[HAY] Tóm tắt Neural Network

Tóm tắt quá trình hoạt động của mạng neural network

1. **Dữ liệu đầu vào**: Dữ liệu đầu vào là các cặp giá trị (Learn, Sleep) và nhãn tương ứng (Is_Pass). Mỗi mẫu dữ liệu được biểu diễn bởi hai đặc trưng: Learn và Sleep.
   Mục tiêu là dự đoán nhãn Is_Pass dựa trên các đặc trưng này.
2. **Layer input**: Layer input có 2 node tương ứng với 2 đặc trưng của dữ liệu đầu vào: Learn và Sleep.
3. **Layer ẩn**: Trong một mạng neural network đơn giản, có thể sử dụng một hoặc nhiều layer ẩn. Trong trường hợp này, chúng ta sẽ sử dụng một layer ẩn.
   Số node trong layer ẩn là một siêu tham số mà bạn có thể điều chỉnh để tối ưu hiệu suất mạng. Trong ví dụ này, chúng ta có thể chọn 4 node cho layer ẩn.
4. **Layer đầu ra**: Layer đầu ra có 1 node, tương ứng với dự đoán Is_Pass.
5. **Quá trình Feedforward**:Mỗi mẫu dữ liệu được đưa vào mạng neural network.
   Mỗi node trong layer input là một đầu vào cho mạng. Các trọng số và hệ số bias được áp dụng và kết hợp để tạo ra giá trị tại các node trong layer ẩn.
   Giá trị đầu ra từ layer ẩn được truyền qua hàm kích hoạt (ví dụ: sigmoid) để tạo ra giá trị tại các node trong layer đầu ra.
   Giá trị dự đoán từ layer đầu ra được sử dụng để tính toán hàm mất mát (ví dụ: MSE) với nhãn thực tế.
6. **Quá trình Backpropagation**:Hàm mất mát được sử dụng để tính toán độ lỗi của mạng.
   Độ lỗi được lan truyền ngược từ layer đầu ra trở lại layer ẩn, dựa trên đạo hàm của hàm kích hoạt và các trọng số của mạng.Các trọng số và hệ số bias được cập nhật bằng cách sử dụng gradient descent để giảm thiểu độ lỗi.
   Quá trình này được lặp lại cho đến khi độ lỗi giảm đến mức chấp nhận được hoặc đạt đến số lần lặp được chỉ định (epochs).
7. **Lặp qua dữ liệu**:Mỗi lần lặp lại (epoch), mỗi mẫu dữ liệu trong tập huấn luyện được sử dụng để cập nhật trọng số của mạng.

⇨ Quá trình này tiếp tục cho đến khi mạng hội tụ hoặc cho đến khi đạt đến số lần lặp được chỉ định.
Quá trình này được lặp lại cho đến khi mạng hội tụ hoặc cho đến khi đạt đến số lần lặp được chỉ định. Sau khi mạng đã được huấn luyện, bạn có thể sử dụng nó để dự đoán nhãn cho các dữ liệu mới.

Kích thước các ma trận liên quan , ví dụ gồm các lớp :

- Input layer : 2 node
- Hidden layer 1 : 3 node
- Hidden layer 2 : 4 node
- Output layer : 1 node

1. Layer Input:
   - Ma trận trọng số $W^{(1)}$ có kích thước 2 * 3 (2 node input và 2 node ẩn)
   - Hệ số bias: $b^{(1)}$ có kích thước 1 * 3
   - $Z^{(1)}$ có kích thước n * 3 (với n là số mẫu trong tập dữ liệu)
   - $A^{(1)}$ có kích thước n * 3
2. Layer ẩn:
   - Ma trận trọng số $W^{(2)}$ có kích thước 3 * 4 (3 node ẩn và 4 node ẩn tiếp theo)
   - Hệ số bias: $b^{(2)}$ có kích thước 1 * 4
   - $Z^{(2)}$ có kích thước n * 4
   - $A^{(2)}$ có kích thước n * 4

3. Layer Output:
    - Ma trận trọng số $W^{(3)}$ có kích thước 4 * 1 (4 node ẩn và 1 node output)
    - Hệ số bias: $b^{(3)}$ có kích thước 1 * 4
    - $Z^{(3)}$ có kích thước n * 1
    - $A^{(3)}$ có kích thước n * 1

⇨ Trong đó n là số mẫu dữ liệu có trong tập dữ liệu

Quá trình hoạt động của neural network bằng đại số tuyến tính . Với số lớp trên .

1. Feedforward:
    - Tính cho layer ẩn thứ 1:
        - $Z^{(1)} = X.W^{(1)} + b^{(1)}$
        - $A^{(1)} = \sigma(Z^{(1)})$
    - Tương tự, tính $Z^{(2)}$ và $A^{(2)}$ cho layer ẩn thứ 2:
        - $Z^{(2)} = A^{(1)}.W^{(2)} + b^{(2)}$
        - $A^{(2)} = \sigma(Z^{(2)})$
    - Cuối cùng, tính $Z^{(3)}$ và $A^{(3)}$ cho layer output:
        - $Z^{(3)} = A^{(2)}.W^{(3)} + b^{(3)}$
        - $A^{(3)} = \sigma(Z^{(3)})$
2. Backpropagation
    - Tính độ lỗi của layer output: $\delta^{(3)}$:
        - $\delta^{(3)} = A^{(3)} - Y$
    - Tính độ lỗi của layer ẩn thứ 2: $\delta^{(2)}$:
        - $\delta^{(2)} = \left(\delta^{(3)} * \left(W^{(3)}\right)^T\right) * \sigma'(Z^{(2)})$
    - Tính độ lỗi của layer ẩn thứ 1: $\delta^{(1)}$:
        - $\delta^{(1)} = \left(\delta^{(2)} * \left(W^{(2)}\right)^T\right) * \sigma'(Z^{(1)})$
    - Cập nhật trọng số và bias:
        - Cập nhật trọng số và bias cho layer output:
            - $W^{(3)} = W^{(3)} - \alpha * \left(A^{(2)}\right)^T * \delta^{(3)}$
            - $b^{(3)} = b^{(3)} - \alpha * \sum \delta^{(3)}$
        - Cập nhật trọng số và bias cho layer ẩn thứ 2 :
            - $W^{(2)} = W^{(2)} - \alpha * \left(A^{(1)}\right)^T * \delta^{(2)}$
            - $b^{(2)} = b^{(2)} - \alpha * \sum \delta^{(2)}$
        - Cuối cùng, cập nhật trọng số và bias cho layer ẩn thứ 1 :
            - $W^{(1)} = W^{(1)} - \alpha * (X)^T * \delta^{(1)}$
            - $b^{(1)} = b^{(1)} - \alpha * \sum \delta^{(1)}$

⇨ Trong đó
- α : learning rate (tốc độ học)
- σ : sigmoid activation function
- σ' : đạo hàm hàm sigmoid
- X là ma trận đầu vào , Y là ma trận nhãn thực tế
- $\delta^{(i)}$: độ lỗi của layer thứ i

Nói thêm về Neural network

1. **Số lớp ẩn và số node của mỗi lớp ẩn trong một mạng neural network phụ thuộc vào nhiều yếu tố khác nhau, bao gồm:**

- Phức tạp của bài toán: Độ phức tạp của bài toán có thể ảnh hưởng đến cần phải có bao nhiêu lớp ẩn và số node trong mỗi lớp. Các bài toán phức tạp thường cần nhiều lớp ẩn và nhiều node hơn để mạng có khả năng học được các mô hình phức tạp.
- Số lượng và đa dạng của dữ liệu đầu vào: Số lượng và đa dạng của các đặc trưng đầu vào cũng có thể ảnh hưởng đến cấu trúc của mạng neural network. Dữ liệu đa dạng hoặc có nhiều đặc trưng có thể đòi hỏi một số lượng lớn hơn các node trong các lớp ẩn để mạng có thể học được các biểu diễn phức tạp của dữ liệu.
- Kích thước của tập dữ liệu: Kích thước của tập dữ liệu đầu vào cũng có thể ảnh hưởng đến cấu trúc của mạng. Trong một số trường hợp, khi có ít dữ liệu, việc sử dụng một mạng quá phức tạp có thể dẫn đến overfitting. Ngược lại, khi có nhiều dữ liệu, có thể sử dụng các mạng lớn hơn mà không gặp vấn đề overfitting.
- Tài nguyên tính toán: Số lượng node trong mỗi lớp ẩn cũng phụ thuộc vào tài nguyên tính toán có sẵn. Mạng lớn với nhiều node yêu cầu nhiều tài nguyên tính toán hơn.
- Thử nghiệm và điều chỉnh: Thường xuyên cần thử nghiệm với các kiến trúc mạng khác nhau và điều chỉnh số lượng lớp ẩn và số node trong mỗi lớp để tìm ra cấu trúc hiệu quả nhất cho bài toán cụ thể.

2. **Sử dụng Neural network**

Mạng neural network có thể áp dụng cho nhiều loại giá trị dự đoán khác nhau và chỉ cần thay đổi hàm kích hoạt (activation function) để phù hợp với loại giá trị y mà bạn muốn dự đoán.

- **Hàm kích hoạt tuyến tính (Linear Activation Function)**: Thích hợp cho các bài toán dự đoán giá trị số, không giới hạn về phạm vi giá trị.
- **Hàm kích hoạt nhị phân (Sigmoid Activation Function)**: Thích hợp cho các bài toán phân loại nhị phân, với đầu ra dự đoán gần 0 hoặc 1.
  Hàm kích hoạt softmax (Softmax Activation Function): Thích hợp cho các bài toán phân loại nhiều lớp (multiclass classification), với đầu ra dự đoán là xác suất của mỗi lớp và tổng các xác suất bằng 1.
- **Hàm kích hoạt ReLU (Rectified Linear Unit)**: Thường được sử dụng cho các mạng neural network sâu (deep neural networks) với hiệu suất tốt, đặc biệt trong các mô hình học sâu (deep learning).

⇨ Và còn nhiều hàm kích hoạt khác nữa mà bạn có thể sử dụng tùy thuộc vào bài toán cụ thể và đặc tính của dữ liệu. Điều này làm cho mạng neural network trở thành một công cụ mạnh mẽ và linh hoạt cho nhiều loại bài toán khác nhau trong học máy.

3. **Số lượng node đầu ra trong mạng neural network**

Số lượng node đầu ra trong mạng neural network phụ thuộc vào loại bài toán mà bạn đang giải quyết và yêu cầu cụ thể của tập dữ liệu. Dưới đây là một số ví dụ cụ thể:

1. **Phân loại nhị phân (Binary Classification):**Trong trường hợp này, chỉ có hai lớp được phân loại, ví dụ: "có bệnh" hoặc "không có bệnh", "spam" hoặc "không spam". Do đó, bạn chỉ cần một node ở output layer để đưa ra dự đoán nhị phân.
2. **Phân loại đa lớp (Multi-Class Classification):** Khi bạn phải phân loại dữ liệu thành nhiều lớp hơn hai, ví dụ: phân loại chủ đề của email thành "thể thao", "chính trị", "kinh doanh", "giải trí", và "khoa học". Trong trường hợp này, số lượng node đầu ra sẽ bằng số lượng lớp cần phân loại.

3. **Dự đoán giá trị số (Regression):** Trong bài toán dự đoán giá trị số, ví dụ: dự đoán giá nhà dựa trên diện tích và số phòng ngủ. Trong trường hợp này, bạn cũng chỉ cần một node ở output layer để dự đoán giá trị liên tục.
4. **Một số bài toán đặc biệt khác:** Có những bài toán yêu cầu sử dụng nhiều node đầu ra hơn, như bài toán Object Detection trong Computer Vision, nơi mỗi node đại diện cho một loại đối tượng có thể xuất hiện trong ảnh.

⇨ Tóm lại, số node đầu ra phụ thuộc vào yêu cầu cụ thể của bài toán và loại dữ liệu bạn đang làm việc.

4. Softmax activation function cho bài toán Phân loại nhiều lớp (Multi-Class Classification) (Nhiều nhãn)

- Trong phân loại đa lớp (Multi-Class Classification), hàm kích hoạt thích hợp cho lớp đầu ra là softmax. Hàm softmax thường được sử dụng để chuyển đổi các điểm số đầu ra thành các xác suất. Nó là một dạng nâng cao của hàm sigmoid, được thiết kế để xử lý các tác vụ phân loại có nhiều lớp.
- Hàm softmax được định nghĩa như sau:

$$\boldsymbol{softmax}(\boldsymbol{z_i}) = \frac{\boldsymbol{e^{z_i}}}{\sum_{\boldsymbol{j=1}}^{\boldsymbol{K}} \boldsymbol{e^{z_j}}}$$

Trong đó :

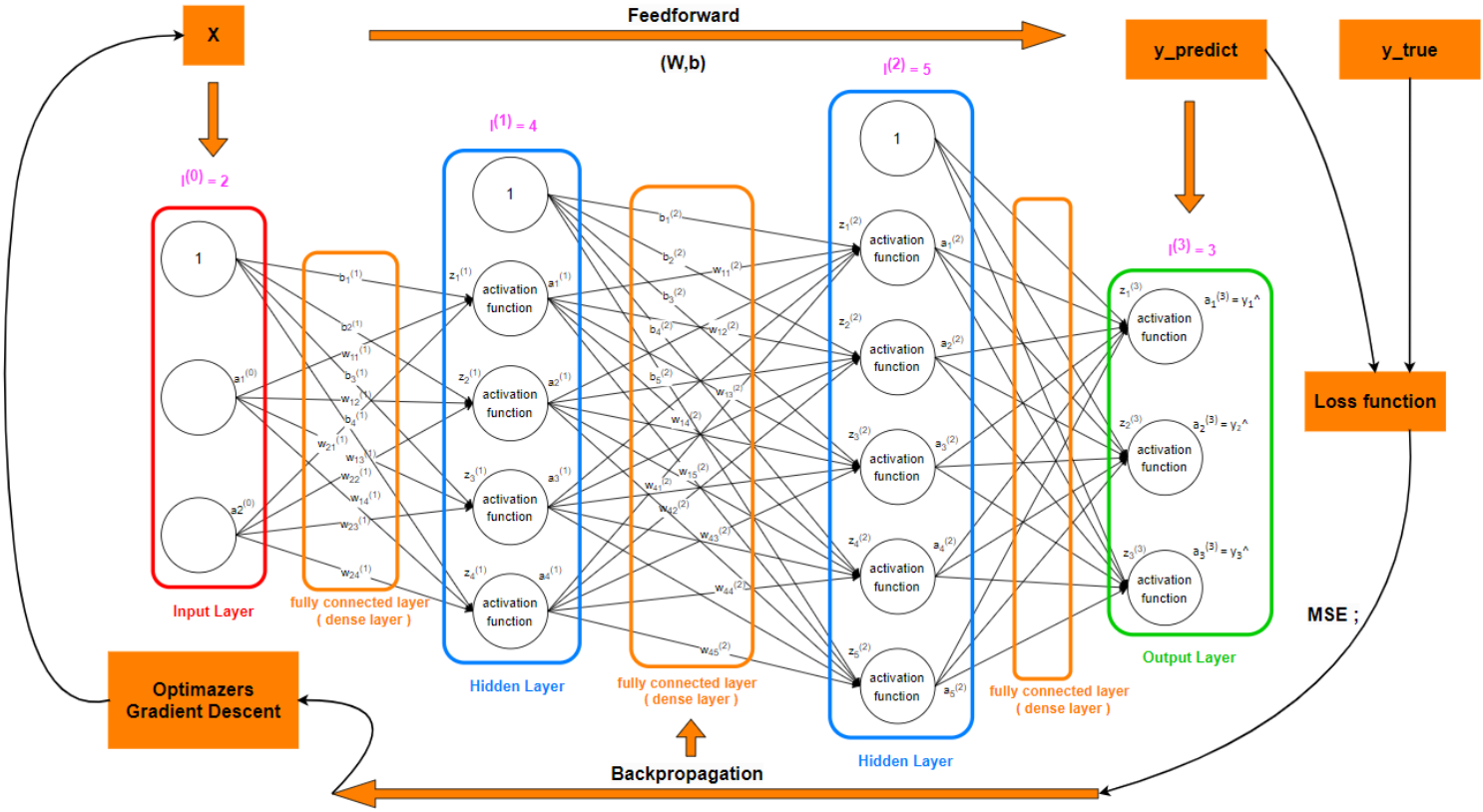
- z_i là điểm số (score) của lớp i (đầu ra của mạng tại node i)
- K là số lớp đầu ra

Hàm softmax thực hiện hai công việc chính:

1. Chuyển đổi các điểm số đầu ra thành các giá trị xác suất bằng cách biến đổi chúng sao cho tổng của tất cả các xác suất bằng 1.
2. Tăng cường sự khác biệt giữa các xác suất, làm cho xác suất cao nhất trở nên rất gần 1 trong khi các xác suất khác gần 0.

⇨ Việc sử dụng softmax trong lớp đầu ra cho phép mô hình phản ánh chính xác xác suất của dữ liệu thuộc vào từng lớp, giúp ta dễ dàng tạo ra dự đoán đúng cho nhiều lớp khác nhau trong bài toán phân loại đa lớp.

Ví dụ về Sơ đồ đồ Neural network



Trong đó :

2. *Neural network*

+ Cấu trúc lớp (layer)

+ Lớp đầu vào (Input layer)

+ Các lớp ẩn (Hidden Layers)

+ Lớp đầu ra (Output layer)

+ Kí hiệu

+ $l^{(i)}$ *là số node của lớp thứ i*

3. *Ban đầu :*

+ Input đầu vào là mảng $x^{(0)} = \begin{bmatrix} x_1^{(0)} & x_2^{(0)} & \dots & x_{l^0}^{(0)} \end{bmatrix}$ *với kích thước là* $1 * l^0$

+ Ở Linear và Logistic regression thì ta có w_0 *nhưng ở neural thì ta sẽ có* b_0 *là hệ số bias tách ra để dễ tính toán .* $w^{(1)}$ *và* $b^{(1)}$ *là lớp fully connected kết nối giữa 2 layer .*

Với $w^{(1)} = \begin{bmatrix} w_{11}^{(1)} & \dots & w_{1l^1}^{(1)} \\ \dots & \dots & \dots \\ w_{l^0 1}^{(1)} & \dots & w_{l^0 l^1}^{(1)} \end{bmatrix}$ *với ma trận là* $l^0 * l^1$ *và* $b^{(1)} = \begin{bmatrix} b_1^{(1)} & b_2^{(1)} & \dots & b_{l^1}^{(1)} \end{bmatrix}$*với ma trận* $1 * l^1$

Ta có $z^{(1)} = x^{(0)} * w^{(1)} + b^{(1)} = \begin{bmatrix} z_1^{(1)} & z_2^{(1)} & \dots & z_{l^1}^{(1)} \end{bmatrix}$

Áp dụng activation fucntion , ví dụ là ***sigmoid*** *cho z ta có :* $a^{(1)} = \sigma(z^{(1)}) = \begin{bmatrix} a_1^{(1)} & a_2^{(1)} & \dots & a_{l^1}^{(1)} \end{bmatrix}$

Ở đây : Trên mỗi layer ta chỉ áp dụng một hàm kích hoạt , nhiều layer có thể áp dụng nhiều hàm kích hoạt ví dụ : layer_1 dùng sigmoid , layer_2 dùng relu , layer_i dùng tanh,...

⇨ ***Sau đó*** $a^{(1)}$ ***lại trở thành input đầu vào cho lớp tiếp theo...***

Chính vì thế nên đổi kí hiệu $a^{(0)} = \left[x^{(0)}\right]^T$ *với ma trận là* $l^0 * 1$ *cho tiện tính toán sau này .*

⇨ ***Tổng quát ra ta sẽ có*** $a^{(i+1)}$ ***với ma trận là*** $l^{(i+1)} * 1$ ***với*** $l^{(i+1)}$ ***là số node trên lớp thứ*** i ***và*** $a^{(i+1)}$ ***là kết quả đầu ra của lớp*** $a^{(i)}$ ***và sẽ là đầu vào của lớp*** $a^{(i+2)}$ ***, trong đó*** $i \in [0, N-3]$ ***, mạng neural network có N lớp (gồm cả input, hidden, output) (***$N \geq 3$)

4. *Sau đó :*

+ Vì $a^{(i)}$ *đã được thay đổi nên* $w^{(i)}$ *cũng phải được thay đổi ta sẽ có*

a. $z^{(i+1)} = a^{(i)} * \left[w^{(i+1)}\right]^T + b^{(i)}$

b. $a^{(i+1)} = activation\ function\left(z^{(i+1)}\right)$

$ex: a^{(i+1)} = \sigma\left(z^{(i+1)}\right);\ if\ activation\ function = sigmoid\ function$

List activation function

1. *Sigmoid Activation Function*

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

2. *Tanh Activation Function*

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

3. *ReLU (Rectified Linear Unit) Activation Function*

$$f(x) = \max(0, x)$$

4. *Leaky ReLU Activation Function*

$$f(x) = \begin{cases} x, & if\ x > 0 \\ constant * x & others \end{cases}$$

5. *Softmax Activation Function (used in output layer for multi-class classification)*

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

6. *Linear Activation Function*

$$f(x) = x$$

7. *Swish Activation Function*

$$f(x) = x * \sigma(x)$$

8. *ELU (Exponential Linear Unit) Activation Function*

$$f(x) = \begin{cases} x, & if\ x \geq 0 \\ \alpha * (e^x - 1) & others \end{cases}$$

Trong đó, α là một số dương cố định.

Biểu diễn dưới dạng ma trận

Feedforward (Lan truyền xuôi)

Tất nhiên trên thực tế ta sẽ không làm việc với một dòng dữ liệu . Đầu vào dữ liệu X sẽ có n hàng và d là số trường dữ liệu hay còn gọi là số tham số đầu vào . Ta có X với ma trận $n * d$. Ta có x_j^i là phần tử ở hàng i cột j .

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & \dots & x_d^2 \\ \dots & \dots & \dots & \dots \\ x_1^n & x_2^n & \dots & x_d^n \end{bmatrix} = \begin{bmatrix} -\left(x^{(1)}\right)^T - \\ -\left(x^{(2)}\right)^T - \\ \dots \\ -\left(x^{(n)}\right)^T - \end{bmatrix}$$

Giải thích : Như đã nói ở trên mỗi hàng dữ liệu $x^{(i)}$đã được chuyển vị lại thành $\left[x^{(i)}\right]^T$

Bởi vì mỗi điểm dữ liệu $x^{(i)}$ là một vector cột với kích thước $d * 1$, chuyển vị để kích thước $1 * d$

Lưu ý : ***$\left[x^{(i)}\right]^T$, i ở đây chính là hàng dữ liệu thứ i . còn i ở Z và A là layer thứ i***

Gọi ma trận $Z^{(i)}$ kích thước $N * l^{(i)}$ trong đó $z_j^{(i)[k]}$ là giá trị thứ j trong layer i sau bước tính tổng linear của dữ liệu thứ k trong dataset . ***Lưu ý : Ở trong X thì i chính là hàng dữ liệu thứ i . Ở trong Z và A thì i chính là layer thứ i và k chính là hàng dữ liệu thứ k trong dataset, tránh nhầm lẫn .***

Tương tự, gọi $A^{(i)}$ kích thước $N * l^{(i)}$ trong đó $a_j^{(i)k}$ là giá trị thứ j trong layer i sau khi áp dụng activation function của dữ liệu thứ k trong dataset

Giải thích : $Z^{(i)}$ và $A^{(i)}$có kích thước $N * l^{(i)}$ trong đó N là số hàng và $l^{(i)}$ là số cột . N sẽ là cố định , N chính là số hàng dữ liệu của dataset , lặp qua N hàng dữ liệu và ta có được N hàng $Z^{(i)}$, còn $l^{(i)}$ sẽ thay đổi vì nó phụ thuộc vào số Node của các lớp layer . Tương tự với $A^{(i)}$.

Ta có :

$$Z^{(i)} = \begin{bmatrix} z_1^{(i)[1]} & z_2^{(i)[1]} & \dots & z_{l^{(i)}}^{(i)[1]} \\ z_1^{(i)[2]} & z_2^{(i)[2]} & \dots & z_{l^{(i)}}^{(i)[1]} \\ \dots & \dots & \dots & \dots \\ z_1^{(i)[n]} & z_2^{(i)[n]} & \dots & z_{l^{(i)}}^{(i)[n]} \end{bmatrix} = \begin{bmatrix} -\left(z^{(i)[1]}\right)^T - \\ -\left(z^{(i)[2]}\right)^T - \\ \dots \\ -\left(z^{(i)[n]}\right)^T - \end{bmatrix} = \begin{bmatrix} \left(x^{[1]}\right)^T * w^{(i)} + \left(b^{(i)}\right)^T \\ \left(x^{[2]}\right)^T * w^{(i)} + \left(b^{(i)}\right)^T \\ \dots \\ \left(x^{[n]}\right)^T * w^{(i)} + \left(b^{(i)}\right)^T \end{bmatrix}$$

$$= \begin{bmatrix} \left(x^{[1]}\right)^T * w^{(i)} + \left(b^{(i)}\right)^T \\ \left(x^{[2]}\right)^T * w^{(i)} + \left(b^{(i)}\right)^T \\ \dots \\ \left(x^{[n]}\right)^T * w^{(i)} + \left(b^{(i)}\right)^T \end{bmatrix} = X * W^{(i)} + \begin{bmatrix} \left(b^{(i)}\right)^T \\ \left(b^{(i)}\right)^T \\ \dots \\ \left(b^{(i)}\right)^T \end{bmatrix} = X * W^{(i)} + b^{(i)}$$

Giải thích :

5. *Có N hàng dữ liệu và M layer , lặp qua N hàng dữ liệu ta sẽ có $Z^{(i)}$ và lặp qua M layer ta sẽ có ma trận Z gồm nhiều $Z^{(i)}$, $i \in [1, M-1]$; $M \geq 3$. (layer luôn bắt đầu bằng 0 nên input layer sẽ là i – 1)*
6. *$\left(x^{[k]}\right)^T$ có kích thước $1 * d$ với d là số đặc trưng (tham số đầu vào) , mặc khác d cũng là $l^{(0)}$, là số node của lớp input đầu vào*

7. $w^{(i)}$ *có kích thước là* $l^{(0)} * \ l^{(1)}$
8. $(b^{(i)})^T$ *có kích thước là* $1 * \ l^{(1)}$
9. *Giả sử ta khai triển* $z^{(i)[1]}$ *: Với i = 1 (lớp đầu tiên sau khi cho input* $x^{(k)}$ *vào*

$$z^{(1)[1]} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_{l^{(0)}}^1 \end{bmatrix} * \begin{bmatrix} w_{11}^{(1)} & \dots & w_{1l^1}^{(1)} \\ \dots & \dots & \dots \\ w_{l^0 1}^{(1)} & \dots & w_{l^0 l^1}^{(1)} \end{bmatrix} + \begin{bmatrix} b_1^1 & b_2^1 & \dots & b_{l^{(1)}}^1 \end{bmatrix}$$

$$z_1^{(1)[1]} = x_1^1 * w_{11}^1 + \ x_2^1 * w_{21}^1 + \dots + \ x_{l^0}^1 * w_{l^0 1}^1 + b_1^1$$

$$z_2^{(1)[1]} = x_1^1 * w_{12}^1 + \ x_2^1 * w_{22}^1 + \dots + \ x_{l^0}^1 * w_{l^0 2}^1 + b_2^1$$

…

$$z_{l^1}^{(1)[1]} = x_1^1 * w_{1l^1}^1 + \ x_2^1 * w_{2l^i}^1 + \dots + \ x_{l^0}^1 * w_{l^0 l^i}^1 + b_{l^1}^1$$

Cuối cùng ta được : $z^{(1)[1]} = \begin{bmatrix} z_1^{(1)[1]} & z_2^{(1)[1]} & \dots & z_{l^1}^{(1)[1]} \end{bmatrix}$ *với kích thước* $1 * \ l^1$

⇨ ***Sau khi có*** $\boldsymbol{z^{(i)[k]}}$ ***ta chỉ cần cho qua activation function nữa là xong có được*** $\boldsymbol{a^{(i)[k]}}$ ***làm đầu vào cho lớp tiếp theo .*** $\boldsymbol{a^{(i)[k]} = \ \sigma\ (z^{(i)[k]})}$ ***hay*** $\boldsymbol{A^{(i)} = \ \sigma\ (Z^{(i)})}$***. Và*** $\boldsymbol{A^{(i)} = \ \hat{Y}}$ ***nếu i là lớp cuối cùng .***

Tổng quát :

Ta sẽ có với i = 0 (input layer)

$$z^{(i+1)[k]} = \begin{bmatrix} x_1^k & x_2^k & \dots & x_{l^{(i)}}^k \end{bmatrix} * \begin{bmatrix} w_{11}^{(i+1)} & \dots & w_{1l^{(i+1)}}^{(i+1)} \\ \dots & \dots & \dots \\ w_{l^i 1}^{(i+1)} & \dots & w_{l^i l^{(i+1)}}^{(i+1)} \end{bmatrix} + \begin{bmatrix} b_1^1 & b_2^1 & \dots & b_{l^{(i+1)}}^1 \end{bmatrix}$$

$$z_1^{(i+1)[k]} = x_1^k * w_{11}^{(i+1)} + \ x_2^k * w_{21}^{(i+1)} + \dots + \ x_{l^i}^1 * w_{l^i 1}^{(i+1)} + b_1^1$$

$$z_2^{(i+1)[k]} = x_1^k * w_{12}^{(i+1)} + \ x_2^k * w_{22}^{(i+1)} + \dots + \ x_{l^i}^1 * w_{l^i 2}^{(i+1)} + b_2^1$$

…

$$z_{l^{i+1}}^{(i+1)[k]} = x_1^k * w_{1l^{i+1}}^{(i+1)} + \ x_2^k * w_{2l^{i+1}}^{(i+1)} + \dots + \ x_{l^i}^k * w_{l^i l^{i+1}}^{(i+1)} + b_{l^{(i+1)}}^1$$

Cuối cùng ta được : $z^{(i+1)[k]} = \begin{bmatrix} z_1^{(i+1)[k]} & z_2^{(i+1)[k]} & \dots & z_{l^{i+1}}^{(i+1)[k]} \end{bmatrix}$ *với kích thước* $1 * \ l^{i+1}$

$$a^{(i+1)[k]} = \begin{bmatrix} \sigma(z_1^{(i+1)[k]}) & \sigma(z_2^{(i+1)[k]}) & \dots & \sigma(z_{l^{i+1}}^{(i+1)[k]}) \end{bmatrix}$$

Ta sẽ có với i ≥ 1 (hidden layer + output layer)

$$z^{(i+1)[k]} = \begin{bmatrix} a_1^{(i)[k]} & a_2^{(i)[k]} & \dots & a_{l^{(i)}}^{(i)[k]} \end{bmatrix} * \begin{bmatrix} w_{11}^{(i+1)} & \dots & w_{1l^{(i+1)}}^{(i+1)} \\ \dots & \dots & \dots \\ w_{l^i 1}^{(i+1)} & \dots & w_{l^i l^{(i+1)}}^{(i+1)} \end{bmatrix} + \begin{bmatrix} b_1^{(i+1)} & b_2^{(i+1)} & \dots & b_{l^{(i+1)}}^{(i+1)} \end{bmatrix}$$

$$z_1^{(i+1)[k]} = a_1^{(i)[k]} * w_{11}^{(i+1)} + \ a_2^{(i)[k]} * w_{21}^{(i+1)} + \dots + \ a_{l^{(i)}}^{(i)[k]} * w_{l^i 1}^{(i+1)} + b_1^{(i+1)}$$

$$z_2^{(i+1)[k]} = a_1^{(i)[k]} * w_{12}^{(i+1)} + \ a_2^{(i)[k]} * w_{22}^{(i+1)} + \dots + \ a_{l^{(i)}}^{(i)[k]} * w_{l^i 2}^{(i+1)} + b_2^{(i+1)}$$

…

$$z_{l^{i+1}}^{(i+1)[k]} = a_1^{(i)[k]} * w_{1l^{i+1}}^{(i+1)} + a_2^{(i)[k]} * w_{2l^{i+1}}^{(i+1)} + \cdots + a_{l^{(i)}}^{(i)[k]} * w_{l^i l^{i+1}}^{(i+1)} + b_{l^{(i+1)}}^{(i+1)}$$

Cuối cùng ta được : $z^{(i+1)[k]} = \left[z_1^{(i+1)[k]} \quad z_2^{(i+1)[k]} \quad \dots \quad z_{l^{i+1}}^{(i+1)[k]}\right]$ *với kích thước* $1 * l^{i+1}$

$$a^{(i+1)[k]} = \left[\sigma(z_1^{(i+1)[k]}) \quad \sigma(z_2^{(i+1)[k]}) \quad \dots \quad \sigma(z_{l^{i+1}}^{(i+1)[k]})\right]$$

⇨ *Từ input X ta có* $\hat{Y}$*, tuy nhiên như 2 bài trước , cần phải tối ưu W và b bằng Gradient descent . Ta sẽ đi tìm đạo hàm của các hệ số đối với hàm loss funcion . Việc tính đạo hàm của các hệ số trong neural network được thực hiện bởi thuật toán* ***backpropagation***.

Backpropagation (Lan truyền ngược)

Model

10. Mô hình 2 – 2 – 1 có nghĩa là 2 node trong input layer, 1 hidden layer có 2 node và output layer có 1 node.
11. Input layer và hidden layer luôn thêm node 1 để tính bias cho layer sau, nhưng không tính vào số lượng node trong layer.
12. Ở mỗi node trong hidden layer và output layer đều thực hiện 2 bước: tính tổng linear và áp dụng activation function.

Loss function

Hàm loss fucntion vẫn dùng giống như trong bài 2 , với mỗi điểm $(x^{(i)}, y_i)$, gọi hàm loss function

$$L = -(y_i * \log_e(\hat{y}_i) + (1 - y_i) * \log_e(1 - \hat{y}_i))$$

$$J = \frac{1}{N} * \sum_{i=1}^{N} L = -\frac{1}{N} * \sum_{i=1}^{N} x_m^{(i)} * (y_i * \log_e(\hat{y}_i) + (1 - y_i) * \log_e(1 - \hat{y}_i))$$

Lưu ý : $\left[x^{(i)}\right]^T$***, i ở đây chính là hàng dữ liệu thứ i . còn i ở Z và A là layer thứ i***

Gradient descent

Lưu ý :

Khi hàm f(x) là hàm 1 biến x , ví dụ $f(x) = 2 * x + 1$. Đạo hàm f đối với biến x là $\frac{df}{dx}$

Khi hàm f(x,y) là hàm nhiều biến, ví dụ $f(x, y) = x^2 + y^2$. Đạo hàm f đối với biến x kí hiệu là $\frac{\partial f}{\partial x}$

$$L = -(y_i * \log_e(\hat{y}_i) + (1 - y_i) * \log_e(1 - \hat{y}_i))$$

Hay $L = -(y_{i+1} * \log_e(\widehat{y_{i+1}}) + (1 - y_{i+1}) * \log_e(1 - \widehat{y_{i+1}}))$

Trong đó :

$$\widehat{y_{i+1}} = a^{(i+1)[k]} = \sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right) = \sigma\left(a_1^{(i)[k]} * w_{1l^{i+1}}^{(i+1)} + a_2^{(i)[k]} * w_{2l^{i+1}}^{(i+1)} + \cdots + a_{l^{(i)}}^{(i)[k]} * w_{l^i l^{i+1}}^{(i+1)} + b_{l^{(i+1)}}^{(i+1)}\right)$$

Chuẩn bị

$$\frac{\partial L}{\partial \hat{y}_i} = \frac{\partial(-(y_i * \log_e(\hat{y}_i) + (1 - y_i) * \log_e(1 - \hat{y}_i)))}{\partial \hat{y}_i} = -\left(\frac{y_i}{\hat{y}_i} - \frac{1 - y_i}{1 - \hat{y}_i}\right) = \frac{\hat{y}_i - y_i}{\hat{y}_i * (1 - \hat{y}_i)}$$

$$\frac{d(\sigma(x))}{dx} = \frac{d\left(\frac{1}{1 + e^{-x}}\right)}{dx} = \frac{d\left(\frac{1}{1 + e^{-x}}\right)}{d(1 + e^{-x})} * \frac{d(1 + e^{-x})}{d(-x)} * \frac{d(-x)}{x} = -\frac{1}{(1 + e^{-x})^2} * e^{-x} * -1$$

$$= \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} * \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}} * \left(1 - \frac{1}{1+e^{-x}}\right) = \sigma(x) * (1 - \sigma(x))$$

Tính đạo hàm L với W^{i+1} và b^{i+1}

Áp dụng quy tắc Chain rule : $\frac{\partial L}{\partial b^{i+1}} = \frac{\partial L}{\partial \widehat{y_{i+1}}} * \frac{\partial \widehat{y_{i+1}}}{\partial b^{i+1}}$

1. $\frac{\partial \widehat{y_{i+1}}}{\partial b^{(i+1)}_{l^{(i+1)}}} = \frac{\partial\left(\sigma\left(a_1^{(i)[k]} * w_{1l^{i+1}}^{(i+1)} + a_2^{(i)[k]} * w_{2l^{i+1}}^{(i+1)} + \cdots + a_{l^{(i)}}^{(i)[k]} * w_{l^i l^{i+1}}^{(i+1)} + b_{l^{(i+1)}}^{(i+1)}\right)\right)}{\partial b^{(i+1)}_{l^{(i+1)}}} = \frac{\partial\left(\sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right)\right)}{\partial z_{l^{i+1}}^{(i+1)[k]}} * \frac{\partial z_{l^{i+1}}^{(i+1)[k]}}{\partial b^{(i+1)}_{l^{(i+1)}}}$

$= \sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right) * \left(1 - \sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right)\right) * 1 = \widehat{y_{i+1}} * (1 - \widehat{y_{i+1}})$

2. $\frac{\partial \widehat{y_{i+1}}}{\partial a^{(i)[k]}_{l^{(i)}}} = \frac{\partial\left(\sigma\left(a_1^{(i)[k]} * w_{1l^{i+1}}^{(i+1)} + a_2^{(i)[k]} * w_{2l^{i+1}}^{(i+1)} + \cdots + a_{l^{(i)}}^{(i)[k]} * w_{l^i l^{i+1}}^{(i+1)} + b_{l^{(i+1)}}^{(i+1)}\right)\right)}{\partial a^{(i)[k]}_{l^{(i)}}} = \frac{\partial\left(\sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right)\right)}{\partial z_{l^{i+1}}^{(i+1)[k]}} * \frac{\partial z_{l^{i+1}}^{(i+1)[k]}}{\partial a^{(i)[k]}_{l^{(i)}}}$

$= \sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right) * \left(1 - \sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right)\right) * \ w_{l^i l^{i+1}}^{(i+1)} = w_{l^i l^{i+1}}^{(i+1)} * \widehat{y_{i+1}} * (1 - \widehat{y_{i+1}})$

3. $\frac{\partial \widehat{y_{i+1}}}{\partial w^{(i+1)}_{l^i l^{i+1}}} = \frac{\partial\left(\sigma\left(a_1^{(i)[k]} * w_{1l^{i+1}}^{(i+1)} + a_2^{(i)[k]} * w_{2l^{i+1}}^{(i+1)} + \cdots + a_{l^{(i)}}^{(i)[k]} * w_{l^i l^{i+1}}^{(i+1)} + b_{l^{(i+1)}}^{(i+1)}\right)\right)}{\partial w^{(i+1)}_{l^i l^{i+1}}} = \frac{\partial\left(\sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right)\right)}{\partial z_{l^{i+1}}^{(i+1)[k]}} * \frac{\partial z_{l^{i+1}}^{(i+1)[k]}}{\partial w^{(i+1)}_{l^i l^{i+1}}}$

$= \sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right) * \left(1 - \sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right)\right) * \ a_{l^{(i)}}^{(i)[k]} = a_{l^{(i)}}^{(i)[k]} * \widehat{y_{i+1}} * (1 - \widehat{y_{i+1}})$

Do đó :

1. $\frac{\partial L}{\partial b^{(i+1)}_{l^{(i+1)}}} = \frac{\partial L}{\partial \widehat{y_{i+1}}} * \frac{\partial \widehat{y_{i+1}}}{\partial b^{(i+1)}_{l^{(i+1)}}} = \frac{\widehat{y_{i+1}} - y_{i+1}}{\widehat{y_{i+1}} * (1 - \widehat{y_{i+1}})} * \ \widehat{y_{i+1}} * (1 - \widehat{y_{i+1}}) = \widehat{y_{i+1}} - \ y_{i+1}$
2. $\frac{\partial L}{\partial a^{(i)[k]}_{l^{(i)}}} = \frac{\partial L}{\partial \widehat{y_{i+1}}} * \frac{\partial \widehat{y_{i+1}}}{\partial a^{(i)[k]}_{l^{(i)}}} = \frac{\widehat{y_{i+1}} - y_{i+1}}{\widehat{y_{i+1}} * (1 - \widehat{y_{i+1}})} * \ w_{l^i l^{i+1}}^{(i+1)} * \ \widehat{y_{i+1}} * (1 - \widehat{y_{i+1}}) = w_{l^i l^{i+1}}^{(i+1)} * \ (\widehat{y_{i+1}} - y_{i+1})$
3. $\frac{\partial L}{\partial w^{(i+1)}_{l^i l^{i+1}}} = \frac{\partial L}{\partial \widehat{y_{i+1}}} * \frac{\partial \widehat{y_{i+1}}}{\partial w^{(i+1)}_{l^i l^{i+1}}} = \frac{\widehat{y_{i+1}} - y_{i+1}}{\widehat{y_{i+1}} * (1 - \widehat{y_{i+1}})} * a_{l^{(i)}}^{(i)[k]} * \ \widehat{y_{i+1}} * (1 - \widehat{y_{i+1}}) = a_{l^{(i)}}^{(i)[k]} * (\widehat{y_{i+1}} - y_{i+1})$

Biểu diễn dưới dạng ma trận

Lưu ý : Đạo hàm của L đối với ma trận W kích thước m * n cũng là một ma trận cùng kích thước m * n .

$$\frac{\partial L}{\partial W}\begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \cdots & \frac{\partial L}{\partial w_{1n}} \\ \cdots & \cdots & \cdots \\ \frac{\partial L}{\partial w_{m1}} & \cdots & \frac{\partial L}{\partial w_{mn}} \end{bmatrix}$$

Do đó : $\frac{\partial J}{\partial w^{i+1}} = \left(a^{(i)[k]}\right)^T * (\hat{Y} - \ Y)$; $\frac{\partial J}{\partial b^{i+1}} = \left(sum\left(\hat{Y} - \ Y\right)\right)^T$; $\frac{\partial J}{\partial a^{(i)[k]}} = \left(w^{i+1}\right)^T * \ (\hat{Y} - \ Y)$

Phép tính **Sum** tính tổng trên các cột của ma trận

$$W = \begin{bmatrix} w_{11}^{(i+1)} & \dots & w_{1l^{(i+1)}}^{(i+1)} \\ \dots & \dots & \dots \\ w_{l^i 1}^{(i+1)} & \dots & w_{l^i l^{(i+1)}}^{(i+1)} \end{bmatrix} => sum(W) = (w_{11}^{(i+1)} + \dots + w_{l^i 1}^{(i+1)}, \dots, w_{1l^{(i+1)}}^{(i+1)} + \dots + w_{l^i l^{(i+1)}}^{(i+1)})$$

$$\big(sum(W)\big)^T = \begin{bmatrix} w_{11}^{(i+1)} + \dots + w_{l^i 1}^{(i+1)} \\ \dots \\ w_{1l^{(i+1)}}^{(i+1)} + \dots + w_{l^i l^{(i+1)}}^{(i+1)} \end{bmatrix}$$

Tính đạo hàm L với W^i và b^i . *Ta sẽ có với i = 0 (input layer)*

Lưu ý : Ở trên ta đạo hàm như sau : $\frac{\partial L}{\partial b^{i+1}} = \frac{\partial L}{\partial \widehat{y_{i+1}}} * \frac{\partial \widehat{y_{i+1}}}{\partial b^{i+1}}$

Còn sau đây ta sẽ đạo hàm theo : $\frac{\partial L}{\partial b^{i+1}} = \frac{\partial L}{\partial a^{(i+1)}} * \frac{\partial a^{(i+1)}}{\partial b^{i+1}}$

Ta có :

$z_{l^{i+1}}^{(i+1)[k]} = x_1^k * w_{1l^{i+1}}^{(i+1)} + x_2^k * w_{2l^{i+1}}^{(i+1)} + \dots + x_{l^i}^k * w_{l^i l^{i+1}}^{(i+1)} + b_{l^{(i+1)}}^1$

Cuối cùng ta được : $z^{(i+1)[k]} = \begin{bmatrix} z_1^{(i+1)[k]} & z_2^{(i+1)[k]} & \dots & z_{l^{i+1}}^{(i+1)[k]} \end{bmatrix}$ *với kích thước* $1 * l^{i+1}$

$a^{(i+1)[k]} = \begin{bmatrix} \sigma(z_1^{(i+1)[k]}) & \sigma(z_2^{(i+1)[k]}) & \dots & \sigma(z_{l^{i+1}}^{(i+1)[k]}) \end{bmatrix}$

1. $\frac{\partial a^{(i+1)}}{\partial b^{i+1}} = \frac{\partial \left(\sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right)\right)}{\partial z_{l^{i+1}}^{(i+1)[k]}} * \frac{\partial z_{l^{i+1}}^{(i+1)[k]}}{\partial b_{l^{(i+1)}}^1} = a^{(i+1)} * (1 - a^{(i+1)})$
2. $\frac{\partial a^{(i+1)}}{\partial w_{l^i l^{i+1}}^{(i+1)}} = \frac{\partial \left(\sigma\left(z_{l^{i+1}}^{(i+1)[k]}\right)\right)}{\partial z_{l^{i+1}}^{(i+1)[k]}} * \frac{\partial z_{l^{i+1}}^{(i+1)[k]}}{\partial w_{l^i l^{i+1}}^{(i+1)}} = x_{l^i}^k * a^{(i+1)} * (1 - a^{(i+1)})$

Mặc khác ta có ở trên

$$\frac{\partial L}{\partial a^{(i)}} = w_{l^i l^{i+1}}^{(i+1)} * (\widehat{y_{i+1}} - y_{i+1})$$ ***Thay i bằng i + 1 ta sẽ được tương tự***

Do đó :

1. $\frac{\partial L}{\partial b^{i+1}} = \frac{\partial L}{\partial a^{(i+1)}} * \frac{\partial a^{(i+1)}}{\partial b^{i+1}} = a^{(i+1)} * \left(1 - a^{(i+1)}\right) * w_{l^{i+2} l^{i+1}}^{(i+2)} * (\widehat{y_{i+1}} - y_{i+1})$
2. $\frac{\partial L}{\partial w_{l^i l^{i+1}}^{(i+1)}} = \frac{\partial L}{\partial a^{(i+1)}} * \frac{\partial a^{(i+1)}}{\partial w_{l^i l^{i+1}}^{(i+1)}} = x_{l^i}^k * a^{(i+1)} * (1 - a^{(i+1)}) * w_{l^{i+2} l^{i+1}}^{(i+2)} * (\widehat{y_{i+1}} - y_{i+1})$

Biểu diễn dưới dạng ma trận

Có thể viết dưới dạng Chain rule : $\frac{\partial J}{\partial W^{(i+1)}} = \frac{\partial J}{\partial A^{(i+1)}} * \frac{\partial A^{(i+1)}}{\partial Z^{(i+1)}} * \frac{\partial Z^{(i+1)}}{\partial W^{(i+1)}}$ (1)

Từ trên ta đã tính được : $\frac{\partial J}{\partial A^{(i+1)}} = \left(\hat{Y} - Y\right) * \left(W^{(i+2)}\right)^T$

Đạo hàm của hàm sigmoid $\frac{d\sigma(x)}{dx} = \sigma(x) * (1 - \sigma(x))$ $và$ $A^{(i+1)} = \sigma(Z^{(i+1)})$, nêu trong (1) có thể hiểu là $\frac{\partial A^{(i+1)}}{\partial Z^{(i+1)}} = A^{(i+1)} * (1 - A^{(i+1)})$.

Cuối cùng $Z^{(i+1)} = X * W^{(i+1)} + b^{(i+1)}$, nên có thể hiểu $\frac{\partial Z^{(i+1)}}{\partial W^{(i+1)}} = X$

Kết hợp lại tất cả ta có : $\boldsymbol{\frac{\partial J}{\partial W^{(i+1)}} = X^T * \left(\left(\left(\hat{Y} - Y\right) * \left(W^{(i+2)}\right)^T\right) \otimes A^{(i+1)} \otimes \left(1 - A^{(i+1)}\right)\right)}$

Thế khi nào thì dùng element-wise(⊗), khi nào dùng nhân ma trận(∗) ?

- Khi tính đạo hàm ngược lại quá trình activation thì dùng ⊗
- Khi có phép tính nhân ma trận thì dùng (*) , nhưng đặc biệt chú ý đến kích thước ma trận và dùng transpose nếu cần thiết . Ví dụ ma trận X kích thước N*3, W kích thước 3*4, Z = X * W sẽ có kích thước N * 4 thì

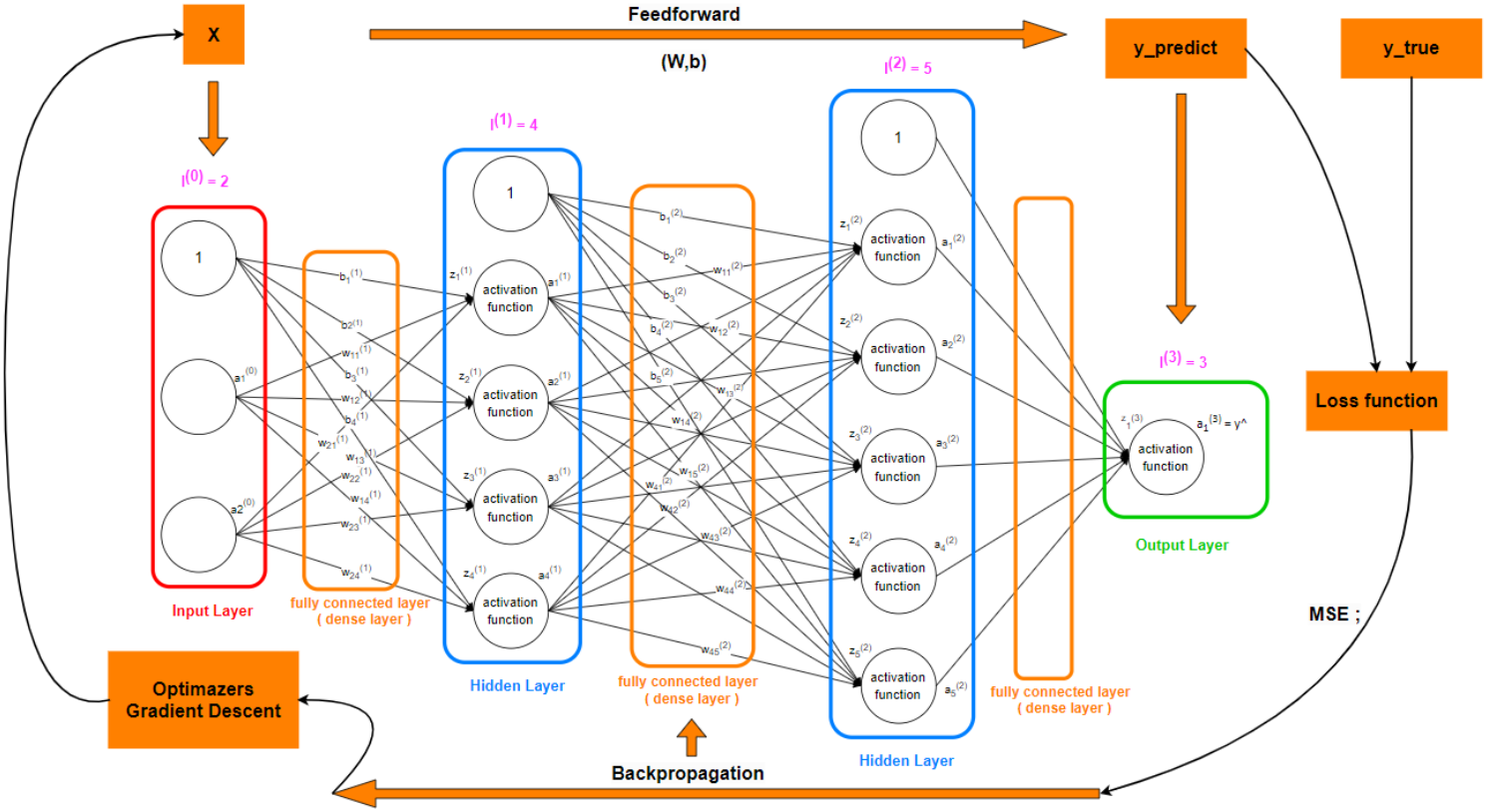
$\frac{\partial J}{\partial W} = X^T * \frac{\partial J}{\partial Z}$ và $\frac{\partial J}{\partial X} = \frac{\partial J}{\partial Z} * W^T$. Tương tự $\boldsymbol{\frac{\partial J}{\partial b^{(i+1)}} = sum\left(\left(\left(\hat{Y} - Y\right) * \left(W^{(i+2)}\right)^T\right) \otimes A^{(i+1)}\right)^T}$

⇨ **Vậy là đã tính xong hết đạo hàm của loss function với các hệ số W và bias b , giờ có thể áp dụng gradient descent để giải bài toán .**

Tóm tắt toàn bộ quá trình

Giả sử ta có mạng neural network với số lớp như sau :

- Input layer : 2 node
- Hidden layer 1 : 4 node
- Hidden layer 2 : 5 node
- Output layer : 1 node



1. Bước 1 : $\frac{\partial J}{\partial \hat{Y}}$, trong đó $\hat{Y} = A^{(3)}$
2. Bước 2 :

$$\frac{\partial J}{\partial \widehat{W^{(3)}}} = \left(A^{(2)}\right)^T * \left(\frac{\partial J}{\partial \hat{Y}} \otimes \frac{\partial A^{(3)}}{\partial Z^{(3)}}\right)$$

$$\frac{\partial J}{\partial \widehat{b^{(3)}}} = \left(sum\left(\frac{\partial J}{\partial \hat{Y}} \otimes \frac{\partial A^{(3)}}{\partial Z^{(3)}}\right)\right)^T$$

$$\frac{\partial J}{\partial \widehat{A^{(2)}}} = \left(\frac{\partial J}{\partial \hat{Y}} \otimes \frac{\partial A^{(3)}}{\partial Z^{(3)}}\right) * \left(W^{(3)}\right)^T$$

3. Bước 3 :

$$\frac{\partial J}{\partial \widehat{W^{(2)}}} = \left(A^{(1)}\right)^T * \left(\frac{\partial J}{\partial A^{(2)}} \otimes \frac{\partial A^{(2)}}{\partial Z^{(2)}}\right)$$

$$\frac{\partial J}{\partial \widehat{b^{(2)}}} = \left(sum\left(\frac{\partial J}{\partial A^{(2)}} \otimes \frac{\partial A^{(2)}}{\partial Z^{(2)}}\right)\right)^T$$

$$\frac{\partial J}{\partial \widehat{A^{(1)}}} = \left(\frac{\partial J}{\partial A^{(2)}} \otimes \frac{\partial A^{(2)}}{\partial Z^{(2)}}\right) * \left(W^{(2)}\right)^T$$

4. Bước 4 :

$$\frac{\partial J}{\partial \widehat{W^{(1)}}} = \left(A^{(0)}\right)^T * \left(\frac{\partial J}{\partial A^{(1)}} \otimes \frac{\partial A^{(1)}}{\partial Z^{(1)}}\right)$$

$$\frac{\partial J}{\partial \widehat{b^{(1)}}} = \left(sum\left(\frac{\partial J}{\partial A^{(1)}} \otimes \frac{\partial A^{(1)}}{\partial Z^{(1)}}\right)\right)^T$$

$$A^{(0)} = X$$

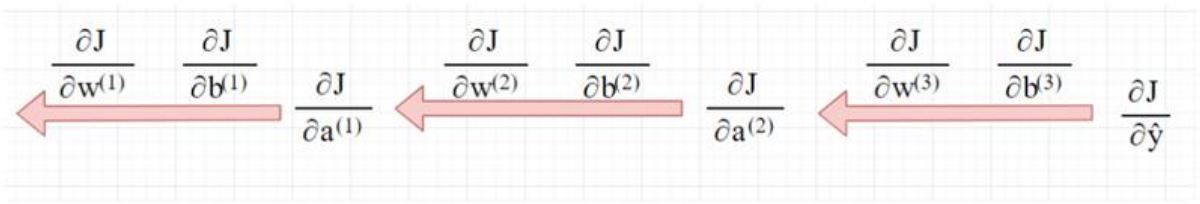
⇨ Nếu network có nhiều layer hơn thì cứ tiếp tục cho đến khi tính được đạo hàm củaloss function J với tất cả hệ số W và bias b .

Nếu hàm hàm acvtivation là sigmoid thì $\frac{\partial A^{(i)}}{\partial Z^{(i)}} = A^{(i)} \otimes \left(1 - A^{(i)}\right)$

Ở bài trước quá trình feedforward

Hình 6.3: Quá trình feedforward

Thì ở bài này quá trình tính đạo hàm ngược lại



Hình 6.4: Quá trình backpropagation

Đấy là vì sao thuật toán được gọi là backpropagation (lan truyền ngược)

Nói thêm

Mạng nơ-ron tích chập (CNN) thường được sử dụng trong xử lý ảnh và nhận diện vì các đặc điểm sau:

1. Cấu trúc chia sẻ trọng số: CNN sử dụng các lớp tích chập để trích xuất đặc trưng từ dữ liệu đầu vào. Các lớp này có các bộ lọc nhỏ di chuyển trên toàn bộ hình ảnh để phát hiện các đặc điểm cụ thể như cạnh, góc, hoặc đường cong. Việc sử dụng trọng số chia sẻ giúp giảm bớt số lượng tham số cần học và tăng tốc độ tính toán.
2. Dữ liệu không gian: Ảnh có cấu trúc không gian, nghĩa là các pixel ở gần nhau thường có mối quan hệ về mặt không gian. CNN có thể hiệu quả trong việc xử lý dữ liệu không gian như ảnh bằng cách giữ lại thông tin về không gian và mối quan hệ giữa các pixel.
3. Scale invariance: CNN có khả năng nhận biết các đặc trưng ở nhiều tỷ lệ khác nhau trong ảnh. Điều này có ý nghĩa là nó có thể nhận diện đối tượng trong ảnh dù chúng có kích thước khác nhau.

CNN và RNN (Mạng nơ-ron hồi tiếp) khác nhau ở cấu trúc và cách chúng xử lý dữ liệu. Trong khi CNN thích hợp cho dữ liệu không gian như ảnh, RNN thích hợp cho dữ liệu tuần tự như văn bản hoặc âm

thanh. CNN thường được sử dụng cho việc phát hiện đặc trưng cục bộ trong dữ liệu không gian, trong khi RNN thích hợp cho việc mô hình chuỗi dữ liệu với các phụ thuộc thời gian.

KNN (K-nearest neighbors) là một thuật toán học máy không được huấn luyện, nghĩa là nó không học ra một mô hình từ dữ liệu huấn luyện mà thay vào đó so sánh các điểm dữ liệu mới với các điểm dữ liệu đã được gán nhãn trong tập huấn luyện và dự đoán dựa trên việc tìm kiếm các điểm gần nhất. KNN không có cấu trúc mạng nơ-ron như CNN hoặc RNN, và nó không yêu cầu việc huấn luyện mô hình.