

Bài 7: Neural Network

Tuần 4A

Nhắc lại về bài toán dự đoán

Cho x  dự đoán y

- Bài toán Phân loại (Classification)
- Bài toán Hồi quy (Regression)

Nhắc lại về bài toán dự đoán

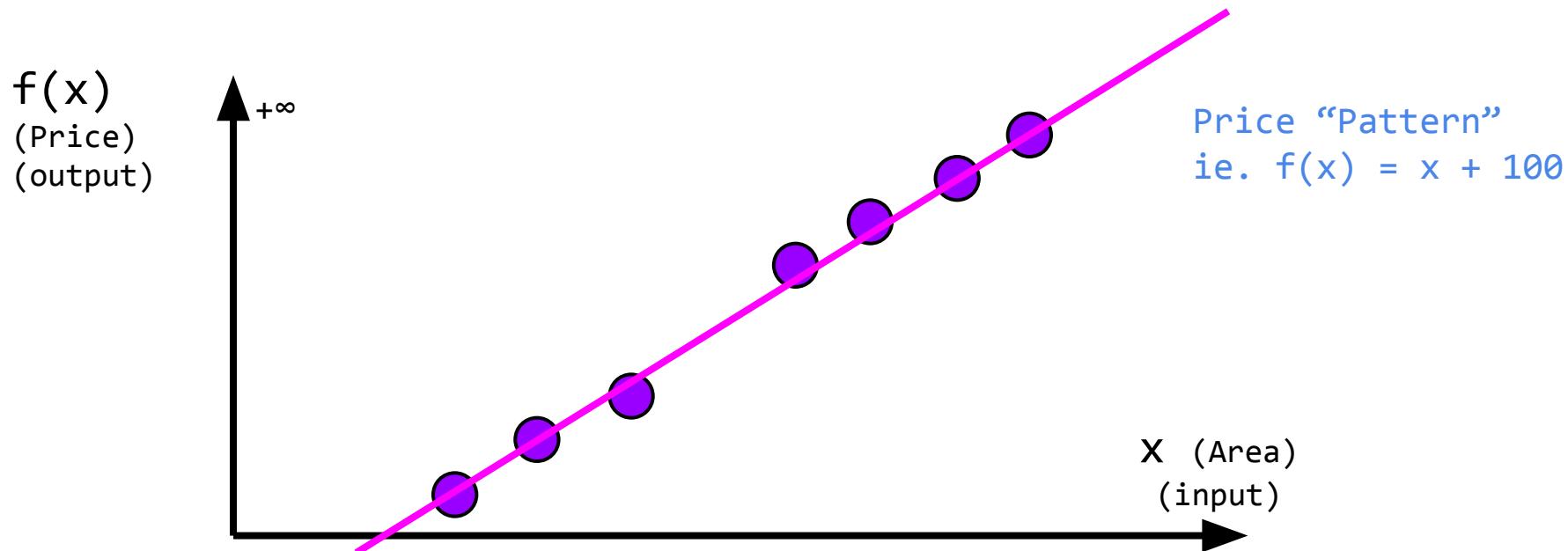
Cho diện tích của căn nhà

Dự đoán giá

Area - x	Price - y
2104	400
1600	330
2400	369
1416	232

Đây là bài toán regression

Nhắc lại về bài toán dự đoán



Đây là bài toán regression

Nhắc lại về bài toán dự đoán

Cho số lượng từ “giảm giá”
trong một email

Dự đoán
spam mail hay không

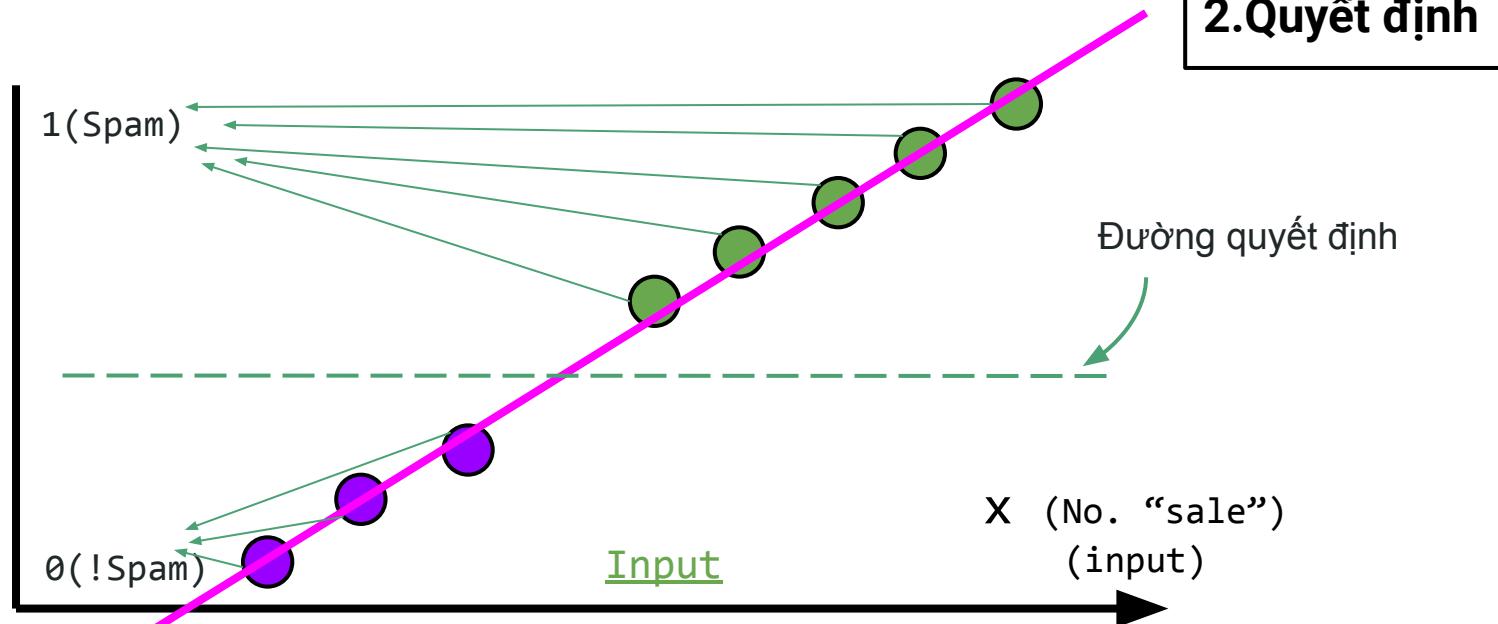
No. “sale” - x	Spam or not - y
10	1
12	1
2	0
6	0

Đây là bài toán classification

Nhắc lại về bài toán dự đoán

$f(x)$
 (Squeezed
 spam value)
Codomain:
 $(0,1)$

Output!



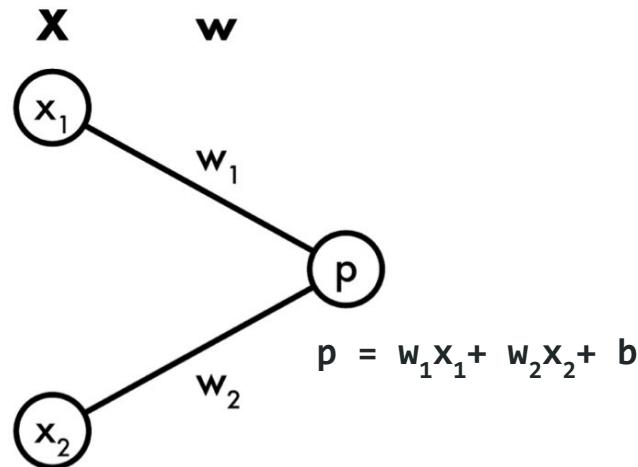
- Dựa trên số lượng từ “giảm giá” trong email, ta có thể dự đoán email đó là spam hay không.
- Bài toán này gồm 2 bước: Suy diễn và Quyết định

Neural Network

- Dạng cơ bản nhất của NN là một **Linear model**, output là **tổ hợp tuyến tính** của các input.
- Mạng **Deep NN** là dạng mở rộng của NN, gồm **Linear model (I)** với **hàm phi tuyến (II)** và **cấu trúc phân cấp (III)** (*hierarchical structure*)

Neural Network

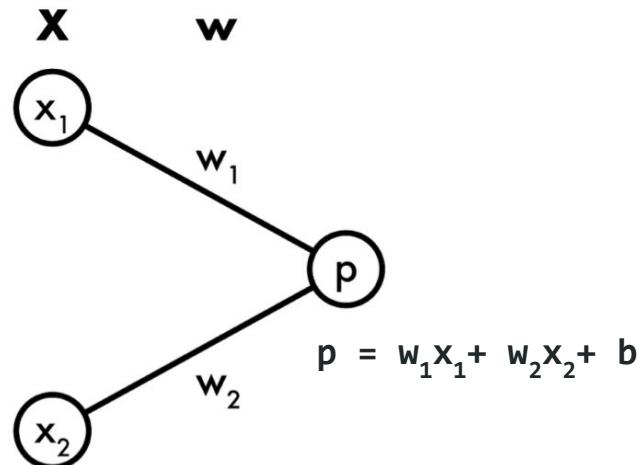
Một đơn vị



- Dạng cơ bản nhất của NN là một Linear model, output là **tổ hợp tuyến tính** của các input.

Neural Network

Một đơn vị

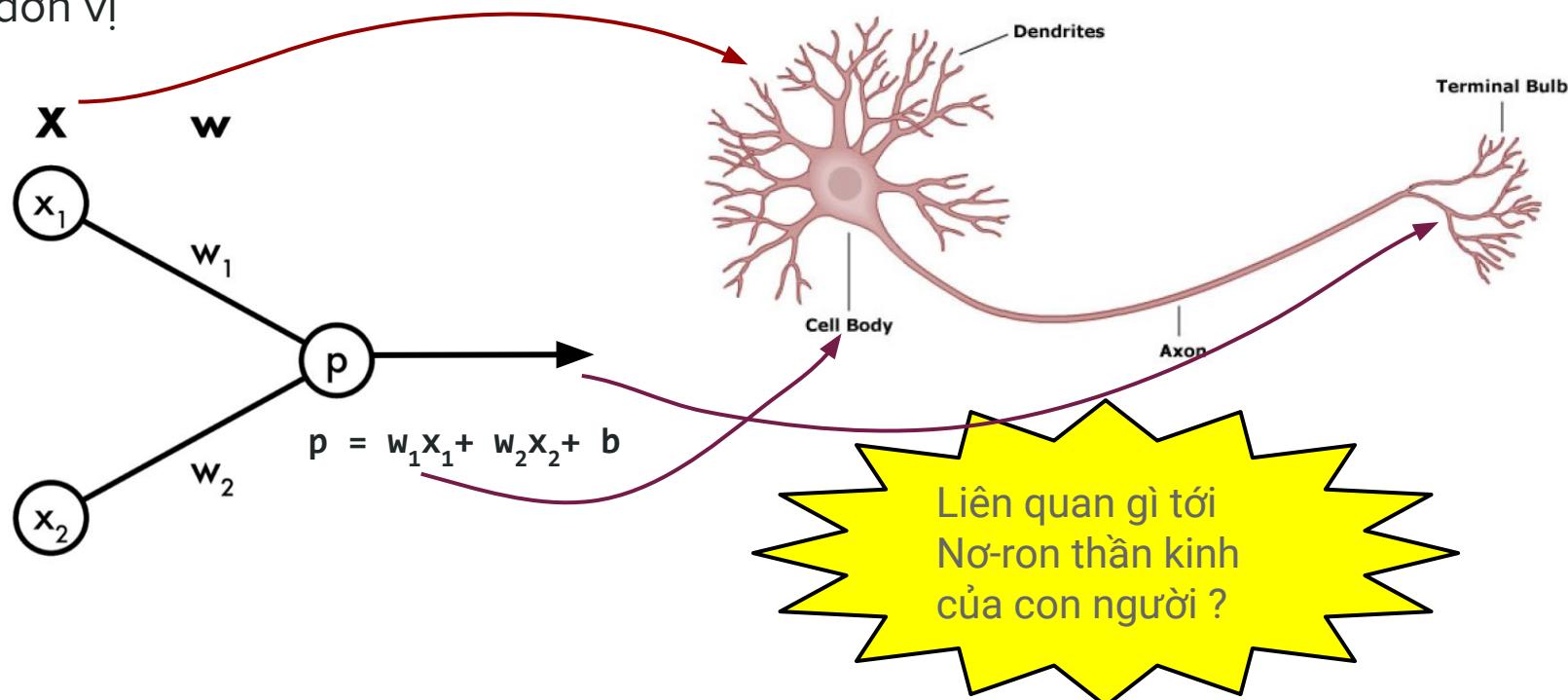


- Dạng cơ bản nhất của NN là một Linear model, output là **tổ hợp tuyến tính** của các input.

Liên quan gì tới
Nơ-ron thần kinh
của con người ?

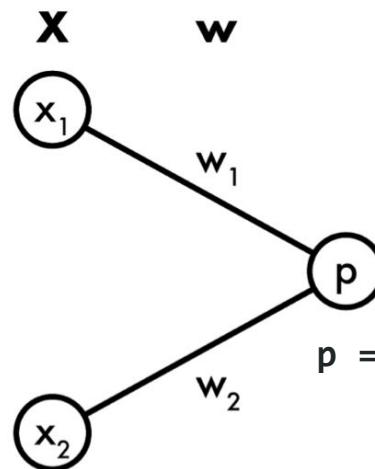
Neural Network

Một đơn vị

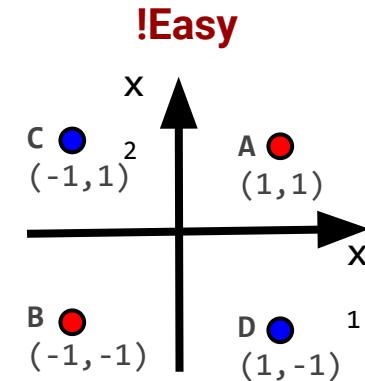
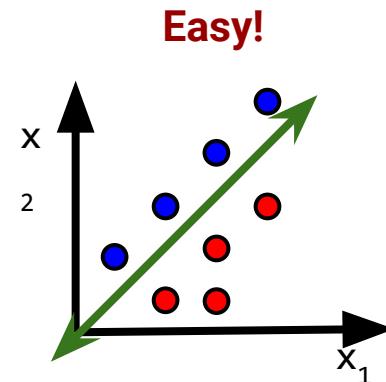


Neural Network

Một đơn vị



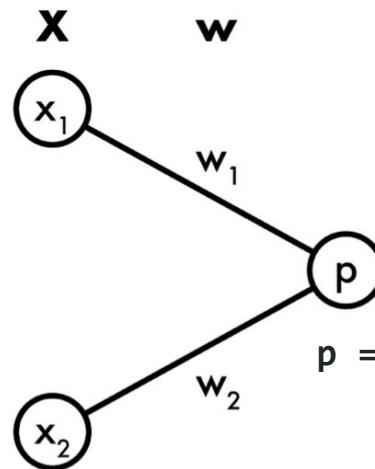
Bài toán Phân loại



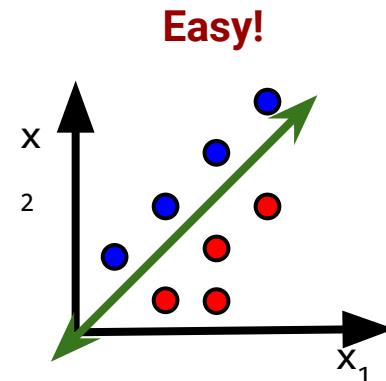
- Linear model (I) cũng như hầu hết các ML models hoạt động tốt khi mối quan hệ giữa input features và output rõ ràng.

Neural Network

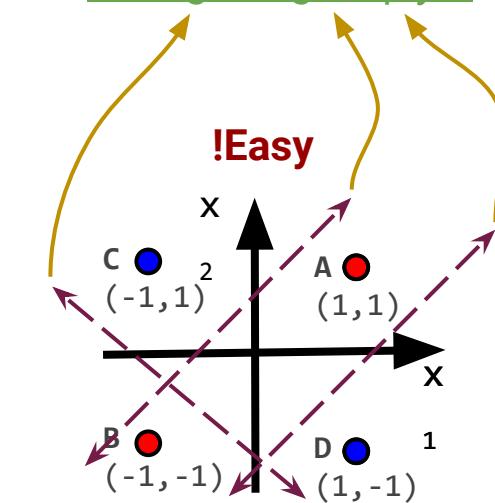
Một đơn vị



Bài toán Phân loại



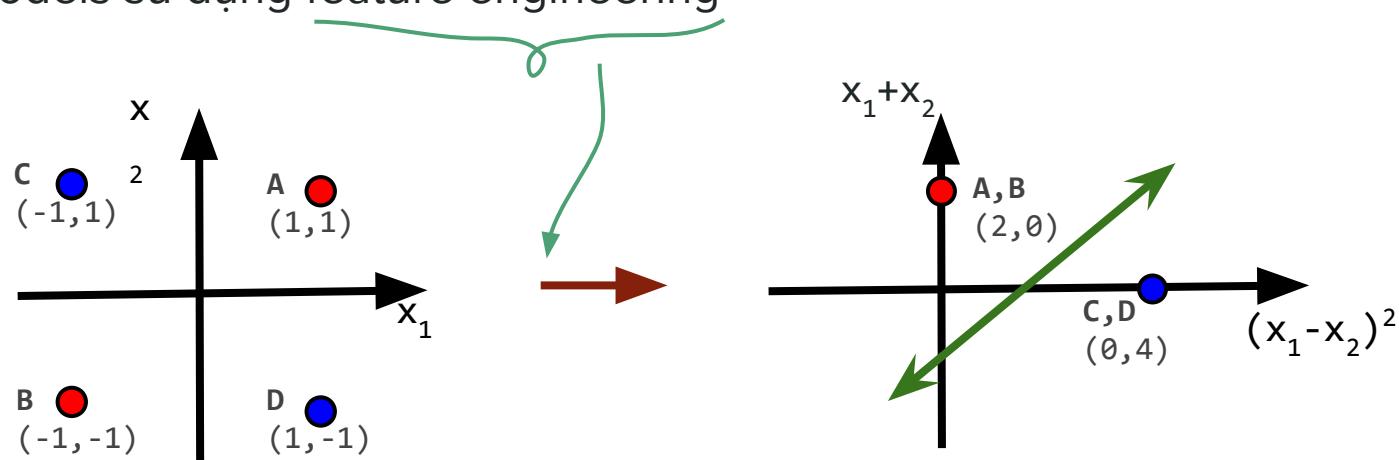
Không thể giải quyết



- Linear model (I) cũng như hầu hết các ML models hoạt động tốt khi mối quan hệ giữa input features và output rõ ràng.

Neural Network

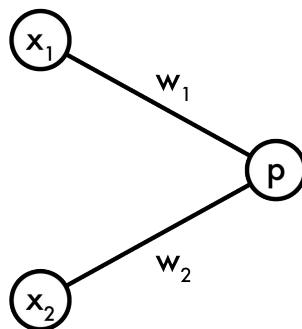
XOR: ML models sử dụng feature engineering



- Các kỹ sư ML thông thường phải tìm bằng cách thử-sai và lựa chọn các features phù hợp cho bài toán ML cụ thể.

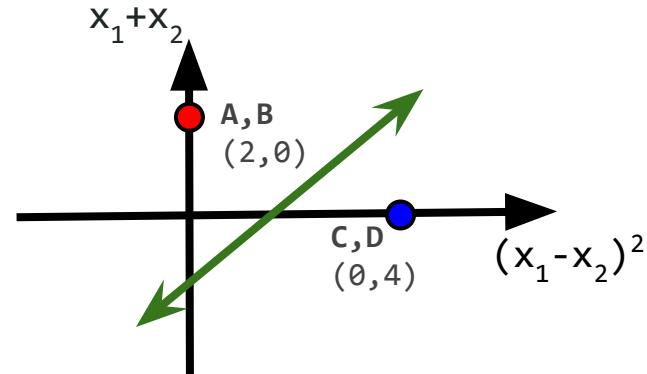
Neural Network

XOR: “automate” feature engineering



$$p = f(w_1x_1 + w_2x_2)$$

features mới

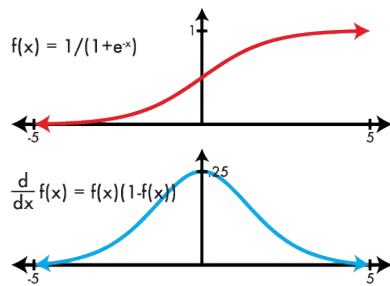


- Cơ bản, bước feature engineering sử dụng tổ hợp tuyến tính của input features, sau đó biến đổi tổ hợp này 1 hàm phi tuyến $f(w_1x_1 + w_2x_2)$ để tạo ra được các features mong muốn.
- Tại sao ta không học luôn quá trình extract features này? -> NN với hàm phi tuyến

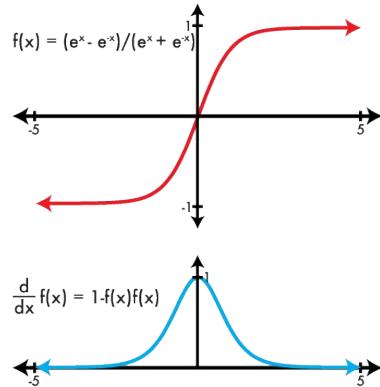
Neural Network

Hàm phi tuyến (Non-linear/Activation function)

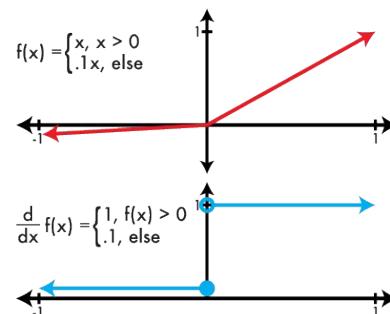
logistic



tanh



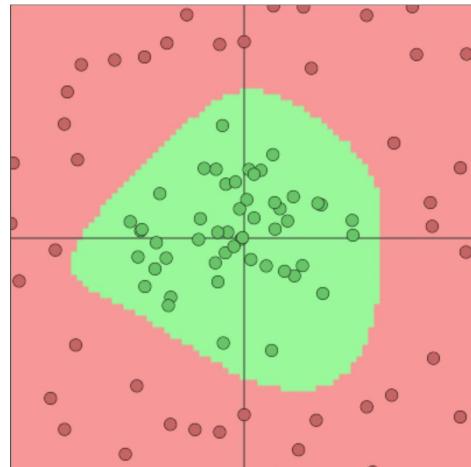
REctified Linear Unit (ReLU)



- Hàm phi tuyến giúp biến đổi không gian input features, tạo ra những features phù hợp giúp nhiệm vụ **Hồi quy/Phân loại** ở bước sau ở nên “straightforward”

Neural Network

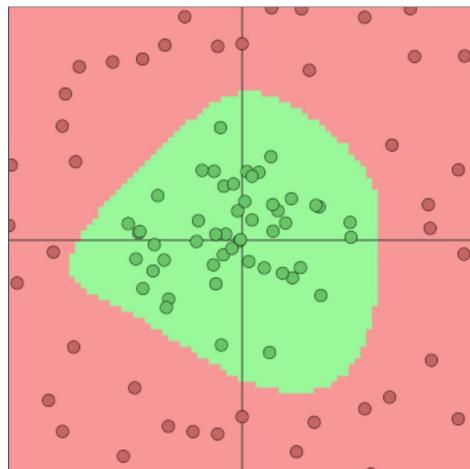
Hàm phi tuyế̉n (Non-linear/Activation function) - Ví dụ



!Easy

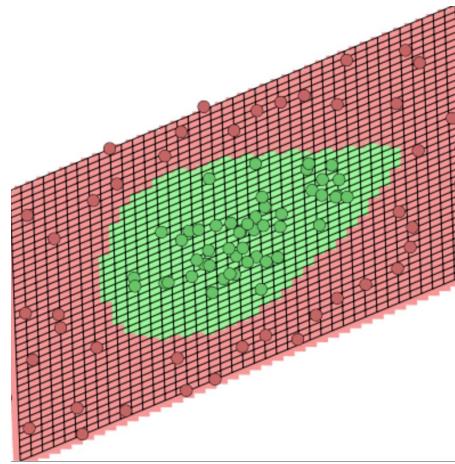
Neural Network

Hàm phi tuyến (Non-linear/Activation function) - Ví dụ



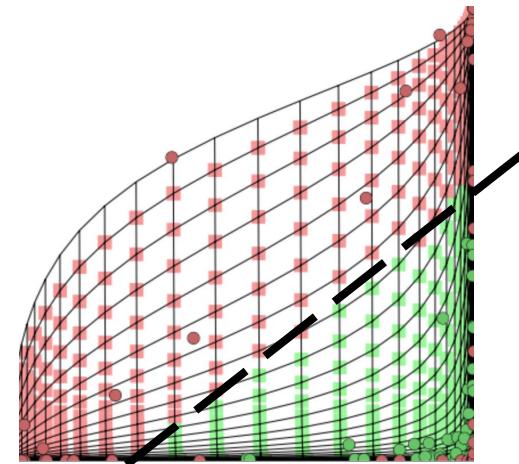
!Easy

w^*
-->



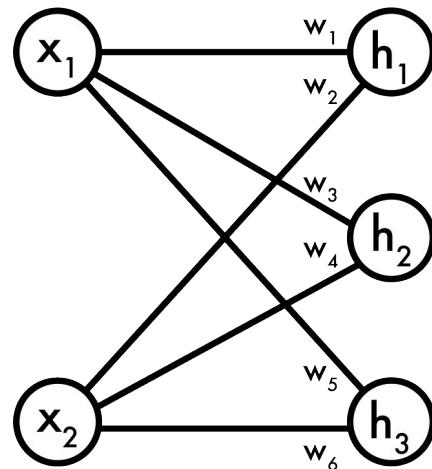
Easy!

$f()$
-->



Neural Network

Một lớp ẩn



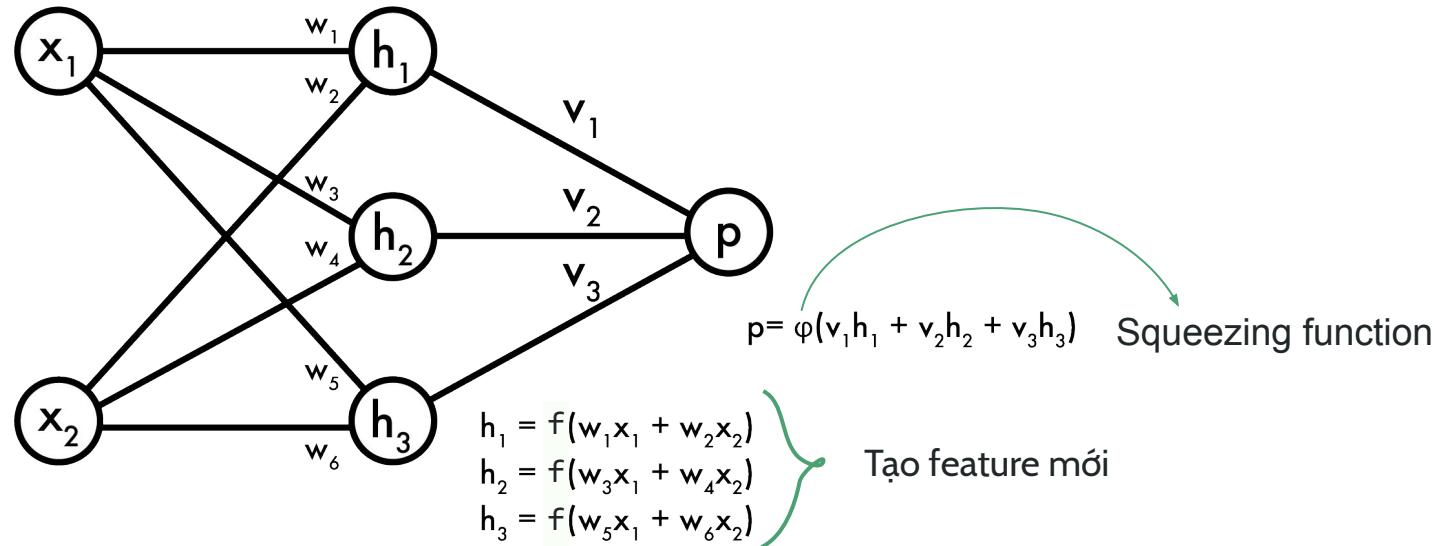
$$\begin{aligned}
 h_1 &= f(w_1x_1 + w_2x_2) \\
 h_2 &= f(w_3x_1 + w_4x_2) \\
 h_3 &= f(w_5x_1 + w_6x_2)
 \end{aligned}$$

Tạo feature mới

- Tạo ra features mới từ raw input, lớp này gọi là hidden layer H

Neural Network

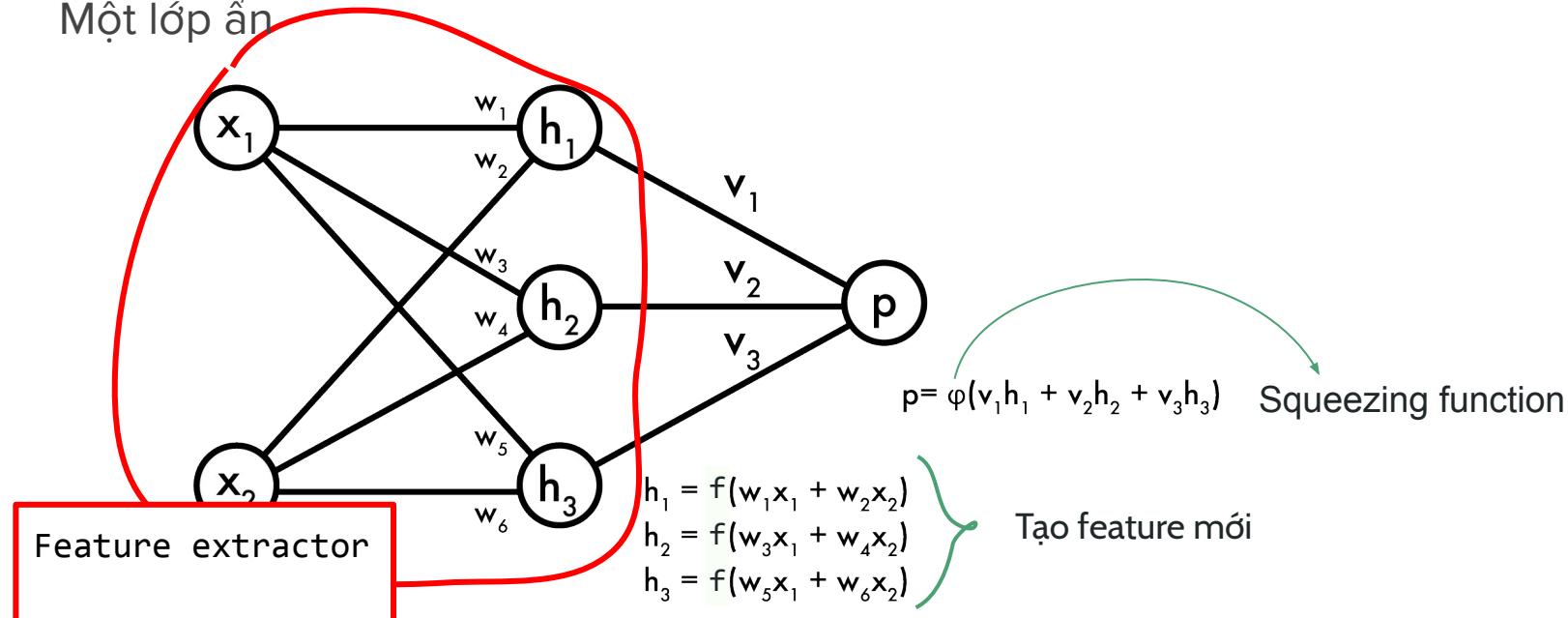
Một lớp ẩn



- Tạo ra features mới từ raw input, lớp này gọi là hidden layer *H*
- Các features này được sử dụng cho nhiệm vụ Hồi quy/Phân loại như ở các slide trước

Neural Network

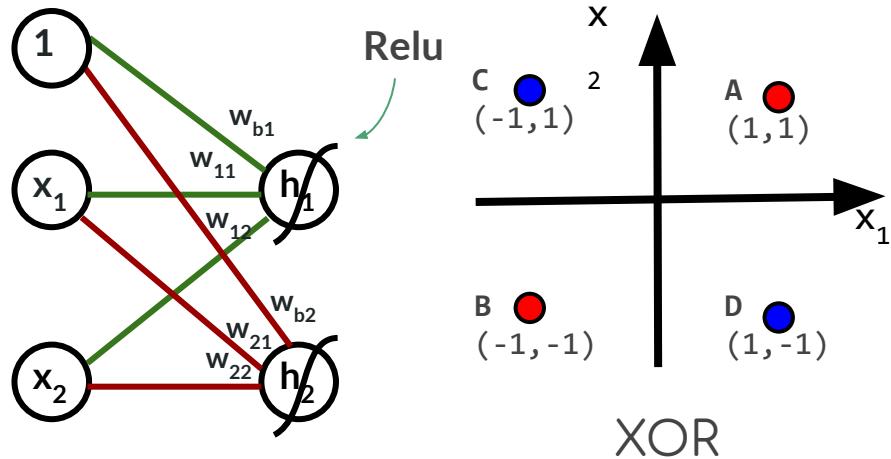
Một lớp ẩn



Công thức tạo Hidden layer: Linear model (I) + hàm phi tuyến (II)

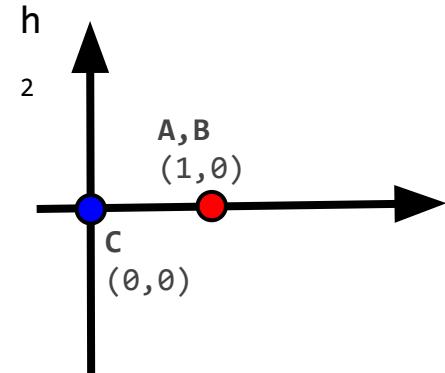
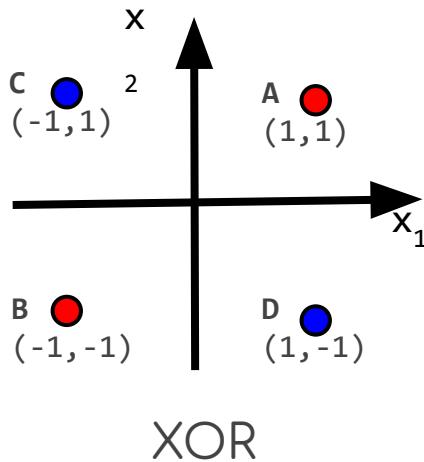
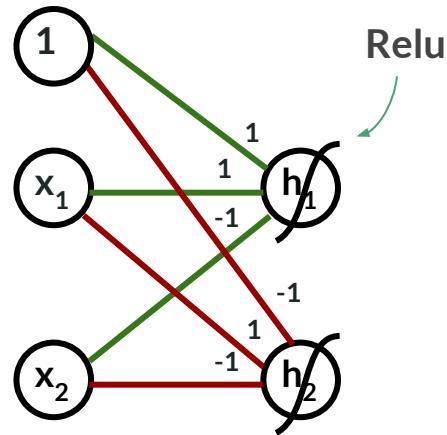
Neural Network

Một lớp ẩn giải quyết XOR problem



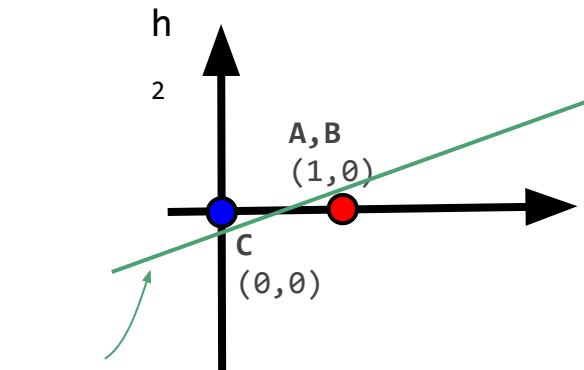
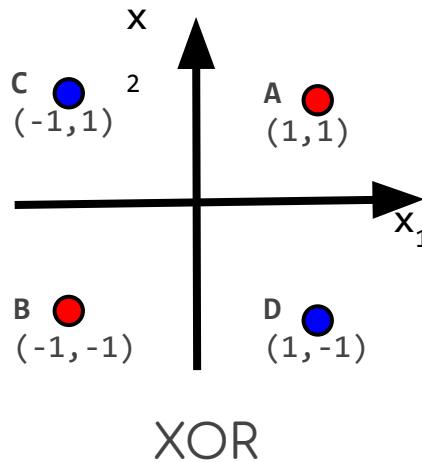
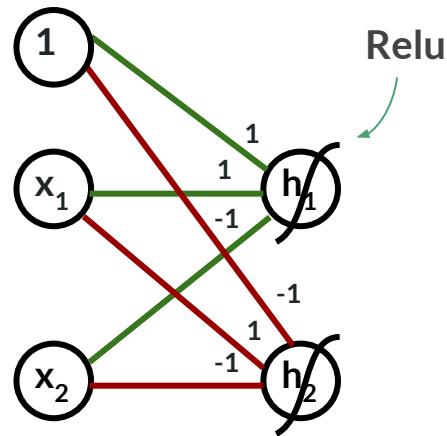
Neural Network

Một lớp ẩn giải quyết XOR problem



Neural Network

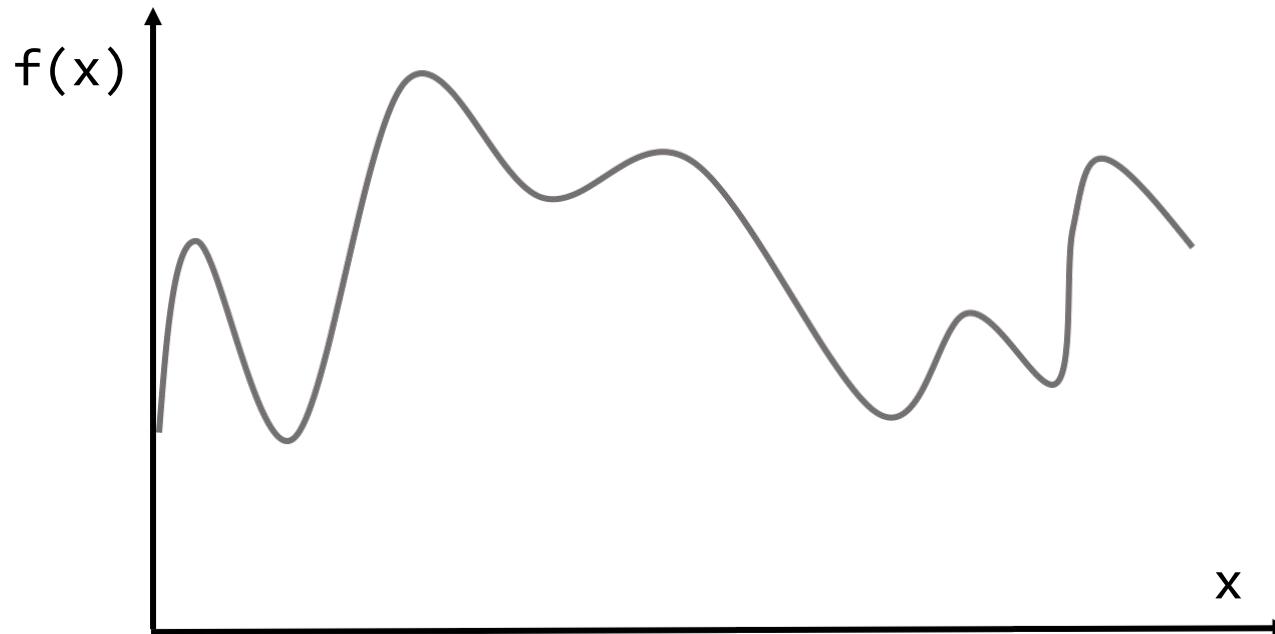
Một lớp ẩn giải quyết XOR problem



Transformed
XOR

Neural Network

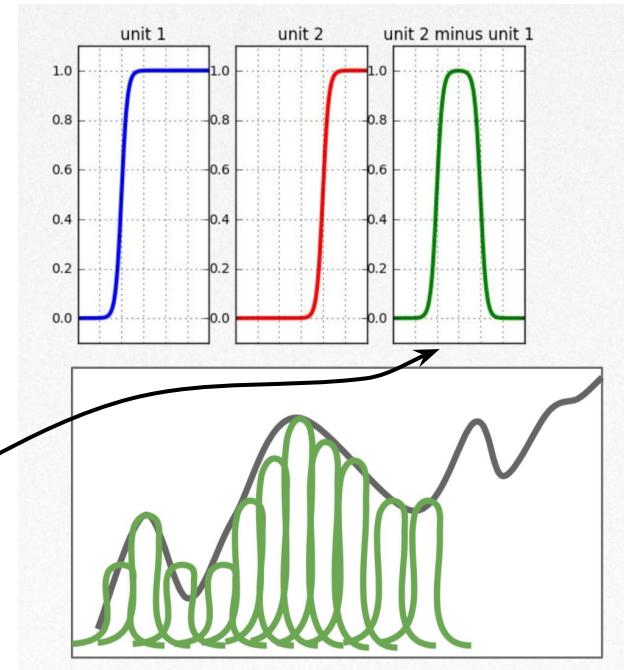
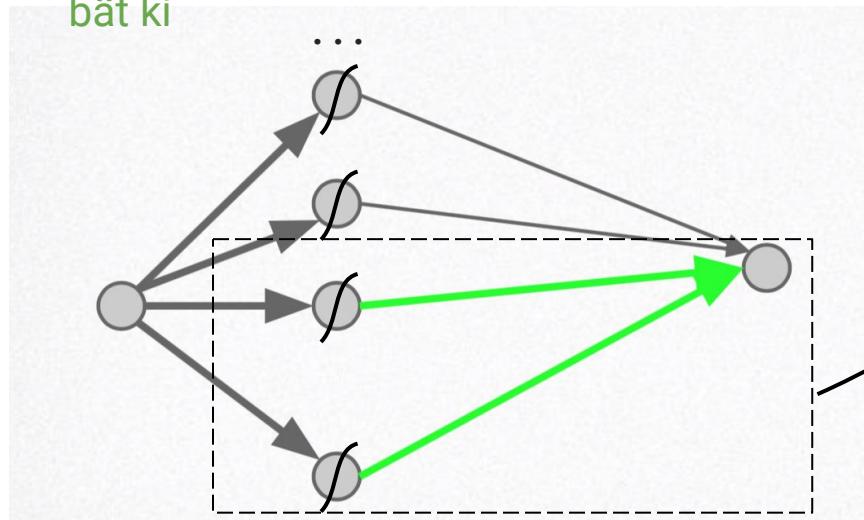
Một lớp ẩn được xem có thể xấp xỉ mọi hàm số (visual proof 1D)



Neural Network

Một lớp ẩn được xem có thể xấp xỉ mọi hàm số (visual proof 1D)

- 1 cặp hidden units (với hàm sigmoid) có thể tạo thành 1 “bump”.
- Tổ hợp các bumps có thể tạo thành 1 hàm số bất kì

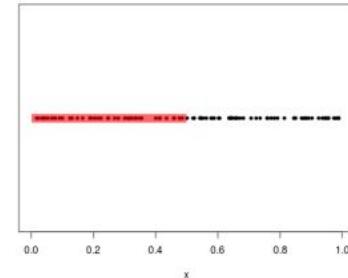


Neural Network

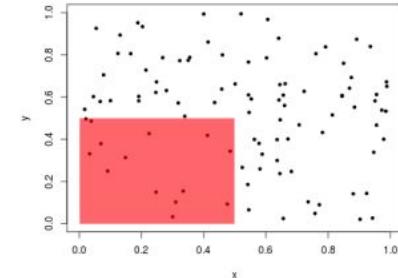
Tại sao cần mạng NN với nhiều hơn 1 hidden layer?

- 1 hidden layer cần rất nhiều hidden units để biểu diễn 1 arbitrary function
- Curse of dimensionality: Hiểu đơn giản, khi số chiều của dữ liệu tăng lên, độ phức tạp của model phải tăng lên theo **cấp số mũ** để đảm bảo khả năng giải quyết bài toán được giữ nguyên.

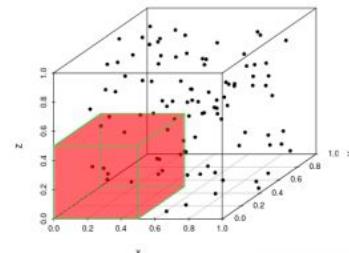
1-D: 42% of data captured.



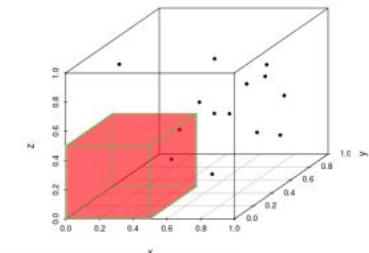
2-D: 14% of data captured.



3-D: 7% of data captured.



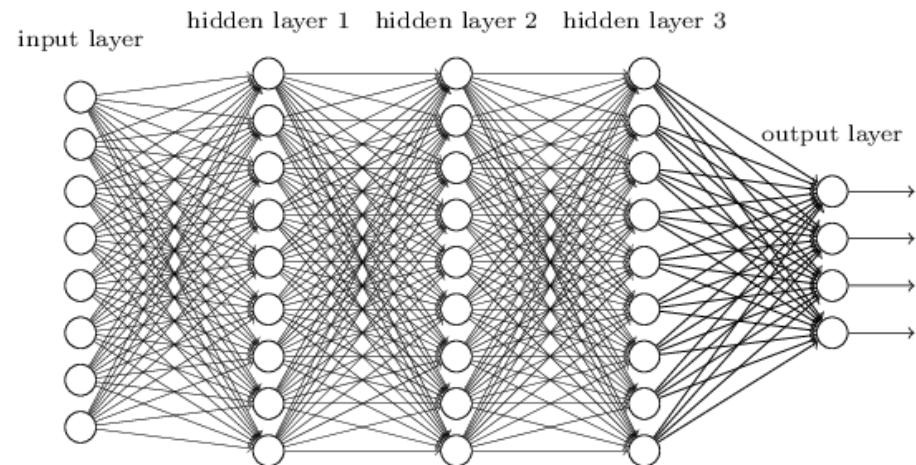
4-D: 3% of data captured.
 $t = 0$



Neural Network

Tại sao cần mạng NN với nhiều hơn 1 hidden layer?

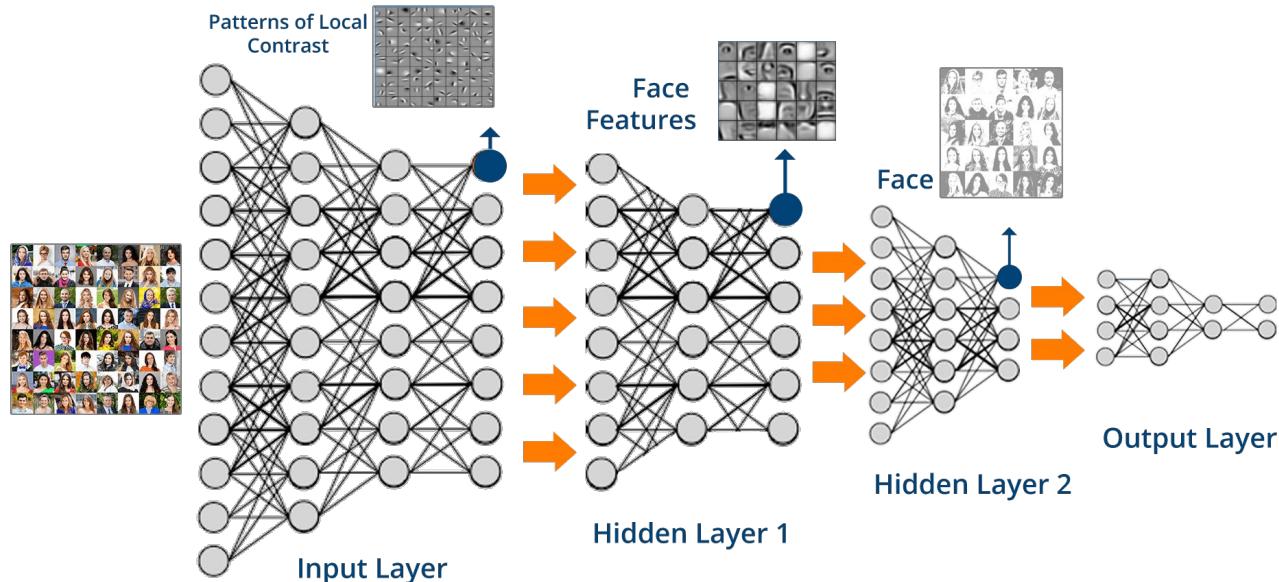
- 1 hidden layer cần rất nhiều hidden units để biểu diễn 1 arbitrary function
- Curse of dimensionality
- Có thể xem mỗi lớp như hàm mapping độc lập. Những lớp sau có thể **tận dụng** lại những hàm mapping đơn giản từ lớp trước đó để tạo ra hàm mapping có khả năng biểu diễn cực kì phức tạp (tính kẽ thừa) -> **cấu trúc phân cấp (III)**



Ex: phép đếm -> phép cộng (đếm nhiều lần)
 -> phép nhân (cộng nhiều lần) -> phép mũ luỹ thừa (nhân nhiều lần)

Neural Network

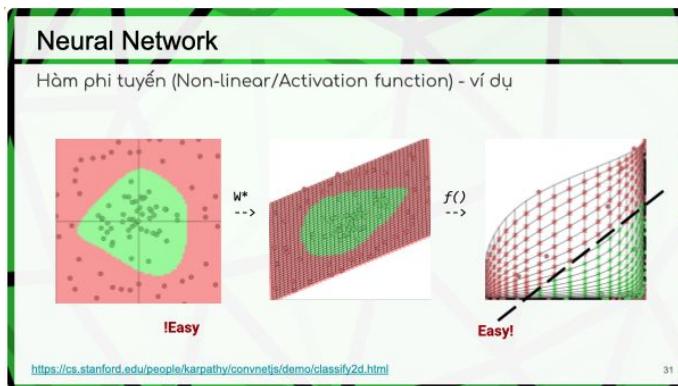
Cấu trúc phân cấp



- Cấu trúc phân cấp giúp phân đoạn hàm mapping phức tạp ra thành từng bước một với những hàm đơn giản hơn. Như ví dụ ở hình trên.

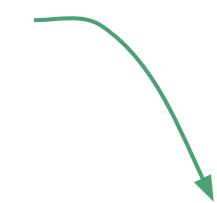
Neural Network

Cấu trúc phân cấp

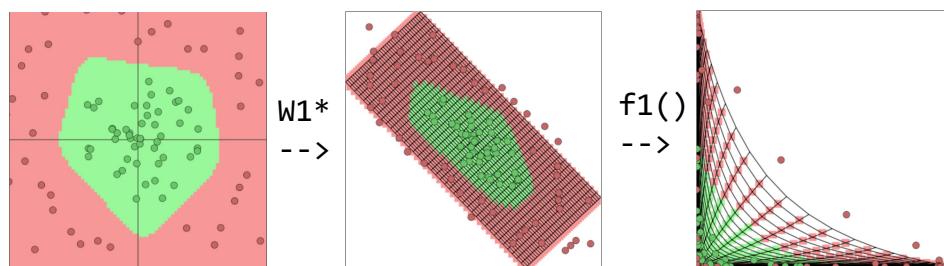
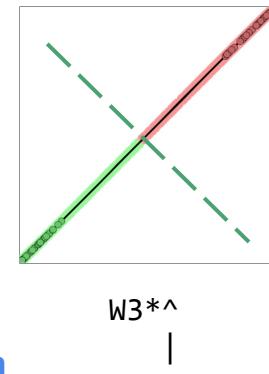


Quá trình biến đổi trở nên đơn giản hơn

NN với 1 lớp ẩn



NN
với nhiều hơn 1 lớp ẩn



Neural Network

Deep Neural Network

- = Linear model (I)
- + Hàm phi tuyến (II)
- + Cấu trúc phân cấp(III)

Neural Network

- Ta đã đề cập tới việc NNs CÓ THỂ xấp xỉ bất kì hàm số nào, với điều kiện đã có được các tham số.
-> Nhưng làm cách nào để tìm ra các tham số đó để NNs để đạt được phép mapping như ý?

Backpropagation algorithm

- Thuật toán lan truyền ngược

Backpropagation algorithm

Partial Derivative (Đạo hàm từng phần)

$$f(x, y) = xy$$

Đạo hàm riêng trên từng biến nói cho chúng ta biết độ nhạy(sensitivity) của hàm số trên biến đó

$$\frac{df(x; y)}{dx} = \lim_{\epsilon \rightarrow 0} \frac{f(x+\epsilon; y) - f(x; y)}{\epsilon} \longrightarrow f(x + \epsilon; y) = f(x; y) + \epsilon \frac{df(x; y)}{dx}$$

$$\frac{df(y; x)}{dy} = \lim_{\epsilon \rightarrow 0} \frac{f(y+\epsilon; x) - f(y; x)}{\epsilon} \longrightarrow f(y + \epsilon; x) = f(y; x) + \epsilon \frac{df(y; x)}{dy}$$

$$x = 6, y = -2 \rightarrow f(x, y) = -12$$

$$\frac{\partial f}{\partial x} = y = -2 \quad \text{Khi tăng giá trị của } x \text{ lên một khoảng nhỏ } \epsilon \text{ thì } f(x,y) \text{ giảm } 2\epsilon$$

$$\frac{\partial f}{\partial y} = x = 6 \quad \text{Tương tự, khi tăng giá trị của } y \text{ lên một khoảng nhỏ } \epsilon \text{ thì } f(x,y) \text{ tăng } 6\epsilon$$

Backpropagation algorithm

Partial Derivative

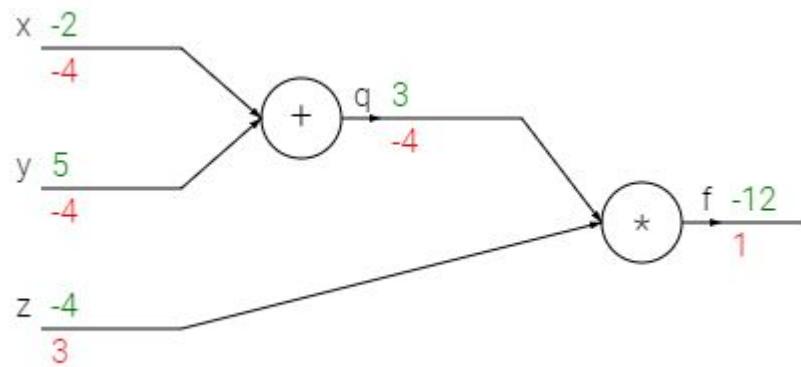
$$f(x, y, z) = (x + y)z$$

$$q = x + y \rightarrow f = qz$$

$$\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1 \quad \frac{\partial f}{\partial q} = z = -4$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x} = -4$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y} = -4 \quad \frac{\partial f}{\partial z} = q = 3$$



Backpropagation algorithm

Partial Derivative

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

How to calculate partial derivative of:

$$f(w, x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

Backpropagation algorithm

Partial Derivative

$$f(w, x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$$a_0 = w_0x_0$$

$$a_1 = w_1x_1$$

$$a_2 = a_0 + a_1$$

$$a_3 = w_2 + a_2$$

$$a_4 = -a_3$$

$$a_5 = e^{a_4}$$

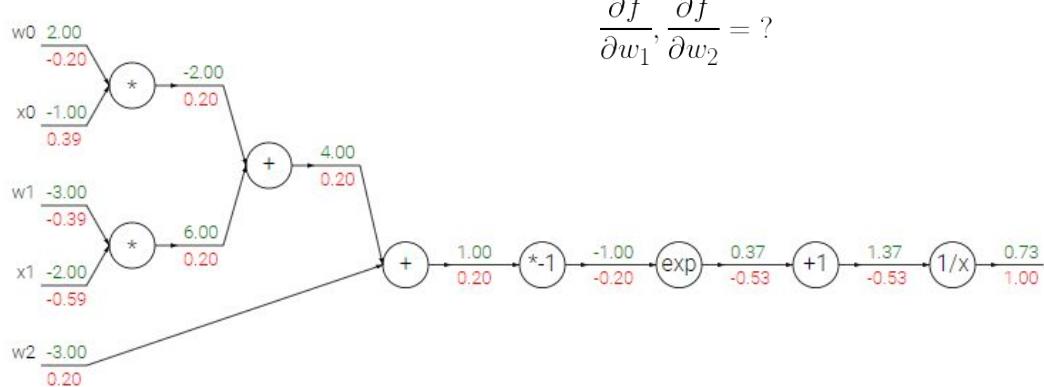
$$a_6 = a_5 + 1$$

$$f(w, x) = \frac{1}{a_6}$$

$$\frac{\partial f}{\partial a_6} = -\frac{1}{a_6^2} = -\frac{1}{(1.37)^2} \approx -0.53$$

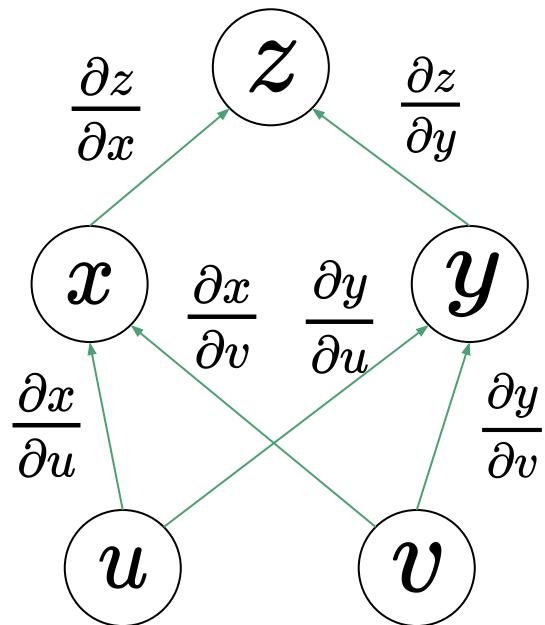
$$\frac{\partial f}{\partial a_5} = \frac{\partial f}{\partial a_6} \frac{\partial a_6}{\partial a_5} = -0.53$$

$$\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2} = ?$$



Backpropagation algorithm

Multivariable Chain Rule



$$\frac{\partial z}{\partial u} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial u} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial u}$$

$$\frac{\partial z}{\partial v} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial v} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial v}$$

Backpropagation algorithm

Multivariable Chain Rule

$$y_1 = g_1(x)$$

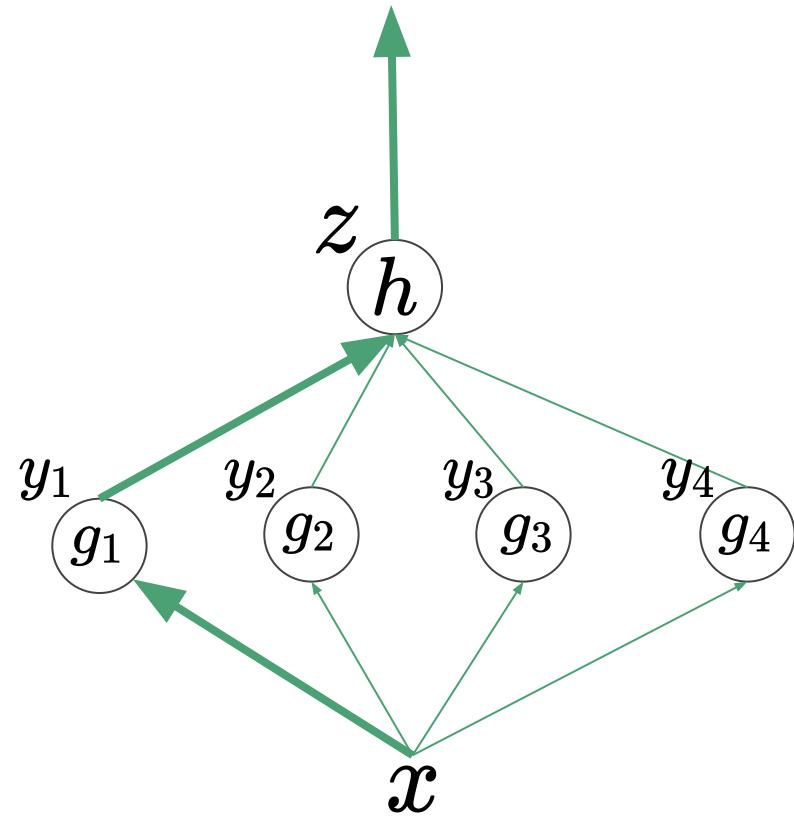
$$y_2 = g_2(x)$$

$$y_3 = g_3(x)$$

$$y_4 = g_4(x)$$

$$z = h(y_1, y_2, y_3)$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} +$$



Backpropagation algorithm

Multivariable Chain Rule

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

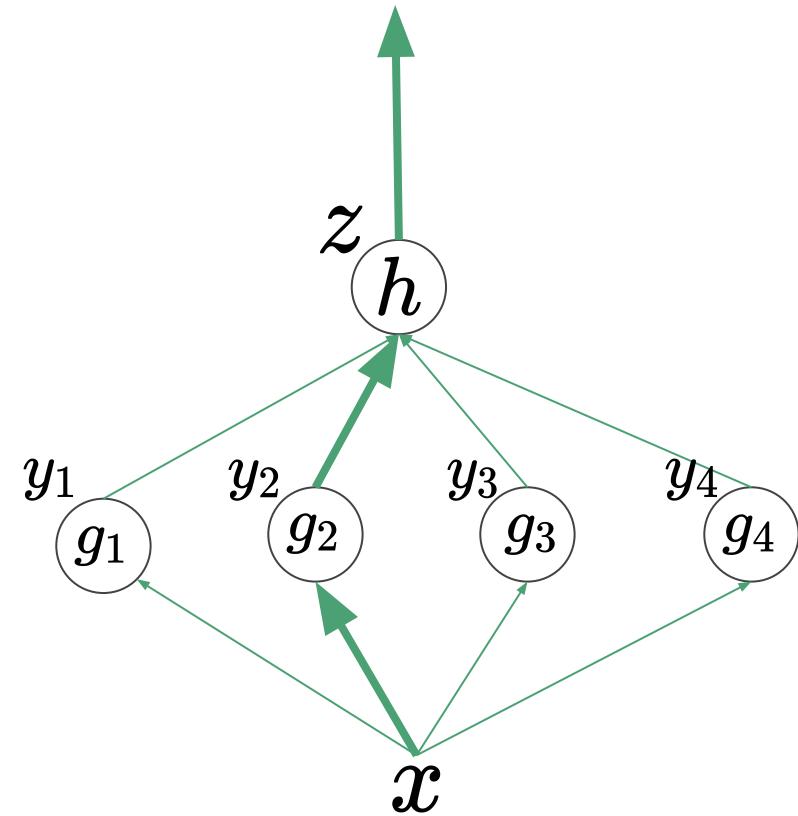
$$y_3 = g_3(x)$$

$$y_4 = g_4(x)$$

$$z = h(y_1, y_2, y_3)$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} +$$

$$\frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x} +$$



Backpropagation algorithm

Multivariable Chain Rule

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

$$y_3 = g_3(x)$$

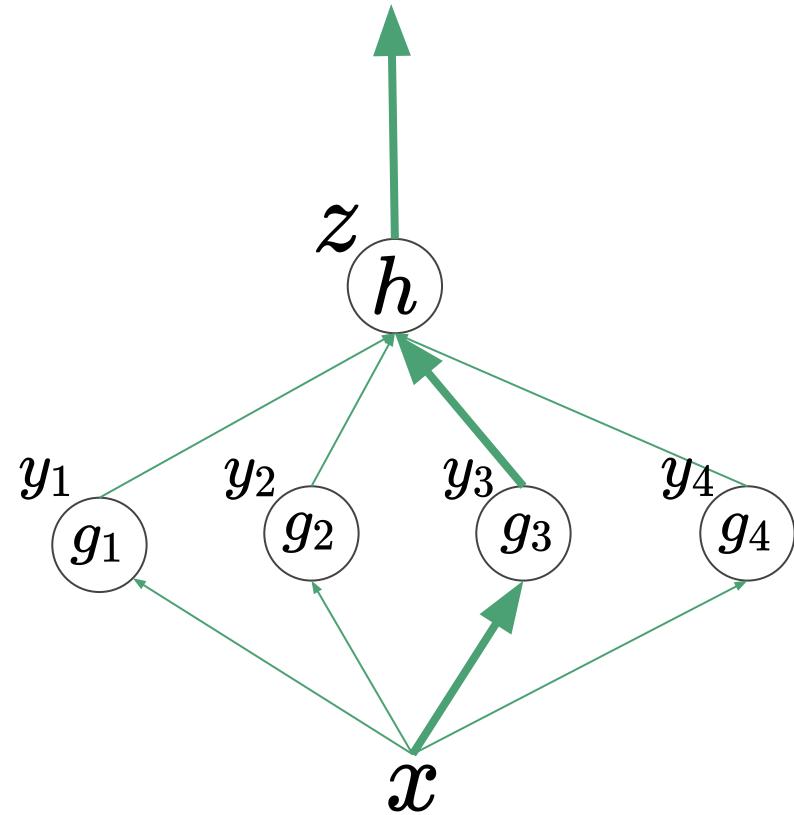
$$y_4 = g_4(x)$$

$$z = h(y_1, y_2, y_3)$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} +$$

$$\frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x} +$$

$$\frac{\partial z}{\partial y_3} \frac{\partial y_3}{\partial x} +$$



Backpropagation algorithm

Multivariable Chain Rule

$$y_1 = g_1(x)$$

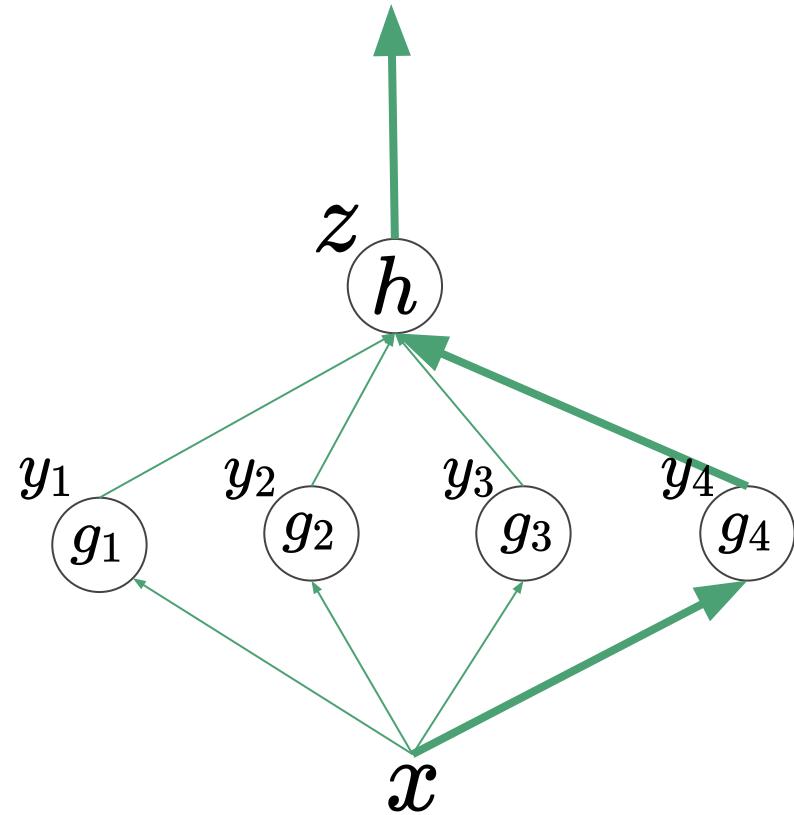
$$y_2 = g_2(x)$$

$$y_3 = g_3(x)$$

$$y_4 = g_4(x)$$

$$z = h(y_1, y_2, y_3)$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x} + \frac{\partial z}{\partial y_3} \frac{\partial y_3}{\partial x} + \frac{\partial z}{\partial y_4} \frac{\partial y_4}{\partial x}$$



Backpropagation algorithm

Quy ước

$$\nabla X = \left[\frac{\partial J}{\partial x_{i,j}} \right]_{i,j=1}^{m,n} = \begin{bmatrix} \frac{\partial J}{\partial x_{1,1}} & \dots & \frac{\partial J}{\partial x_{n,1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial x_{1,m}} & \dots & \frac{\partial J}{\partial x_{m,n}} \end{bmatrix}$$

$$(\nabla X)_{i,:} = \left[\frac{\partial J}{\partial x_{i,1}} \quad \dots \quad \frac{\partial J}{\partial x_{i,n}} \right]$$

$$(\nabla X)_{i,j} = \frac{\partial J}{\partial x_{i,j}}$$

Backpropagation algorithm

Copy Node

$$y_{(ij)1} = y_{(ij)2} = y_{(ij)3} = \text{copy}(x_{ij})$$

$$\rightarrow \frac{\partial y_{(ij)1}}{\partial x_{ij}} = \frac{\partial y_{(ij)2}}{\partial x_{ij}} = \frac{\partial y_{(ij)3}}{\partial x_{ij}} = 1$$

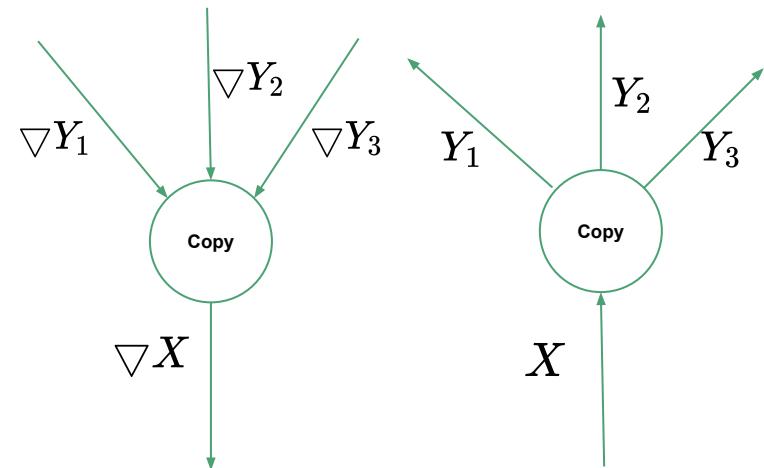
$$\nabla x_{ij} = \frac{\partial J}{\partial x_{ij}} = \sum_{k=1}^3 \frac{\partial J}{\partial y_{(ij)k}} \frac{\partial (y_{ij})_k}{\partial x_{ij}}$$

$$= \sum_{k=1}^3 \frac{\partial J}{\partial (y_{ij})_k}$$

$$= \sum_{k=1}^3 \nabla (y_{ij})_k$$

$$\rightarrow \nabla X = \sum_{k=1}^3 \nabla Y_k$$

$$Y_1 = Y_2 = Y_3 = X$$



X

x_{ij}

Y_1

$y_{(ij)1}$

Backpropagation algorithm

Addition Node

$$y_{ij} = x_{(ij)1} + x_{(ij)2}$$

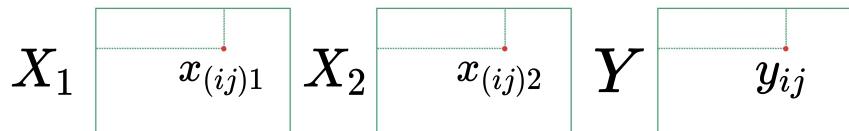
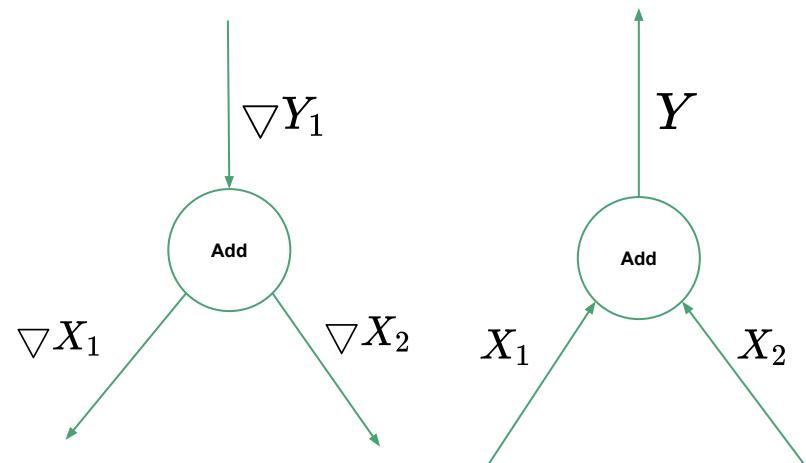
$$\rightarrow \frac{\partial y_{(ij)}}{\partial x_{(ij)1}} = \frac{\partial y_{(ij)}}{\partial x_{(ij)2}} = 1$$

$$\begin{aligned}\nabla x_{(ij)1} &= \frac{\partial J}{\partial x_{(ij)1}} = \frac{\partial J}{\partial y_{ij}} \frac{\partial y_{ij}}{\partial x_{(ij)1}} \\ &= \frac{\partial J}{\partial y_{ij}} = \nabla y_{ij}\end{aligned}$$

$$\rightarrow \nabla X_1 = \nabla Y$$

$$\rightarrow \nabla X_2 = \nabla Y$$

$$Y = X_1 + X_2$$



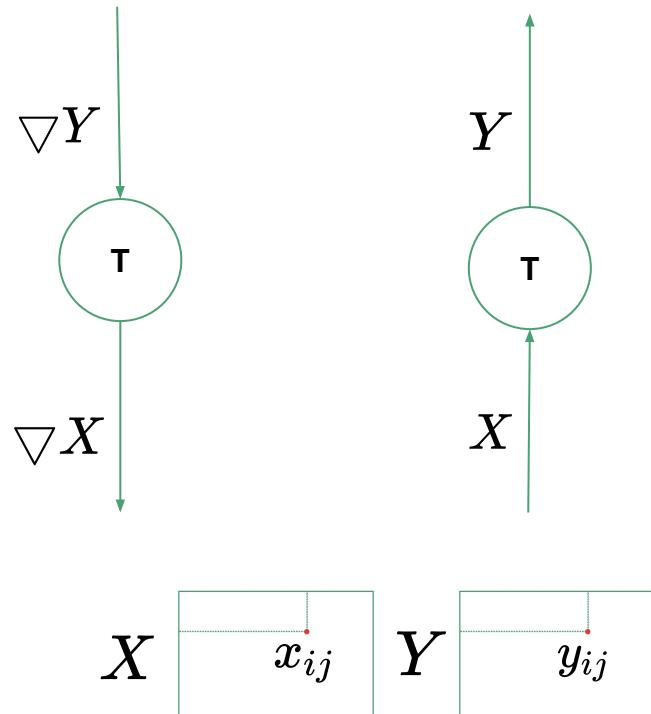
Backpropagation algorithm

Transpose Node

$$\nabla X = (\nabla Y)^T$$

Chứng minh công thức này như thế nào?

$$Y = X^T$$



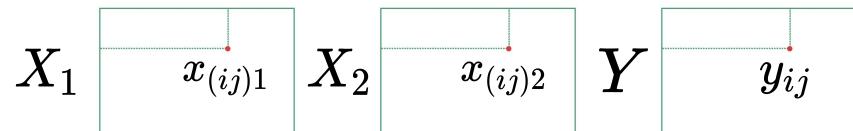
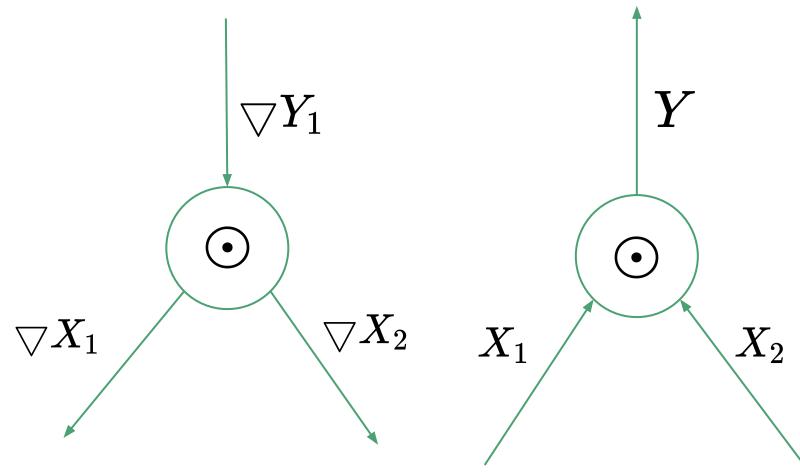
Backpropagation algorithm

Element wise multiplication Node

$$\begin{aligned}\nabla X_1 &= \nabla Y X_2 \\ \nabla X_2 &= \nabla Y X_1\end{aligned}$$

Chứng minh công thức này như thế nào?

$$Y = X_1 \odot X_2$$



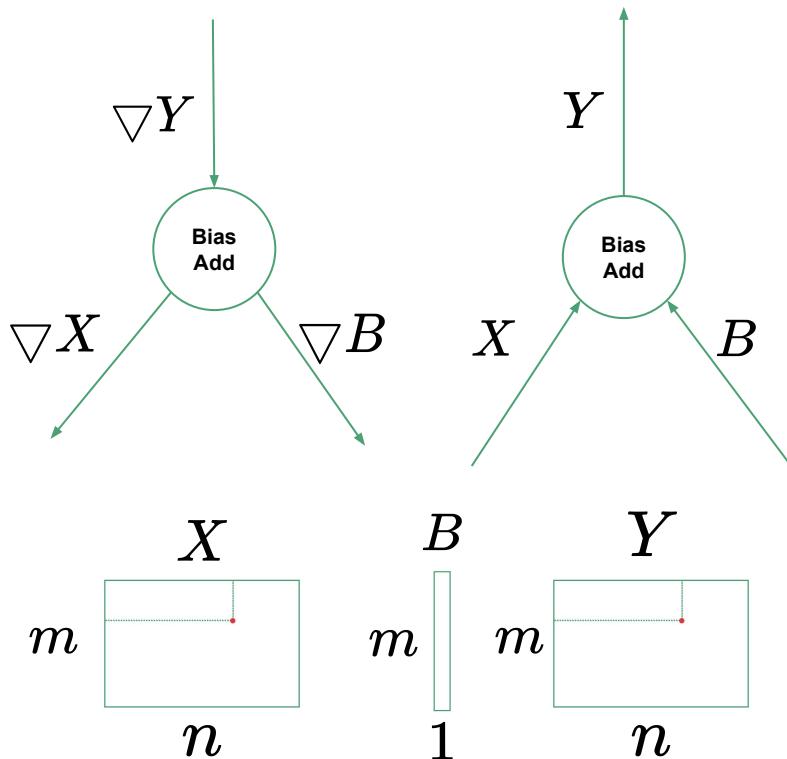
Backpropagation algorithm

Add Bias Node

$$\begin{aligned}\nabla X &= \nabla Y \\ \nabla B &= \sum_{j=1}^n \nabla Y_j\end{aligned}$$

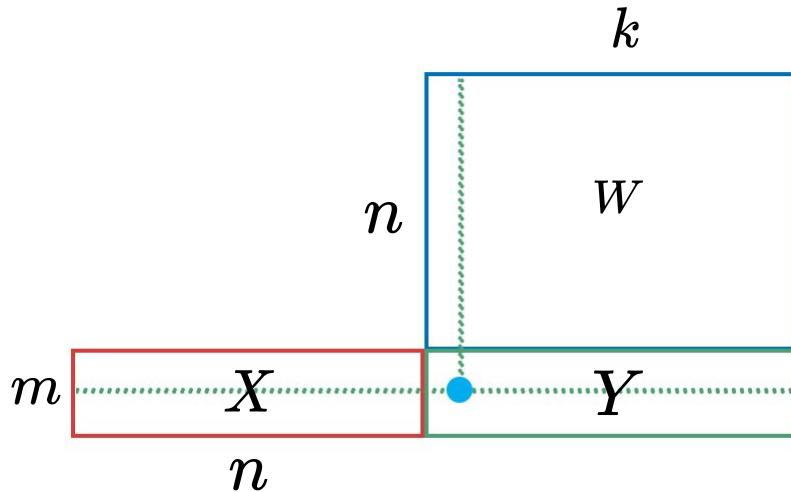
Chứng minh công thức này như thế nào?

$$Y = X + B$$

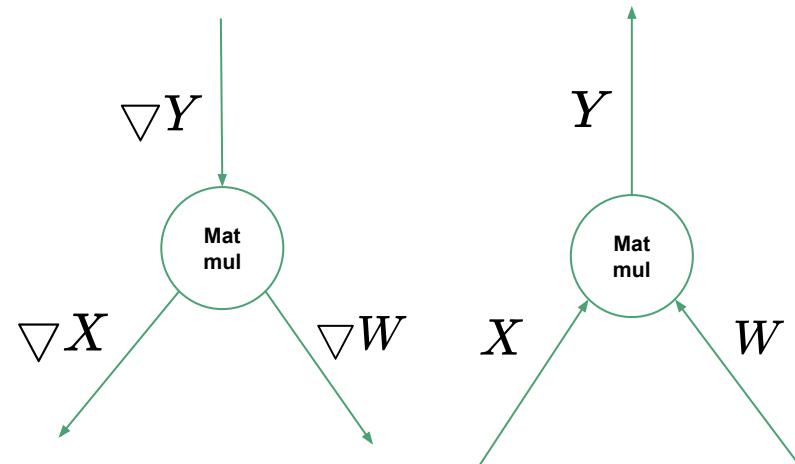


Backpropagation algorithm

Matmul Node



$$Y = XW$$

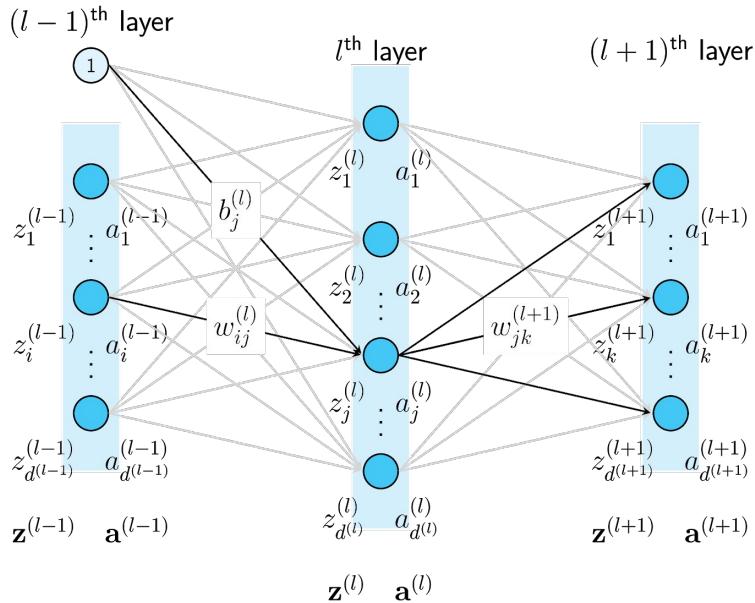


$$\boxed{\nabla X = \nabla Y W^T}$$

Chứng minh công thức này như thế nào?

Backpropagation algorithm

Fully connected Network



$$\mathbf{e}^{(l+1)} = [e_1^{(l+1)}, e_2^{(l+1)}, \dots, e_{d^{(l+1)}}^{(l+1)}]^T \in \mathbb{R}^{d^{(l+1)} \times 1}$$

$\mathbf{w}_{j:}^{(l+1)}$ Hàng thứ j của ma trận $\mathbf{W}^{(l+1)}$

$$\begin{aligned} \frac{\partial J}{\partial w_{ij}^{(l)}} &= \frac{\partial J}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}} \\ &= e_j^{(l)} a_i^{(l-1)} \end{aligned}$$

$$\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$$

$$\mathbf{b}^{(l)} \in \mathbb{R}^{d^{(l)} \times 1}$$

$$z_j^{(l)} = \mathbf{w}_j^{(l)T} \mathbf{a}^{(l-1)} + b_j^{(l)}$$

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)T} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$$

$$\begin{aligned} e_j^{(l)} &= \frac{\partial J}{\partial z_j^{(l)}} = \frac{\partial J}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \\ &= \left(\sum_{k=1}^{d^{(l+1)}} \frac{\partial J}{\partial z_k^{(l+1)}} \cdot \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} \right) f'(z_j^{(l)}) \\ &= \left(\sum_{k=1}^{d^{(l+1)}} e_k^{(l+1)} w_{jk}^{(l+1)} \right) f'(z_j^{(l)}) \\ &= \left(\mathbf{w}_{j:}^{(l+1)} \mathbf{e}^{(l+1)} \right) f'(z_j^{(l)}) \end{aligned}$$

Backpropagation algorithm

Learning Process

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

Backpropagation algorithm

Learning Process

- **Bước 4:** Lan truyền ngược

Đổi với output layer

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

Đổi với hidden layer, bỏ **4.1** và thay đổi **4.2**:

- **4.2** Tính sai số gradient: $\delta_k^{(l)} = a_k^{(l)} \times (1 - a_k^{(l)}) \times \sum_{i=1}^n \delta_i^{(l+1)} \times w_{ik}^{(l+1)}$

Backpropagation algorithm

Learning Process

- Quá trình lan truyền ngược được thực hiện 1 lần sau mỗi lần Lan truyền thuận
- Thực hiện lặp **Bước 3** và **Bước 4** nhiều lần cho đến khi mô hình hội tụ (converge)
 - Đặt ra số lần
 - Đặt ra độ chính xác mong muốn
- Quá trình lặp **Bước 3** và **Bước 4** được gọi là quá trình huấn luyện (training process) và được thực hiện trên toàn bộ tập huấn luyện
- Sau khi dừng quá trình huấn luyện, ta sẽ có được tập các giá trị weights tối ưu cho bài toán
- Tập weights này sẽ được giữ nguyên (freeze) và mang vào sử dụng