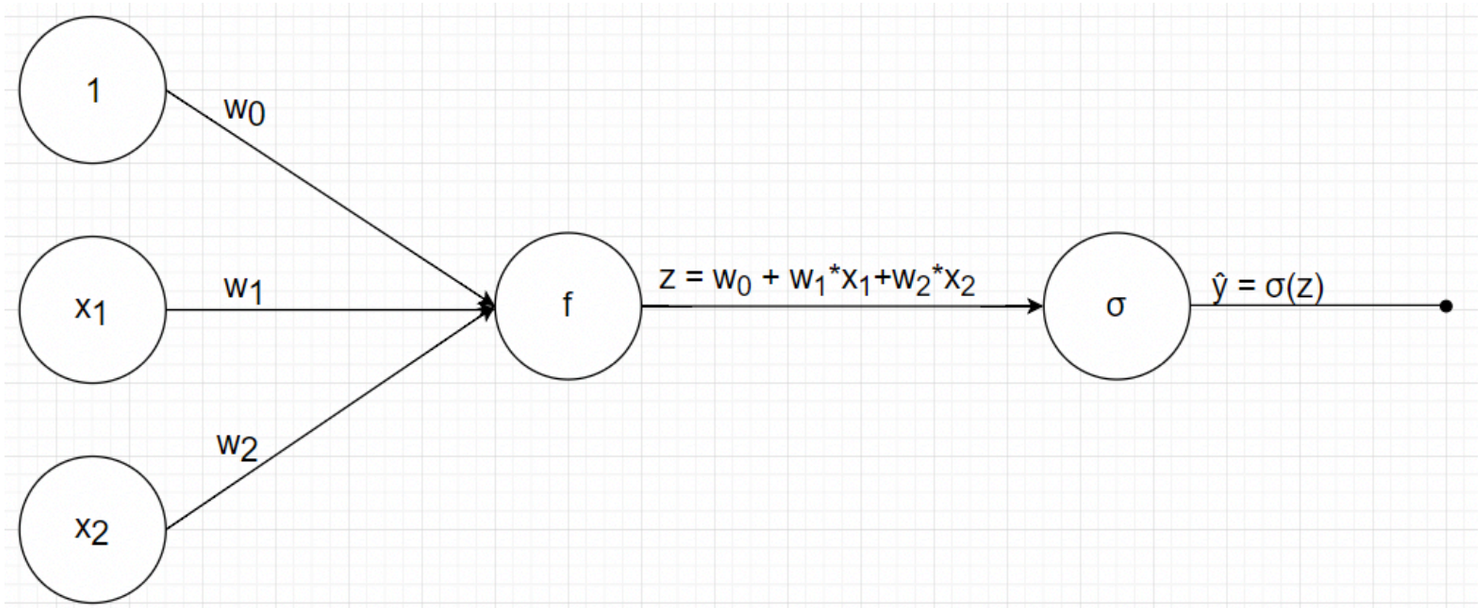
BACK PROPAGATION trong mạng neural

Người thực hiện: Trần Ngọc Bảo Duy - 51702091

Mạng neural đơn giản logistic regression



Đây là một mạng nơ-ron đơn giản sử dụng, 1 input và 1 output

với hàm z = w0 + w1*x1 + w2*x2 ta có thể viết là: z = ∑i(wi * xi)

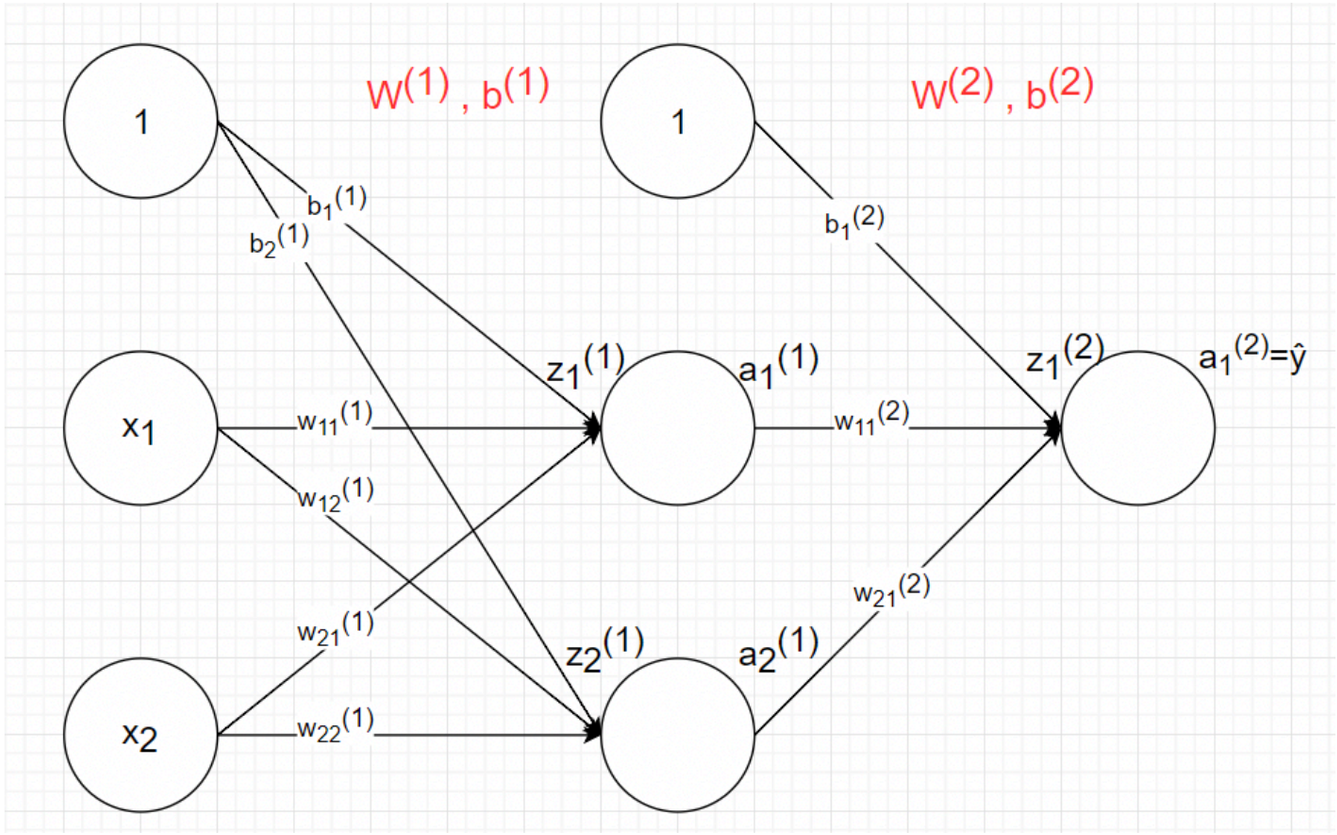
$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial J}{\partial A^{(1)}} * \frac{\partial A^{(1)}}{\partial Z^{(1)}} * \frac{\partial Z^{(1)}}{\partial W^{(1)}}$$

và **hàm σ activate function** là hàm sigmoid - output một không (1-0) để xác định có phải Y chúng ta cần không $\hat{y}=\sigma(z)$

wi là bộ tham số chúng ta cần xây dựng để:

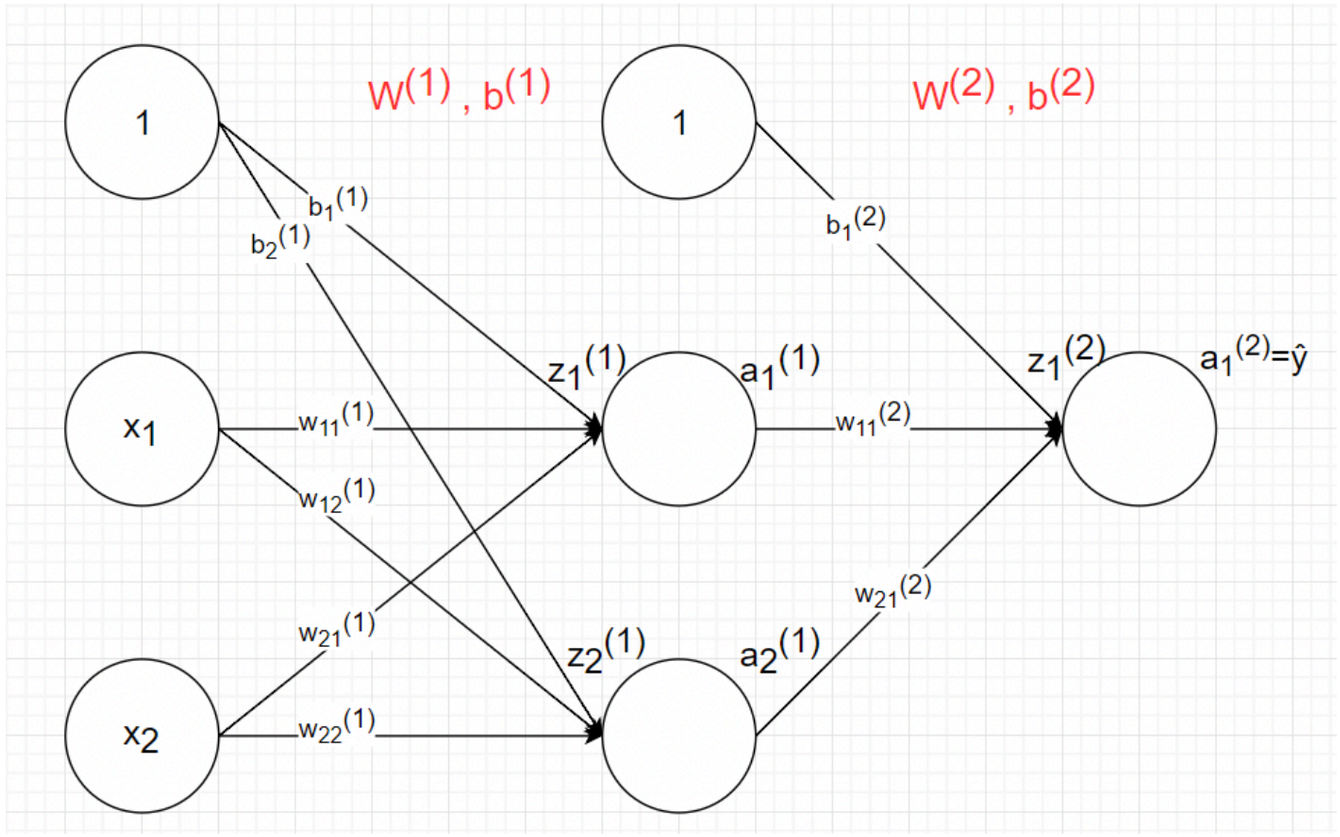
từ Xi (tham số đầu vào) chúng ta có thể predict ra được Yi (kết quả dự đoán - output) cần thiết

Mạng neural 2 layer



- Mô hình trên có 2 layer (số lượng layer của mô hình không tính input layer)
- Mô hình: 2-2-1, nghĩa là 2 node trong input layer, 1 hidden layer có 2 node và output layer có 1 node.
- Input layer và hidden layer luôn thêm node 1 để tính bias cho layer sau, nhưng không tính vào số lượng node trong layer
- Ở mỗi node trong hidden layer và output layer đều thực hiện 2 bước: tính tổng linear và áp dụng activation function.
- Các hệ số và bias tương ứng được ký hiệu như trong hình

Mạng neural 2 layer



- $z_1^{(1)} = b_1^{(1)} + x_1 * w_{11}^{(1)} + x_2 * w_{21}^{(1)}$
- $a_1^{(1)} = \sigma(z_1^{(1)})$
- $z_2^{(1)} = b_2^{(1)} + x_1 * w_{12}^{(1)} + x_2 * w_{22}^{(1)}$
- $a_2^{(1)} = \sigma(z_2^{(1)})$
- $z_1^{(2)} = b_1^{(2)} + a_1^{(1)} * w_{11}^{(2)} + a_2^{(1)} * w_{21}^{(2)}$
- $\hat{y} = a_1^{(2)} = \sigma(z_1^{(2)})$

- Cũng như mô hình logistic regression chúng ta tính gradient descend để cập nhật w cũng phải thông qua từng layer và các node liên quan, vì vậy không thể tính trực tiếp mà phải tính ở dạng chain rule. Hay có thể viết gọn lại theo công thức bên dưới

$$
\begin{aligned}
Z^{(1)} &= X * W^{(1)} + b^{(1)} \\
A^{(1)} &= \sigma(Z^{(1)}) \\
Z^{(2)} &= A^{(1)} * W^{(2)} + b^{(2)} \\
\hat{Y} &= A^{(2)} = \sigma(Z^{(2)})
\end{aligned}
$$

Mạng neural 2 layer (Giải thích)

$$
\begin{aligned}
Z^{(1)} &= X * W^{(1)} + b^{(1)} \\
A^{(1)} &= \sigma(Z^{(1)}) \\
Z^{(2)} &= A^{(1)} * W^{(2)} + b^{(2)} \\
\hat{Y} &= A^{(2)} = \sigma(Z^{(2)})
\end{aligned}
$$

- Giá trị predict sẽ là giá trị của từng layer khi back propagration
- Y = activation function của Z tại layer cuối
- Z layer cuối lại tính theo CT: Activation function của layer trước đó nhân với hệ số W và bias có nghĩa là: z tại layer

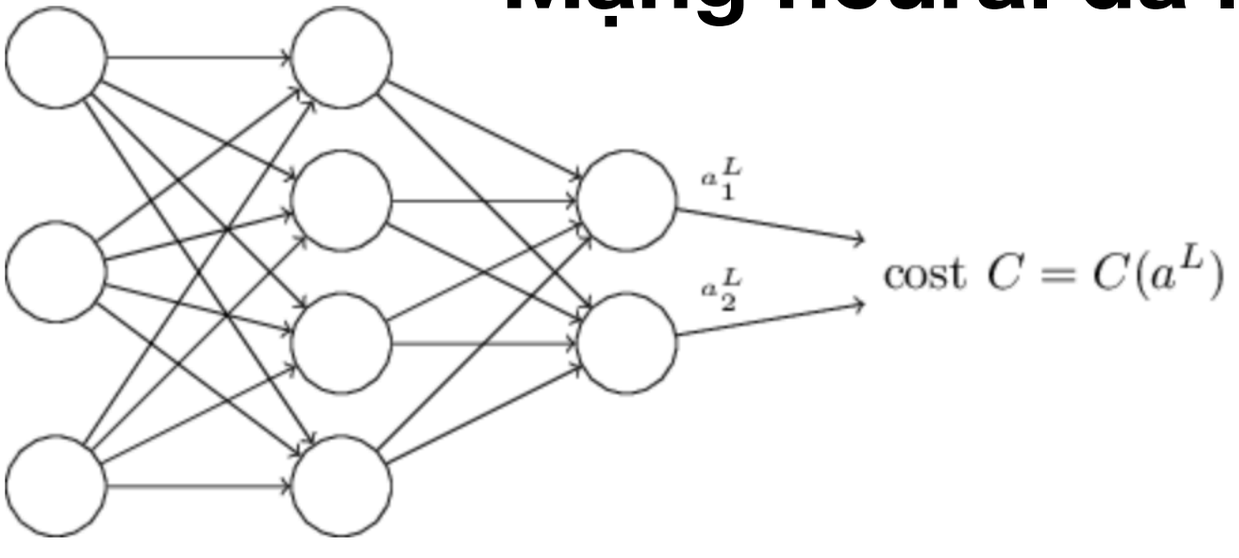
$$z(l+1) = a(l) * w(l+1) + b(l+1)$$

- Và cứ tiếp tục như thế cho đến Z đầu tiên của mạng neural được tính bằng X thay vì activation function

$$z(1) = X * w(2) + b(1)$$

(*) layer đầu tiên

Mạng neural đa layer (L layer)



$$C = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2,$$

Ở đây chúng ta gọi C là Cost function mà chúng ta dùng để đánh giá mức độ predict chính xác của mạng neural C càng nhỏ thì mức độ chính xác càng cao (ở đây chúng ta dùng mean square error để đánh giá) tức là dùng giá trị thật trừ đi giá trị predict sau đó lấy trung bình hay gọi là MSE vì nó dễ dàng tính khi đã có các giá trị cụ thể

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Trong mạng neural giữa các layer với nhau chúng ta sẽ tính loss function của chúng để cuối cùng tìm ra được cost function cho cả mạng neural

Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{j\,k}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

Các công thức cần chú ý để tính được cost function, chúng ta biết rằng khi tính loss function ở bài linear regression thì chúng ta cần phải có gradient descent để cập nhật lại các trọng số w dựa vào đạo hàm theo w, để xây dựng phương trình đánh giá, phân loại theo công thức, nhưng không thể nào tính trực tiếp cho đạo hàm của C function trên w của từng node trong từng layer vì vậy người ta đã thêm vào các đạo hàm trung gian (chain rule) để có thể tính được đạo hàm này cũng là công thức BP4

$$w^l_{jk} = w^l_{jk} \text{ - learning rate * } \frac{\partial C}{\partial w^l_{jk}}$$

Summary: the equations of backpropagation

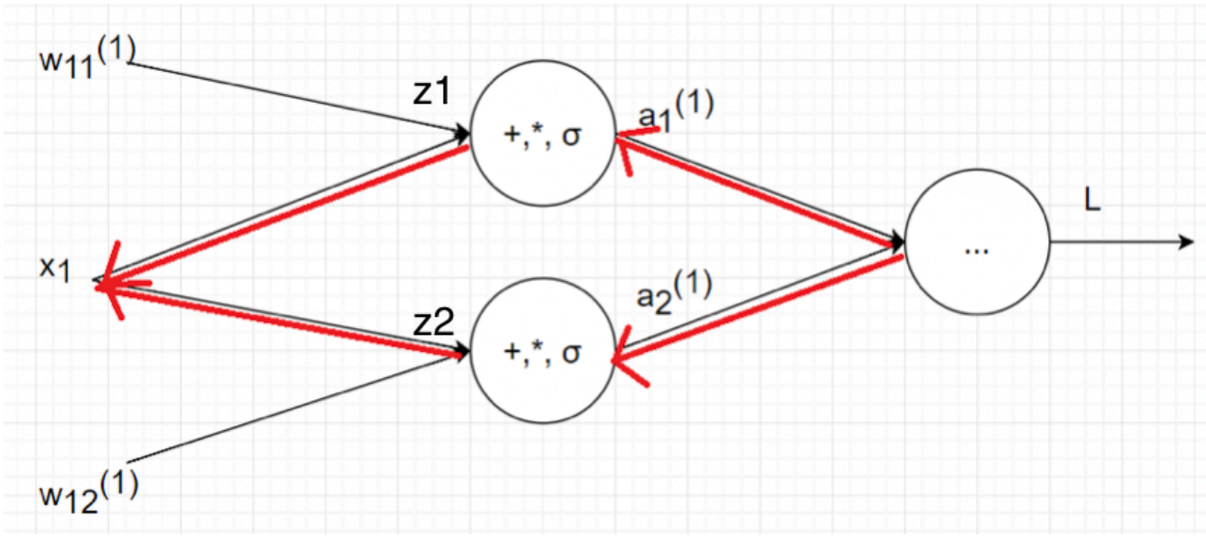
$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{j\,k}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

Ta có thể thấy công thức đầu tiên chính là hàm loss dành cho LAYER cuối cùng được viết dưới dạng nhân ma trận, như hình bên dưới chúng ta cần backpropagation từ L về activation function cuối cùng sau đó qua hàm linear z đến các node của layer L-1.



$$\delta^L_j = \frac{\partial C}{\partial a^L_j} \sigma'(z^L_j).$$

Với

$\frac{\partial C}{\partial a^L_j}$ đánh giá sự thay đổi của hàm mất mát tương ứng với layer đó

$\sigma'(z^L_j)$ đánh giá sự thay đổi giá trị của activation function của layer tương ứng

Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{j\,k}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

Công thức BP2 là hàm loss của các layer bên trong mạng neural:

Activation của layer hiện tại sẽ = (hệ số W* loss function) của layer kế tiếp * với đạo hàm của activation tại z của layer đó

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l),$$

vậy là độ lỗi - hay hàm loss layer phía trước sẽ được tính dựa vào hàm loss của layer phía sau cũng với các hệ số W

Và cứ như thế ta backward cho tới layer đầu tiên

Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

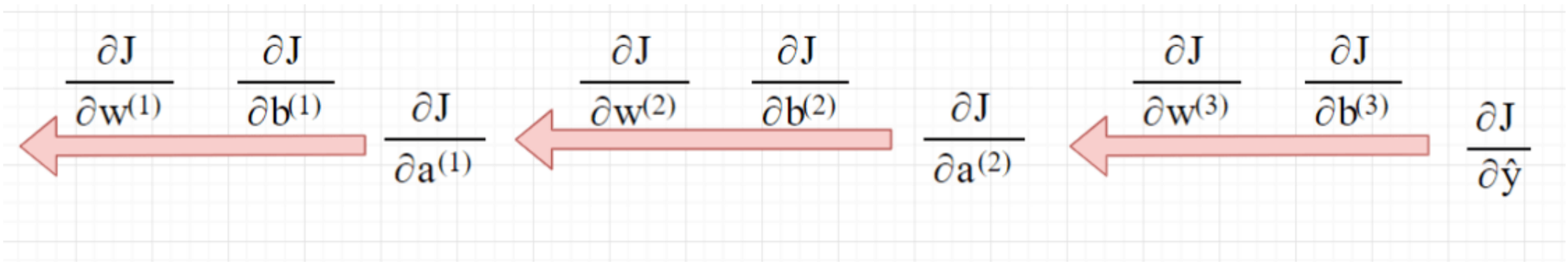
$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{j\,k}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

Công thức BP3 chính là đạo hàm của C theo hệ số bias

công thức này sẽ bằng với hàm loss của từng node của 1 layer cố định ứng với bias tại layer đó $\frac{\partial C}{\partial b^l_j} = \delta^l_j.$

Tại mỗi layer chúng ta cần khởi tạo hệ số bias để không cho mô hình tiến về 0, vì khi đó mô hình sẽ không còn ý nghĩa vì vậy chúng ta cần phải tính luôn đạo hàm cho bias rồi sau đó gộp vào w để quá trình backward diễn ra tốt đẹp



Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{j\,k}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

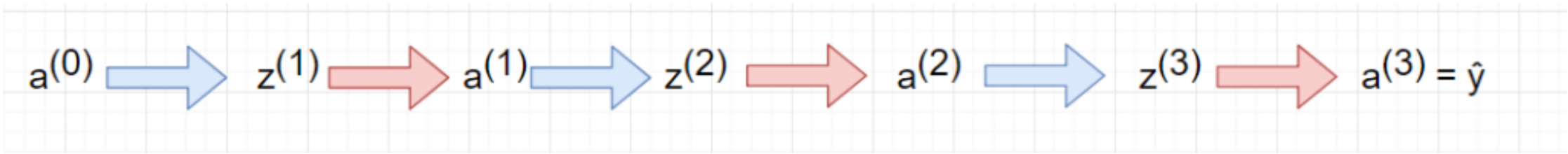
Công thức BP4 chính là đạo hàm của C theo w là công thức chúng ta cần để cập nhật các hệ số w theo từng layer

công thức này sẽ bằng hàm activation của node k layer phía trước * công thức BP3

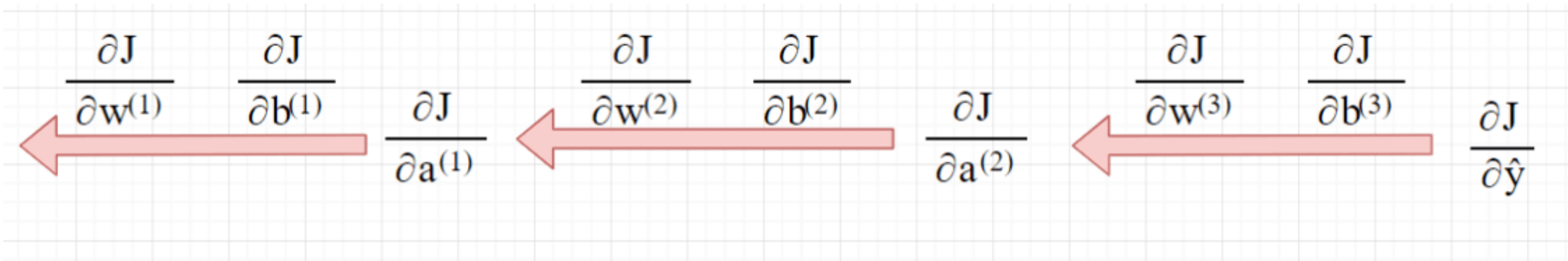
$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial J}{\partial A^{(1)}} * \frac{\partial A^{(1)}}{\partial Z^{(1)}} * \frac{\partial Z^{(1)}}{\partial W^{(1)}}$$

Như công thức trên này thì khi chúng ta đạo hàm theo W phải thông qua các activation và Z mới có thể tìm ra được đạo hàm của C theo W tại từng node trong từng layer

Tổng kết



Quá trình feed forward trong mạng neural với 2 layer



Quá trình backwark trong mạng neural với 2 layer

Như vậy chúng ta sẽ tính theo các công thức và đạo hạm đã trình bày phía trên để có thể cập nhật các hệ số W, giúp mô hình có kết quả output tốt hơn, nhưng bên cạnh đó việc chọn các hàm activation giữa các layer cũng quan trọng không kém

Reference

Sách Deep Learning cơ bản của thầy Nguyễn Thanh Tuấn

Khoá học Machine Learning của thầy Vũ Hữu Tiệp
https://machinelearningcoban.com/2017/02/24/mlp/

Back propagation trong mạng neural
http://neuralnetworksanddeeplearning.com/chap2.html?fbclid=IwAR3biprzq-_QAi7_s4wel9hUC-askWsYpZ98PjgU4ZSwYPMqUoj6rkjAoro

Và các tài liệu thầy đã reference cho em