g(x) = \frac{1}{x} + 1$

→ **Giải:**

Ta có công thức: $\big[f\big(g(x)\big)\big]' = f'\big(g(x)\big) \times g'(x)$

- $f'(x) = 4x$
- $f'\big(g(x)\big) = 4\left(\frac{1}{x} + 1\right)$
- $g'(x) = \frac{-1}{x^2}$

Suy ra $\big[f\big(g(x)\big)\big]' = f'\big(g(x)\big) \times g'(x) = 4\left(\frac{1}{x} + 1\right) \times \frac{-1}{x^2}$

1 Ôn tập đạo hàm

- **Bài tập 3**: Đạo hàm hàm số sau:

$$c(s(x)) = \frac{1}{2}\left(y - \left(\frac{1}{1+e^{-x}}\right)\right)^2$$

1 Ôn tập đạo hàm

Đạo hàm 3 hàm hợp?

$$\left[f\left(g(h(x))\right) \right]' = \ ?$$

$$\left[f\left(g(h(x))\right) \right]' = f'\left(g(h(x))\right) \times g'(h(x)) \times h'(x)$$

1 Ôn tập đạo hàm

- **Ví dụ**: Đạo hàm hàm số $f\left(g\left(h(x)\right)\right)$:

Biết rằng: $f(x) = 2x^2 - 3$ và $g(x) = \frac{1}{x} + 1$ và $h(x) = 4e^x - x$

1 Ôn tập đạo hàm

- **Ví dụ**: Đạo hàm hàm số ~~$f(g(x))$~~: $f\left(g\left(h(x)\right)\right)$:

Biết rằng: $f(x) = 2x^2 - 3$ và $g(x) = \frac{1}{x} + 1$ và $h(x) = 4e^x - x$

→ **Giải:**

$$\left[f\left(g\left(h(x)\right)\right)\right]' = f'\left(g\left(h(x)\right)\right) \times g'\left(h(x)\right) \times h'(x)$$

$$\left[\boldsymbol{f\left(g(x)\right)}\right]' = \boldsymbol{f'\left(g(x)\right) \times g'(x)} = \mathbf{4}\left(\frac{\mathbf{1}}{\boldsymbol{x}} + \mathbf{1}\right) \times \frac{\mathbf{-1}}{\boldsymbol{x^2}}$$

$h(x)$ $h(x)$ $\times h'(x)$

$$= \mathbf{4}\left(\frac{\mathbf{1}}{\boldsymbol{4e^x - x}} + \mathbf{1}\right) \times \frac{\mathbf{-1}}{\boldsymbol{(4e^x - x)^2}} \times \boldsymbol{(4e^x - 1)}$$

1 Ôn tập đạo hàm

Bài tập 4: Đạo hàm hàm số sau:

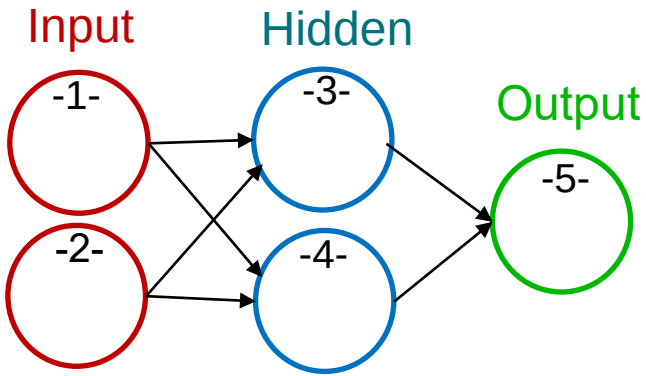
$$c\left(s\left(o(w_1)\right)\right) = \frac{1}{2}\left(y - \left(\frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + b)}}\right)\right)^2$$

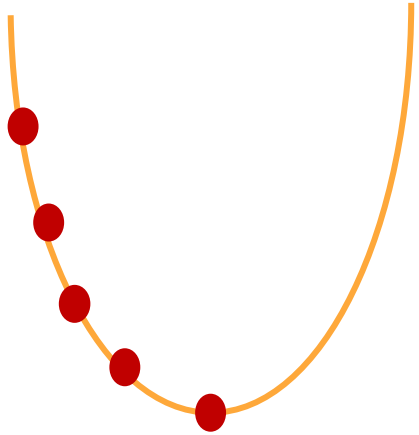
1 Ôn tập đạo hàm

Giải

1 Neural Network

- Xét mạng neuron sau





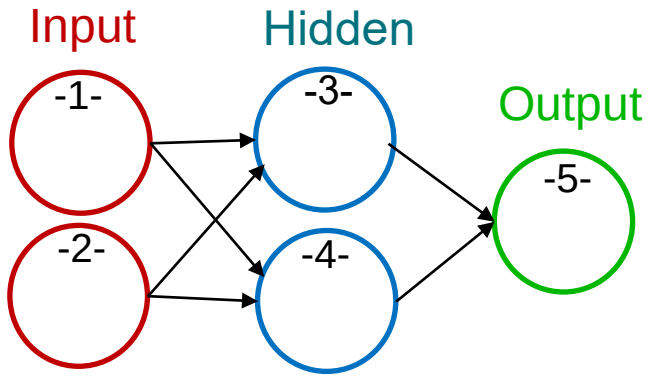
→ Đặt ra hàm cost (hàm độ lỗi):

$$cost = \frac{1}{2}(y - \hat{y})^2$$

Cost càng thấp, mạng neuron sẽ có params càng tốt

1 Neural Network

- Xét mạng neuron sau



$$cost = \frac{1}{2}(y - \hat{y})^2$$

Cách tính output $\hat{y}_5$:

$$\hat{y}_5 = sigmoid(w_{35}h_3 + w_{45}h_4 + b_5) = \frac{1}{1 + e^{-(w_{35}h_3 + w_{45}h_4 + b_5)}}$$

$$cost = \frac{1}{2}\left(y - \frac{1}{1 + e^{-(w_{35}h_3 + w_{45}h_4 + b_5)}}\right)^2$$

→ Đạo hàm để tìm cực tiểu

1 Neural Network

$$cost = \frac{1}{2}\left(y - \left(\frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + b)}}\right)\right)^2$$

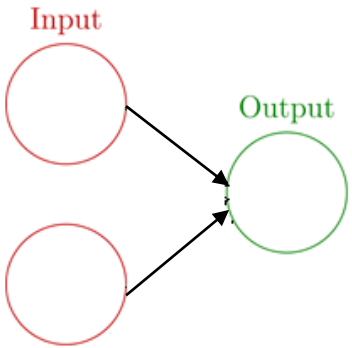
$$c\left(s\left(o(w_1)\right)\right) = \frac{1}{2}\left(y - \left(\frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + b)}}\right)\right)^2$$

$$cost = \frac{1}{2}\left(y - \frac{1}{1 + e^{-(w_{35} h_3 + w_{45} h_4 + b_5)}}\right)^2$$

1 Neural Network

Ta có kết quả đạo hàm cần tìm:

$$= -(y - \hat{y}) \times \hat{y} \times (1 - \hat{y}) \times \boldsymbol{x_1}$$



- **3.1:** Tính độ lỗi: $e_k = y_k - \hat{y}_k$
- **3.2:** Tính sai số gradient: $\delta_k = \hat{y}_k \times (1 - \hat{y}_k) \times e_k$
- **3.3:** Tính giá trị cần thay đổi: $\Delta w_{jk} = \boldsymbol{h_j} \times \delta_k$
- **3.4:** Cập nhật trọng số: $w_{jk} = w_{jk} + \alpha \times \Delta w_{jk}$
- **3.5:** Cập nhật bias: $b_k = b_k + \alpha \times \delta_k$

1 Dummy's questions

1. Nếu đổi hàm cost thì công thức có thay đổi không?
2. Nếu đổi activation function thì công thức có thay đổi không?
3. Lý do sử dụng hàm sigmoid làm activation function?
4. Backpropagation là gì, nó khác gì với gradient descent?
5. Tại sao trừ cho đạo hàm nhiều lần thì mô hình "thông minh hơn"?
6. Câu hỏi phỏng vấn: Giải thích gradient descent

1 Formal formulas

1. **Input x:** Set the corresponding activation a^1 for the input layer.

2. **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.

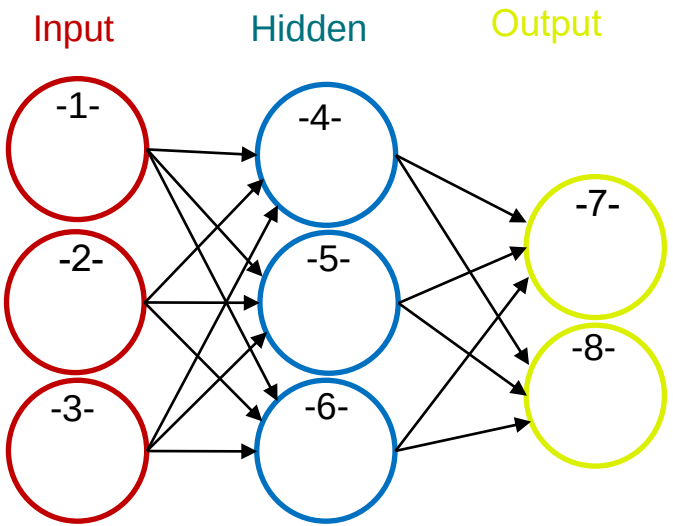
3. **Output error δ^L:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.

4. **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.

5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

1 Neural Network

- Sử dụng phép nhân ma trận để lan truyền, thay vì scalar
- Trọng số được ký hiệu $w^{(l)}$ để chỉ ma trận trọng số sau layer l
- Bias được ký hiệu $b^{(l)}$
- $\Delta w^{(l)}$ là giá trị gradient (dùng để cập nhật ma trận trọng số)
- Dùng phép "trừ"



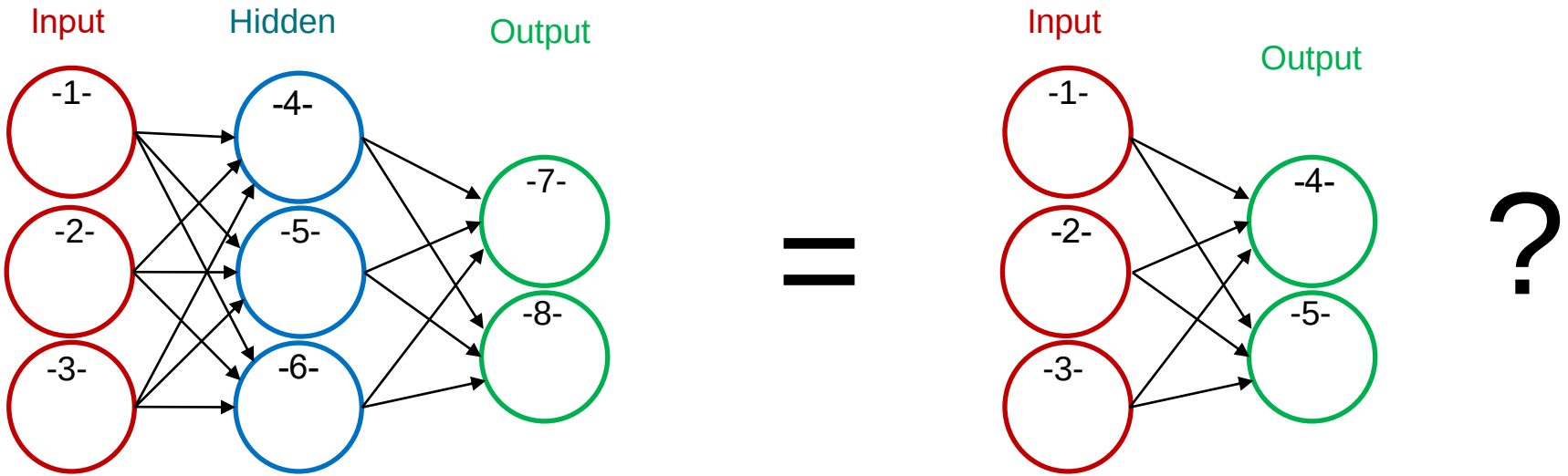
Nội dung

1. Chứng minh công thức lan truyền ngược
2. Chức năng của activation function
3. Một số activation function
4. Một số hàm cost thông dụng
5. Biến thể của gradient descent
6. Momentum
7. Regularization trong mạng neuron

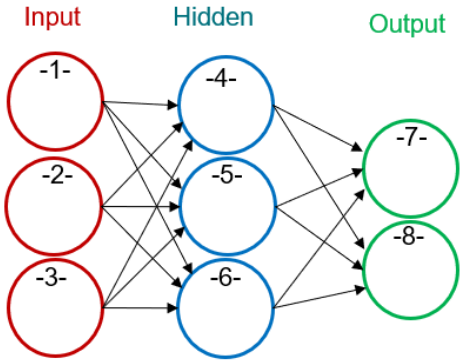
2 Remind: Bài tập về nhà

Bài 2 (optional): Hãy chứng minh:

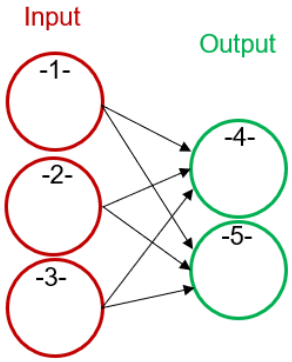
Với parameters tùy ý, nếu không có activation function, thì hai models sau hoàn toàn tương tự nhau



2 Remind: Bài tập về nhà



=



?

$$h = \mathrm{w}^{(1)}x + b^{(1)}$$

$$\mathrm{y} = \mathrm{w}^{(2)}h + b^{(2)}$$

$$\leftrightarrow \mathrm{y} = \mathrm{w}^{(2)}\left(\mathrm{w}^{(1)}x + b^{(1)}\right) + b^{(2)}$$

$$\leftrightarrow \mathrm{y} = \mathrm{w}^{(2)}\mathrm{w}^{(1)}x + \mathrm{w}^{(2)}b^{(1)} + b^{(2)}$$

w b

$$y = \mathrm{w}x + b$$

Chọn $\mathrm{w} = \mathrm{w}^{(2)}\mathrm{w}^{(1)}$

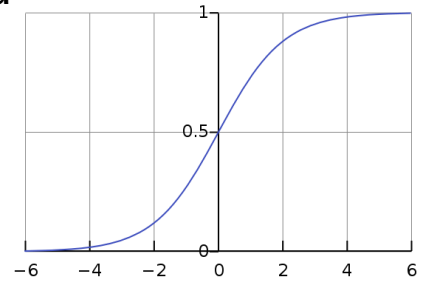
$$b = \mathrm{w}^{(2)}b^{(1)} + b^{(2)}$$

3 Activation function

- Giúp mạng trở nên phi tuyến → Giải quyết được nhiều bài toán
- Có nhiều activation functions:

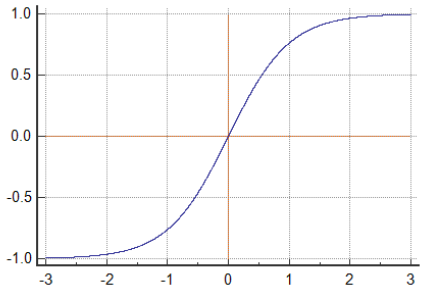
- Sigmoid

$$\frac{1}{1+e^{-x}}$$



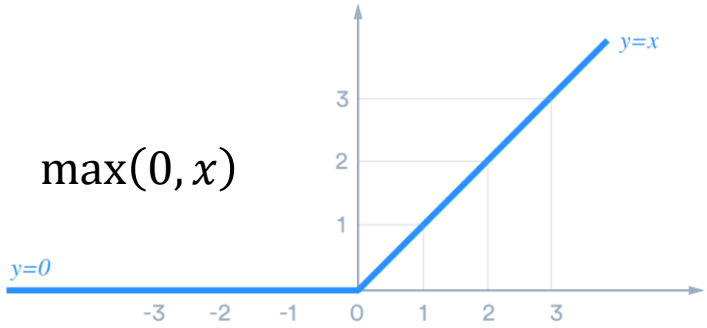
- Tanh

$$\frac{e^{2x}-1}{e^{2x}+1}$$



- ReLU (Rectified Linear Unit)

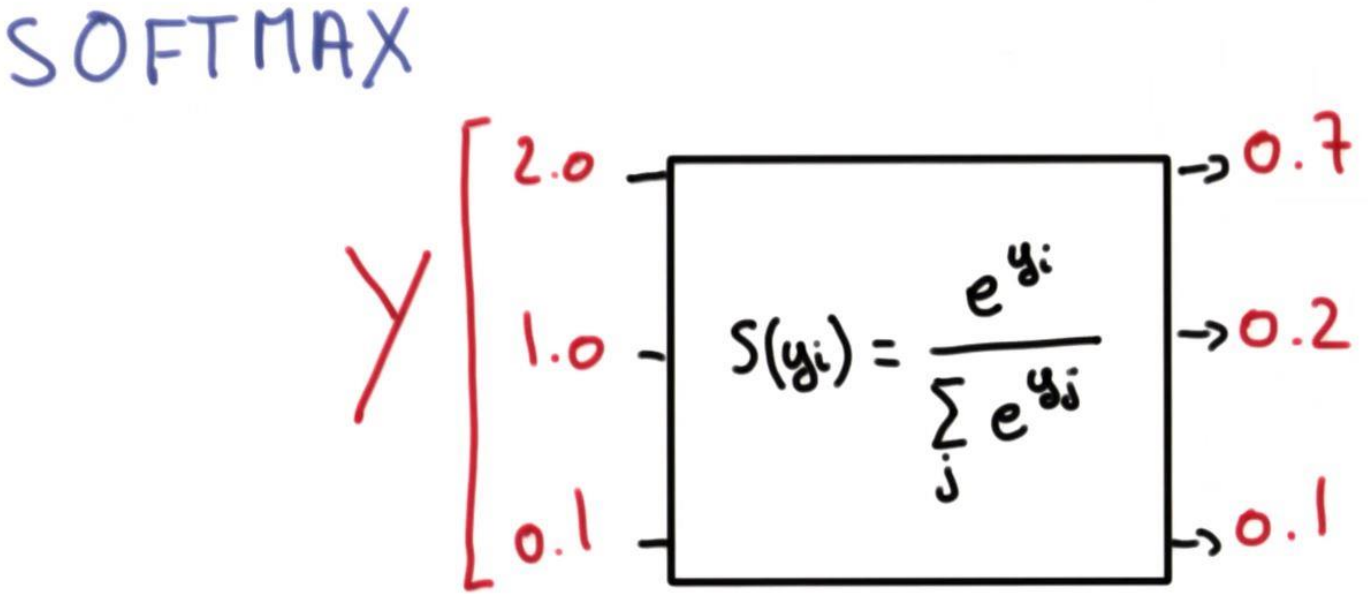
$$\max(0, x)$$



- Softmax

$$\text{softmax}(x_1) = \frac{e^{x_1}}{e^{x_1} + e^{x_2} + \cdots + e^{x_n}}$$

3 Activation function



3 Activation function

- **Bài tập 5**: Tính đạo hàm hàm Tanh, ReLU và softmax

1 Dummy's questions

1. Tại sao sigmoid làm cho mô hình khó hội tụ hơn các activation functions khác?
2. Activation function nào xịn xò nhất?
3. Tại sao không sử dụng hàm max thay vì hàm softmax?

Nội dung

1. Chứng minh công thức lan truyền ngược
2. Chức năng của activation function
3. Một số activation function
4. Một số hàm cost thông dụng
5. Biến thể của gradient descent
6. Momentum
7. Regularization trong mạng neuron

4 Hàm cost

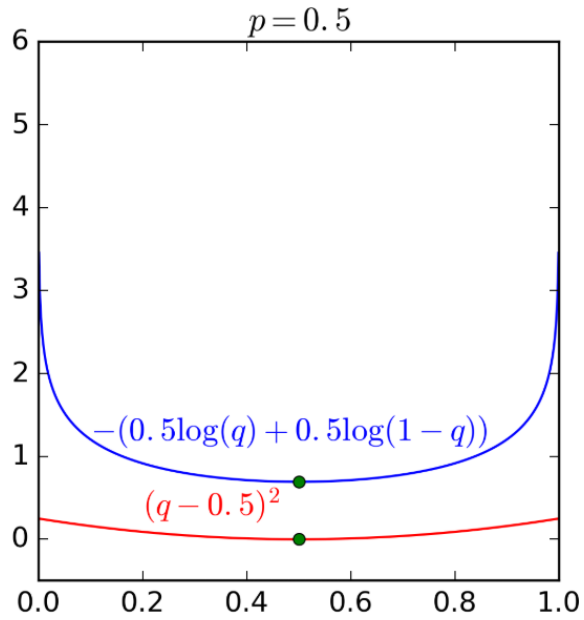
- Mean squared error

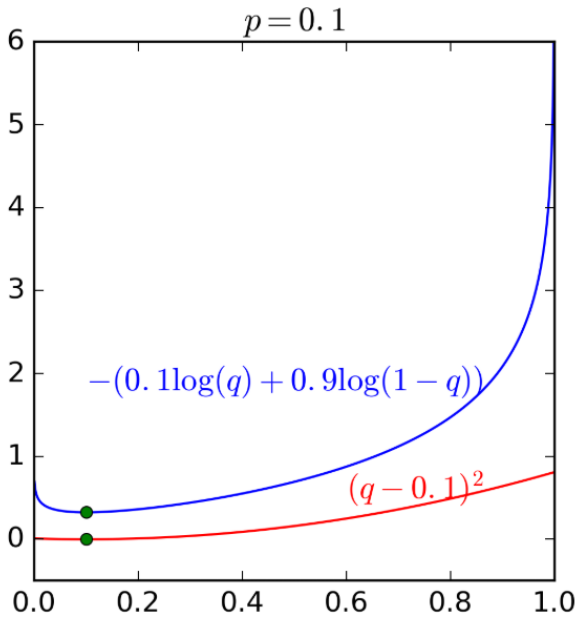
$$cost = \frac{1}{m}\sum_{i}^{m}\frac{1}{2} \times \left(y_i - \hat{y}_i\right)^2$$

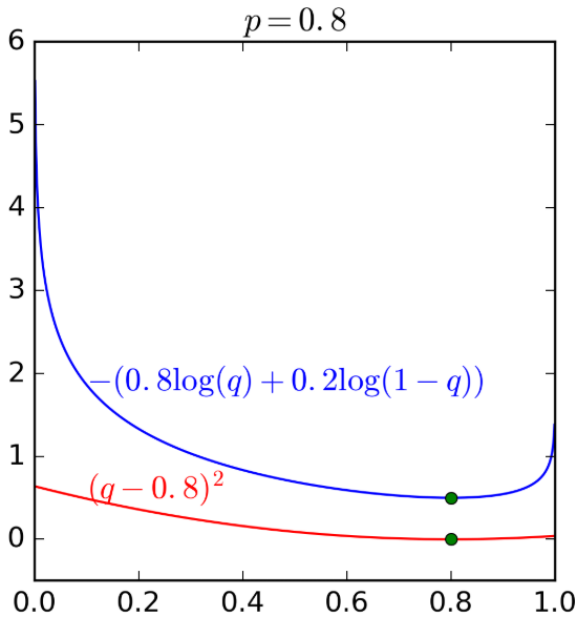
- Cross entropy

$$cost = -\frac{1}{m}\sum_{i}^{m}\left(\ln(\hat{y}_i) \times y_i + \ln(1-\hat{y}_i) \times (1-y_i)\right)$$

4 Ý nghĩa


$p = 0.5$
$-(0.5\log(q) + 0.5\log(1-q))$
$(q-0.5)^2$


$p = 0.1$
$-(0.1\log(q) + 0.9\log(1-q))$
$(q-0.1)^2$


$p = 0.8$
$-(0.8\log(q) + 0.2\log(1-q))$
$(q-0.8)^2$

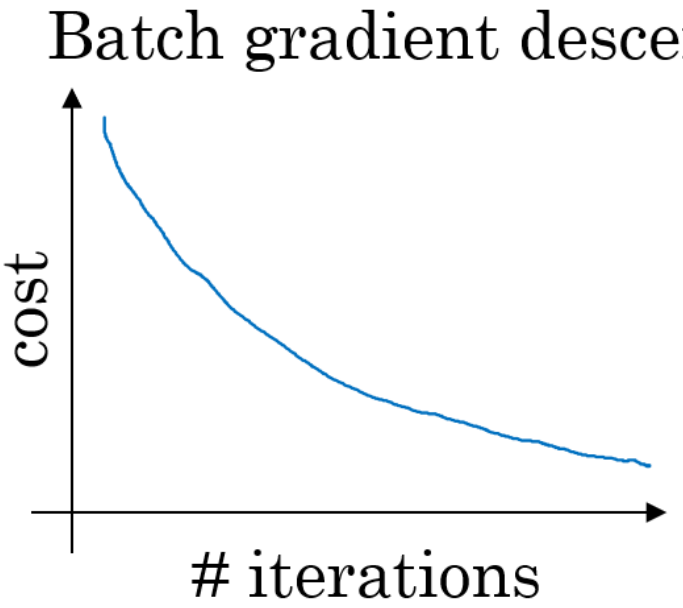
Nội dung

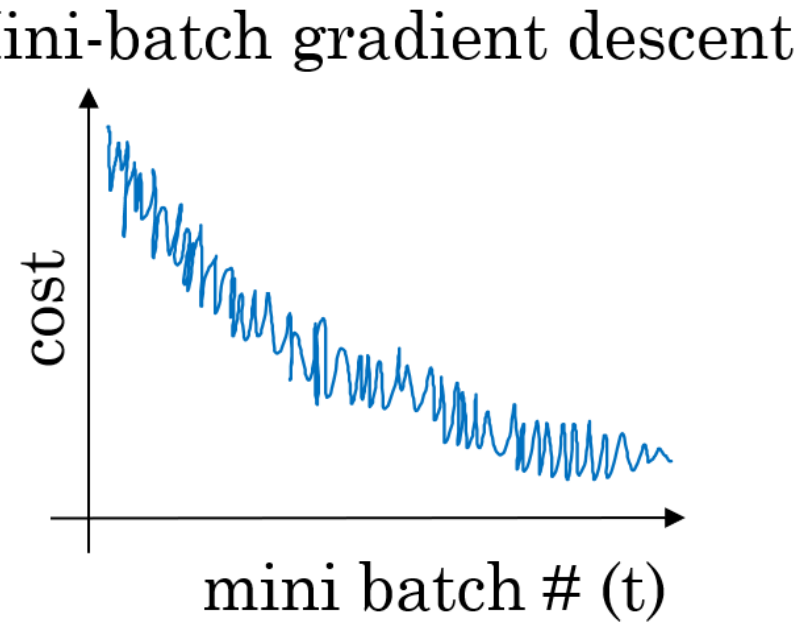
1. Chứng minh công thức lan truyền ngược
2. Chức năng của activation function
3. Một số activation function
4. Một số hàm cost thông dụng
5. Biến thể của gradient descent
6. Momentum
7. Regularization trong mạng neuron

5 Biến thể của gradient descent

- **Batch Gradient Descent**: khi cập nhật, chúng ta sử dụng **tất cả** các điểm dữ liệu
  - Đôi khi không hiệu quả khi có quá nhiều dữ liệu huấn luyện
  - Không hiệu quả với online learning
- **Stochastic Gradient Descent**: Sử dụng duy nhất một điểm dữ liệu mỗi lần huấn luyện
  - Cần shuffle tập dữ liệu mối khi huấn luyện
  - Hội tụ nhanh
- **Minibatch Gradient Descent**: sử dụng một số lượng nhỏ k mẫu dữ liệu (nhỏ hơn tổng số dữ liệu) mỗi lần huấn luyện
  - Chọn k sao cho $10 < k < 1000$. Ví dụ $k = 32, 64, 128, \ldots$
  - Cần shuffle tập dữ liệu mối khi huấn luyện

5 Biến thể của gradient descent





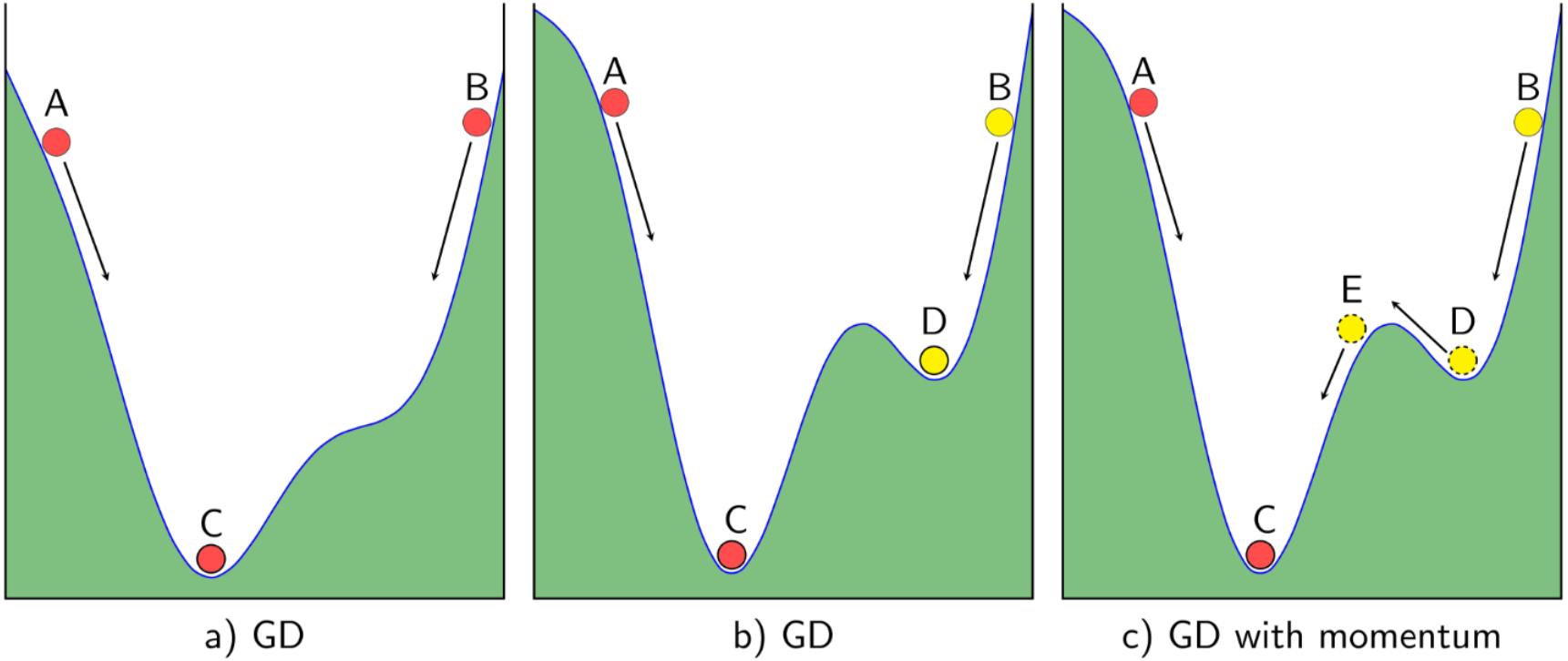
- Nếu dữ liệu nhỏ (< 200 mẫu): dùng batch gradient descent
- Nếu dữ liệu lớn, dùng mini-batch gradient descent
- Đảm bảo dữ liệu phù hợp với machine stats

Nội dung

1. Chứng minh công thức lan truyền ngược
2. Chức năng của activation function
3. Một số activation function
4. Một số hàm cost thông dụng
5. Biến thể của gradient descent
6. Momentum
7. Regularization trong mạng neuron

6 Momentum

Sử dụng momentum để tránh local minima



6 Momentum

- Biểu diễn momentum dưới dạng toán

  $\Delta w_{jk} = -\hat{y}_k \times (1 - \hat{y}_k) \times (y_k - \hat{y}_k) \times \mathrm{h}_j$

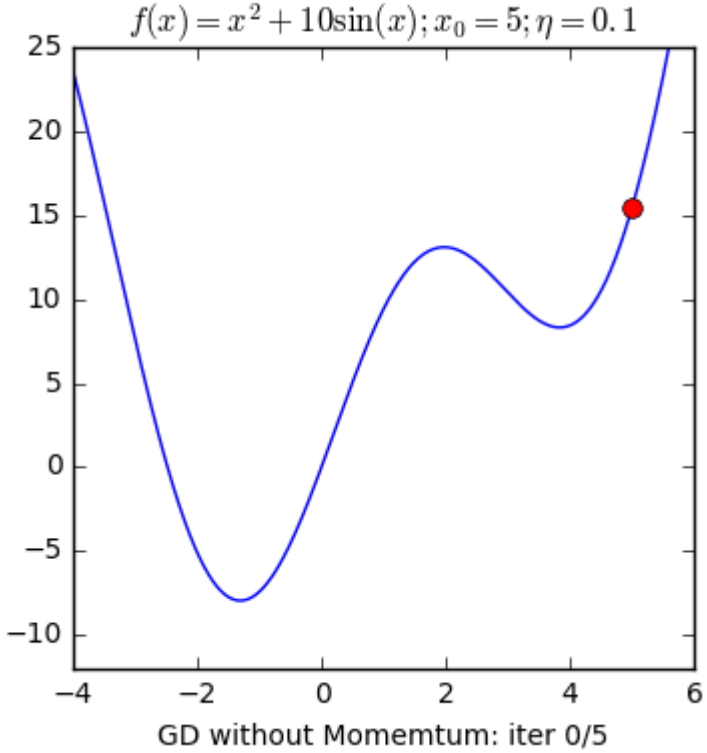
  Cập nhật trọng số: $w_{jk} = w_{jk} - \alpha \times \Delta w_{jk}$

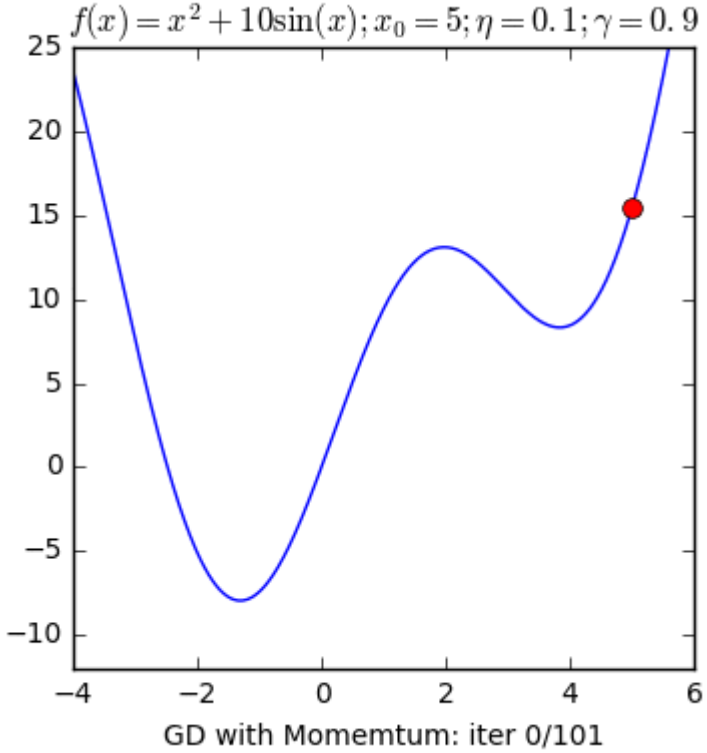
  Thêm biến vận tốc: $\mathrm{v}^{(t)} = \gamma \times \mathrm{v}^{(t-1)} + \alpha \times \Delta w_{jk}$

  Khi đó: $w_{jk} = w_{jk} - \mathrm{v}^{(t)}$

- γ thường được chọn với giá trị 0.9
- Chọn vận tốc đầu $\mathrm{v}^{(0)} = 0$

6 Momentum


$f(x) = x^2 + 10\sin(x); x_0 = 5; \eta = 0.1$
25
20
15
10
5
0
−5
−10
−4
−2
0
2
4
6
GD without Momemtum: iter 0/5

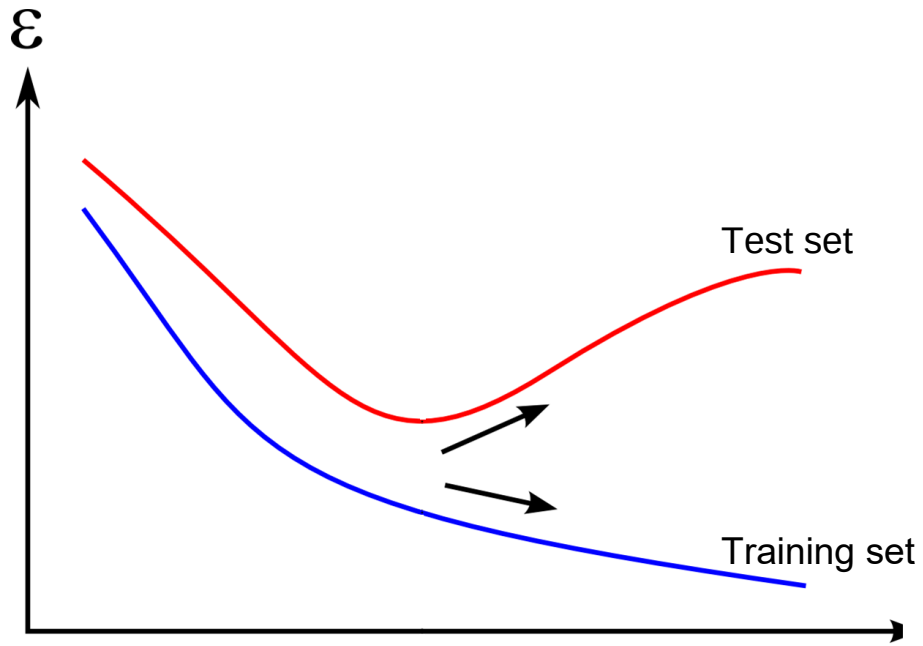

$f(x) = x^2 + 10\sin(x); x_0 = 5; \eta = 0.1; \gamma = 0.9$
25
20
15
10
5
0
−5
−10
−4
−2
0
2
4
6
GD with Momemtum: iter 0/101

Nội dung

1. Chứng minh công thức lan truyền ngược
2. Chức năng của activation function
3. Một số activation function
4. Một số hàm cost thông dụng
5. Biến thể của gradient descent
6. Momentum
7. Regularization trong mạng neuron

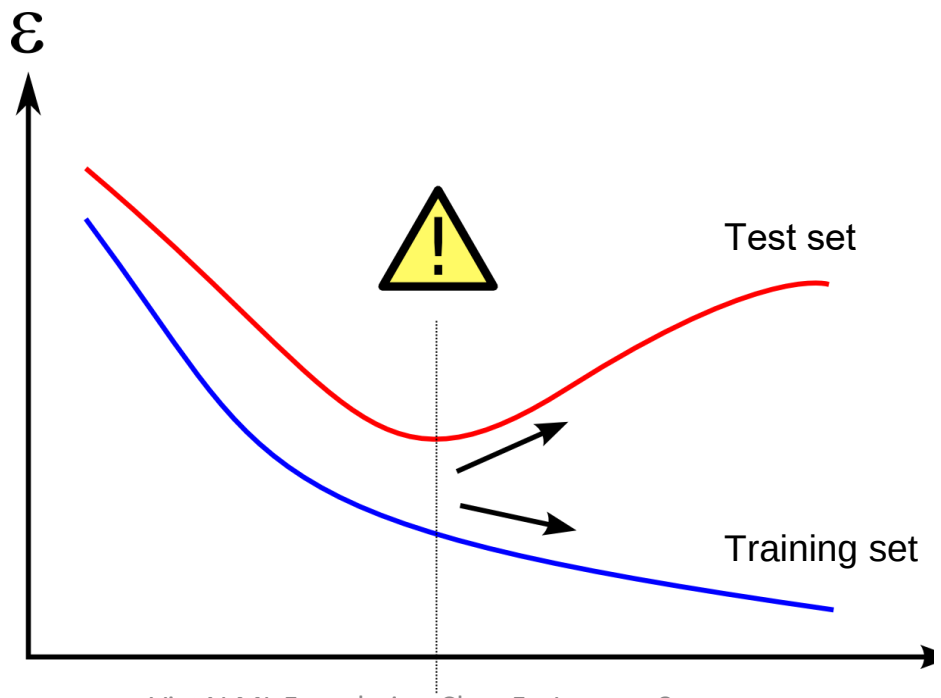
6 Dấu hiệu của Overfitting

- Overfitting



6 Early Stopping

- Dừng huấn luyện trước khi xảy ra dấu hiệu overfitting



6 Regularization: L2 Norm

- L2 Norm:

$$\text{cost} = \text{cost} + \underbrace{\frac{\lambda}{2m} \times \sum_{l=1}^{L-1} \left(\|w^{(l)}\|_F \right)^2}_{\textbf{Regularization term}}$$

λ: regularization parameter

m: tổng số mẫu dữ liệu

L: tổng số lớp

$$\Delta w^{(l)} = kq \text{ từ backprop} + \frac{\lambda}{m} w^{(l)}$$

$$w^{(l)} = w^{(l)} - \alpha \times \Delta w^{(l)}$$

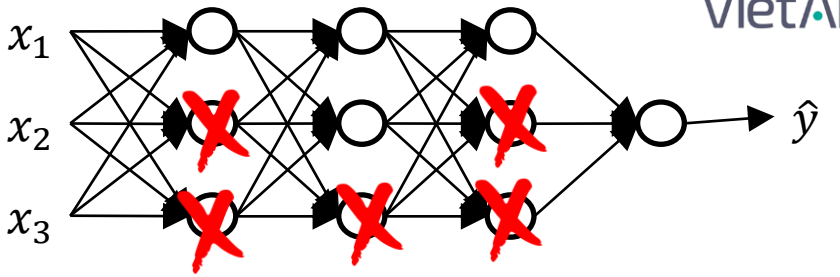
Công thức L2 Norm:
Frobenius norm

$$\|\mathbf{A}\|_{\mathrm{F}} = \sqrt{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \mathbf{A}_{ij}^2}$$

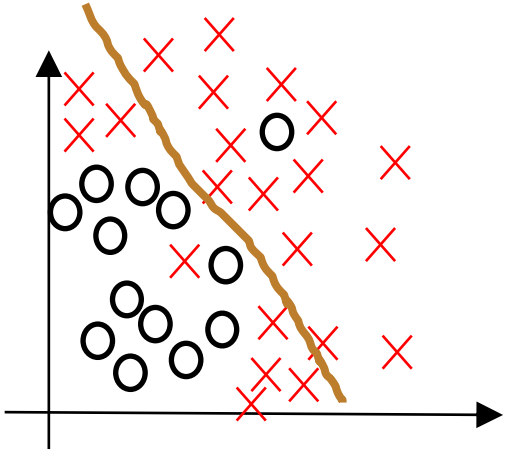
6 Regularization: L2 Norm

$$\Delta w^{(l)} = kq \text{ từ backprop} + \frac{\lambda}{m} w^{(l)}$$

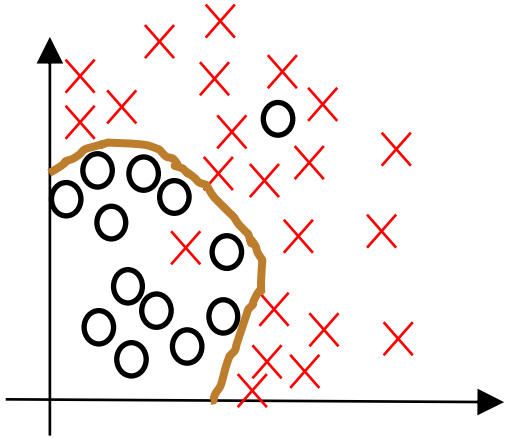
$$w^{(l)} = \Delta w^{(l)} - \alpha \times \Delta w^{(l)}$$



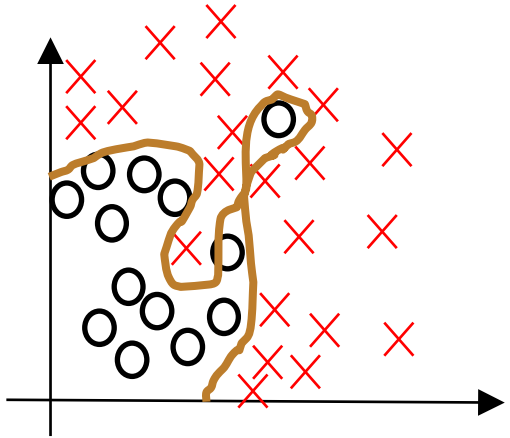
Làm cho $w^{(l)}$ nhỏ hơn, nếu λ càng lớn → $w^{(l)}$ tiến về 0



underfitting



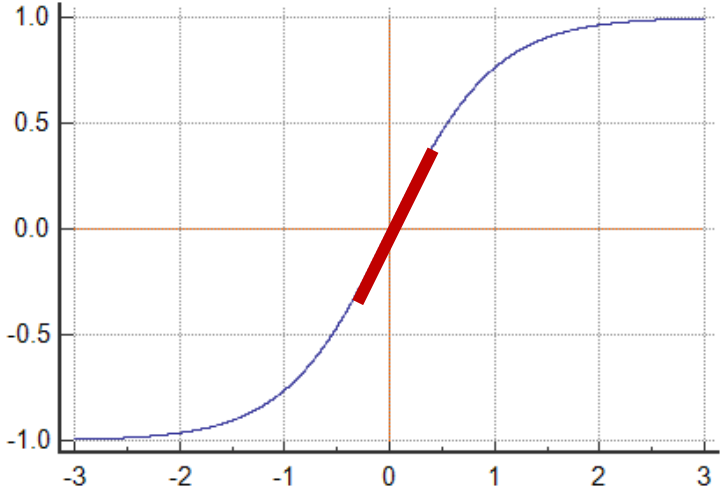
"OK"



overfitting

6 Regularization: L2 Norm

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

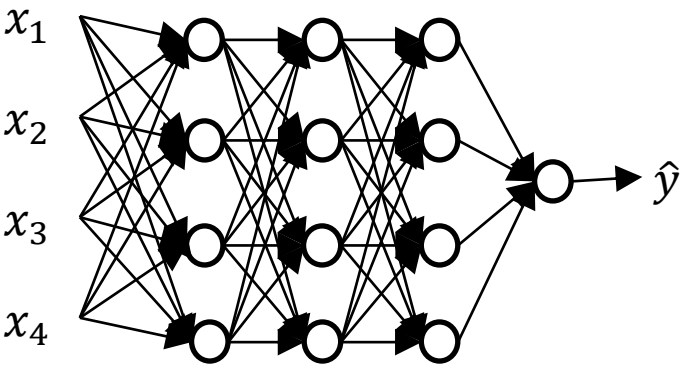


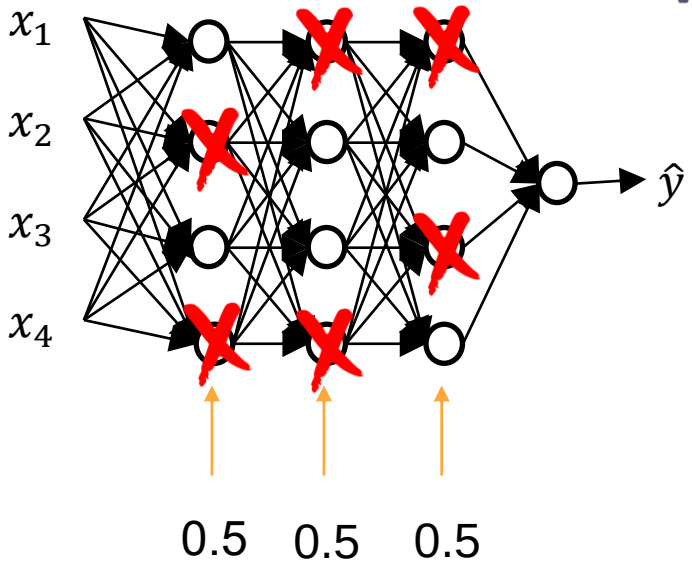
nếu λ càng lớn → $w^{(l)} \approx 0$ → $z^{(l)} = w^{(l)}x^{(l-1)} + b^{(l)} \approx 0$

Giá trị $\tanh(z)$ thay đổi nằm trong khoảng màu đỏ
→ gần như đường thẳng

Mỗi layer ≈ tuyến tính (linear) hơn

6 Regularization: Dropout





- Định nghĩa xác suất giữ lại (keep probability) để giữ lại neurons
- Dropout được thực hiện trên mỗi batch → các neurons bị tắt sẽ khác nhau mỗi vòng lặp
- Khi predict: tắt chế độ dropout (keep_prob = 1 cho tất cả)

6 Regularization: Inverted Dropout

- Quá trình cài đặt dropout:

```
import numpy as np
# Giả sử a là giá trị có sẵn trên các neuron lớp đang xét
keep_prob = 0.8
d = np.random.rand(a.shape[0],a.shape[1]) < keep_prob
a = a*d
# Inverted dropout: giữ nguyên giá trị trung bình của a
a = a/keep_prob
```

- Giả sử có 50 neurons → 10 neurons bị tắt (giá trị = 0)

$$tb(a) = tb(a) - \frac{20}{100} \times tb(a) = tb(a) - 0.2 \times tb(a) = 0.8 \times tb(a) = tbm(a)$$

- Bù lại khi thực hiện inverted dropout:

$$tbm(a) = tbm(a) : \frac{80}{100} = tbm(a) \times \frac{100}{80} = tbm(a) \times \frac{80}{80} + tbm(a) \times \frac{20}{80} = tb(a)$$

6 Regularization: Inverted Dropout

Lý do dropout hoạt động:

- Không thể tin tưởng hoàn toàn trên duy nhất một feature, cần phải lan rộng giá trị weights