CSE: **Faculty of Computer Science and Engineering**
Thuyloi University

MẠNG NƠ RON NHÂN TẠO

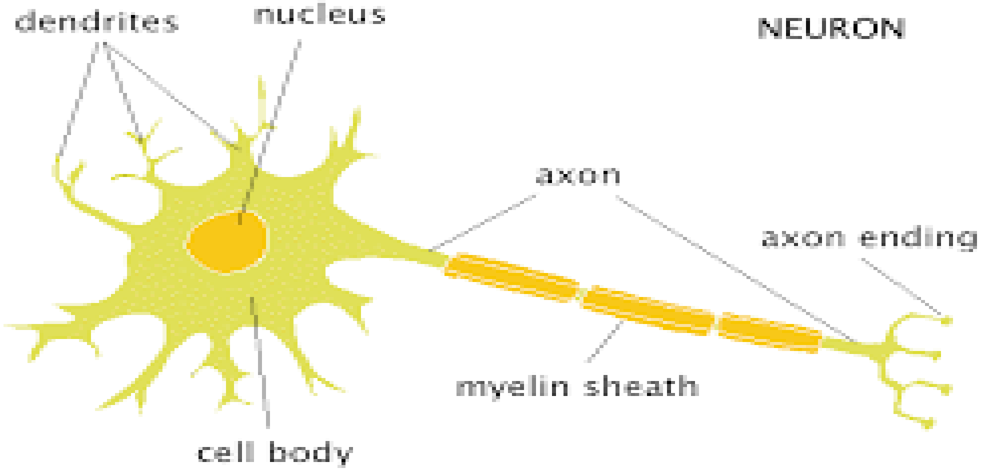
Neural Network

TS. Nguyễn Thị Kim Ngân

Mạng nơron nhân tạo

- Kĩ thuật tính toán mềm dựa trên mô phỏng hoạt động của não bộ (mạng nơron thần kinh)

Nơron thần kinh



- Soma: thân nơron, tiếp nhận hoặc phát ra các xung động thần kinh
- Dendrites: dây thần kinh vào, đưa tín hiệu tới nơron
- Axon: đầu dây thần kinh ra, nối với dây thần kinh vào hoặc tới nhân tế bào của nơron khác thông qua khớp nối
- Synapse: khớp để kích hoạt hoặc kích thích thông tin

Nơron nhân tạo

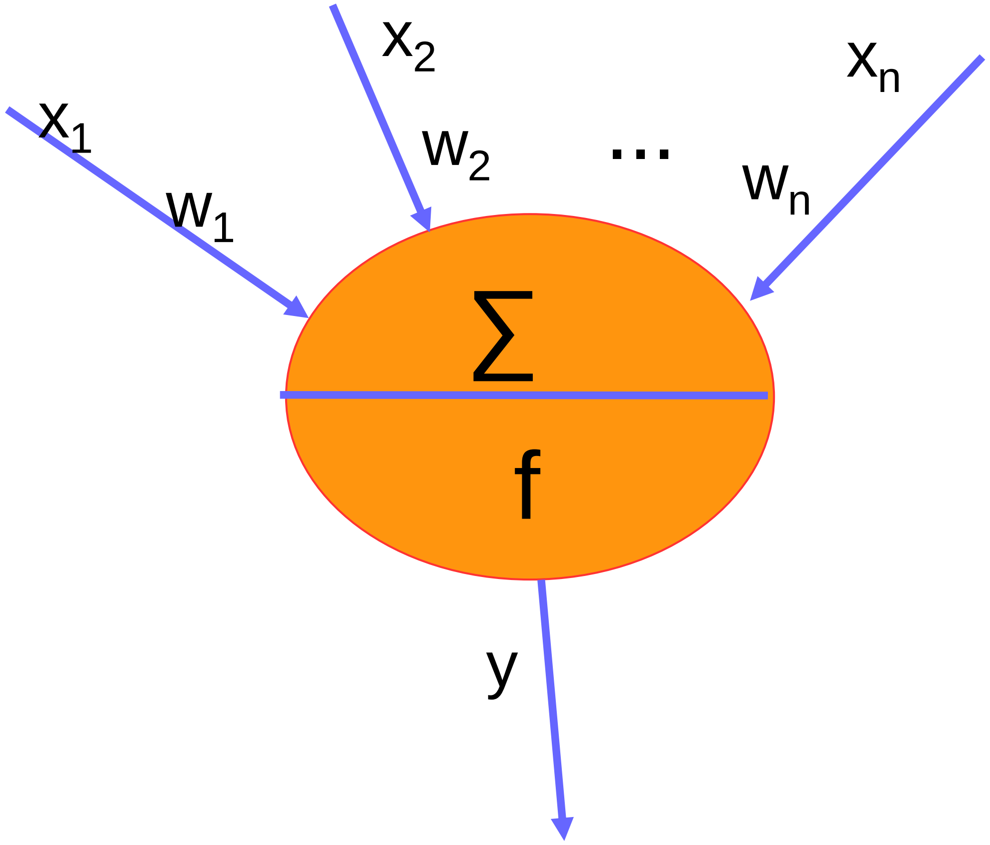
- Tổng thông tin vào của 1 nơron

  $Net = \sum w_i x_i$

- Đầu ra

  $y = f(Net) = f(\sum w_i x_i)$

f được gọi là hàm truyền (transfer function) hay kích hoạt (activation function)



Một số hàm kích hoạt phổ biến

Hàm sigmoid

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

Hàm tanh

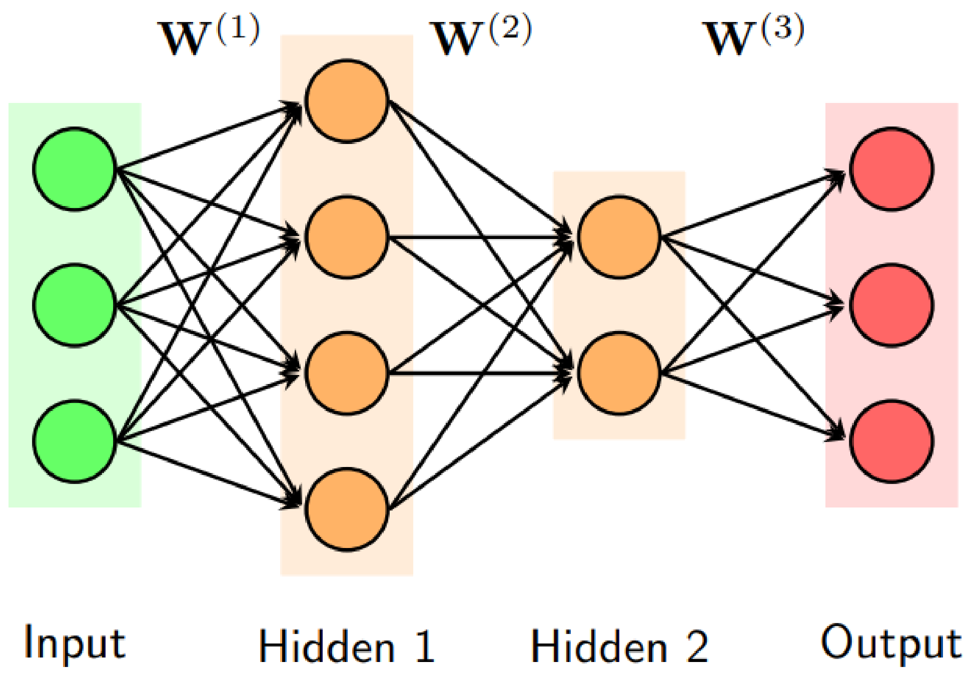
$$f(x) = \tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Hàm ReLu

f(x) = max(0,x)

Cấu trúc mạng nơron

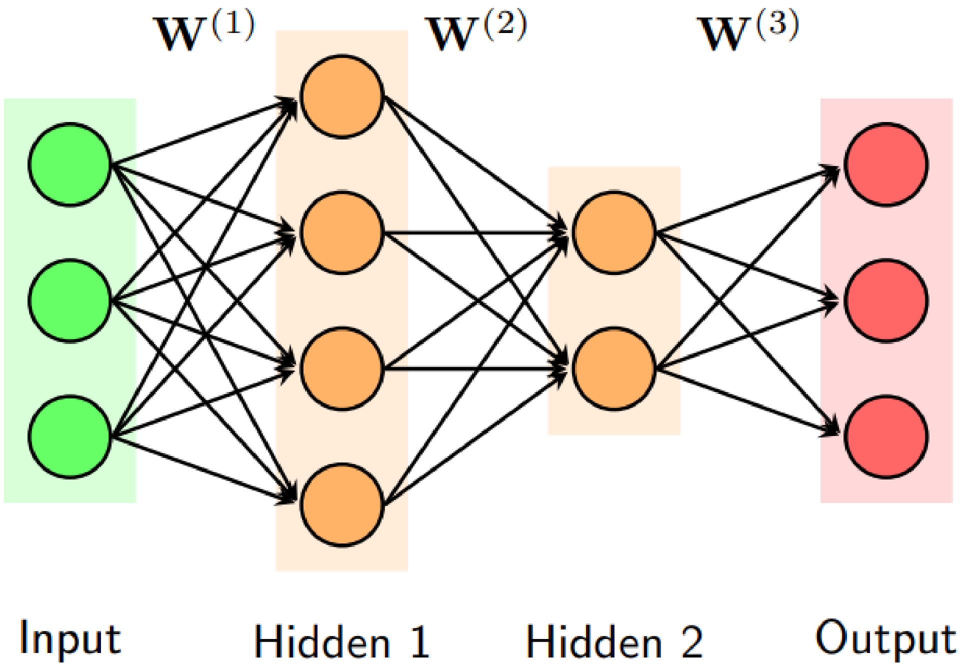
- Lớp (layer) là tập hợp chứa các neuron theo chiều dọc
- Mỗi neural network có 3 loại layer chính:
    - Lớp đầu vào (input layer)
    - Lớp ẩn (hidden layer)
    - Lớp đầu ra (output layer)



- Số lượng layer bằng số hidden layers cộng với 1
- Số lượng layer trong một MLP được ký hiệu là L
    - Trong hình bên, L=3

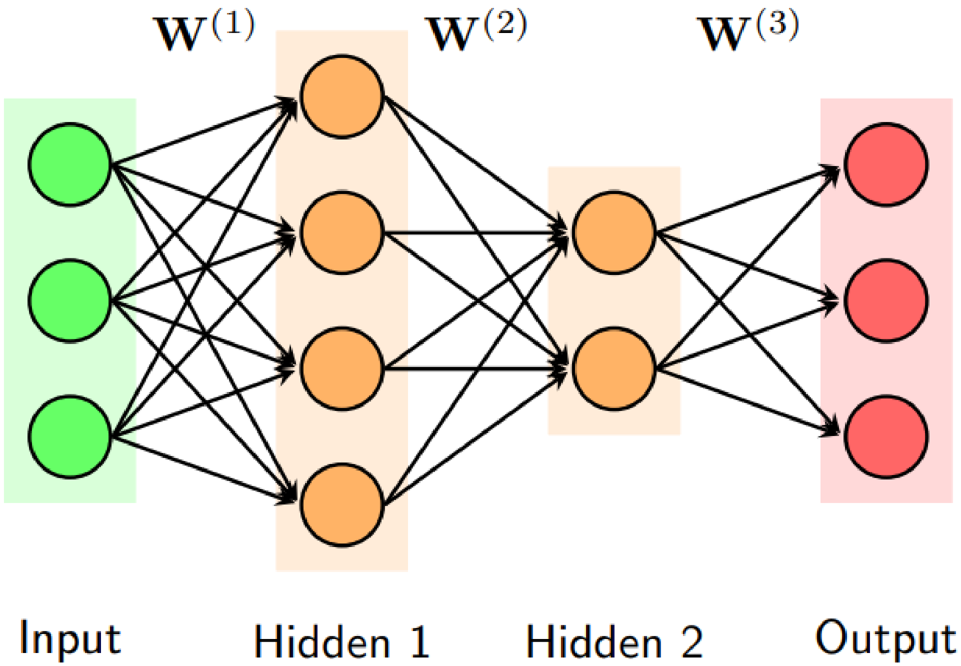
Cấu trúc mạng nơron

- Nơ-ron (Unit/Node): Mỗi node hình tròn được gọi là một nơ-ron
- Mỗi nơ-ron chứa một giá trị số thực

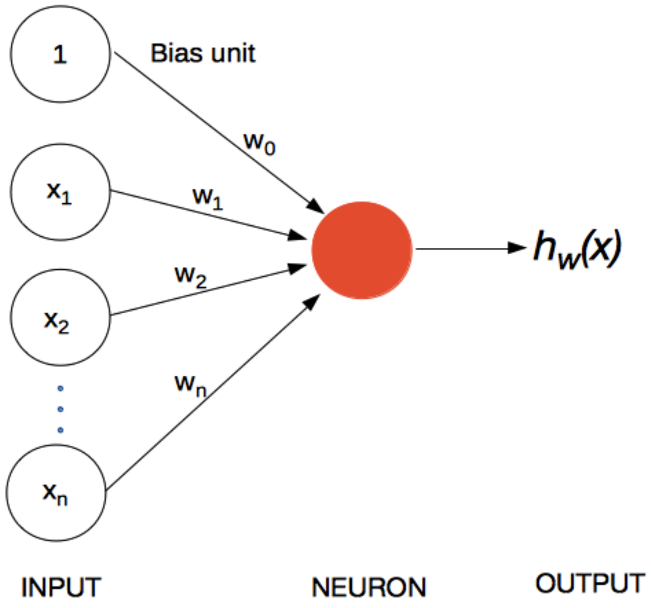


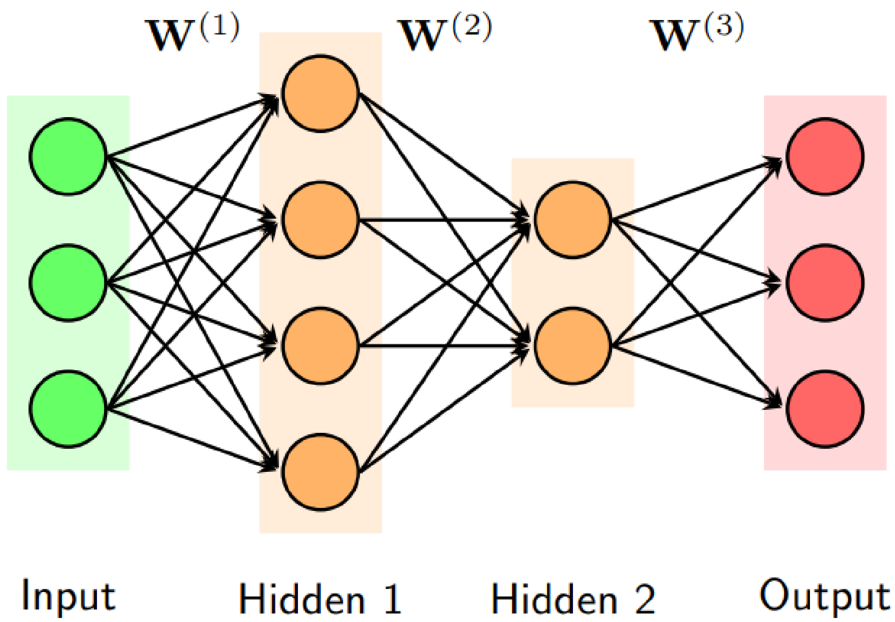
Cấu trúc mạng nơron

- Trọng số (weight) là giá trị nằm trên mỗi cạnh nối các neuron với nhau
- Mỗi giá trị trọng số là một số thực



Mô hình mạng nơron



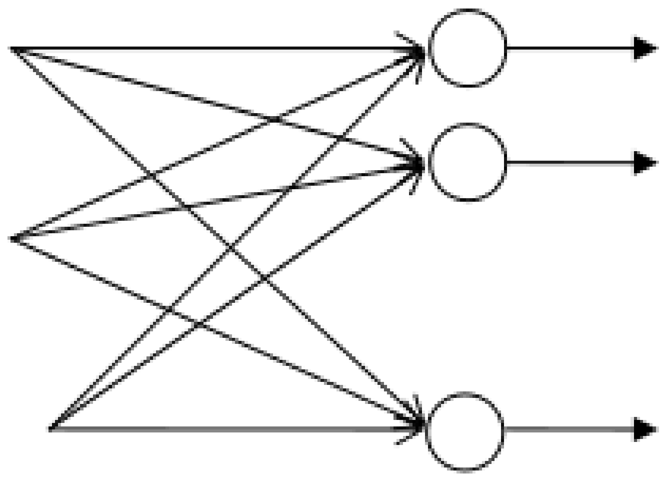


Perceptron

Multilayer neural network

Mô hình mạng nơron

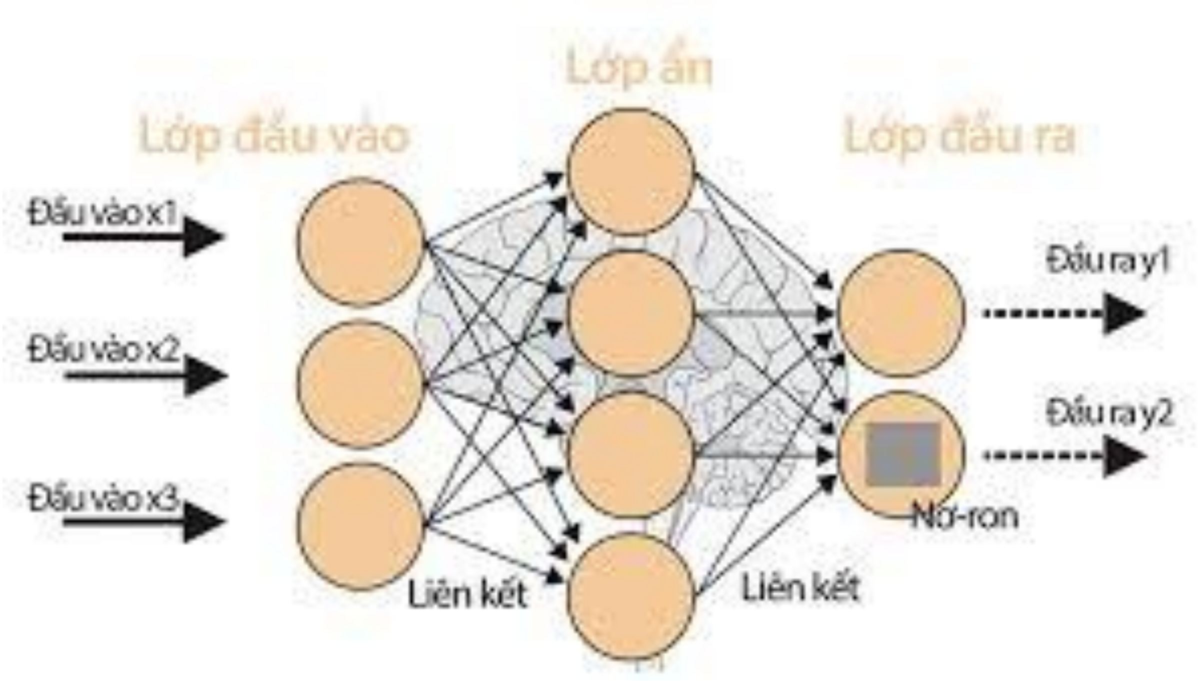
- Không chu trình: mạng truyền thẳng (feed forward NN)



Mạng truyền thẳng 1 lớp

Mô hình mạng nơron

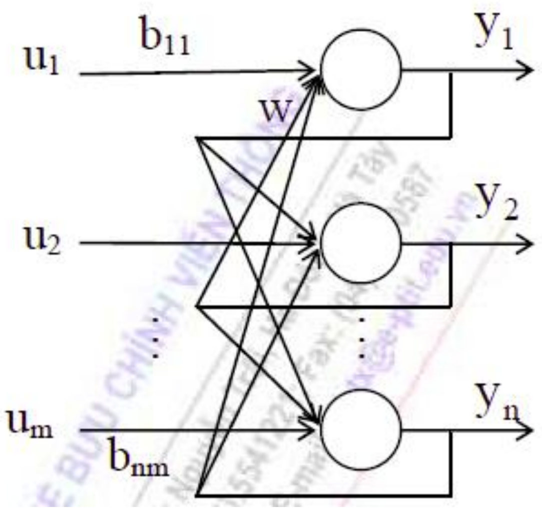
- Không chu trình: mạng truyền thẳng (feed forward NN)



Mạng truyền thẳng 3 lớp

Mô hình mạng nơron

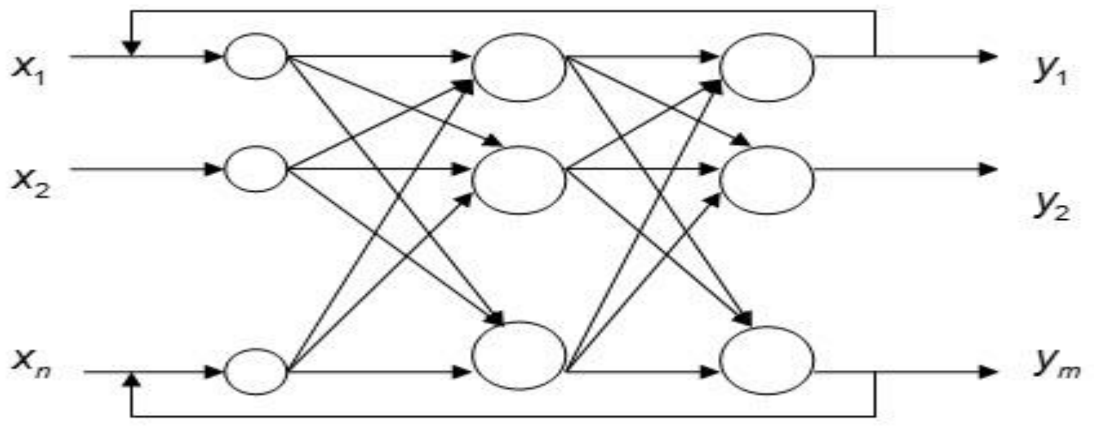
- Có chu trình: mạng hồi quy (recurrent NN)



Mạng hồi quy 1 lớp

Mô hình mạng nơron

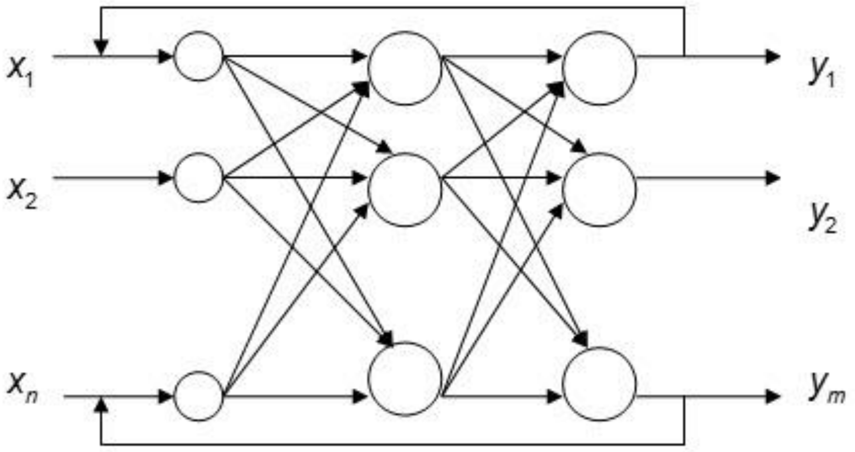
- Có chu trình: mạng hồi quy (recurrent NN). Tín hiệu đầu ra tại lớp i được dung làm đầu vào cho một số lớp 1, 2, …i



Mạng hồi quy nhiều lớp

Mô hình mạng nơron

- Mạng hồi quy nhiều lớp(recurrent neural network)
  - Chứa các liên kết ngược có sự kết nối giữa neural đầu ra với neural đầu vào
  - Lưu lại các trạng thái đầu ra trước đó
  - Trạng thái đầu ra tiếp theo không chỉ phụ thuộc vào các tín hiệu đầu vào mà còn phụ thuộc vào các trạng thái trước đó của mạng



Thuật toán

- **Input:** Tập dữ liệu (X_{train}, y_{train}), kiến trúc mạng nơ ron (số lớp ẩn, số nơ ron của mỗi lớp ẩn), hàm kích hoạt (activation function), hàm mất mát (loss function)
- **Output:** Bộ vector trọng số của các liên kết giữa các nơ ron (W) để hàm mất mát đạt giá trị tối ưu
- **Method:**
  - Phương pháp phổ biến nhất để tối ưu MLP vẫn là Gradient Descent (GD)
  - Để áp dụng GD, chúng ta cần tính được gradient của hàm mất mát theo từng ma trận trọng số W^l, và vector bias b^l

Thiết kế mạng nơron

- Thiết kế mạng
  - Thủ công
  - Tự động: bằng thuật toán học để tín hiệu đầu ra của mạng thu được giống như mong muốn
    - Học cấu trúc: tìm ra cấu trúc mạng (số lớp, số nơron/lớp) hợp lý
    - Học tham số: tìm ra các trọng số liên kết hợp lý (giả sử cấu trúc của mạng là cố định)
- Các kiểu học cho học tham số
  - Có thầy, có giám sát (supervised learning)
  - Không thầy, không có giám sát (unsupervised learning)
  - Tăng cường (enhancement learning)

Các kí hiệu sử dụng

Mỗi output của một unit (trừ các input units)

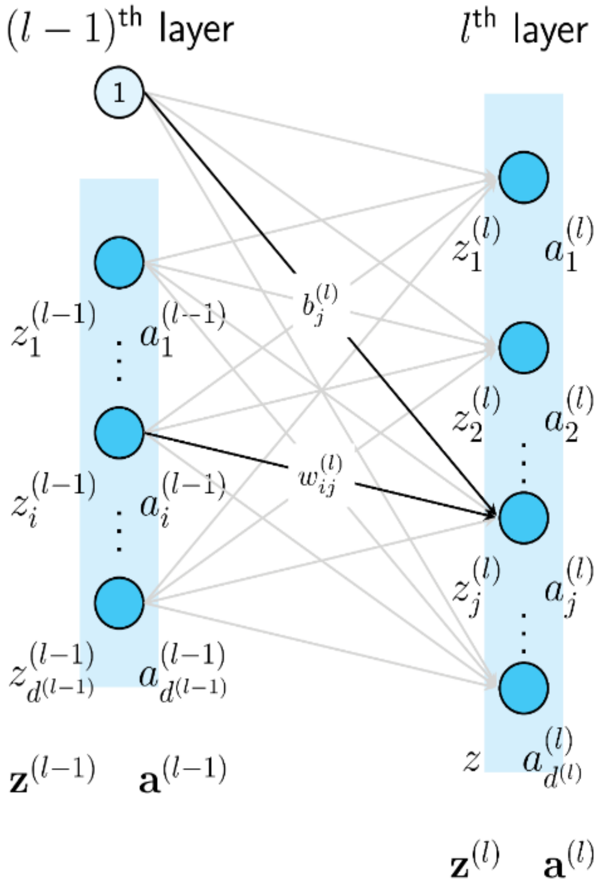
được tính dựa vào công thức:

$$a_i^{(l)} = f(\mathbf{w}_i^{(l)T}\mathbf{a}^{(l-1)} + b_i^{(l)})$$

Trong đó f(.) là một activation function

Ở dạng vector, biểu thức bên trên được viết:

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



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

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

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

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

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

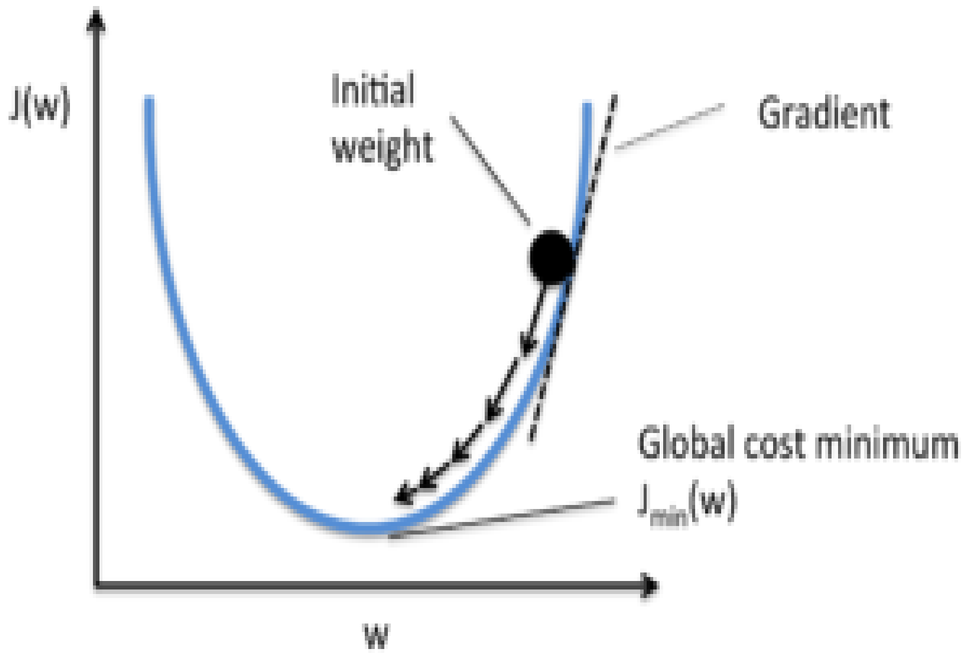
Phương pháp tối ưu mạng Nơron

- Phương pháp phổ biến nhất để tối ưu MLP vẫn là Gradient Descent (GD)
- Để áp dụng GD, chúng ta cần tính được gradient của hàm mất mát theo từng ma trận trọng số W^l, và vector bias b^l

Phương pháp giảm gradient

- Hàm lỗi (tổng chêch lệch giữa đầu ra thu được và đầu ra mong muốn) là 1 hàm f(w) của các trọng số liên kết
- Cần tìm ra 1 trọng số w mà tại đó hàm lỗi là nhỏ nhất
- Hàm lỗi sẽ giảm dần mỗi khi học 1 dữ liệu mẫu. Tức là đầu ra thu được của mạng sẽ tiến sát dần đến đầu ra mong muốn

$W_{i+1}=W_i-f'(W_i , x)$



Feedforward

Feedforward: Tính toán được thực hiện từ đầu đến cuối của network

Tnh *predicted output* (y_hat) với một input x

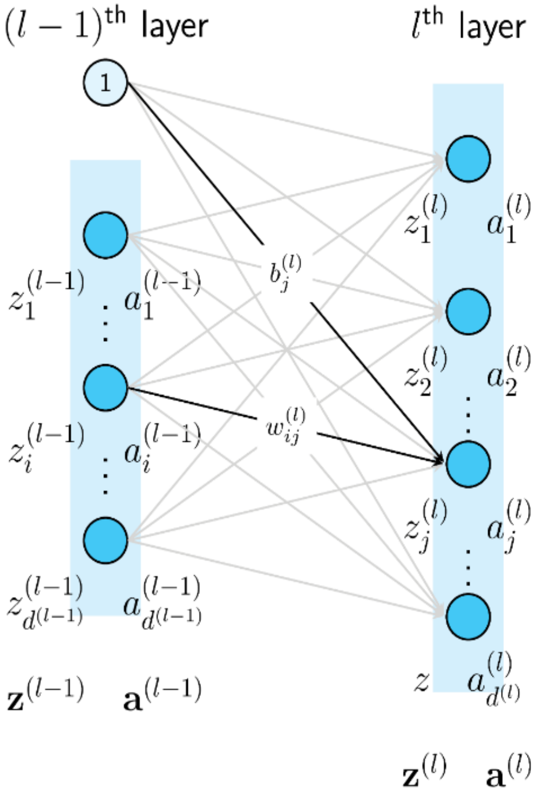
$$\mathbf{a}^{(0)} = \mathbf{x}$$

$$z_i^{(l)} = \mathbf{w}_i^{(l)T}\mathbf{a}^{(l-1)} + b_i^{(l)}$$

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)T}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \quad l = 1, 2, \ldots, L$$

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)}), \quad l = 1, 2, \ldots, L$$

$$\hat{\mathbf{y}} = \mathbf{a}^{(L)}$$



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

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

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

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

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

Backpropagation

Giả sử J(W,b, X,Y) là một hàm mất mát của bài toán

W, b: là tập tập hợp tất cả các ma trận trọng số giữa các layers và biases của mỗi layer

X,Y: là cặp dữ liệu huấn luyện với mỗi cột tương ứng với một điểm dữ liệu

Để có thể áp dụng các gradient-based methods, cần tính được:

$$\frac{\partial J}{\partial \mathbf{W}^{(l)}}; \frac{\partial J}{\partial \mathbf{b}^{(l)}}, \quad l = 1, 2, \dots, L$$

Backpropagation

Đạo hàm theo tham số w của tầng output (layer Lth)

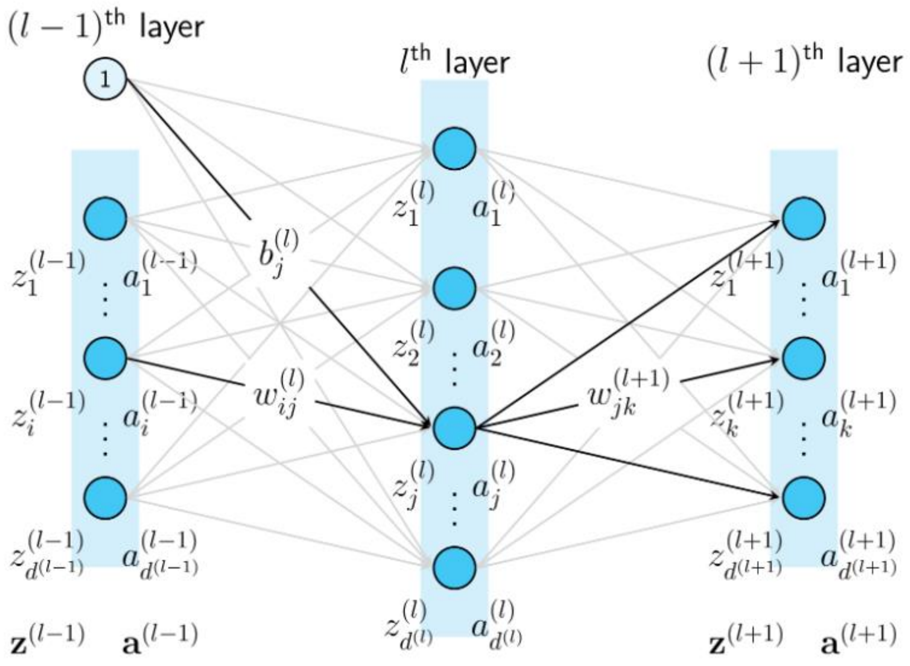
$$\frac{\partial J}{\partial w_{ij}^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \cdot \frac{\partial z_j^{(L)}}{\partial w_{ij}^{(L)}} = e_j^{(L)} a_i^{(L-1)}$$

$$e_j^{(L)} = \frac{\partial J}{\partial z_j^{(L)}}$$

$$\frac{\partial z_j^{(L)}}{\partial w_{ij}^{(L)}} = a_i^{(L-1)}$$

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

$$\frac{\partial J}{\partial b_j^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \cdot \frac{\partial z_j^{(L)}}{\partial b_j^{(L)}} = e_j^{(L)}$$



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

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

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

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

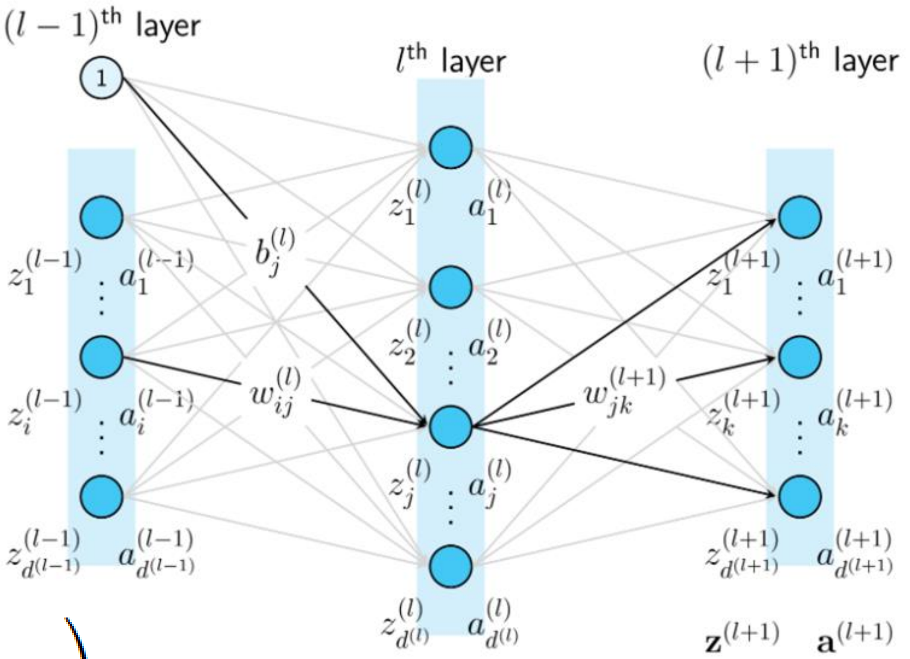
$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$

Backpropagation

Đạo hàm theo tham số w của tầng thứ *l*

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \frac{\partial J}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}}$$

$$= e_j^{(l)} a_i^{(l-1)}$$



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

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

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

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

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

$$e_j^{(l)} = \frac{\partial J}{\partial z_j^{(l)}} = \frac{\partial J}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}}$$

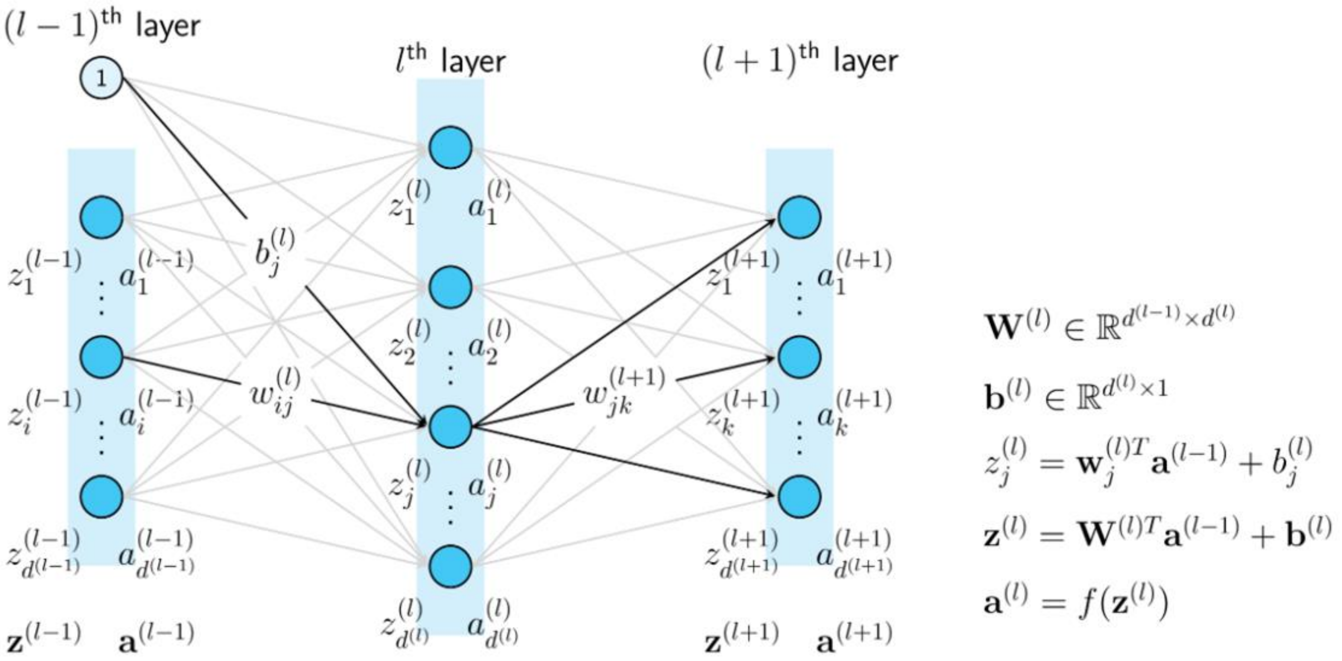
$$= \left(\sum_{k=1}^{d^{(l+1)}} \frac{\partial J}{\partial z_k^{(l+1)}} \cdot \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} \right) f^{(l)\prime}(z_j^{(l)}) = \left(\sum_{k=1}^{d^{(l+1)}} e_k^{(l+1)} w_{jk}^{(l+1)} \right) f^{(l)\prime}(z_j^{(l)})$$

vì $a_j^{(l)} = f^{(l)}(z_j^{(l)})$.

Backpropagation

Đạo hàm theo tham số b của tầng thứ *l*

$$\frac{\partial J}{\partial b_j^{(l)}} = e_j^{(l)}$$



Thuật toán Backpropagation

Thuật toán 16.1: Backpropagation tới $w_{ij}^{(l)}, b_i^{(l)}$

1. *Bước feedforward: Với 1 giá trị đầu vào* $\mathbf{x}$*, tính giá trị đầu ra của network, trong quá trình tính toán, lưu lại các giá trị activation* $\mathbf{a}^{(l)}$ *tại mỗi layer.*
2. *Với mỗi unit j ở output layer, tính*

$$e_j^{(L)} = \frac{\partial J}{\partial z_j^{(L)}}; \quad \frac{\partial J}{\partial w_{ij}^{(L)}} = a_i^{(L-1)} e_j^{(L)}; \quad \frac{\partial J}{\partial b_j^{(L)}} = e_j^{(L)} \tag{16.19}$$

3. *Với $l = L-1, L-2, ..., 1$, tính:*

$$e_j^{(l)} = \left(\mathbf{w}_{j:}^{(l+1)} \mathbf{e}^{(l+1)}\right) f'(z_j^{(l)}) \tag{16.20}$$

4. *Cập nhật đạo hàm cho từng hệ số*

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = a_i^{(l-1)} e_j^{(l)}; \quad \frac{\partial J}{\partial b_j^{(l)}} = e_j^{(l)} \tag{16.21}$$

Thuật toán Backpropagation

Thuật toán 16.2: Backpropagation tới $\mathbf{W}^{(l)}$ và vector bias $\mathbf{b}^{(l)}$

1. *Bước feedforward: Với một giá trị đầu vào* $\mathbf{x}$, *tính giá trị đầu ra của network, trong quá trình tính toán, lưu lại các activation* $\mathbf{a}^{(l)}$ *tại mỗi layer.*
2. *Với output layer, tính*

$$\mathbf{e}^{(L)} = \nabla_{\mathbf{z}^{(L)}} J \in \mathbb{R}^{d^{(L)}}; \ \nabla_{\mathbf{W}^{(L)}} J = \mathbf{a}^{(L-1)} \mathbf{e}^{(L)T} \in \mathbb{R}^{d^{(L-1)} \times d^{(L)}}; \ \nabla_{\mathbf{b}^{(L)}} J = \mathbf{e}^{(L)}$$

3. *Với* $l = L-1, L-2, ..., 1$, *tính:*

$$\mathbf{e}^{(l)} = \left(\mathbf{W}^{(l+1)} \mathbf{e}^{(l+1)}\right) \odot f'(\mathbf{z}^{(l)}) \in \mathbb{R}^{d^{(l)}} \tag{16.22}$$

   trong đó $\odot$ *là element-wise product hay Hadamard product tức lấy từng thành phần của hai vector nhân với nhau để được vector kết quả.*
4. *Cập nhật đạo hàm cho các ma trận trọng số và vector bias:*

$$\nabla_{\mathbf{W}^{(l)}} J = \mathbf{a}^{(l-1)} \mathbf{e}^{(l)T} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}; \quad \nabla_{\mathbf{b}^{(l)}} J = \mathbf{e}^{(l)} \tag{16.23}$$

Thuật toán Backpropagation

Thuật toán 16.3: Backpropagation tới $\mathrm{W}^{(l)}$ và bias $\mathrm{b}^{(l)}$ (mini-batch)

1. *Bước feedforward: Với toàn bộ dữ liệu (batch) hoặc một nhóm dữ liệu (mini-batch) đầu vào* $\mathbf{X}$, *tính giá trị đầu ra của network, trong quá trình tính toán, lưu lại các activation* $\mathbf{A}^{(l)}$ *tại mỗi layer. Mỗi cột của* $\mathbf{A}^{(l)}$ *tương ứng với một cột của* $\mathbf{X}$, *tức một điểm dữ liệu đầu vào.*
2. *Với output layer, tính*

$$\mathbf{E}^{(L)} = \nabla_{\mathbf{Z}^{(L)}} J; \quad \nabla_{\mathbf{W}^{(L)}} J = \mathbf{A}^{(L-1)} \mathbf{E}^{(L)T}; \quad \nabla_{\mathbf{b}^{(L)}} J = \sum_{n=1}^{N} \mathbf{e}_n^{(L)} \tag{16.24}$$

3. *Với* $l = L-1, L-2, \ldots, 1$, *tính:*

$$\mathbf{E}^{(l)} = \left(\mathbf{W}^{(l+1)} \mathbf{E}^{(l+1)}\right) \odot f'(\mathbf{Z}^{(l)}) \tag{16.25}$$

trong đó $\odot$ *là element-wise product hay Hadamard product tức lấy từng thành phần của hai ma trận nhân với nhau để được ma trận kết quả.*

4. *Cập nhật đạo hàm cho ma trận trọng số và vector biases:*

$$\nabla_{\mathbf{W}^{(l)}} J = \mathbf{A}^{(l-1)} \mathbf{E}^{(l)T}; \quad \nabla_{\mathbf{b}^{(l)}} J = \sum_{n=1}^{N} \mathbf{e}_n^{(l)} \tag{16.26}$$