BACKPROPAGATION

Thuật toán quan trọng trong huấn luyện mạng nơ-ron nhân tạo, giúp tối ưu các trọng số của mạng

---

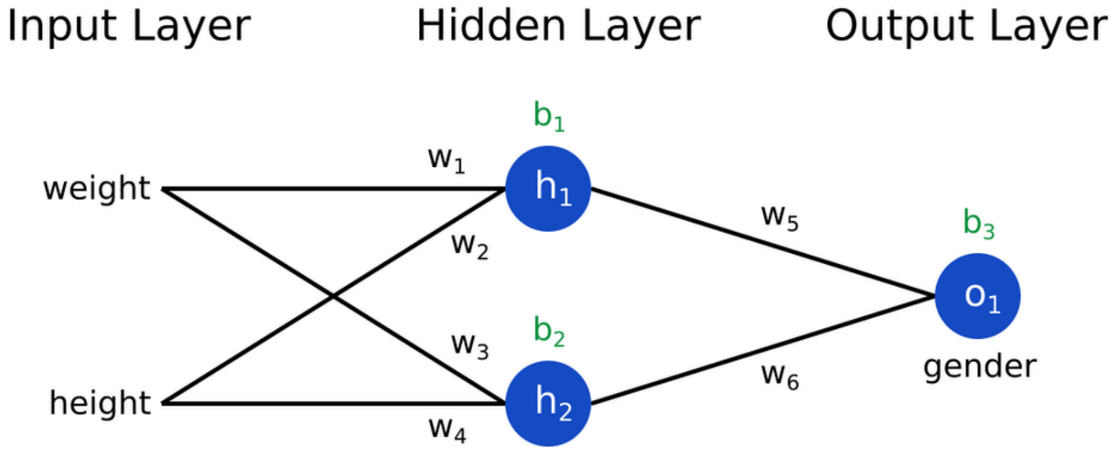
NGƯỜI TRÌNH BÀY
Phạm Công Minh

MENTOR
Phạm Đình Hải

MỤC LỤC

Giới thiệu về Neural Network

- Là mô hình học máy lấy cảm hứng từ não người
- Gồm 3 lớp cấu trúc: input → hiddent layer → output
- Các tham số chính:
  - Trọng số (weight): đánh giá mức độ quan trọng của từng input với output
  - Hệ số (Biase): giúp điều chỉnh output
  - Hàm activation: chuyển input đầu vào thành output đầu ra → giúp học được các mối quan hệ phức tạp
- Ứng dụng: nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên...



Nhược điểm

- Mạng nhân tạo 1 lớp:
  - Không có hidden layer → khó học được các quan hệ phi tuyến tính
  - Độ chính xác thấp với dữ liệu thực tế
- Mạng nhân tạo nhiều lớp:
  - Chi phí tính toán lớn do nhiều layer, nhiều tham số
  - Có thể học các bài toán phi tuyến tính nhưng khó tối ưu trọng số
  - Chưa được ứng dụng nhiều trong thực tế

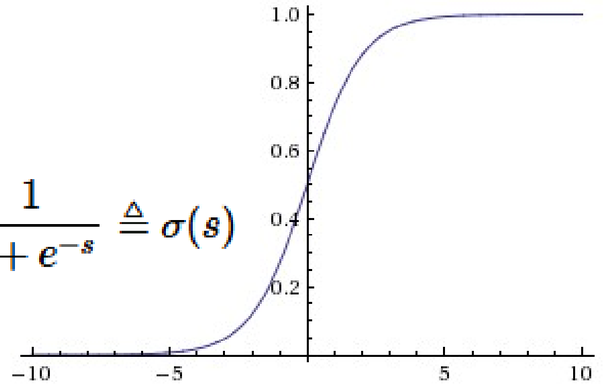
→ Thuật toán Backpropagation ra đời

HÀM PHI TUYẾN TÍNH

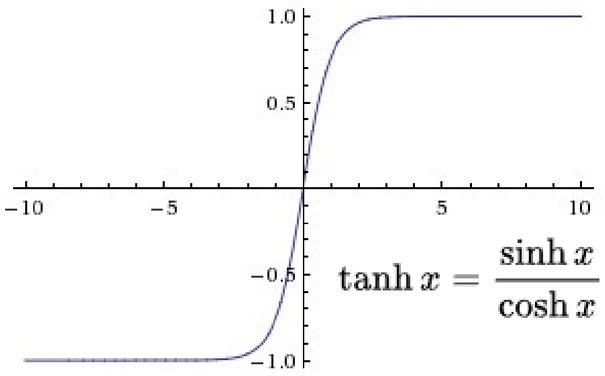
- **Định nghĩa:**
  - Hàm phi tuyến tính (Non-linear function) là hàm mà đồ thị của nó không phải đường thẳng, quan hệ giữa đầu vào và đầu ra không tuyến tính.
  - Hàm phi tuyến tính còn được gọi là hàm kích hoạt trong Neural Network
- **Vai trò :**
  - Giúp mạng học được quan hệ phức tạp.
  - Nếu không có hàm kích hoạt → mạng chỉ là mô hình tuyến tính, không thể giải quyết bài toán phức tạp.
- **Một số hàm cơ bản:**
  - Sigmoid và Tanh
  - Relu

HÀM PHI TUYẾN TÍNH

- **Hàm Sigmoid và Tanh**
  - **Nhược điểm:** vanishing gradient khi x lớn hoặc nhỏ.



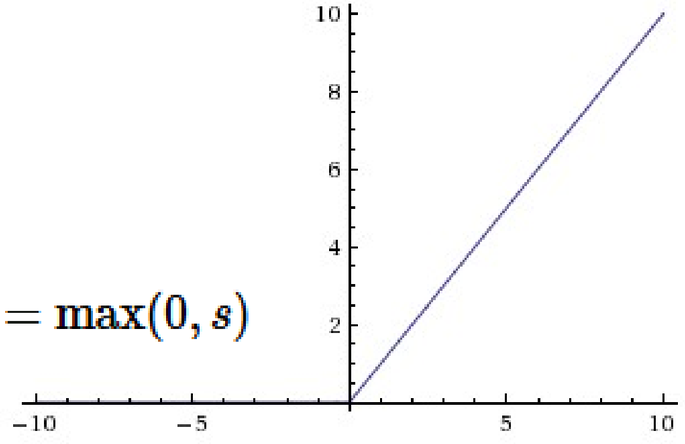
$$f(s) = \frac{1}{1+e^{-s}} \triangleq \sigma(s)$$



$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Hình 5: Hàm sigmoid (trái) và tanh (phải).

- **Hàm Relu**
  - **Ưu điểm:** Tính toán nhanh, gradient không bị mất khi x > 0.
  - **Nhược điểm:** chết nơ-ron khi x < 0 lâu dài.



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

Hình 6: Hàm ReLU

BACKPROPAGATION

Giúp mạng neural học tốt hơn bằng cách điều chỉnh trọng số để giảm sai số dự đoán.

Mục đích

- Cập nhật trọng số để giảm sai số dự đoán.

Các bước thực hiện:

1. Feedforward: Tính toán đầu ra.
2. Tính lỗi (Loss).
3. Lan truyền ngược (Backpropagation): Tính gradient bằng quy tắc chuỗi.
4. Cập nhật trọng số (Gradient Descent).

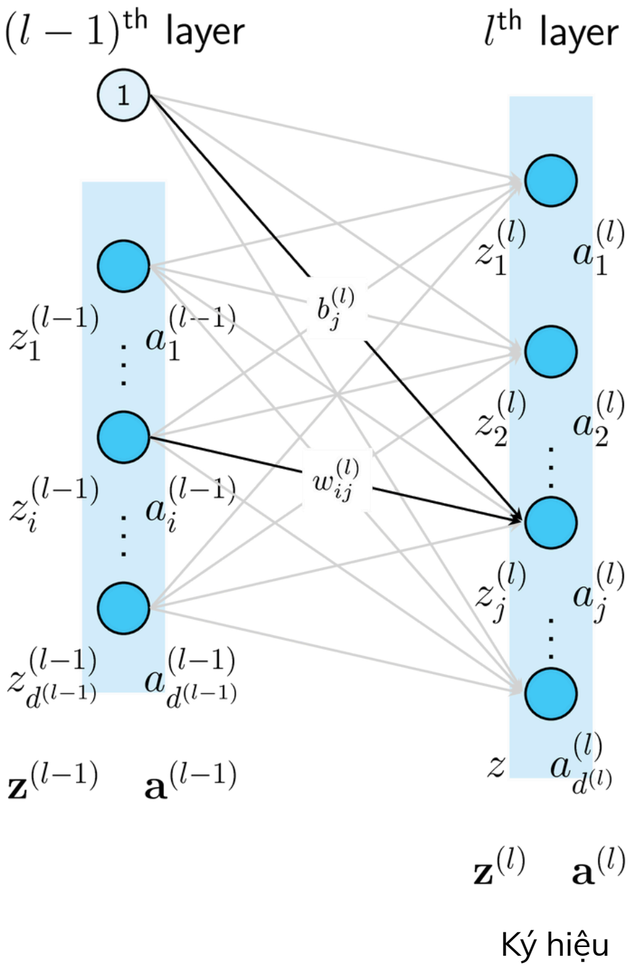
Ý nghĩa:

- Cho phép huấn luyện mạng nhiều lớp.

BACKPROPAGATION

Ký hiệu:

- W (Weights): Thể hiện các kết nối từ layer (l -1) tới layer (l)
- B (Bias) : Tham số bổ sung được cộng vào input trước khi áp dụng hàm kích hoạt
- Z : Đầu vào input của hidden layer
- a : Giá trị của mỗi node sau khi áp dụng hàm kích hoạt



Ký hiệu

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

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

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

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

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

BACKPROPAGATION

Thuật toán:

1. Tính Feedforward đầu ra
2. Tính hàm loss
3. Gradient ở output layer
4. Gradient ở middle layer
5. Cập nhật trọng số W

$$
\begin{aligned}
\mathbf{a}^{(0)} &= \mathbf{x} \\
z_i^{(l)} &= \mathbf{w}_i^{(l)T}\mathbf{a}^{(l-1)} + b_i^{(l)} \\
\mathbf{z}^{(l)} &= \mathbf{W}^{(l)T}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \;\; l = 1, 2, \ldots, L \\
\mathbf{a}^{(l)} &= f(\mathbf{z}^{(l)}), \;\; l = 1, 2, \ldots, L \\
\hat{\mathbf{y}} &= \mathbf{a}^{(L)}
\end{aligned}
$$

Ký hiệu

$$
\begin{aligned}
J(\mathbf{W}, \mathbf{b}, \mathbf{X}, \mathbf{Y}) &= \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{y}_n - \hat{\mathbf{y}}_n\|_2^2 \\
&= \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{a}_n^{(L)}\|_2^2
\end{aligned}
$$

Hàm loss

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

Gradient của hàm loss theo trọng số W

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

Tính E(j) tổng quát cho lớp input và hiddent

$$w := w - \eta \cdot \frac{\partial J}{\partial w}$$

Cập nhật W

$$\frac{\partial J}{\partial z_j^{(L)}} = \frac{\partial J}{\partial a_j^{(L)}} \cdot \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} = \frac{\partial J}{\partial a_j^{(L)}} \cdot f'(z_j^{(L)})$$

Tính E(j) tại lớp output

BACKPROPAGATION CHO MINI-BATCH

Batch được sử dụng khi dữ liệu ít, trong thực tế mini-batch được sử dụng phổ biến

- Ưu điểm:
  - Tăng tốc độ huấn luyện
  - Tiết kiệm bộ nhớ

KẾT QUẢ

```
Epoch 0, Loss: 3050.031455
Epoch 100, Loss: 1759.086815
Epoch 200, Loss: 1779.039991
Epoch 300, Loss: 1833.582806
Epoch 400, Loss: 1942.155698
Epoch 500, Loss: 2118.129977
Epoch 600, Loss: 2344.455655
Epoch 700, Loss: 2586.815979
Epoch 800, Loss: 2822.189910
Epoch 900, Loss: 3041.199610
Test Accuracy: 0.9659

Process finished with exit code 0
```