TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN

MẠNG NORON NHÂN TẠO và GIẢI THUẬT DI TRUYỀN

Neural Network & Genetic Algorithm

Biên soạn: ThS.Phạm Đình Tàu
pdtai@ntt.edu.vn
0985.73.39.39

CHƯƠNG 6

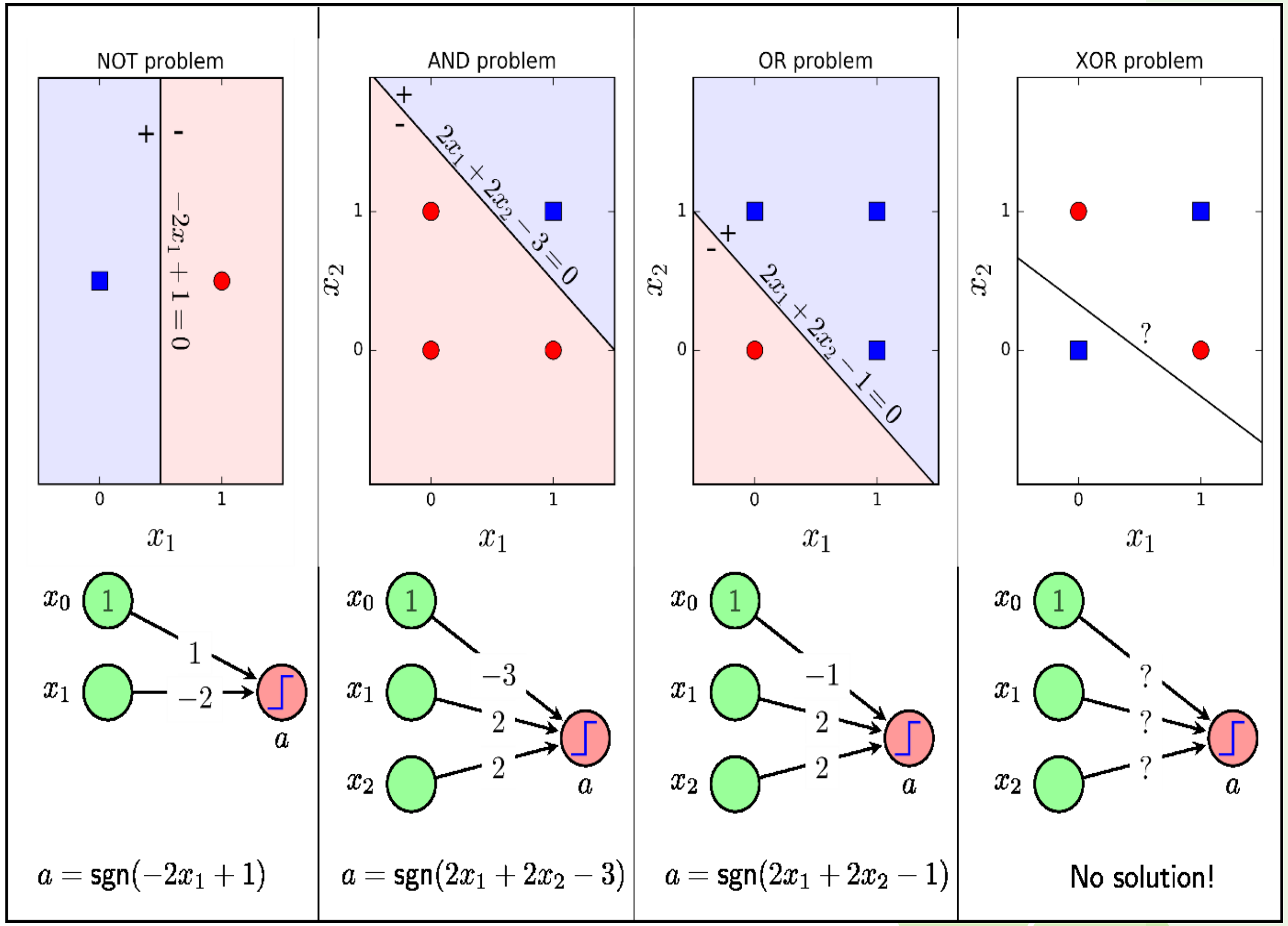
Phương pháp huấn luyện MLP (Multi Layer Perceptron)

MỤC TIÊU

- ✓ **Giới thiệu một giải pháp để tính toán đạo hàm của hàm mục tiêu với mạng đa lớp sử dụng kỹ thuật lan truyền ngược (Backpropagation)**
- ✓ **Thực hành thông qua một ví dụ**

HẠN CHẾ CỦA MẠNG PERCEPTRON 1 LỚP

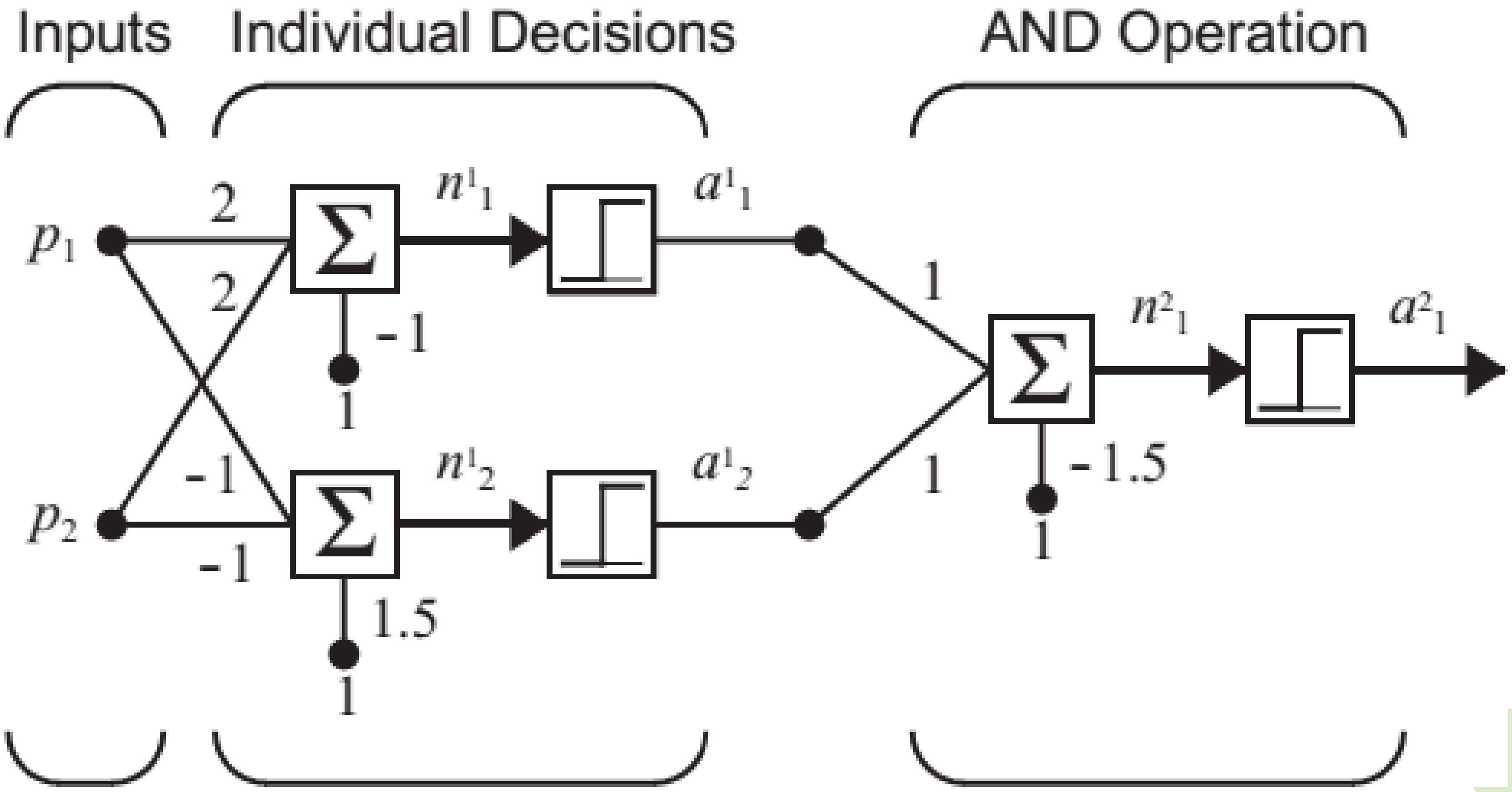
- **Perceptron**: mạng 1 lớp đầu vào và một lớp đầu ra, không có *hidden layer*
- Mạng perceptron không thể phân loại các dữ liệu có phân tách phi tuyến (*XOR function)*



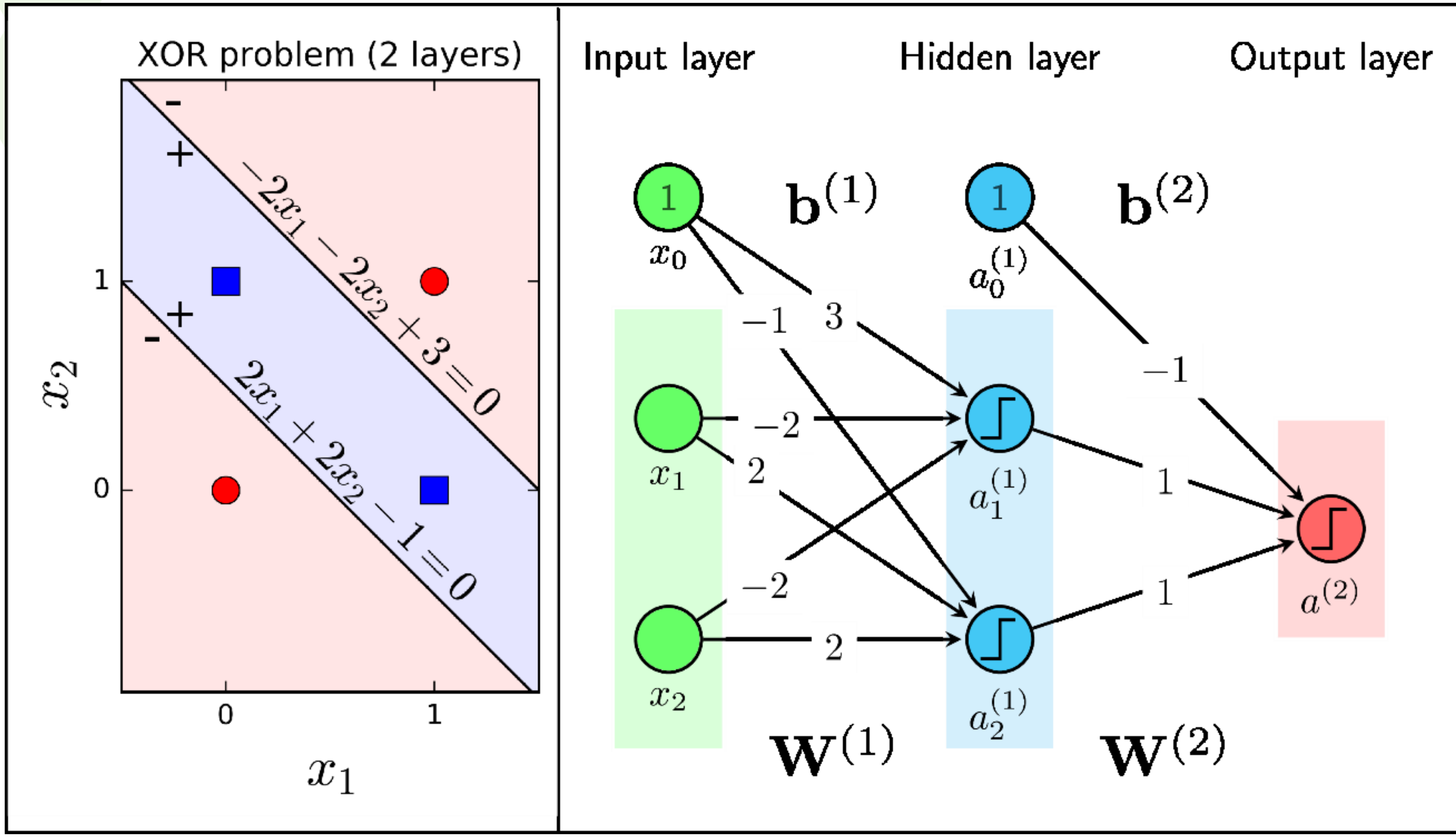
KHẮC PHỤC HẠN CHẾ CỦA PERCEPTRON 1 LỚP

- Thực tế có nhiều mạng đa lớp có thể giải quyết bài toán XOR
- Một lời giải:
  - *Thiết kế 2 neuron để tạo ra hai bao đóng*
  - *Thêm một neuron sau đó để kết hợp hai bao đóng lại thành một (toán tử AND)*

GIẢI PHÁP CHO TOÁN TỬ XOR



GIẢI PHÁP CHO TOÁN TỬ XOR



$$\mathbf{W}^{(1)} = \begin{bmatrix} -2 & 2 \\ -2 & 2 \end{bmatrix}; \quad \mathbf{b}^{(1)} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad \mathbf{b}^{(2)} = [-1]$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}; \mathbf{a}^{(1)} = \begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \end{bmatrix}$$

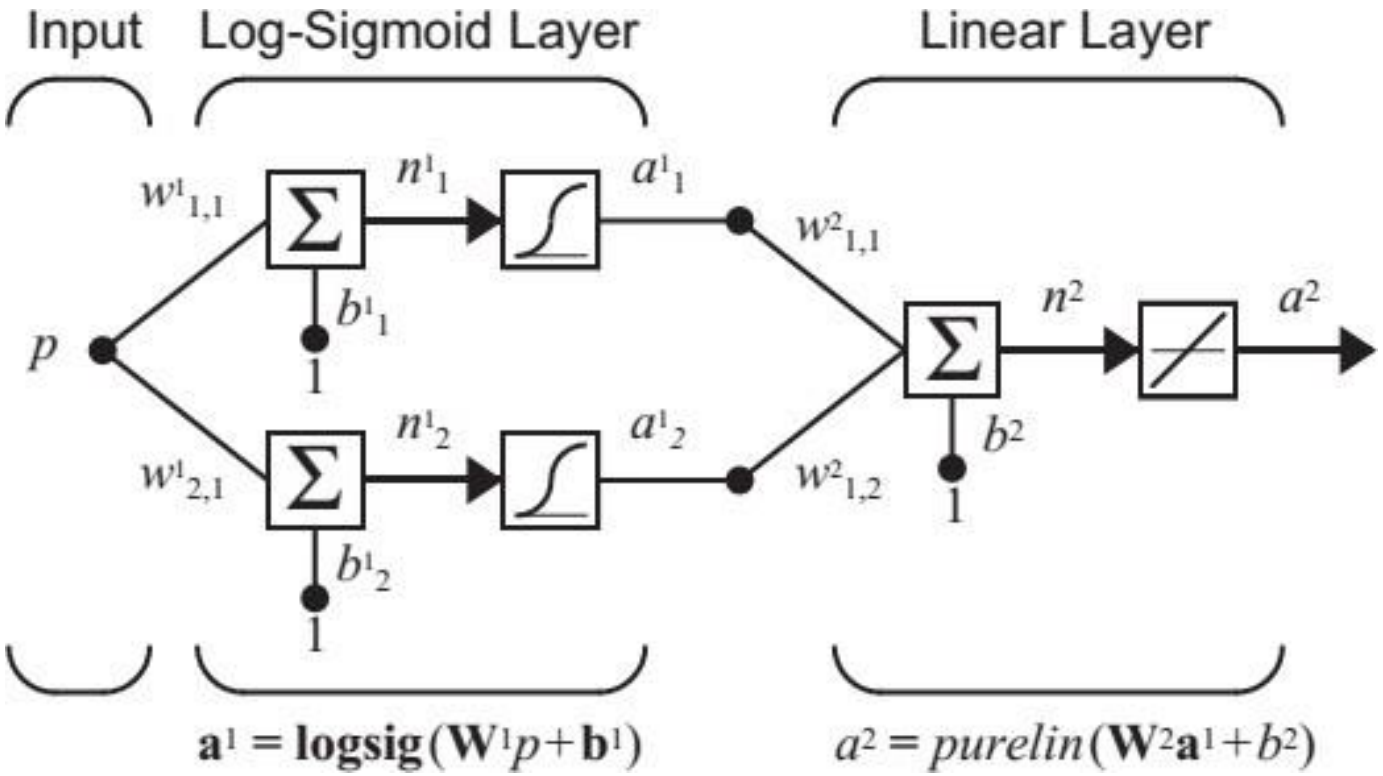
$$\mathbf{a}^{(1)} = f\left(\mathbf{W}^{(1)T}\mathbf{x} + \mathbf{b}^{(1)}\right)$$

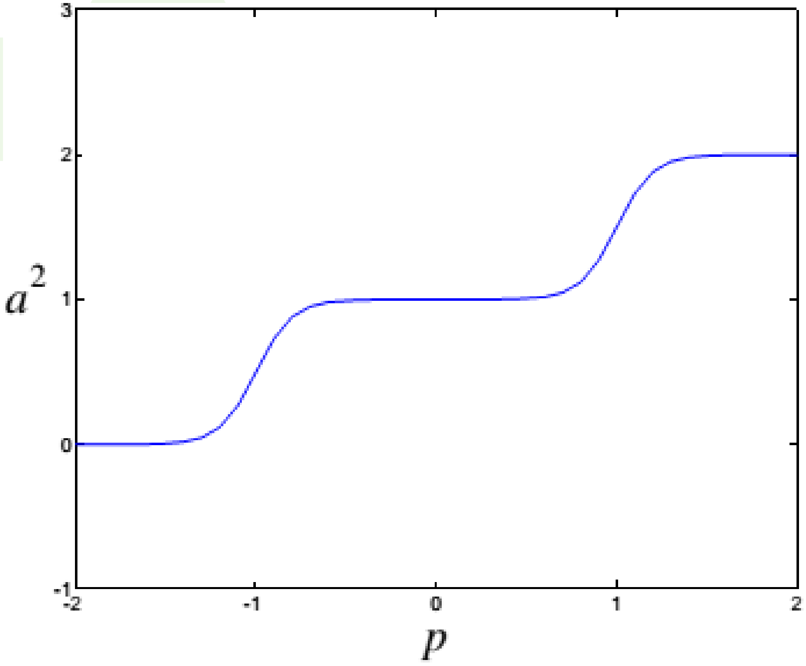
$$a^{(2)} = f\left(\mathbf{W}^{(2)T}\mathbf{a}^{(1)} + \mathbf{b}^{(2)}\right)$$

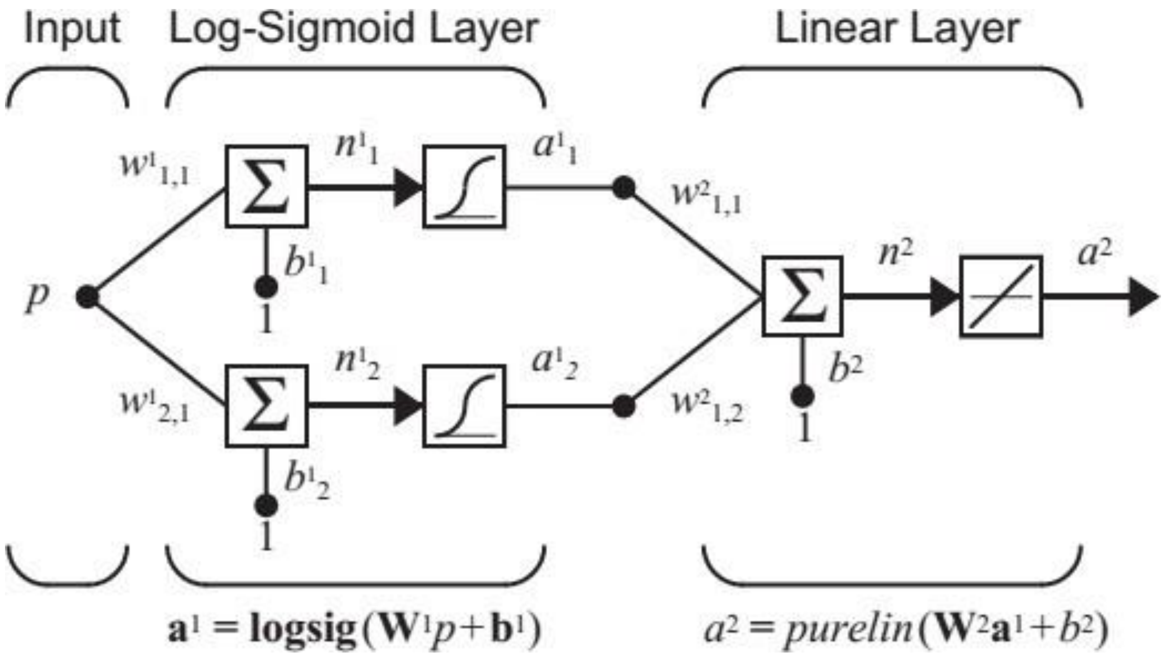
$f(.) = \mathrm{sgn}(.)$ (element-wise)

XẤP XỈ HÀM (FUNCTION APPROXIMATION)

- Ngoài ứng dụng trong phân lớp, các mạng neuron đa lớp cũng được sử dụng để tạo ra các hàm xấp xỉ trong các hệ thống điều khiển







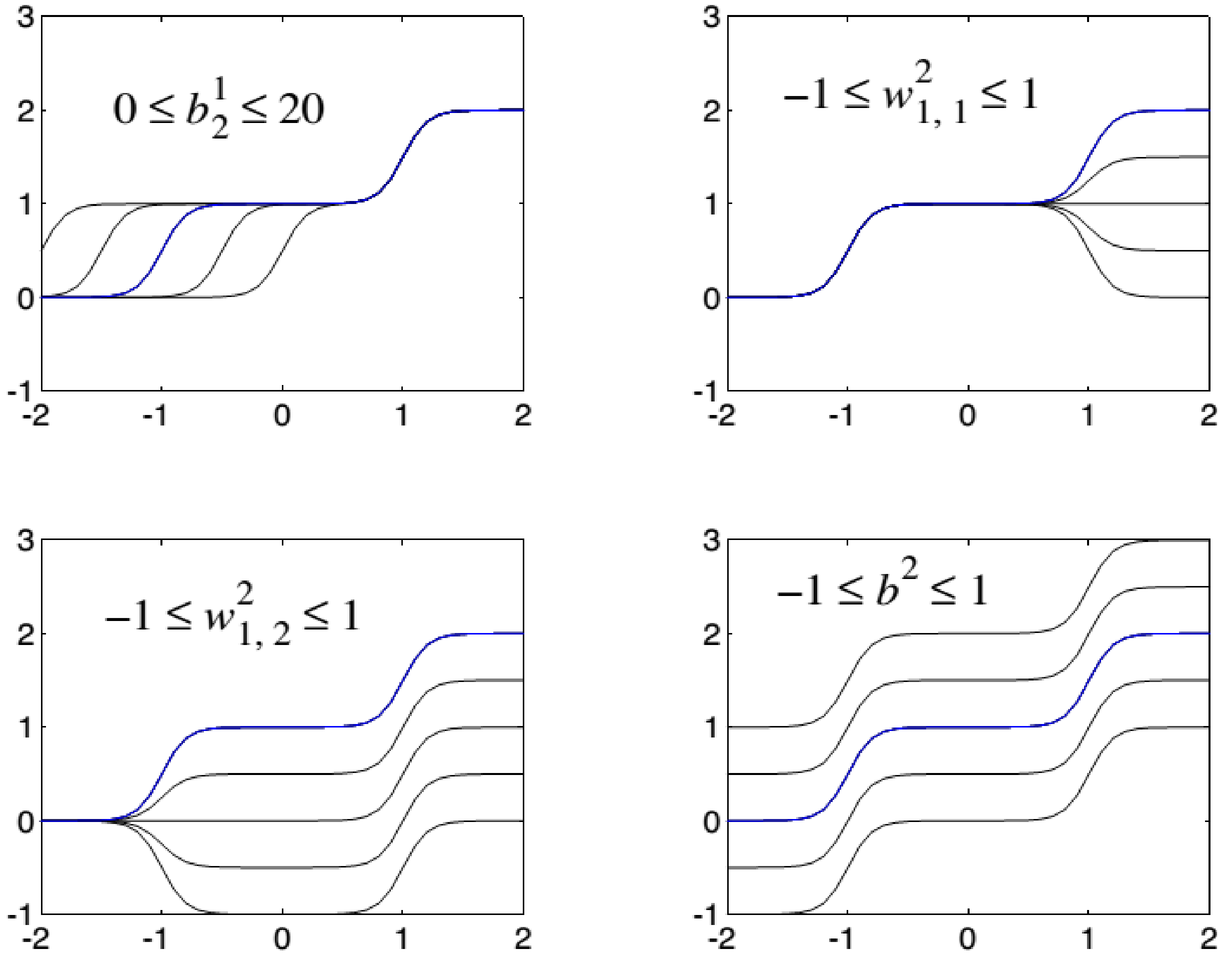
- Giả thiết các thông số của mạng như sau

$$w^1_{1,1} = 10,\ w^1_{2,1} = 10,\ b^1_1 = -10,\ b^1_2 = 10,$$

$$w^2_{1,1} = 1,\ w^2_{1,2} = 1,\ b^2 = 0.$$

XẤP XỈ HÀM (FUNCTION APPROXIMATION)

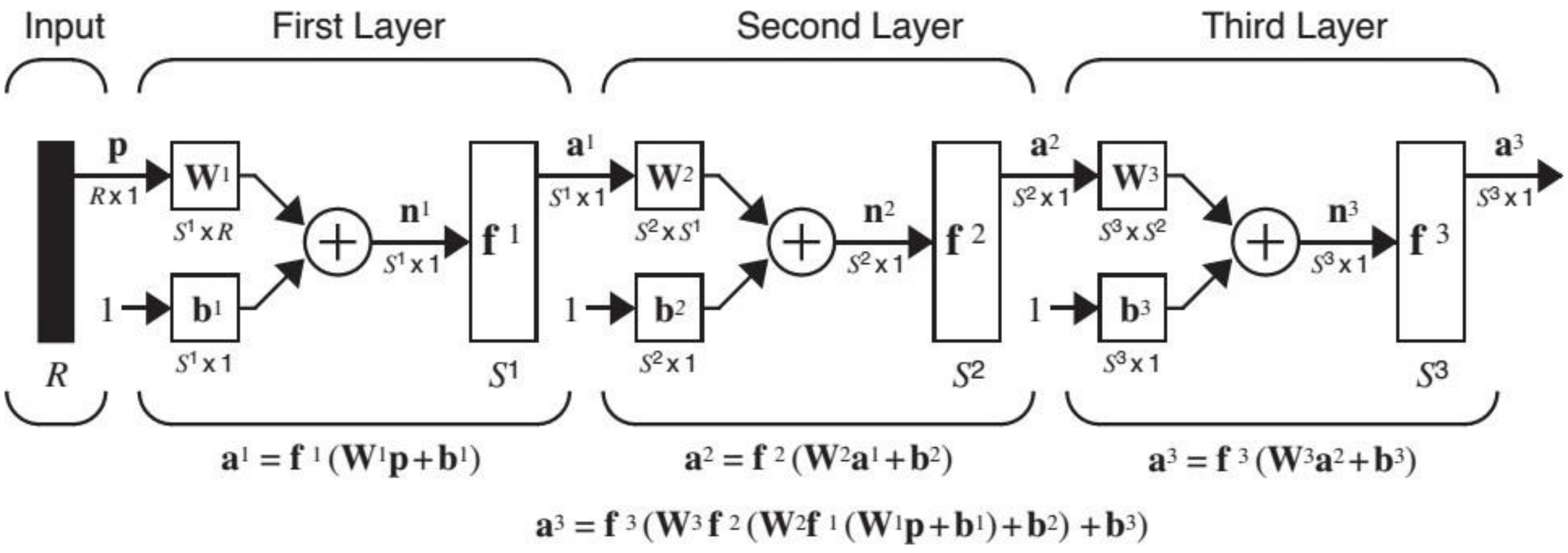
- Đây là xấp xỉ hàm hai bước. Nếu ta thêm các trọng số thì sẽ làm thay đổi hình dáng của đáp ứng.



MẠNG ĐA TẦNG

- Mạng với 1 tầng ẩn có thể biểu diễn bất kỳ hàm Boolean nào
- Khả năng của mạng nhiều tầng đã được khám phá từ lâu, tuy nhiên chỉ đến những năm 80 người ta mới biết cách để huấn luyện các mạng này
- Mạng nhiều tầng, mỗi tầng có hàm kích hoạt tuyến tính thì vẫn chỉ có thể biểu diễn các hàm tuyến tính
- Để biểu diễn các hàm phi tuyến thì các hàm kích hoạt phải là hàm phi tuyến

MULTILAYER NETWORK



$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^{m} + \mathbf{b}^{m+1}) \qquad m = 0, 2, \ldots, M-1$$

$$\mathbf{a}^{0} = \mathbf{p}$$

$$\mathbf{a} = \mathbf{a}^{M}$$

HUẤN LUYỆN MẠNG MLP

Training Set

$$\{\mathbf{p}_1,\mathbf{t}_1\}, \{\mathbf{p}_2,\mathbf{t}_2\}, \dots, \{\mathbf{p}_Q,\mathbf{t}_Q\}$$

Mean Square Error

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2]$$

Vector Case

$$F(\mathbf{x}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})]$$

Approximate Mean Square Error (Single Sample)

$$\hat{F}(\mathbf{x}) = (\mathbf{t}(k)-\mathbf{a}(k))^T(\mathbf{t}(k)-\mathbf{a}(k)) = \mathbf{e}^T(k)\mathbf{e}(k)$$

Approximate Steepest Descent

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha\frac{\partial\hat{F}}{\partial w_{i,j}^m} \qquad b_i^m(k+1) = b_i^m(k) - \alpha\frac{\partial\hat{F}}{\partial b_i^m}$$

GIẢI THUẬT HUẤN LUYỆN MẠNG MLP

- Giải thuật GD được sử dụng
- Tuy nhiên việc tính toán Gradient của hàm mục tiêu trên tất cả các tầng là rất phức tạp
- Giải thuật Backpropagation cho một giải pháp hiệu quả và đơn giản để tính toán đạo hàm của hàm mục tiêu theo trọng số và bias ở các tầng khác nhau

http://neuralnetworksanddeeplearning.com/chap2.html

GIẢI THUẬT HUẤN LUYỆN MẠNG MLP

- Sử dụng kỹ thuật lan truyền ngược
- Kỹ thuật này được nghiên cứu đề xuất nhiều lần bởi các nhà khoa học
  - *Bryson an Ho [1969]*
  - *Werbos [1974]*
  - *Parker [1985]*
  - *Rumelhart et al. [1986]*

CHAIN RULE

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw}$$

Example

$$f(n) = \cos(n) \qquad n = e^{2w} \qquad f(n(w)) = \cos(e^{2w})$$

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw} = (-\sin(n))(2e^{2w}) = (-\sin(e^{2w}))(2e^{2w})$$

Application to Gradient Calculation

$$\frac{\partial \hat{F}}{\partial w_{i,j}^{m}} = \frac{\partial \hat{F}}{\partial n_{i}^{m}} \times \frac{\partial n_{i}^{m}}{\partial w_{i,j}^{m}} \qquad \frac{\partial \hat{F}}{\partial b_{i}^{m}} = \frac{\partial \hat{F}}{\partial n_{i}^{m}} \times \frac{\partial n_{i}^{m}}{\partial b_{i}^{m}}$$

GRADIENT CALCULATION

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m$$

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1} \qquad \frac{\partial n_i^m}{\partial b_i^m} = 1$$

Sensitivity

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m}$$

Gradient

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \qquad \frac{\partial \hat{F}}{\partial b_i^m} = s_i^m$$

STEEPEST DESCENT

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(\mathrm{k}) - \alpha s_i^m a_j^{m-1} \qquad b_i^m(k+1) = b_i^m(k) - \alpha s_i^m$$

$$\boxed{\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \qquad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m}$$

$$\mathbf{s}^m \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \dfrac{\partial \hat{F}}{\partial n_1^m} \\ \dfrac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \dfrac{\partial \hat{F}}{\partial n_{S^m}^m} \end{bmatrix}$$

Next Step: Compute the Sensitivities (Backpropagation)

JACOBIAN MATRIX

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^{m}} \equiv \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix}$$

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial\left(\sum_{l=1}^{S^m} w_{i,l}^{m+1} a_l^m + b_i^{m+1}\right)}{\partial n_j^m} = w_{i,j}^{m+1}\frac{\partial a_j^m}{\partial n_j^m}$$

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = w_{i,j}^{m+1}\frac{\partial f^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1}\dot{f}^m(n_j^m)$$

$$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}$$

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^{m}} = \mathbf{W}^{m+1}\dot{\mathbf{F}}^m(\mathbf{n}^m)$$

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{S^m}^m) \end{bmatrix}$$

BACKPROPAGATION (Sensitivities)

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}}$$

$$\boxed{\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}}$$

The sensitivities are computed by starting at the last layer, and then propagating backwards through the network to the first layer.

$$\mathbf{s}^M \to \mathbf{s}^{M-1} \to \ldots \to \mathbf{s}^2 \to \mathbf{s}^1$$

INITIALIZATION (last layer)

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial (\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^M} (t_j - a_j)^2}{\partial n_i^M} = -2(t_i - a_i)\frac{\partial a_i}{\partial n_i^M}$$

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = \dot{f}^M(n_i^M)$$

$$s_i^M = -2(t_i - a_i)\dot{f}^M(n_i^M)$$

$$\boxed{\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a})}$$

SUMMARY

Forward Propagation

$$\mathbf{a}^0 = \mathbf{p}$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \qquad m = 0, 2, \ldots, M-1$$

$$\mathbf{a} = \mathbf{a}^M$$

Backpropagation

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a})$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1} \qquad m = M-1, \ldots, 2, 1$$

Weight Update

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T \qquad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m$$

Thank you !