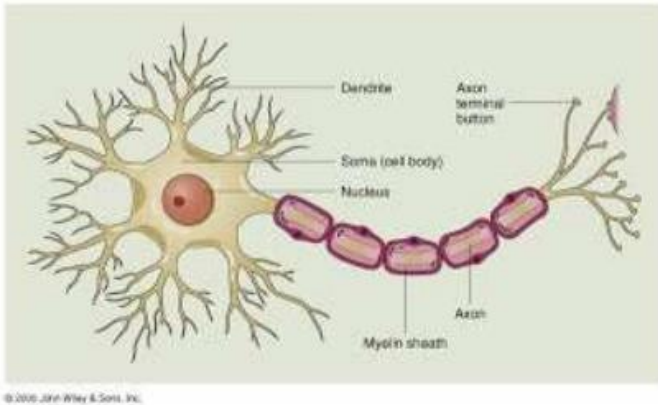


Bài 2:

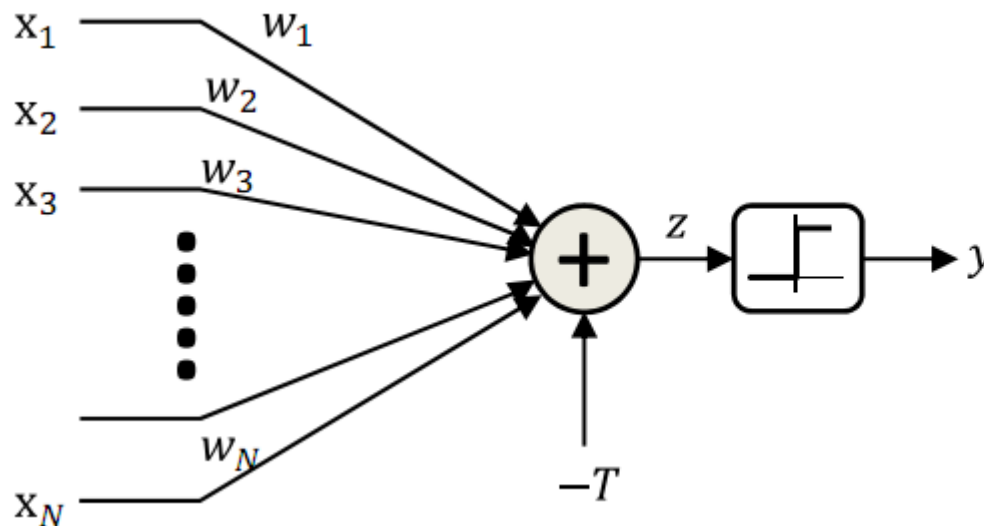
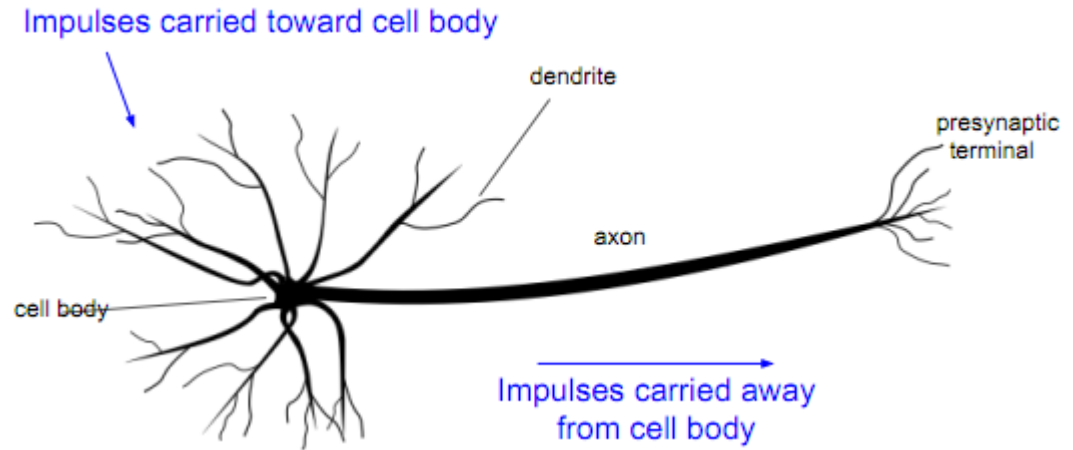
Giới thiệu về mạng nơ-ron

Mạng nơ-ron và bộ não

- Mạng nơ-ron mô phỏng cấu trúc kết nối của não người
- Não người tạo bởi nhiều nơ-ron liên kết với nhau



Perceptron

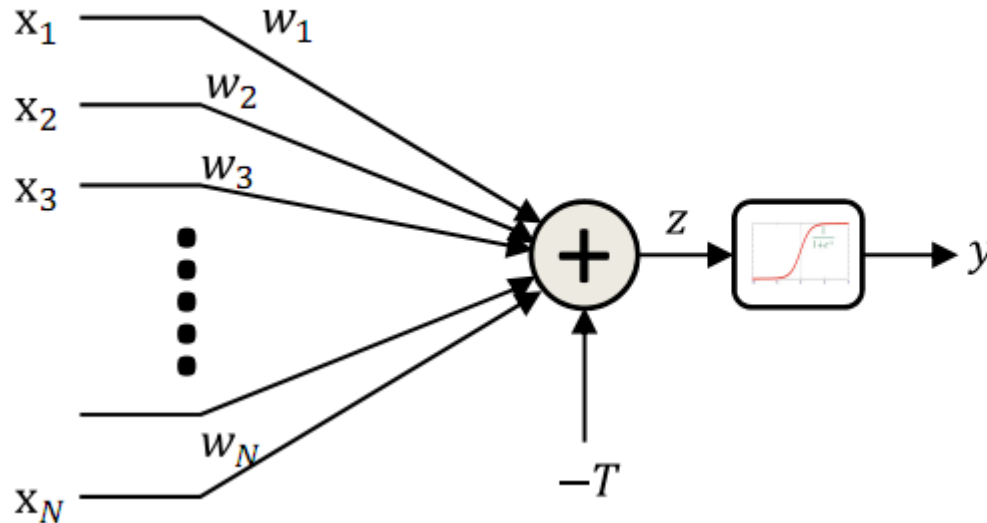


$$z = \sum_i w_i x_i - T$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{else} \end{cases}$$

- Bắt xung “fire” nếu tổng có trọng số của các đầu vào với “bias” T không âm

Perceptron mềm (logistic)

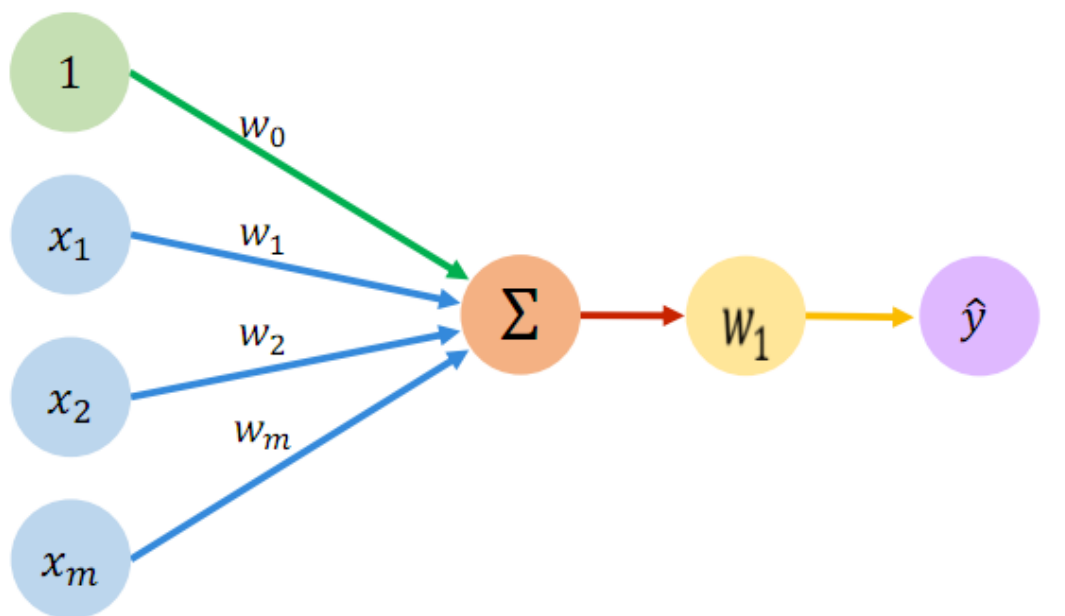


$$z = \sum_i w_i x_i - T$$

$$y = \frac{1}{1 + \exp(-z)}$$

- Sử dụng một hàm khả vi thay cho hàm xung
- Hàm kích hoạt sigmoid được dùng để xấp xỉ hàm xung
- Hàm kích hoạt là hàm tác động lên tổng có trọng số của các dữ liệu vào

Perceptron mềm (logistic)



Inputs Weights Sum Non-Linearity Output

Output

Linear combination of inputs

$$\hat{y} = g \left(w_0 + \sum_{i=1}^m x_i w_i \right)$$

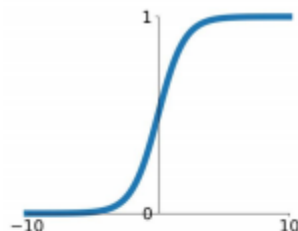
Non-linear activation function

Bias

Một số hàm kích hoạt thường gặp

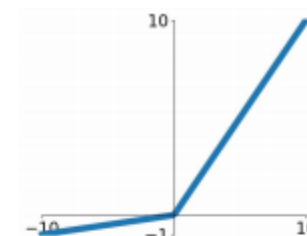
Sigmoid

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



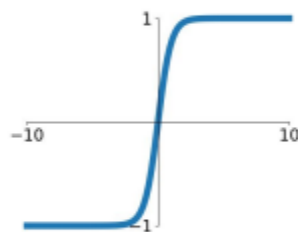
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

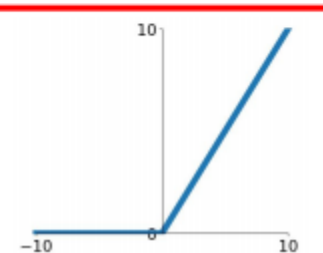


Maxout

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

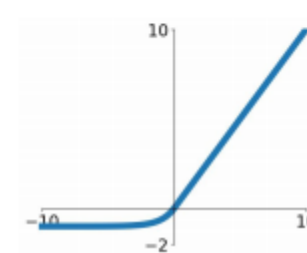
ReLU

$$\max(0, x)$$



ELU

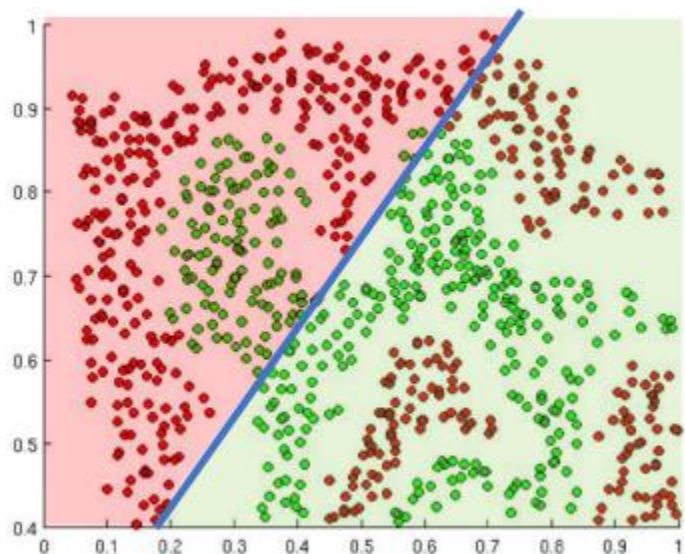
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



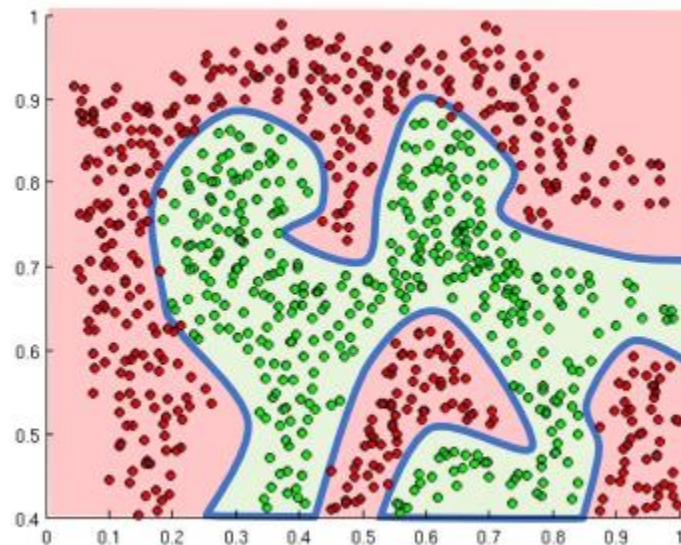
- ReLU là lựa chọn mặc định tốt cho nhiều bài toán
- Hiện nay xu hướng dùng một số hàm kích hoạt hiện đại hơn như ReLU6, swish, mish

Tầm quan trọng của hàm kích hoạt

- Mục đích sử dụng hàm kích hoạt là đưa các lớp phi tuyến vào mạng nơ-ron

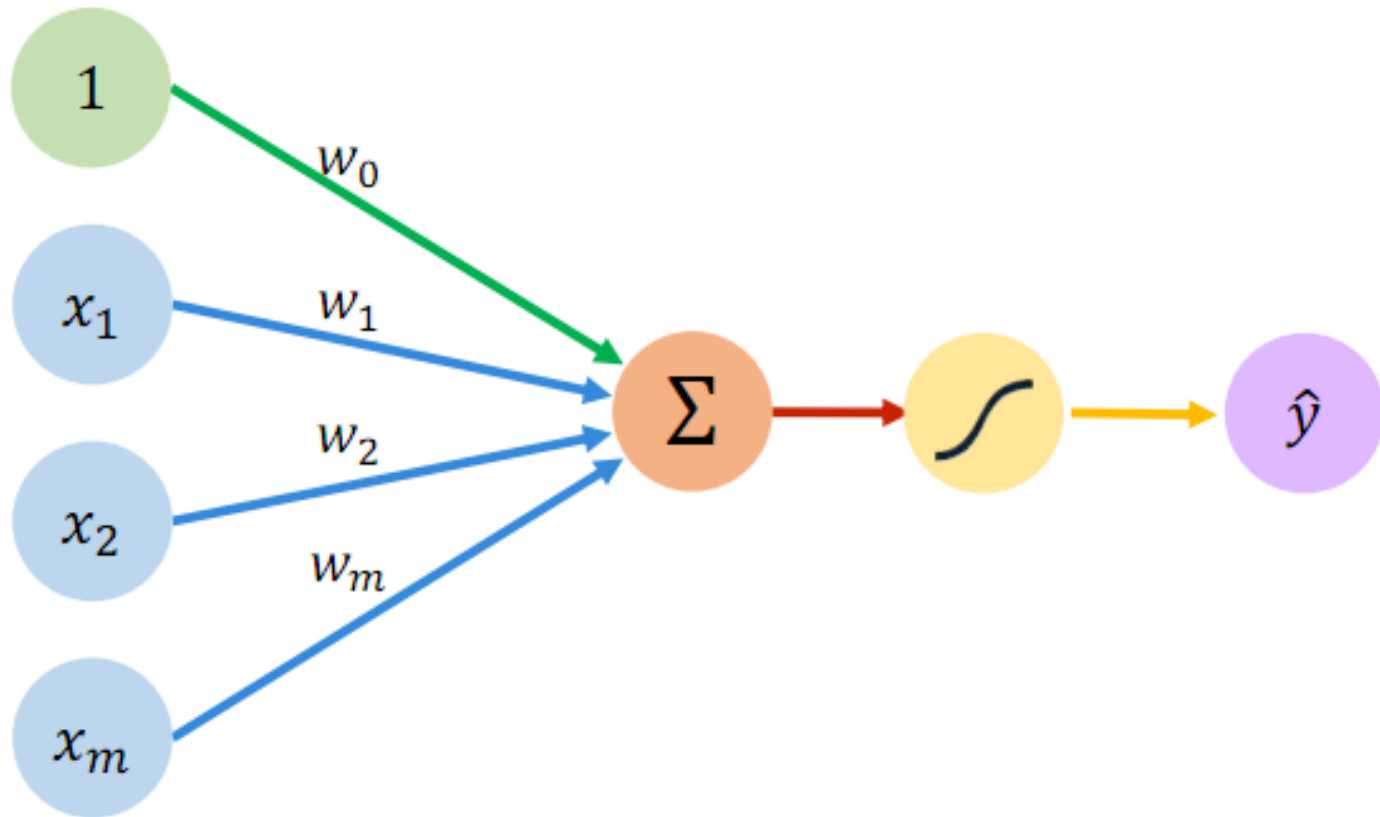


Hàm kích hoạt tuyến tính luôn sinh ra đường phân cách tuyến tính bất kể mạng có lớn cỡ nào



Các lớp phi tuyến cho phép chúng ta xấp xỉ các hàm phức tạp

Perceptron đơn giản hóa



Inputs

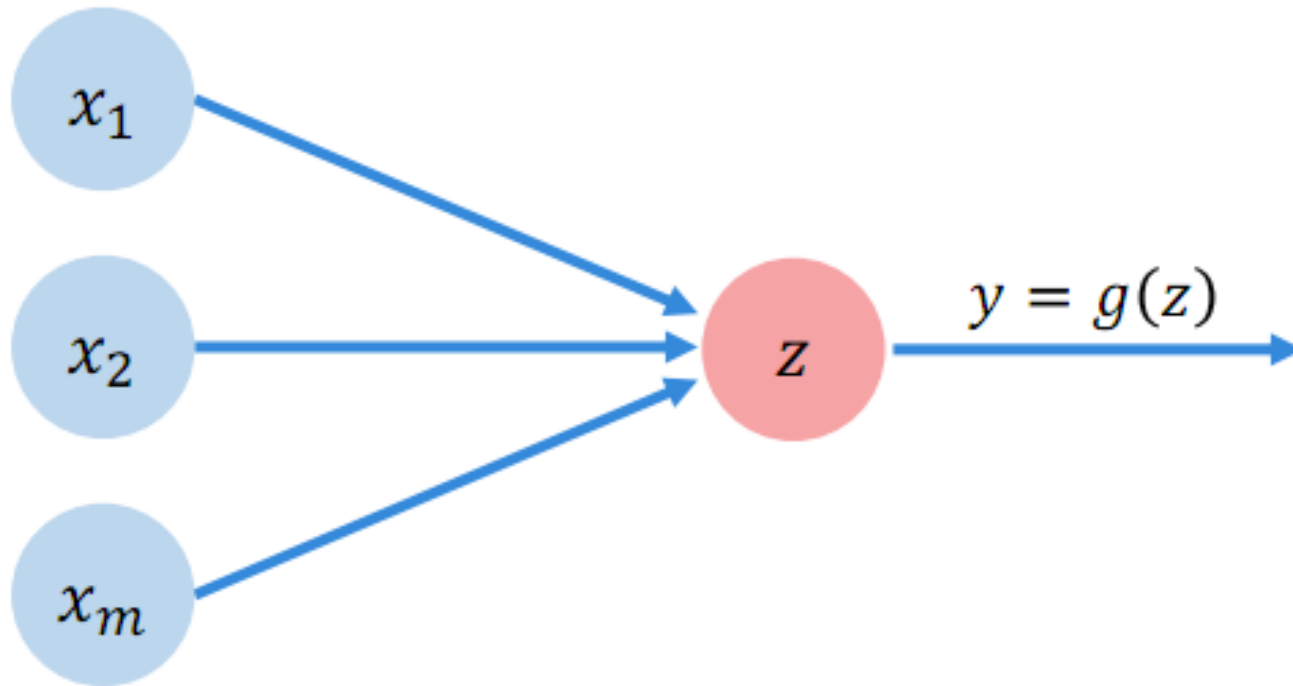
Weights

Sum

Non-Linearity

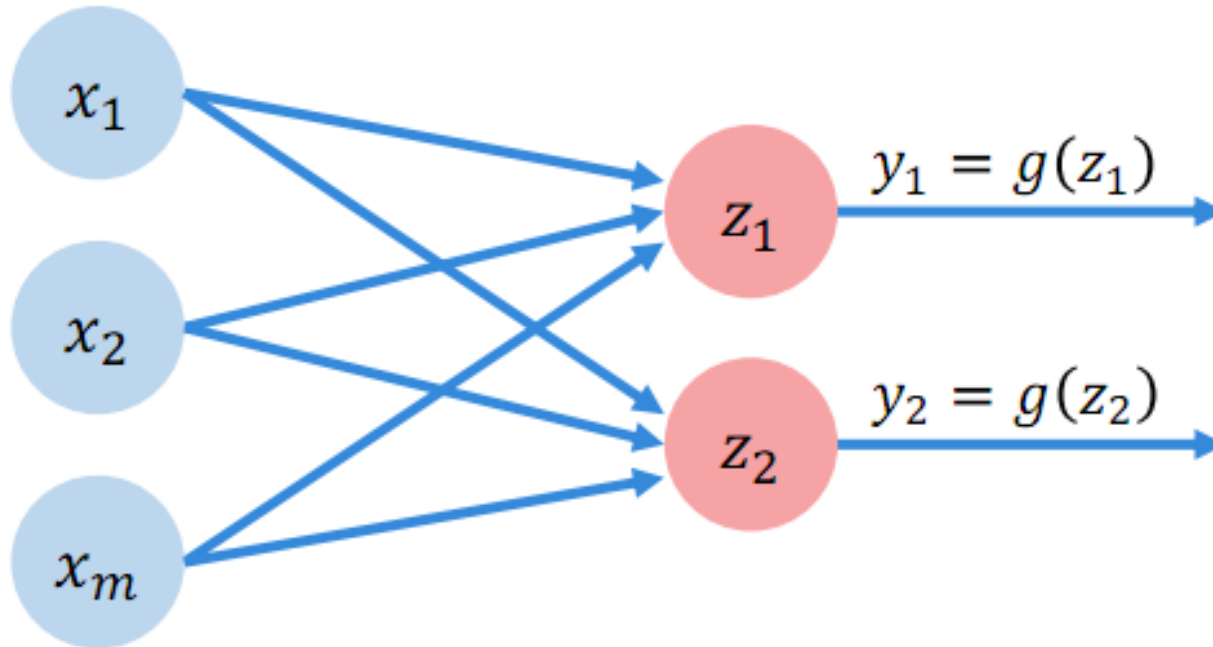
Output

Perceptron đơn giản hóa



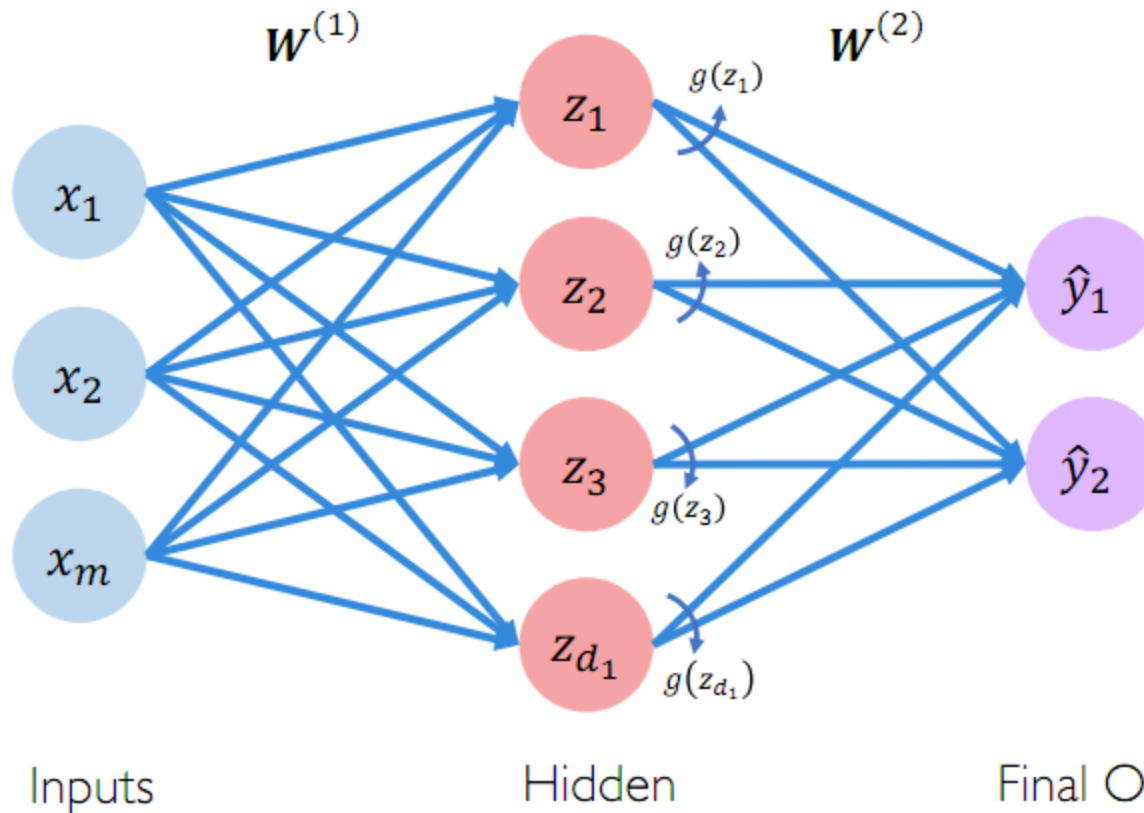
$$z = w_0 + \sum_{j=1}^m x_j w_j$$

Perceptron nhiều đầu ra



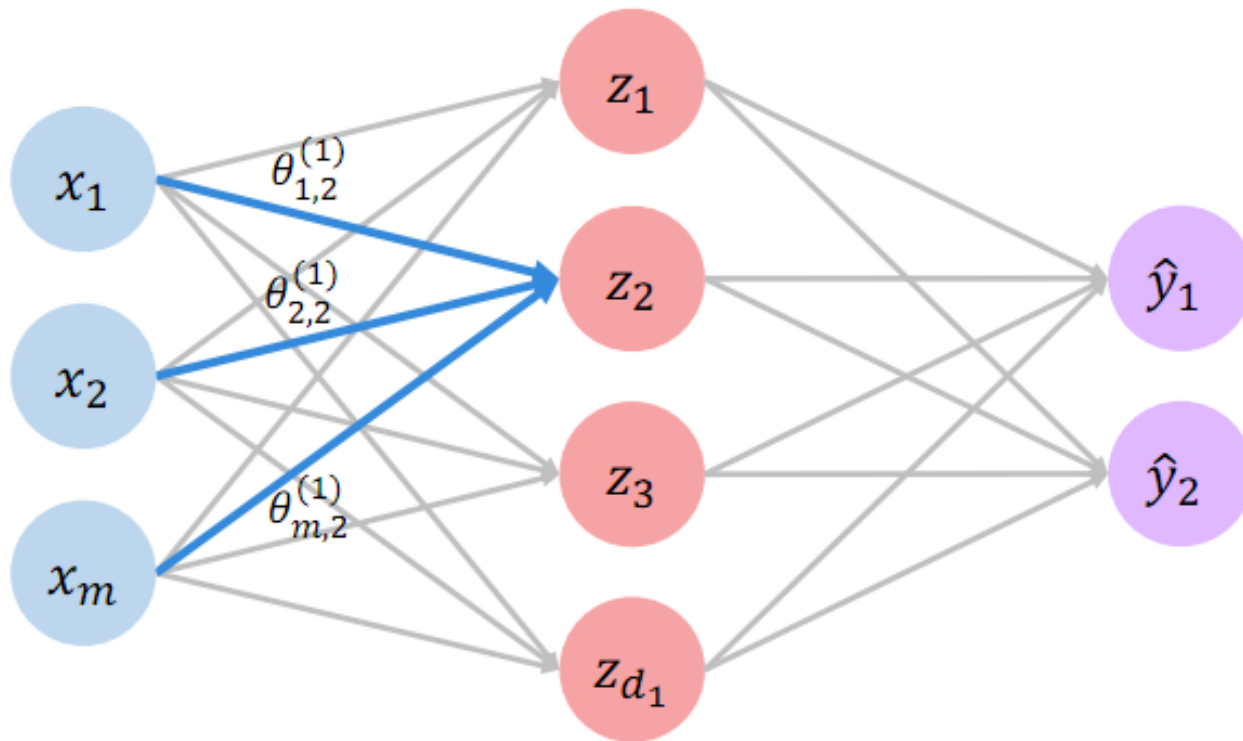
$$z_i = w_{0,i} + \sum_{j=1}^m x_j w_{j,i}$$

Mạng nơ-ron một lớp ẩn



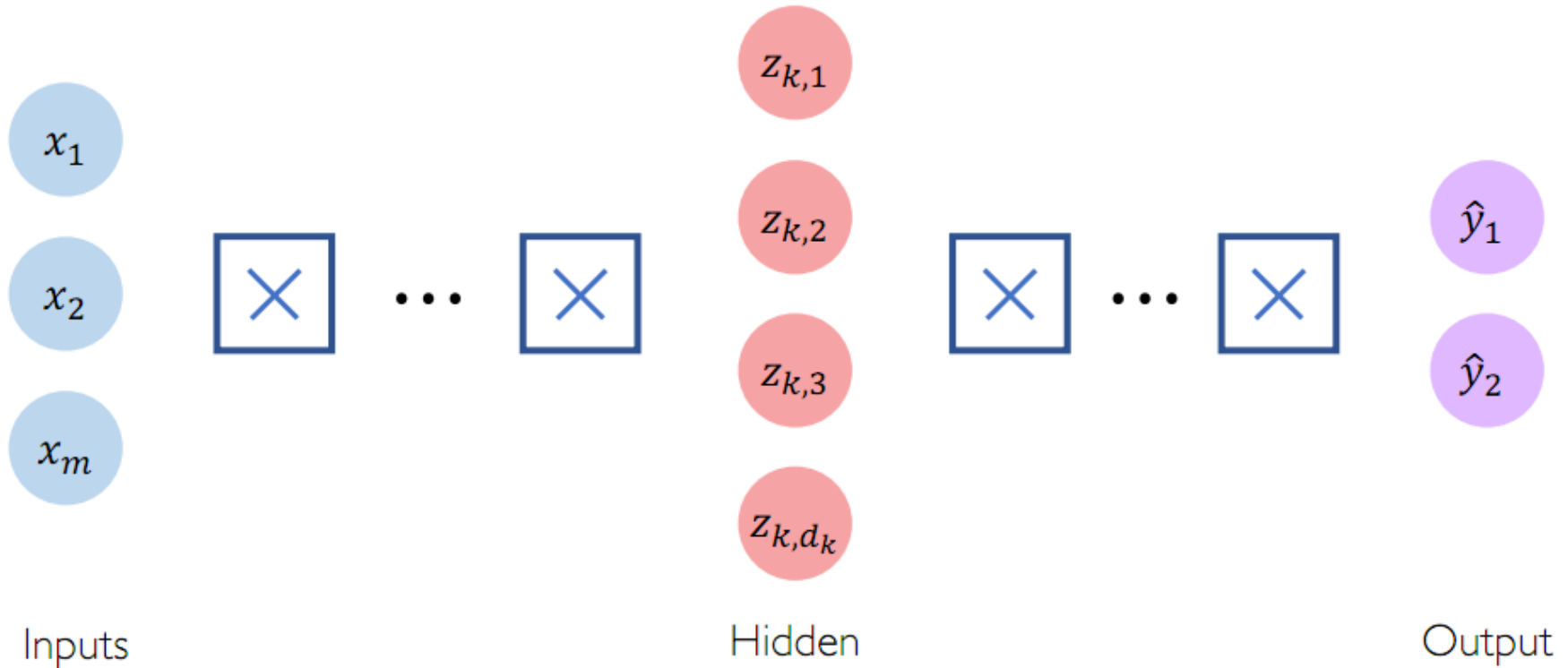
$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^m x_j w_{j,i}^{(1)} \quad \hat{y}_i = g \left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} z_j w_{j,i}^{(2)} \right)$$

Mạng nơ-ron một lớp ẩn



$$\begin{aligned} z_2 &= w_{0,2}^{(1)} + \sum_{j=1}^m x_j w_{j,2}^{(1)} \\ &= w_{0,2}^{(1)} + x_1 w_{1,2}^{(1)} + x_2 w_{2,2}^{(1)} + x_m w_{m,2}^{(1)} \end{aligned}$$

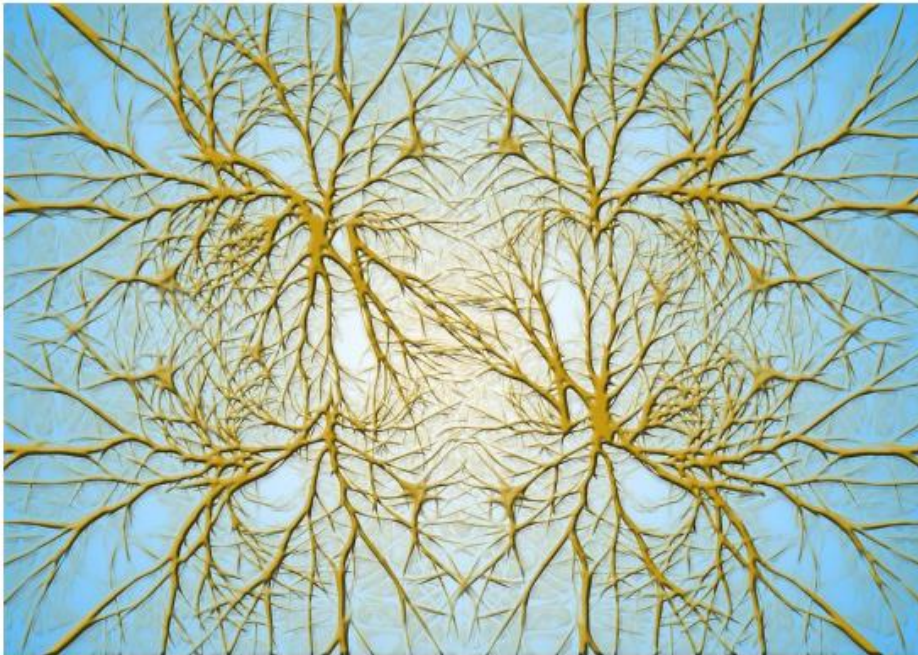
Mạng nơ-ron nhiều lớp



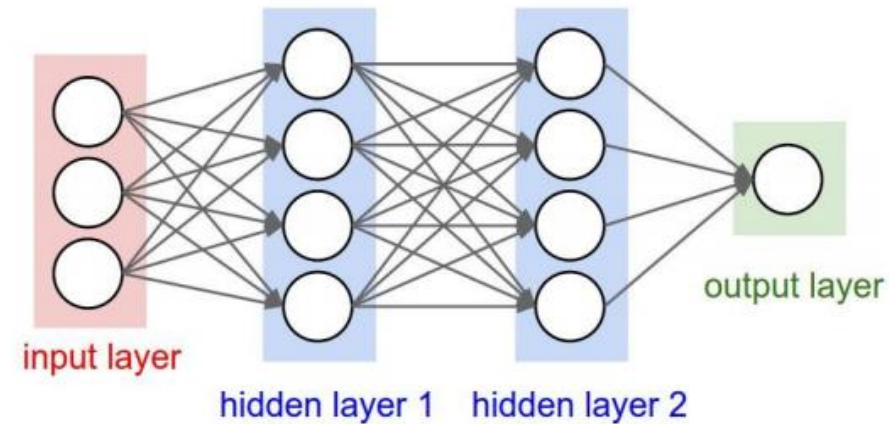
$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{d_{k-1}} g(z_{k-1,j}) w_{j,i}^{(k)}$$

Mạng nơ-ron và bộ não

Nơ-ron sinh học:
Kết nối phức tạp



Mạng nơ-ron nhân tạo:
Các nơ-ron tổ chức
thành các lớp (layers)
để tăng hiệu quả tính
toán nhờ song song hóa



Định lý xấp xỉ tổng quát

- Theorem (Universal Function Approximators). Một mạng nơ-ron từ hai lớp trở lên với số lượng nơ-ron đủ lớn có thể xấp xỉ bất kỳ hàm liên tục nào với độ chính xác tùy ý



$$n_1(x) = \text{Relu}(-5x - 7.7)$$

$$n_2(x) = \text{Relu}(-1.2x - 1.3)$$

$$n_3(x) = \text{Relu}(1.2x + 1)$$

$$n_4(x) = \text{Relu}(1.2x - .2)$$

$$n_5(x) = \text{Relu}(2x - 1.1)$$

$$n_6(x) = \text{Relu}(5x - 5)$$

$$Z(x) = -n_1(x) - n_2(x) - n_3(x) \\ + n_4(x) + n_5(x) + n_6(x)$$

Universal Function Approximation Theorem*



Hornik theorem 1: Whenever the activation function is *bounded and nonconstant*, then, for any finite measure μ , standard multilayer feedforward networks can approximate any function in $L^p(\mu)$ (the space of all functions on R^k such that $\int_{R^k} |f(x)|^p d\mu(x) < \infty$) arbitrarily well, provided that sufficiently many hidden units are available.

Hornik theorem 2: Whenever the activation function is *continuous, bounded and non-constant*, then, for arbitrary compact subsets $X \subseteq R^k$, standard multilayer feedforward networks can approximate any continuous function on X arbitrarily well with respect to uniform distance, provided that sufficiently many hidden units are available.

- In words: Given any continuous function $f(x)$, if a 2-layer neural network has enough hidden units, then there is a choice of weights that allow it to closely approximate $f(x)$.

Cybenko (1989) "Approximations by superpositions of sigmoidal functions"

Hornik (1991) "Approximation Capabilities of Multilayer Feedforward Networks"

Leshno and Schocken (1991) "Multilayer Feedforward Networks with Non-Polynomial Activation Functions Can Approximate Any Function"

Tại sao cần mạng nhiều lớp?

- Mạng nơ-ron nhiều lớp (thậm chí chỉ cần duy nhất một lớp ẩn!) là hàm xấp xỉ tổng quát
 - Mạng nơ-ron có thể biểu diễn hàm bất kỳ nếu nó đủ rộng (số nơ-ron trong một lớp đủ nhiều), đủ sâu (số lớp đủ lớn). Nếu muốn giảm độ sâu của mạng trong nhiều trường hợp sẽ phải bù lại bằng cách tăng chiều rộng lên lũy thừa lần!
 - Mạng nơ-ron một lớp ẩn có thể cần tới số lượng nơ-ron cao gấp lũy thừa lần so với một mạng nhiều tầng
 - Mạng nhiều lớp cần số lượng nơ-ron ít hơn rất nhiều so với các mạng nông (shallow networks) để cùng biểu diễn một hàm số giống nhau
- ➔ Mạng nhiều lớp giá trị hơn

Cực tiểu hóa hàm mục tiêu

- Tìm trọng số của mạng để hàm mục tiêu đạt giá trị cực tiểu

$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(x^{(i)}; \mathbf{W}), y^{(i)})$$

$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} J(\mathbf{W})$$



Remember:


$$\mathbf{W} = \{\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \dots\}$$

Cực tiểu hóa hàm mục tiêu

- Thuật toán Gradient Descent


Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$


```
 weights = tf.random_normal(shape, stddev=sigma)
```

2. Loop until convergence:

3. Compute gradient, $\frac{\partial J(W)}{\partial W}$

```
 grads = tf.gradients(ys=loss, xs=weights)
```

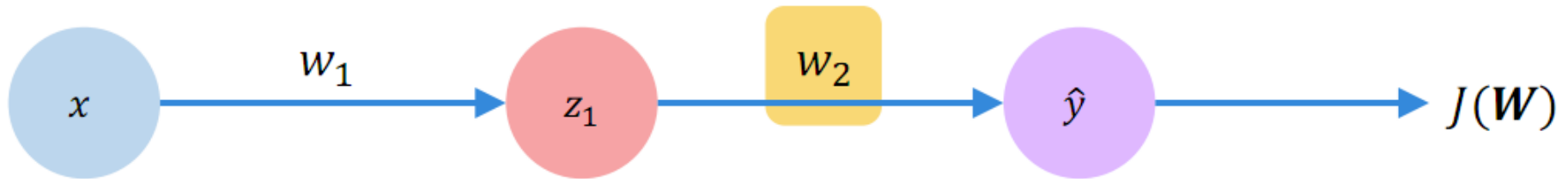
4. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(W)}{\partial W}$

```
 weights_new = weights.assign(weights - lr * grads)
```

5. Return weights

Giải thuật lan truyền ngược

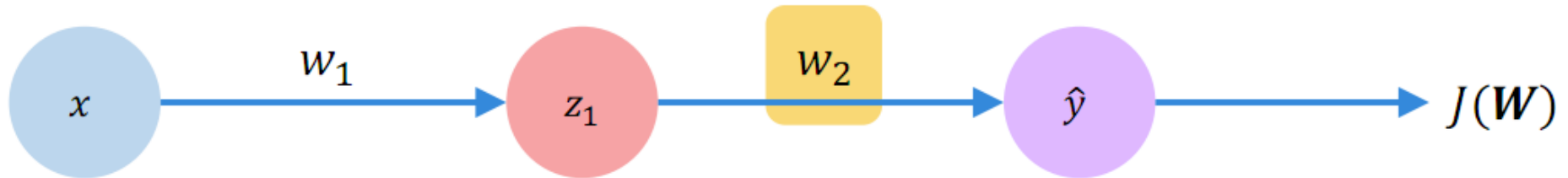
- Đánh giá sự thay đổi nhỏ ở một trọng số nào đó ảnh hưởng như thế nào tới hàm mục tiêu của mạng?



How does a small change in one weight (ex. w_2) affect the final loss $J(\mathbf{W})$?

Giải thuật lan truyền ngược

- Đánh giá sự thay đổi nhỏ ở một trọng số nào đó ảnh hưởng như thế nào tới hàm mục tiêu của mạng?

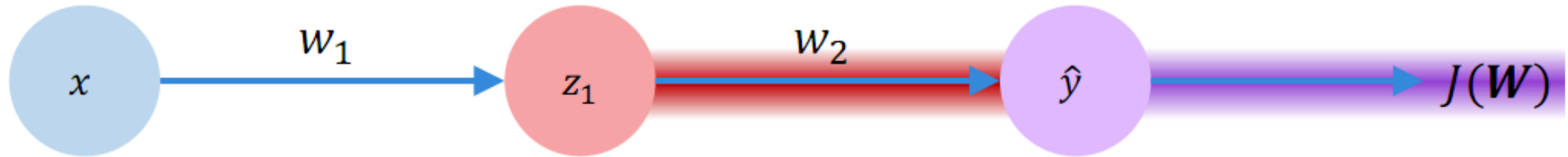


$$\frac{\partial J(W)}{\partial w_2} =$$

Let's use the chain rule!

Giải thuật lan truyền ngược

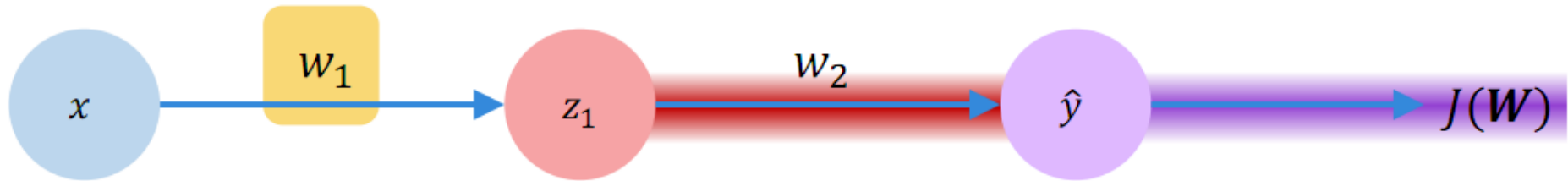
- Đánh giá sự thay đổi nhỏ ở một trọng số nào đó ảnh hưởng như thế nào tới hàm mục tiêu của mạng?



$$\frac{\partial J(W)}{\partial w_2} = \underbrace{\frac{\partial J(W)}{\partial \hat{y}}}_{\text{purple}} * \underbrace{\frac{\partial \hat{y}}{\partial w_2}}_{\text{red}}$$

Giải thuật lan truyền ngược

- Đánh giá sự thay đổi nhỏ ở một trọng số nào đó ảnh hưởng như thế nào tới hàm mục tiêu của mạng?

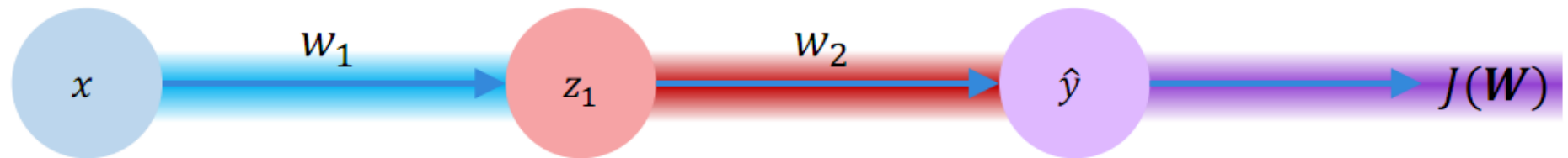


$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_1}$$

↑
↑
 Apply chain rule! Apply chain rule!

Giải thuật lan truyền ngược

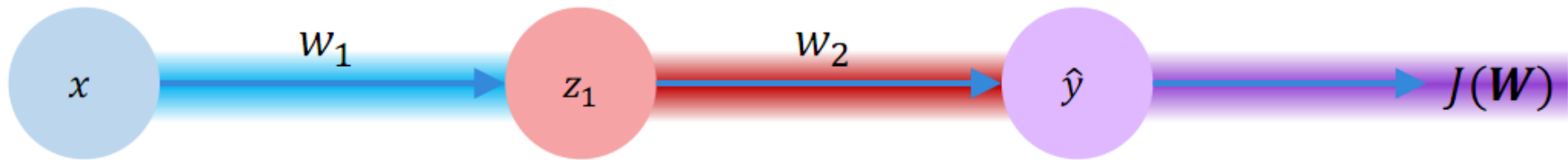
- Đánh giá sự thay đổi nhỏ ở một trọng số nào đó ảnh hưởng như thế nào tới hàm mục tiêu của mạng?



$$\frac{\partial J(W)}{\partial w_1} = \underbrace{\frac{\partial J(W)}{\partial \hat{y}}}_{\text{purple}} * \underbrace{\frac{\partial \hat{y}}{\partial z_1}}_{\text{red}} * \underbrace{\frac{\partial z_1}{\partial w_1}}_{\text{blue}}$$

Giải thuật lan truyền ngược

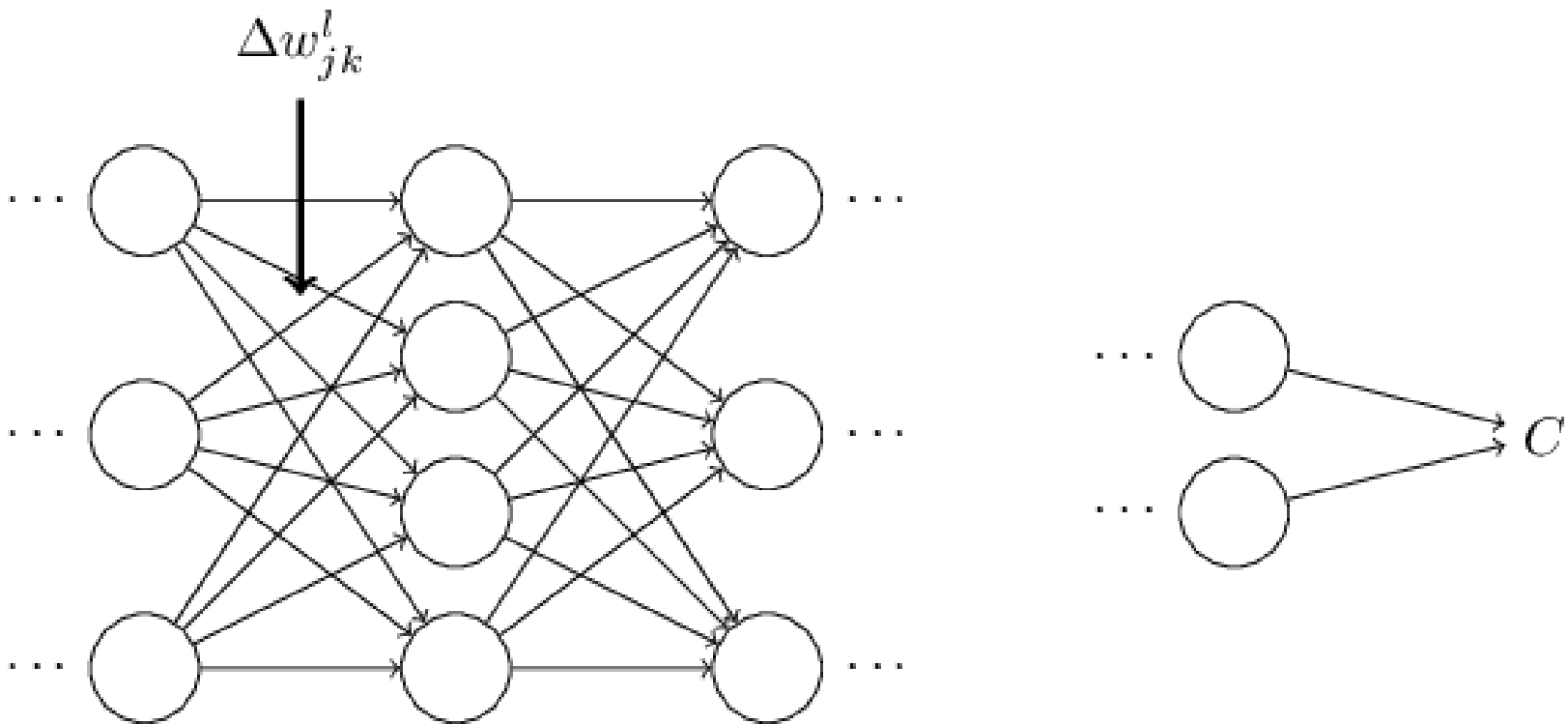
- Lặp lại cách ước lượng này cho tất cả các trọng số trong mạng dựa trên gradients đã tính ở các lớp trước



$$\frac{\partial J(W)}{\partial w_1} = \underbrace{\frac{\partial J(W)}{\partial \hat{y}}}_{\text{purple}} * \underbrace{\frac{\partial \hat{y}}{\partial z_1}}_{\text{red}} * \underbrace{\frac{\partial z_1}{\partial w_1}}_{\text{blue}}$$

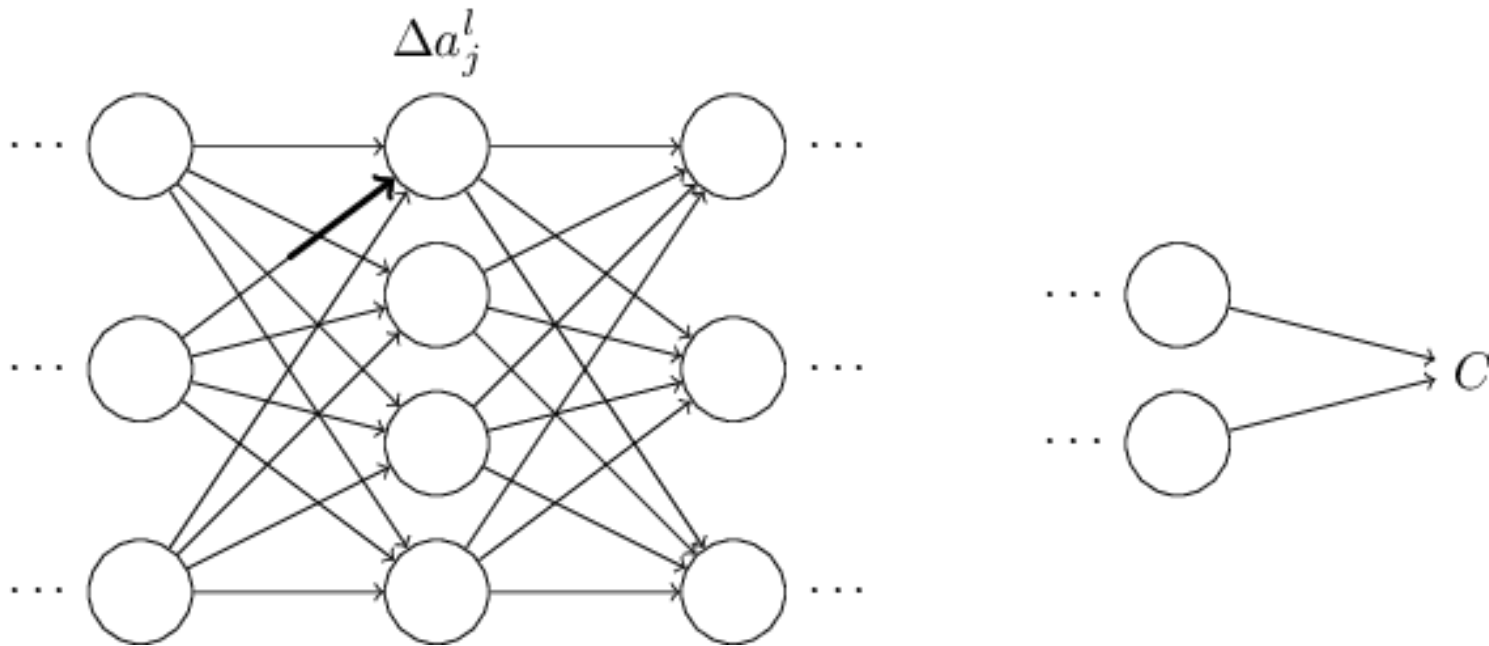
Giải thuật lan truyền ngược

- Giả sử có sự thay đổi nhỏ Δw_{jk}^l giá trị của trọng số w_{jk}^l ở lớp thứ l



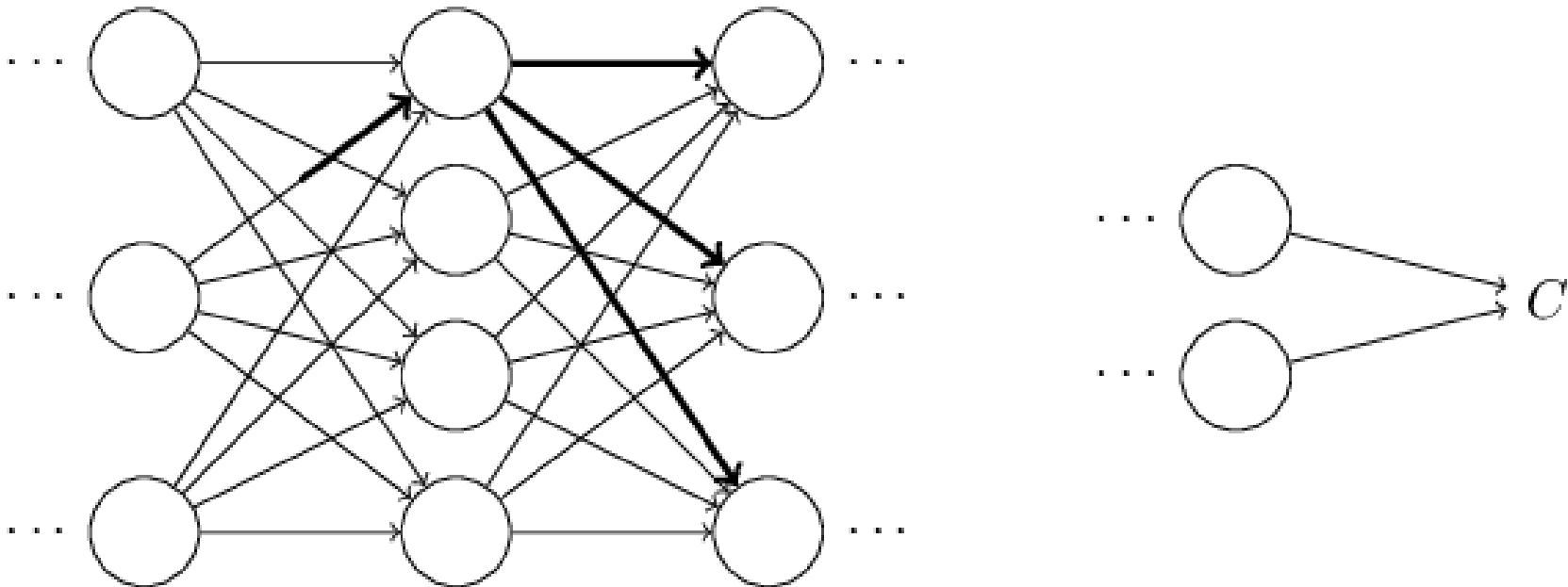
Giải thuật lan truyền ngược

- Sự thay đổi sẽ ảnh hưởng tới giá trị đầu ra của hàm kích hoạt nơ-ron tương ứng



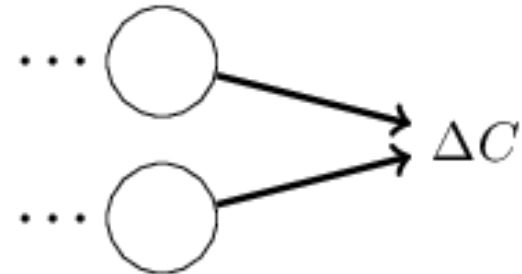
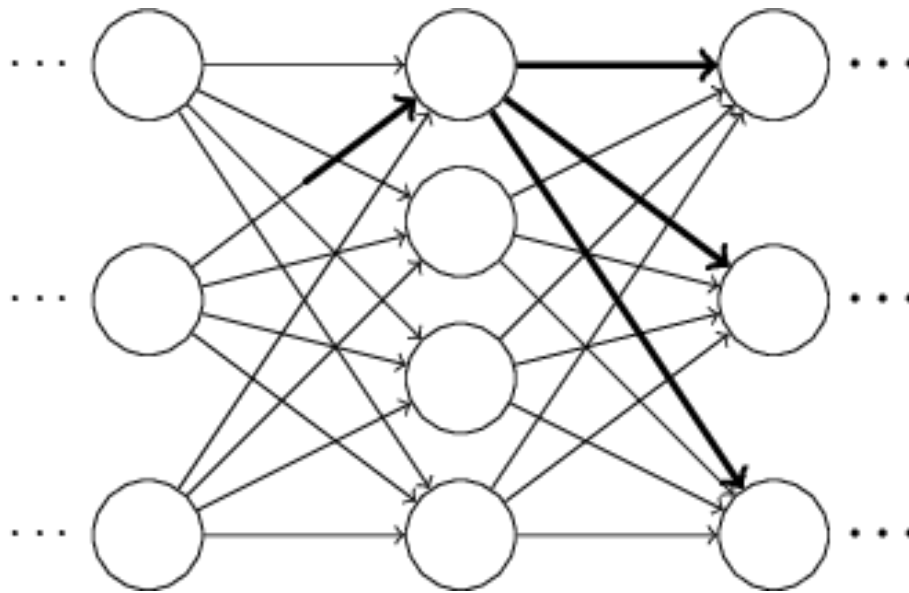
Giải thuật lan truyền ngược

- Và sau đó sẽ làm thay đổi giá trị đầu ra của tất cả các hàm kích hoạt ở các lớp ngay phía sau



Giải thuật lan truyền ngược

- Sự thay đổi sẽ lan truyền tiếp tới các lớp sau nữa và cuối cùng sẽ ảnh hưởng tới hàm mục tiêu, gây ra một lượng thay đổi ΔC



$$\Delta C \approx \frac{\partial C}{\partial w_{jk}^l} \Delta w_{jk}^l$$

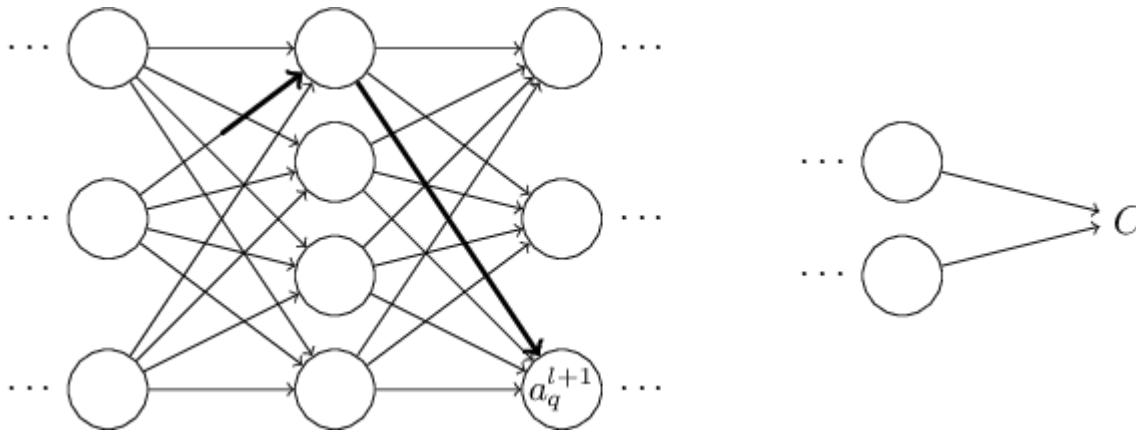
Giải thuật lan truyền ngược

- Như vậy có thể tính đạo hàm riêng của hàm mục tiêu đối với trọng số w_{jk}^l bằng cách theo dõi xem sự thay đổi của trọng số Δw_{jk}^l từng bước ảnh hưởng đến sự thay đổi của hàm mục tiêu ra sao
- Đầu tiên Δw_{jk}^l làm thay đổi hàm kích hoạt của nơ-ron tương ứng một lượng Δa_j^l

$$\Delta a_j^l \approx \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

Giải thuật lan truyền ngược

- Sự thay đổi của hàm kích hoạt a_j^l tiếp tục ảnh hưởng tới các hàm kích hoạt ở lớp kế tiếp



$$\Delta a_q^{l+1} \approx \frac{\partial a_q^{l+1}}{\partial a_j^l} \Delta a_j^l$$

$$\Delta a_q^{l+1} \approx \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

Giải thuật lan truyền ngược

- Sự thay đổi Δa_q^{l+1} tiếp tục ảnh hưởng các hàm kích hoạt phía sau và lan tới hàm mục tiêu.
- Ta có thể tưởng tượng ra một đường đi trong mạng từ w_{jk}^l tới hàm mục tiêu C , theo đó sự thay đổi Δw_{jk}^l sẽ dần dần ảnh hưởng tới các hàm kích hoạt trong đường đi và lan tới C . Giả sử đường đi chứa các hàm kích hoạt $a_j^l, a_q^{l+1}, \dots, a_n^{L-1}, a_m^L$ (L là số lớp của mạng). Khi đó ta có công thức:

$$\Delta C \approx \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_q^{l+1}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

Giải thuật lan truyền ngược

- Sự thay đổi Δa_q^{l+1} tiếp tục ảnh hưởng các hàm kích hoạt phía sau và lan tới hàm mục tiêu.
- Ta có thể tưởng tượng ra một đường đi trong mạng từ w_{jk}^l tới hàm mục tiêu C , theo đó sự thay đổi Δw_{jk}^l sẽ dần dần ảnh hưởng tới các hàm kích hoạt trong đường đi và lan tới C . Giả sử đường đi chứa các hàm kích hoạt $a_j^l, a_q^{l+1}, \dots, a_n^{L-1}, a_m^L$ (L là số lớp của mạng). Khi đó ta có công thức:

$$\Delta C \approx \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

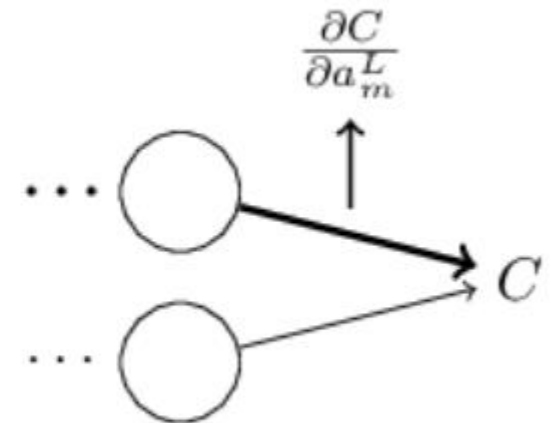
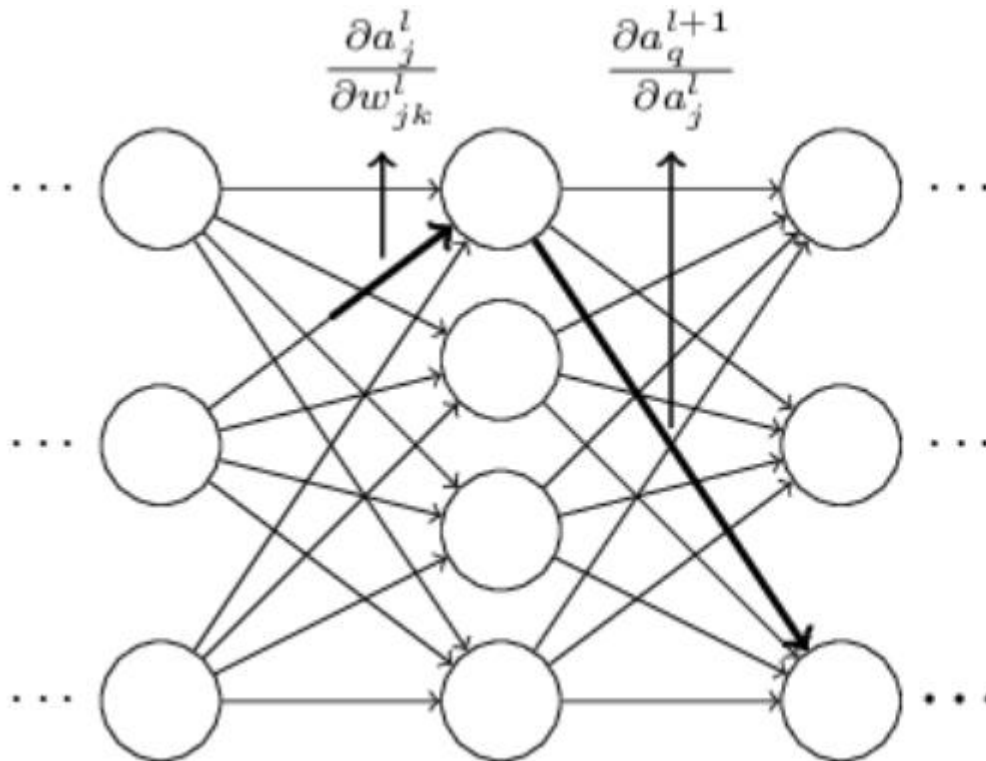
- Hiển nhiên có nhiều đường đi như vậy. Hàm mục tiêu sẽ bị thay đổi theo tất cả các đường đi:

$$\Delta C \approx \sum_{mnp \dots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

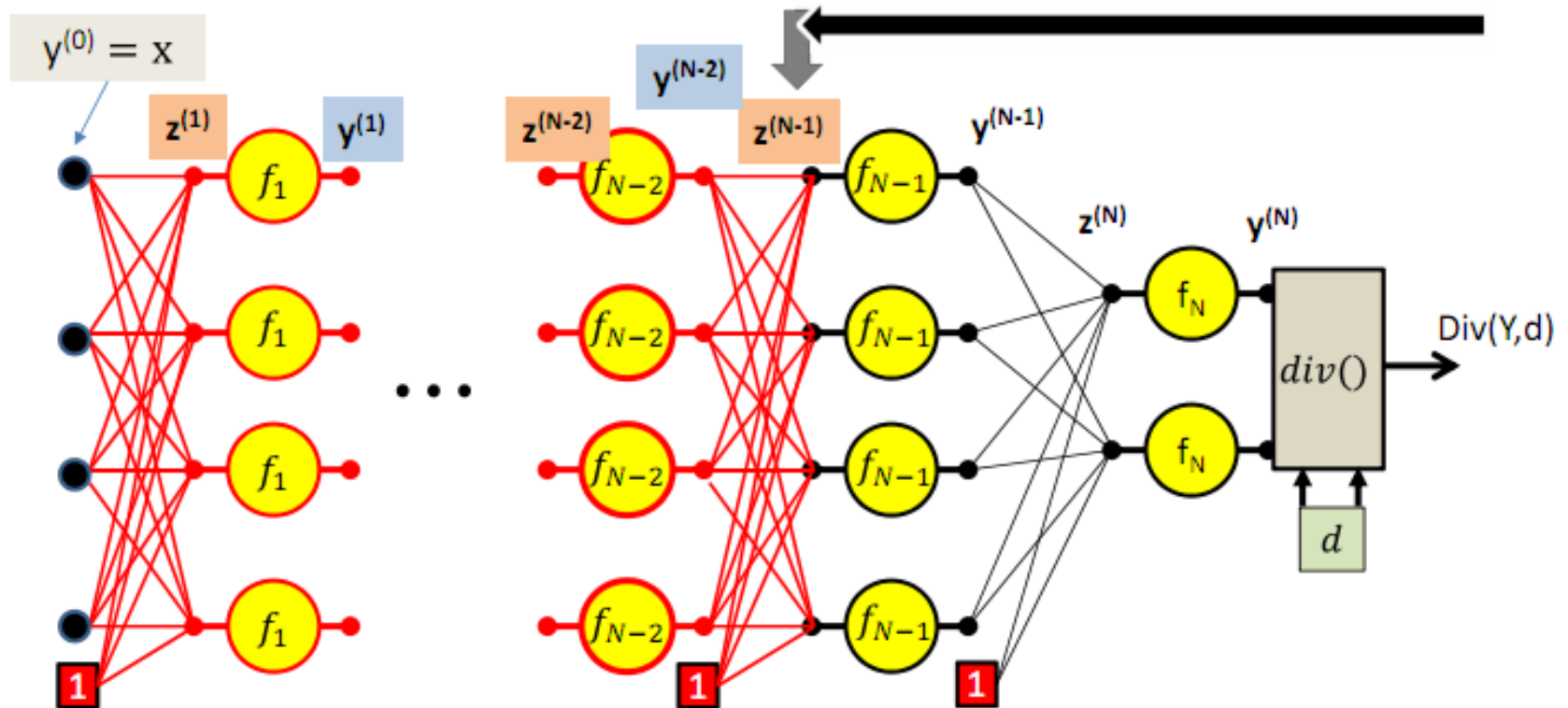
Giải thuật lan truyền ngược

- Cuối cùng ta thu được công thức:

$$\frac{\partial C}{\partial w_{jk}^l} = \sum_{mnp \dots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \dots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l}$$



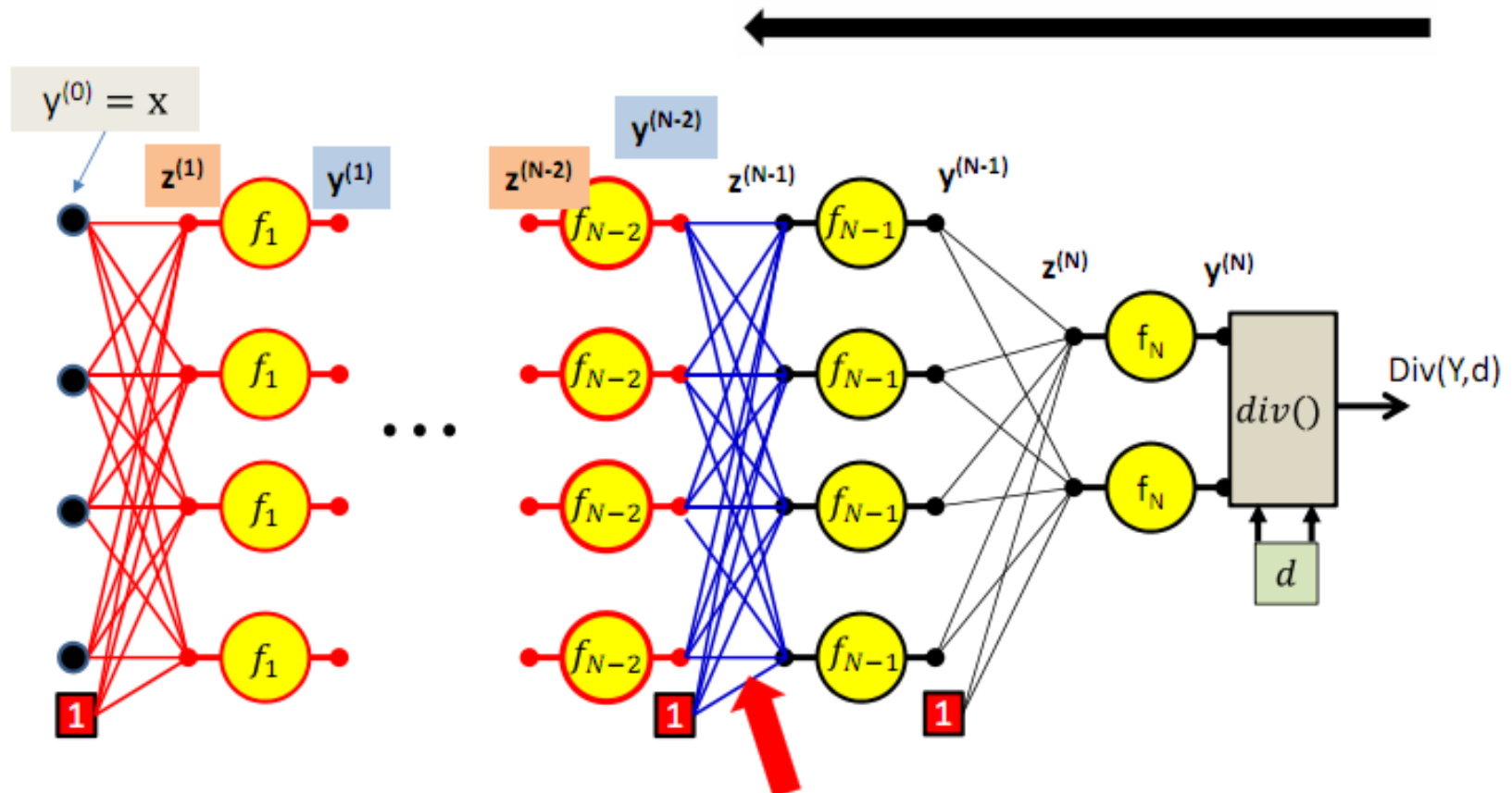
Giải thuật lan truyền ngược



We continue our way backwards in the order shown

$$\frac{\partial \text{Div}}{\partial z_i^{(N-1)}} = f'_{N-1}(z_i^{(N-1)}) \frac{\partial \text{Div}}{\partial y_i^{(N-1)}}$$

Giải thuật lan truyền ngược

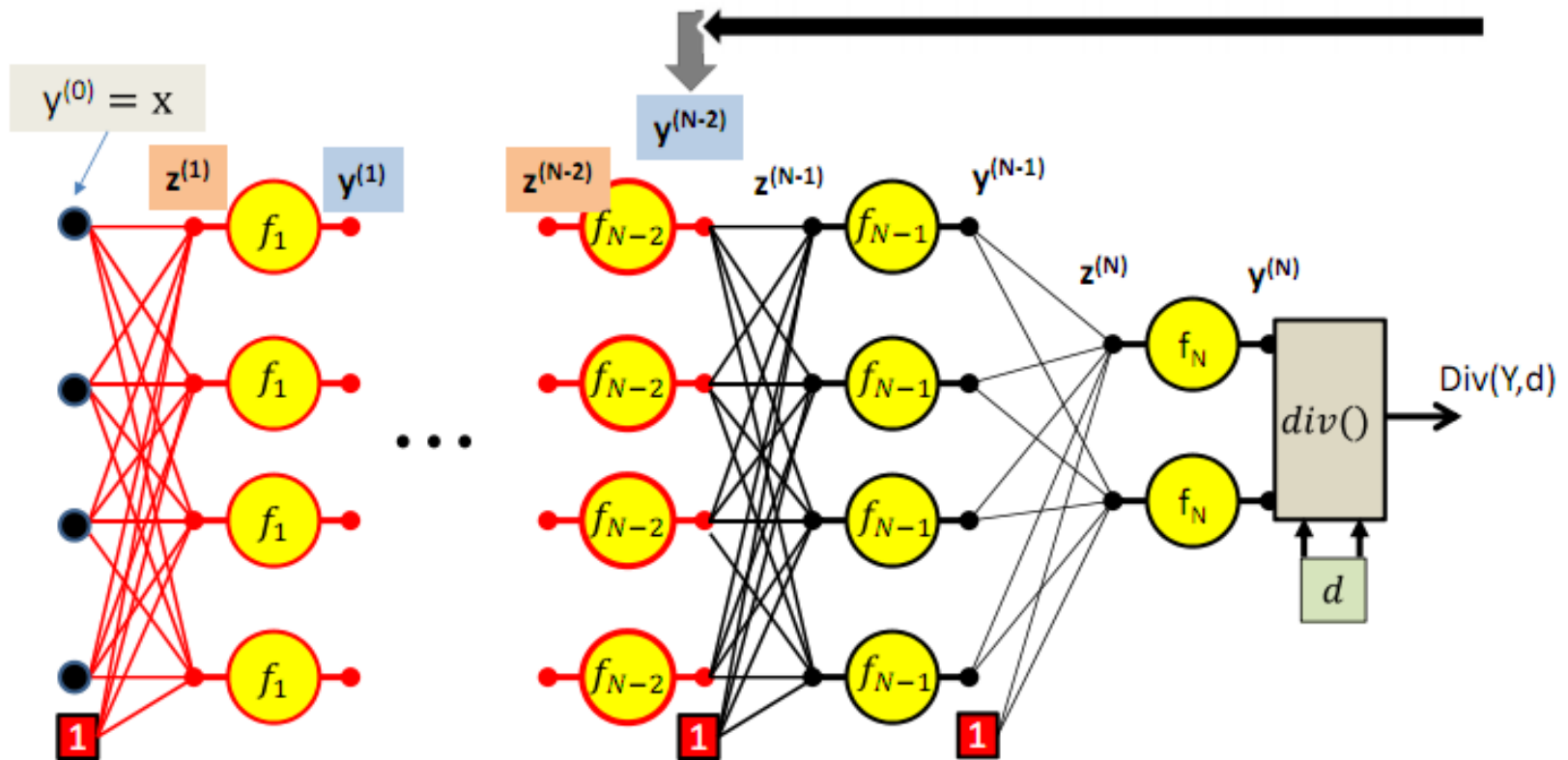


We continue our way backwards in the order shown

$$\frac{\partial Div}{\partial w_{ij}^{(N-1)}} = y_i^{(N-2)} \frac{\partial Div}{\partial z_j^{(N-1)}}$$

For the bias term $y_0^{(N-2)} = 1$

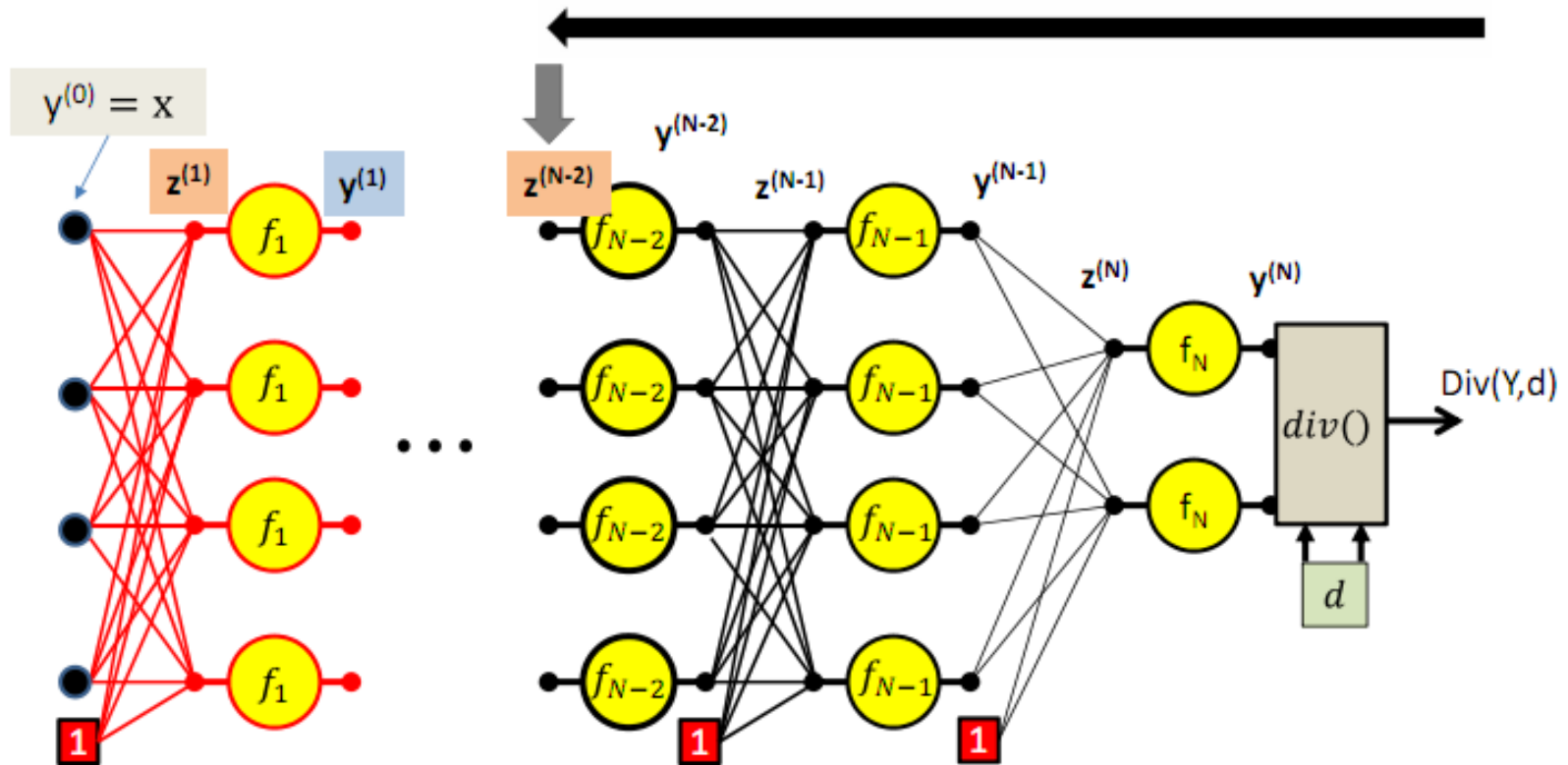
Giải thuật lan truyền ngược



We continue our way backwards in the order shown

$$\frac{\partial Div}{\partial y_i^{(N-2)}} = \sum_j w_{ij}^{(N-1)} \frac{\partial Div}{\partial z_j^{(N-1)}}$$

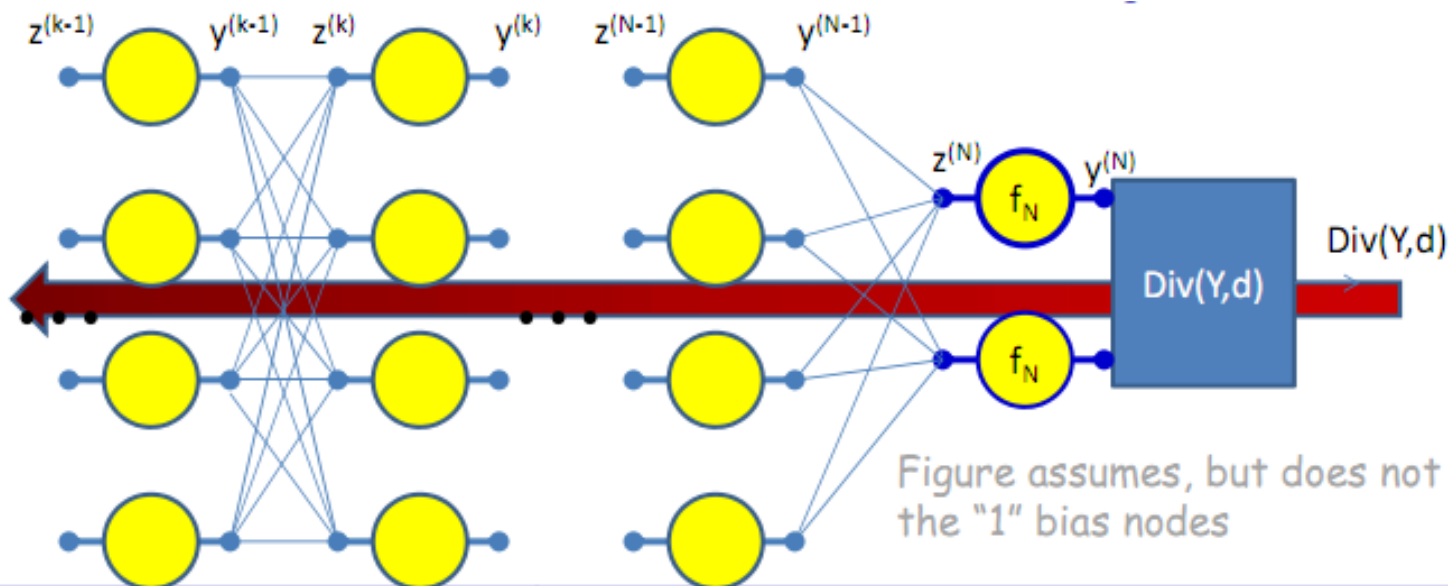
Giải thuật lan truyền ngược



We continue our way backwards in the order shown

$$\frac{\partial \text{Div}}{\partial z_i^{(N-2)}} = f'_{N-2}(z_i^{(N-2)}) \frac{\partial \text{Div}}{\partial y_i^{(N-2)}}$$

Giải thuật lan truyền ngược



Initialize: Gradient w.r.t network output

$$\frac{\partial Div}{\partial y_i} = \frac{\partial Div(Y, d)}{\partial y_i^{(N)}}$$

$$\frac{\partial Div}{\partial z_i^{(N)}} = f'_k(z_i^{(N)}) \frac{\partial Div}{\partial y_i^{(N)}}$$

For $k = N - 1..0$

For $i = 1: \text{layer width}$

$$\frac{\partial Div}{\partial y_i^{(k)}} = \sum_j w_{ij}^{(k+1)} \frac{\partial Div}{\partial z_j^{(k+1)}} \quad \frac{\partial Div}{\partial z_i^{(k)}} = f'_k(z_i^{(k)}) \frac{\partial Div}{\partial y_i^{(k)}}$$

$$\forall j \quad \frac{\partial Div}{\partial w_{ij}^{(k+1)}} = y_i^{(k)} \frac{\partial Div}{\partial z_j^{(k+1)}}$$