Mạng neuron
và ứng dụng trong xử lý tín hiệu

Giảng viên

Trần Thị Thanh Hải

International Research Institute MICA

Multimedia, Information, Communication & Applications

UMI 2954

Hanoi University of Science and Technology

1 Dai Co Viet - Hanoi - Vietnam

Bài 4:
Mạng Perceptron

Source: Neural network design

Mục tiêu

- **Câu hỏi của bài 3:**
  - Làm thế nào để xác định được ma trận trọng số khi số đầu vào lớn
  - Trong trường hợp nào không thể hiện thị được ranh giới ?
- **Mục tiêu**
  - Mô tả giải thuật huấn luyện mạng
  - Định nghĩa thể nào là luật học
  - Phát triển các luật học cho mạng
  - Thảo luận về ưu nhược điểm của mạng Perceptron một lớp

Giới thiệu chung

- **Năm1943**, hai nhà khoa học Warren McCulloch and Walter Pitts đã giới thiệu mô hình neuron nhân tạo lần đầu tiên
- **Đặc trưng cơ bản của mô hình neuron là**
  - Tổng trọng số của tín hiệu đầu vào được so sánh với một giá trị ngưỡng
  - Nếu tổng lớn hơn hoặc bằng ngưỡng thì đầu ra là 1
  - Nếu tổng nhỏ hơn ngưỡng thì đầu ra là 0
- **Họ cho rằng:**
  - Bất kỳ một mạng các neuron như vậy đều có thể tạo ra bất kỳ một hàm toán hay logic
  - Không giống neuron sinh học, các tham số của mạng phải được thiết kế vì không có pp học nào

Giới thiệu chung

- **Năm 1950**, Frank Rosenblatt và một số nhà khoa học đã phát triển một lớp các mạng neuron gọi là **Perceptron**
- **Các neuron trong mạng này giống như các neuron của** Warren McCulloch and Walter Pitts
- **Đóng góp chính**
  - Đưa vào luật học để huấn luyện các mạng perceptron
  - Đã chứng minh rằng với luật học như vậy có có thể hội tụ về một tập các tham số đúng nếu như tồn tại các trọng số như thế của mạng
  - Luật học đơn giản, tự động
  - Có thể học với các bộ tham số khởi tạo ngẫu nhiên

Giới thiệu chung

- **Tuy nhiên mạng perceptron có một số hạn chế**
  - Không có khả năng cài đặt được một số hàm cơ bản
  - Nó được giải quyết với khi được thiết kế thành nhiều lớp
- **Ngày nay mạng Perceptron vẫn là một mạng có ý nghĩa quan trọng vì nó nhanh và tin cậy**
- **Việc hiểu các thao tác của mạng Perceptron sẽ giúp hiểu các mạng phức tạp hơn**

Luật học

- **Là một thủ tục để thay đổi các trọng số và bias của mạng (giải thuật huấn luyện mạng)**
- **Có nhiều luật học khác nhau**
  - Học có giám sát
  - Học không có giám sát
  - Học tăng cường

Học có giám sát

- **Được cung cấp một tập dữ liệu thể hiện đáp ứng của mạng với các dữ liệu đầu vào**

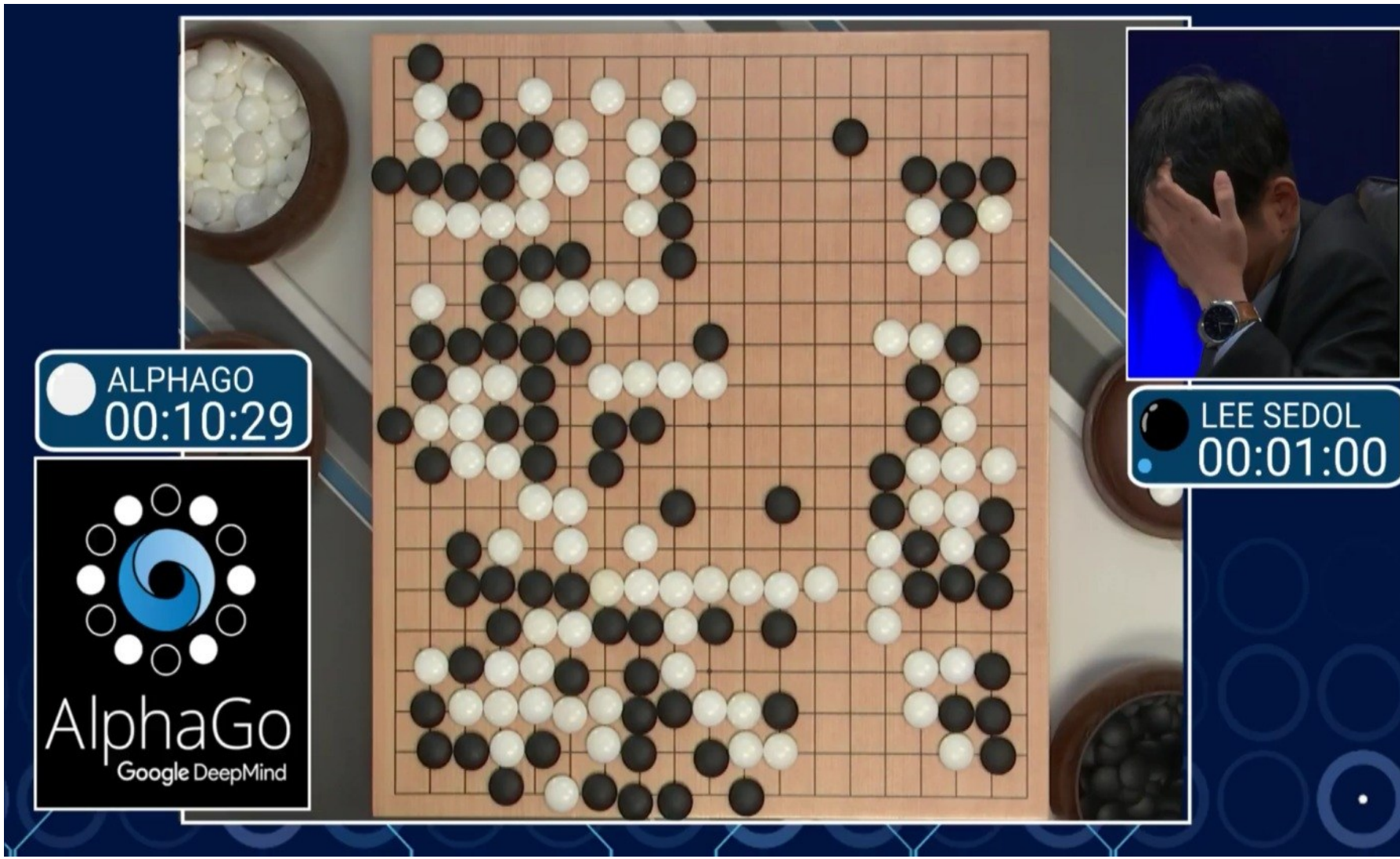
$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

- **Trong đó:**
  - P_i là đầu vào, t_i là **target** đầu ra
  - Do đầu vào được đưa vào mạng, **đáp ứng** của mạng được so sánh với đầu ra t_i
- **Luật học:** làm thay đổi trọng số và bias của mạng để cho đáp ứng đầu ra gần với target nhất

Học tăng cường

- Tương tự như học có giám sát
- Tuy nhiên thay vì đưa ra target, cung cấp một giá trị score thể hiện hiệu năng của mạng với đầu vào cho trước
- Học này kém thông dụng hơn học giám sát, thường được áp dụng trong các hệ thống điều khiển

Học tăng cường

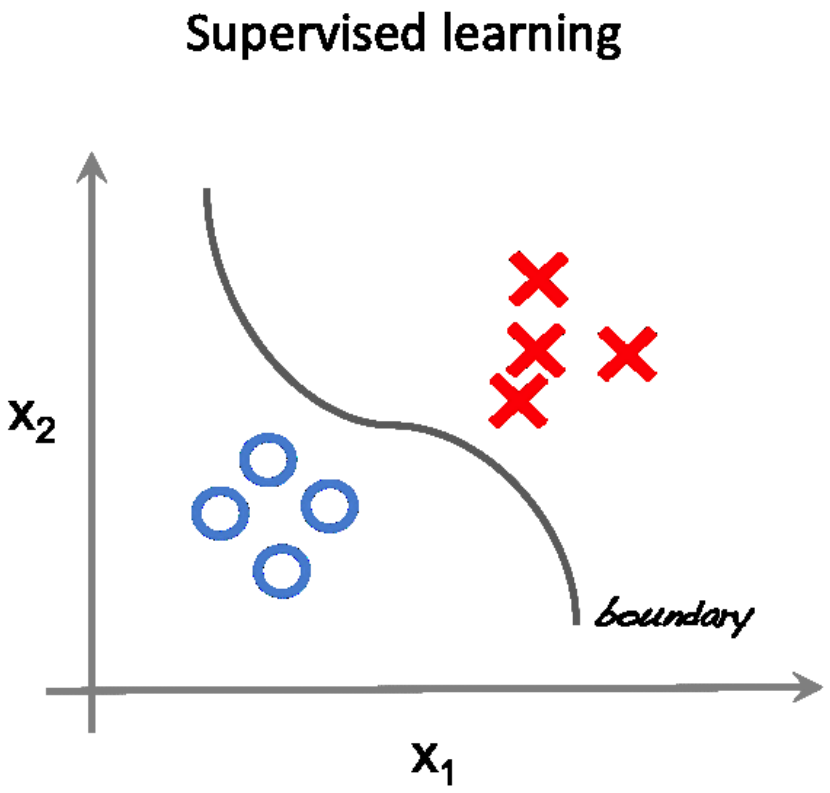


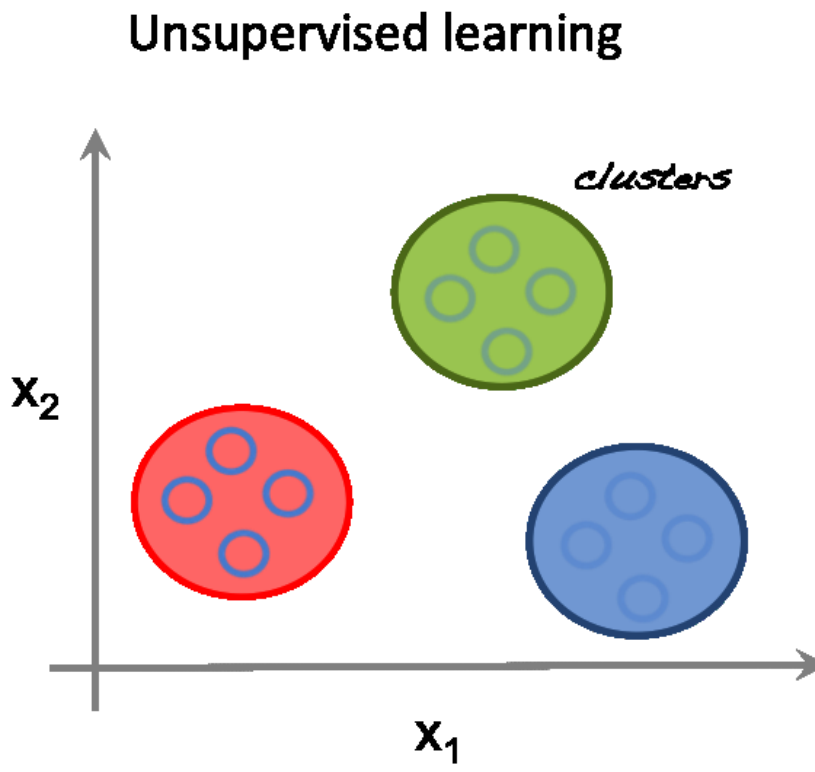
Học tăng cường

- Cờ vây có độ phức tạp cực kỳ cao
- Tổng số nước đi : 10^{761} cờ vua (10^{120})
- Không thể áp dụng IBM DeepBlue (thuật toán thắng người trong môn cờ vua 20 năm trước đây)
- **AlphaGo:**
  - Supervised: dữ liệu từ ván cờ do con người chơi với nhau được đưa vào huấn luyện
  - Reinforcement: tự chơi với chính nó để tìm ra nước đi mới

Học không giám sát

- Chỉ cung cấp đầu vào, không cung cấp đầu ra
- Mạng học để phân cụm dữ liệu





Supervised or unsupervised ?

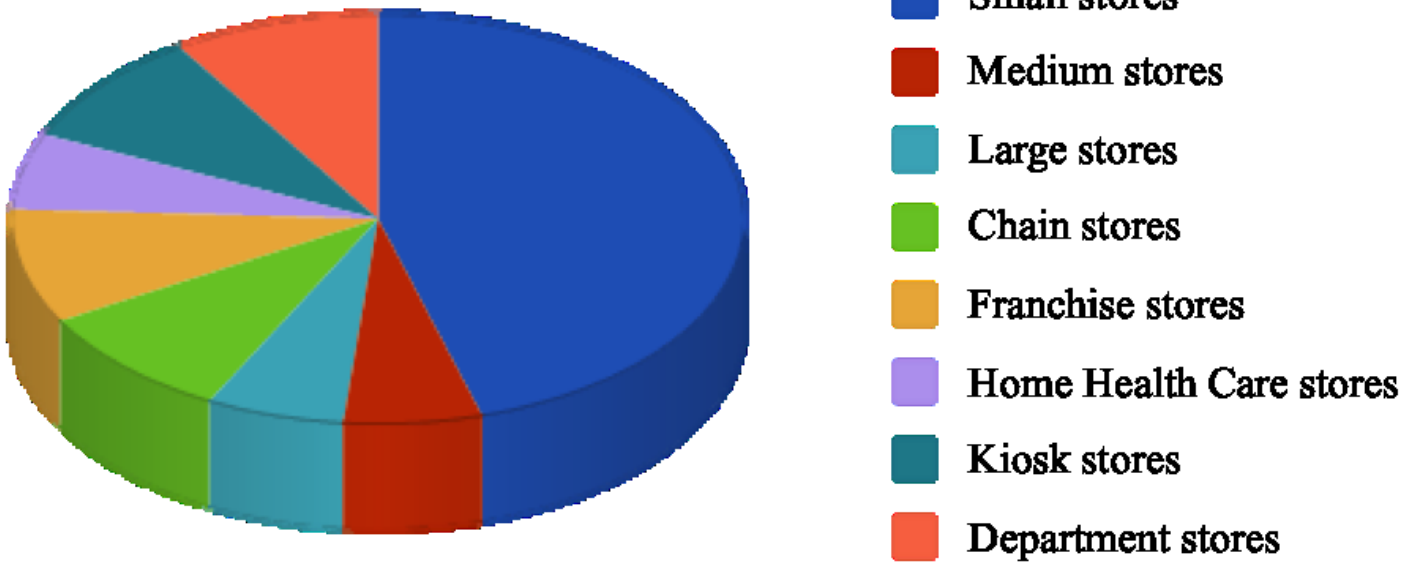
Face recognition

Supervised or unsupervised ?

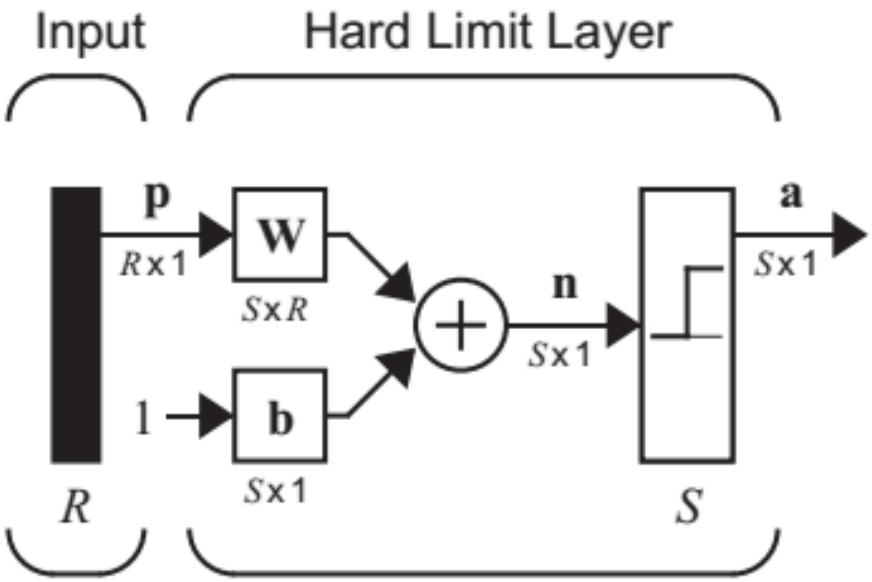
Learning style

Supervised or unsupervised ?

Market segmentation



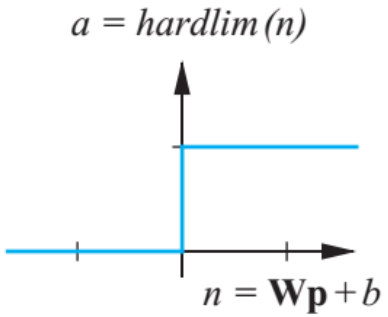
Kiến trúc mạng Perceptron



$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b})$$

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$${}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix} \qquad \mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

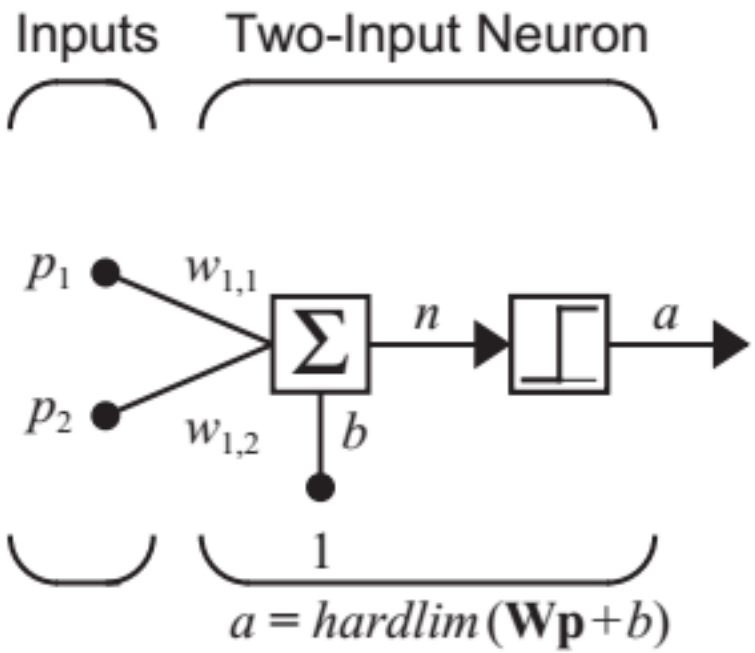


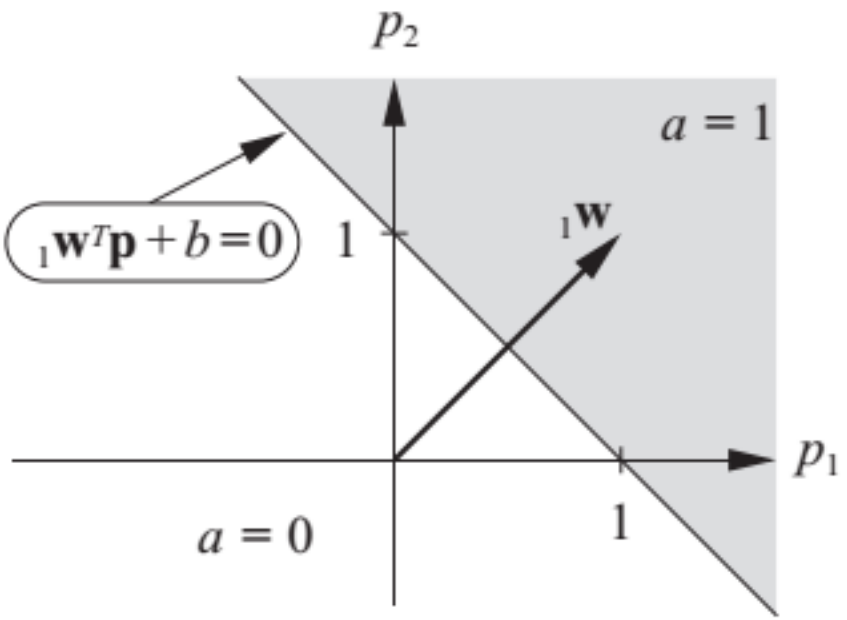
$$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\mathbf{p} + b_i)$$

$$a = hardlim(n) = \begin{cases} 1 & if \quad n \geq 0 \\ 0 & otherwise. \end{cases}$$

Mạng perceptron hai đầu vào

$w_{1,1} = 1 \qquad w_{1,2} = 1 \qquad b = -1$



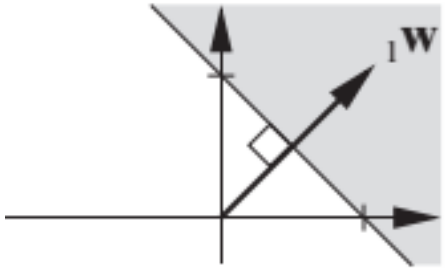


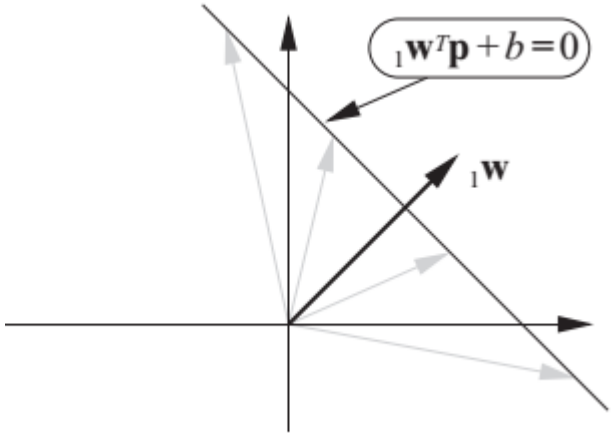
$$a = hardlim(_1\mathbf{w}^{\mathrm{T}}\mathbf{p} + b) = hardlim(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

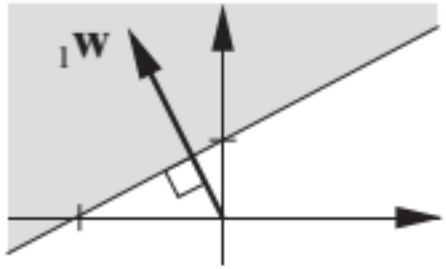
Bao đóng quyết định

$$ {}_1\mathbf{w}^T\mathbf{p} + b = 0 \qquad {}_1\mathbf{w}^T\mathbf{p} = -b $$

- Tất cả các điểm nằm trên bao đóng đều có cùng tích trong với vector trọng số
- Có cùng điểm chiếu lên vector trọng số
- Nằm trên đường thẳng vuông góc với vector trọng số

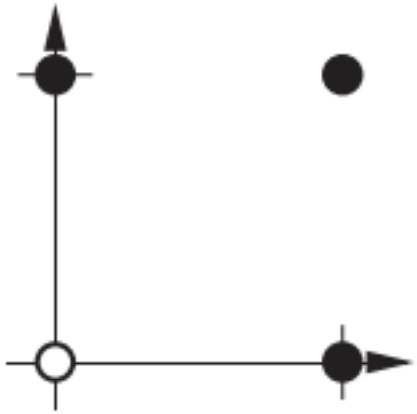




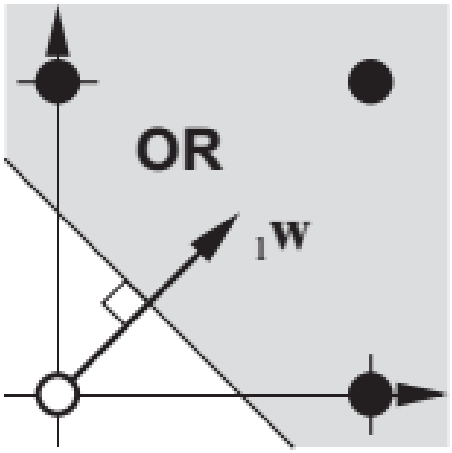


Ví dụ - Toán tử OR

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\} \quad \left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 1\right\}$$



Ví dụ - Toán tử OR



- Vector trọng số phải vuông góc với bao đóng

$$_1\mathbf{w} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

- Lấy một điểm trên bao đóng để tìm bias

$$_1\mathbf{w}^T\mathbf{p} + b = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} + b = 0.25 + \mathrm{b} = 0 \quad \Rightarrow \quad b = -0.25$$

Mạng perceptron nhiều neurons

- Mỗi neuron có một bao đóng quyết định riêng

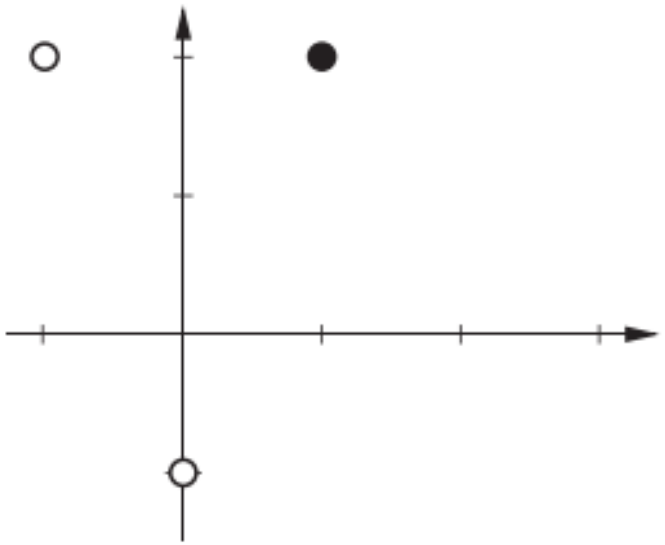
$$_i\mathbf{w}^T\mathbf{p} + b_i = 0$$

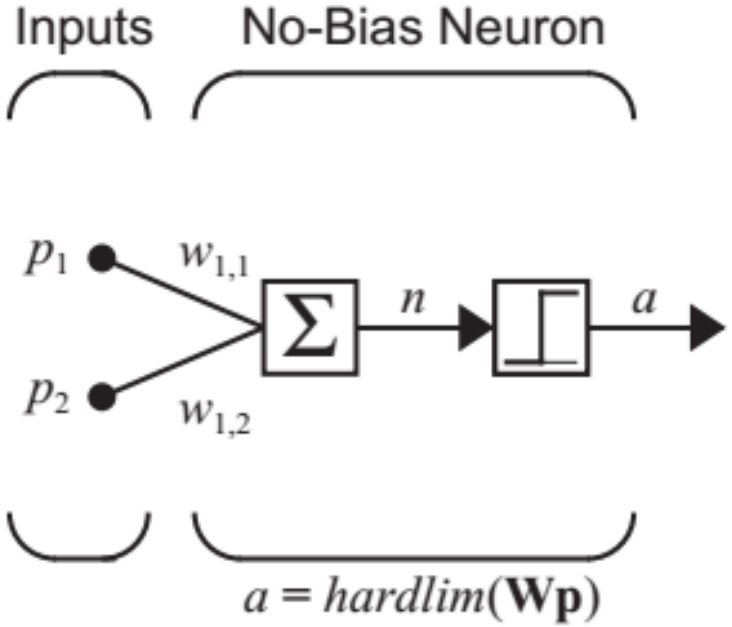
- Mỗi neuron có thể phân các vector đầu vào thành hai loại
- Như vậy một mạng S neuron có thể phân các vector đầu vào thành 2^S

Một ví dụ

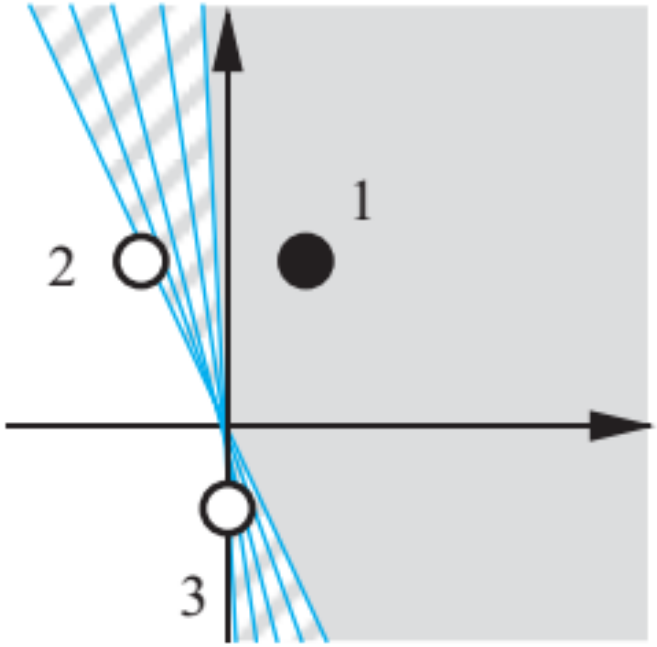
$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

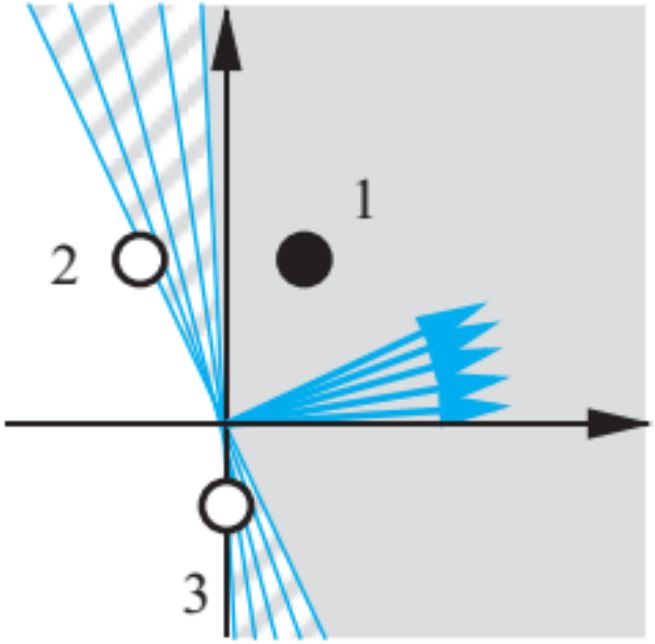
$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}$$





Ranh giới phân tách

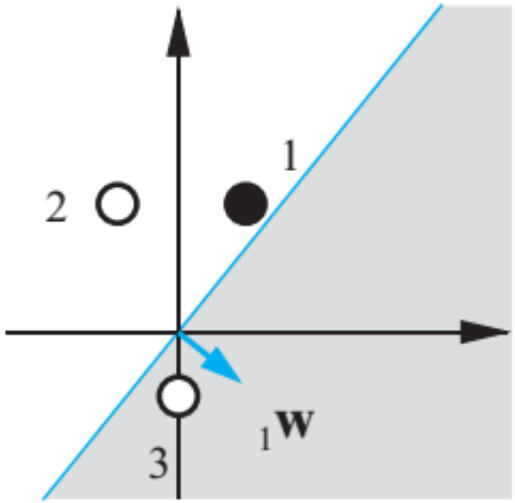




- Khi bias = 0, ranh giới đi qua gốc tọa độ
- Tồn tại nhiều đường ranh giới cho phép phân tách các điểm
- Độ dài của vector trọng số không quan trọng

Khởi tạo

- Mạng không có bias (b = 0)
- Khởi tạo ngẫu nhiên trọng số



$$_1\mathbf{w} = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix}$$

- Đưa đầu vào vào mạng => phân loại sai

$$a = hardlim(_1\mathbf{w}^T\mathbf{p}_1) = hardlim\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$$

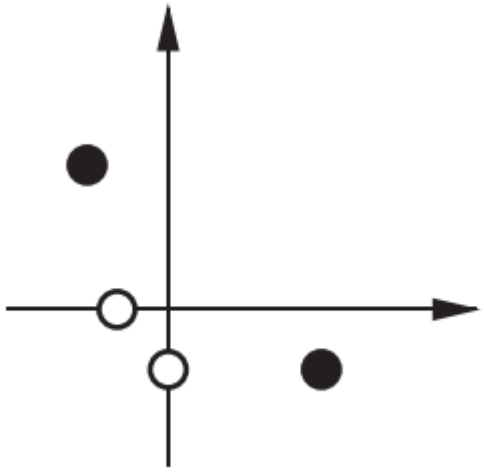
$$a = hardlim(-0.6) = 0$$

- Cần phải thay đổi trọng số để nó chỉ đến gần p1 hơn

Ý tưởng

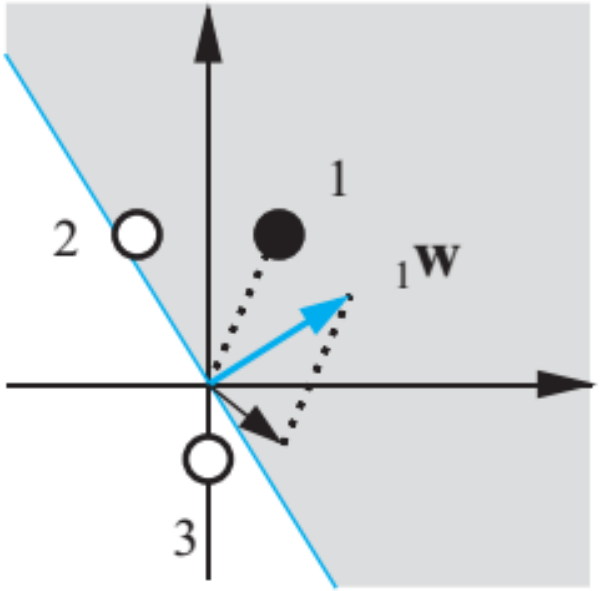
- Cho w = p_1: khi đó đảm bảo là sẽ phân loại đúng vector p_1
- Tuy nhiên hoàn toàn có thể xảy ra các trường hợp mà đó không phải là đáp án đúng
- Nếu cộng thêm p_1 vào w, điều này làm cho w hướng đến gần p_1 hơn

If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$



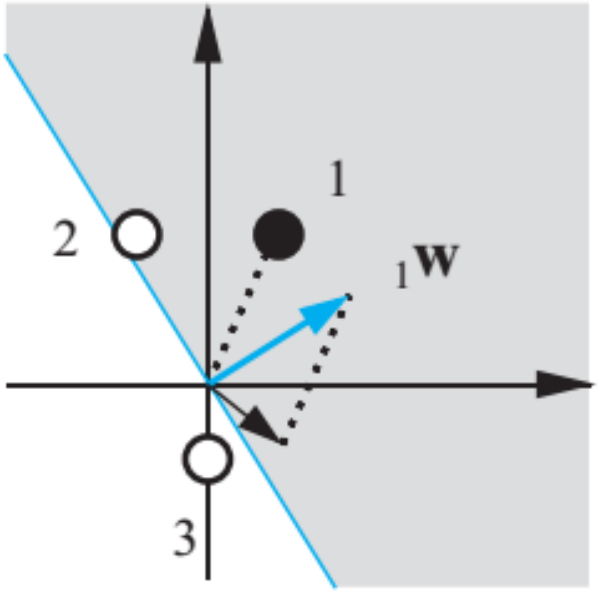
Ý tưởng

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} $$



Ý tưởng

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} $$

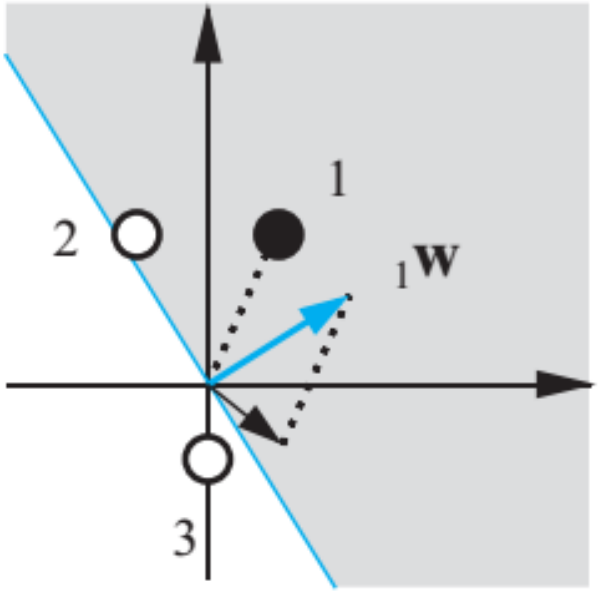


$$ a = hardlim({}_1\mathbf{w}^T\mathbf{p}_2) = hardlim\left(\begin{bmatrix} 2.0 & 1.2 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix} \right) $$

$$ = hardlim(0.4) = 1 . $$

Ý tưởng

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} $$

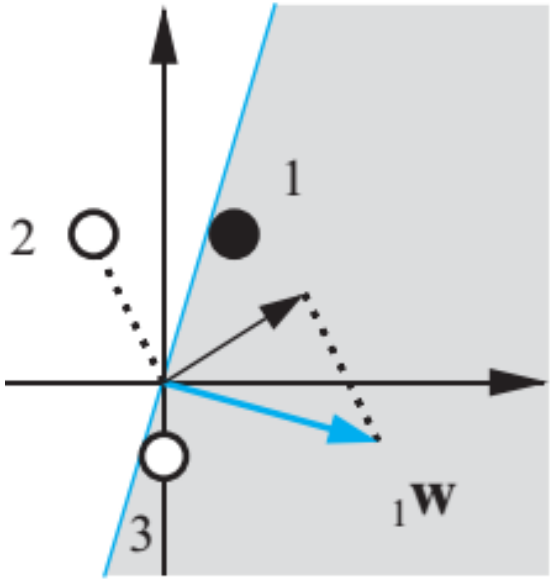


If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} $$

$$ a = hardlim({}_1\mathbf{w}^T\mathbf{p}_3) = hardlim\left(\begin{bmatrix} 3.0 & -0.8 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right) $$

$$ = hardlim(0.8) = 1 . $$



$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix} $$

Các khả năng

If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$.

If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$.

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$.

Tổng quát hóa

$$e = t - a$$

If $e = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$.

If $e = -1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$.

If $e = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$.

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p} $$

Huấn luyện perceptron nhiều neuron

- **Các luật trên cập nhật trọng số cho một neuron**
- **Luật này có thể được khái quát hóa cho mạng perceptron nhiều neuron**
- **Ví dụ**
  - Cập nhật hàng thứ i của ma trận trọng số

$$_i\mathbf{w}^{new} = {}_i\mathbf{w}^{old} + e_i\mathbf{p}$$

  - Cập nhật phần tử thứ i của vector bias

$$b_i^{new} = b_i^{old} + e_i$$

Huấn luyện perceptron nhiều neuron

- Dưới ký pháp ma trận, luật học chuyển thành:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^{T}$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$$

Ví dụ

- **Bài toán nhận dạng cam và táo trong bài 3**

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\-1\\-1\end{bmatrix}, t_1 = \begin{bmatrix}0\end{bmatrix}\right\} \qquad \left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\\-1\end{bmatrix}, t_2 = \begin{bmatrix}1\end{bmatrix}\right\}$$

- **Khởi tạo:**
  - Các trọng số của mạng và vector bias sẽ được khởi tạo ngẫu nhiên với các giá trị nhỏ

$$\mathbf{W} = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix}, \; b = 0.5$$

Ví dụ

- **Bước đầu:**

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}1\\-1\\-1\end{bmatrix} + 0.5\right)$$

$$= hardlim(2.5) = 1$$

- **Tính toán lỗi:** $e = t_1 - a = 0 - 1 = -1$
- **Cập nhật lại trọng số**

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix} + (-1)\begin{bmatrix}1 & -1 & -1\end{bmatrix}$$

$$= \begin{bmatrix}-0.5 & 0 & 0.5\end{bmatrix}.$$

- **Cập nhật lại bias:**

$$b^{new} = b^{old} + e = 0.5 + (-1) = -0.5$$

Ví dụ

- **Bước lặp thứ 2 với mẫu thứ 2:**

$$a = hardlim(\mathbf{W}\mathbf{p}_2 + b) = hardlim\left(\begin{bmatrix}-0.5 & 0 & 0.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + (-0.5)\right)$$

$$= hardlim(-0.5) = 0$$

- **Tính toán lỗi:** $e = t_2 - a = 1 - 0 = 1$
- **Cập nhật:**

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}-0.5 & 0 & 0.5\end{bmatrix} + 1\begin{bmatrix}1 & 1 & -1\end{bmatrix} = \begin{bmatrix}0.5 & 1 & -0.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = -0.5 + 1 = 0.5$$

Ví dụ

- **Bước lặp thứ 3 với mẫu thứ 1:**

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}0.5 & 1 & -0.5\end{bmatrix}\begin{bmatrix}1\\-1\\-1\end{bmatrix} + 0.5\right)$$

$$= hardlim(0.5) = 1$$

$$e = t_1 - a = 0 - 1 = -1$$

- **Cập nhật:**

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}0.5 & 1 & -0.5\end{bmatrix} + (-1)\begin{bmatrix}1 & -1 & -1\end{bmatrix}$$

$$= \begin{bmatrix}-0.5 & 2 & 0.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = 0.5 + (-1) = -0.5$$

Ví dụ

- Nếu tiếp tục các bước lặp, có thể tìm được ma trận trọng số và vector bias sao cho cả hai mẫu đầu vào đều được phân loại đúng
- Giải thuật sẽ hội tụ đến một lời giải
- Chú ý rằng bao đóng quyết định không giống với lời giải ở bài 3, tuy nhiên cả hai vẫn phân loại đúng các mẫu đầu vào

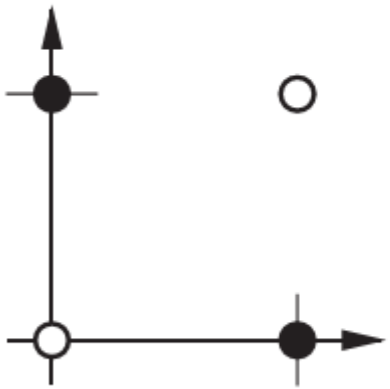
Chứng minh tính hội tụ của mạng

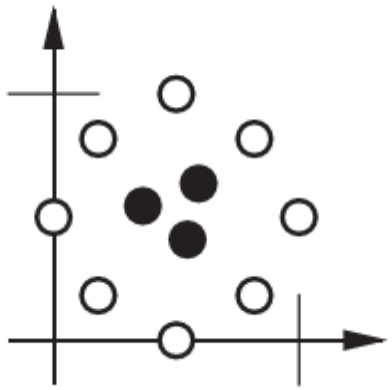
- **Mặc dù luật học đơn giản nhưng lại rất hiệu quả và đảm bảo mạng luôn hội tụ sau một số bước lặp** (không chứng minh)
- **Câu hỏi đặt ra: Mạng perceptron có thể giải quyết những bài toán nào ?**
  - Phân tách tuyến tính (linearly separable)
  - Ví dụ: cổng logic AND
- **Tuy nhiên trong thực tế, rất nhiều bài toán dữ liệu không phân tách tuyến tính**

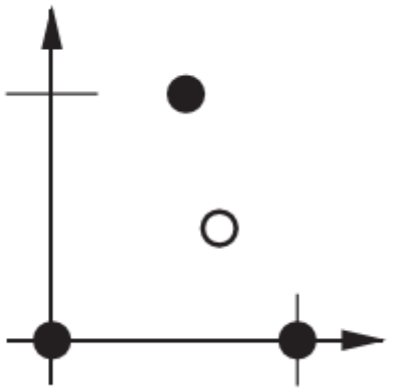
Khi dữ liệu không phân tách tuyến tính

- **Ví dụ cổng logic XOR**

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\}\left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\}\left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 0\right\}$$







Hàm mục tiêu

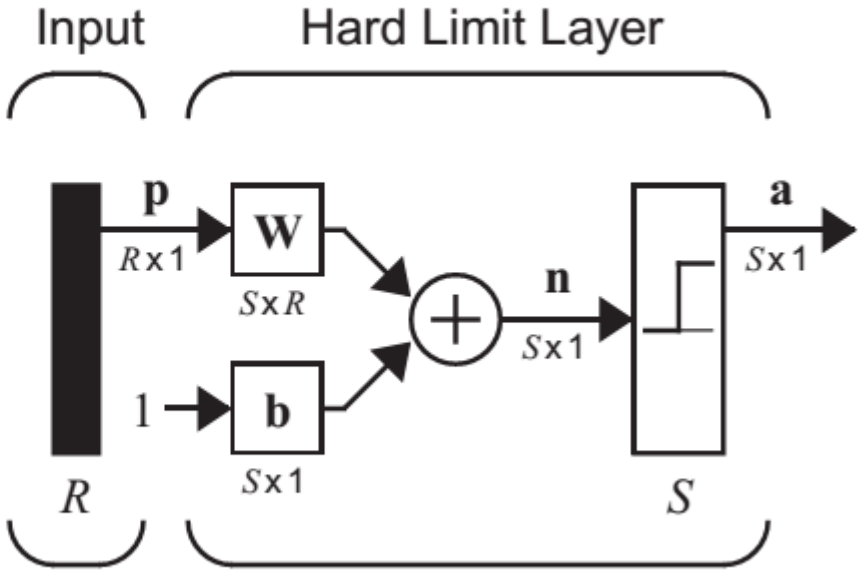
- **Làm thế nào để đánh giá chất lượng của một mô hình mạng đối với một tập dữ liệu ?**
- **Thước đo đánh giá:**
  - Độ chính xác (accuracy) = #correct / Total instances
  - Độ sai số (error) = #misclassified / Total instance = 1 – acc
- **Trên thực tế với dữ liệu thực (target / output)**
  - Lỗi mẫu (pattern error) = target – output
    - Các lỗi có thể loại trừ lẫn nhau: $\sum |t_i - o_i|$ (L_1 loss)
    - Cách sử dụng thông thường: lỗi bình phương:$\sum (t_i - o_i)^2$ (L_2 loss)
  - Tổng lỗi bình phương: Σ Pattern Errors = $\Sigma \Sigma (t_i - o_i)^2$

Tổng kết

- **Kiến trúc mạng perceptron**

Perceptron Architecture

$$\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$



$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b})$$

$${}_i\mathbf{w}^T\mathbf{p} + b_i = 0$$

$$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\mathbf{p} + b_i)$$

MSE và RMSE

- **Mean squared error (lỗi bình phương trình bình)**
  - MSE = SSE / n (với n là số lượng mẫu trong tập dữ liệu)
- **Root mean squared error (square root of SSE)**
  - RMSE = $\sqrt{MSE}$

Mục đích của việc học

- **Nguyên lý học:** Với tập dữ liệu cho trước, cập nhật các tham số của mạng sao cho sai số đầu ra với đích là nhỏ nhất (MSE hoặc RMSE)
- **Quá trình học của mạng:** là giải bài toán tối ưu (hay tìm cực trị của hàm mục tiêu đa biến) trong không gian đa chiều
- **Việc học thường không thể giải trực tiếp** (do dữ liệu lớn, việc phân tách dữ liệu phức tạp) nên phải sử dụng các thuật toán lặp để tìm nghiệm cận tối ưu

Quá trình học

- **Quá trình học kết thúc khi nào ?**
  - Khi số bước lặp đủ lớn
  - Khi hàm mục tiêu đạt giá trị đủ nhỏ

Tổng kết

- **Luật học:**

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^{T}$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$$

where $\mathbf{e} = \mathbf{t} - \mathbf{a}$.

Thực hành – Bài 1

- **Bài toán:** Cho hai cụm dữ liệu thuộc về hai lớp. Dữ liệu là các điểm trong không gian 2 chiều. Các lớp phân tách tuyến tính.
- **Nhiệm vụ:** tạo một mạng perceptron để phân lớp dữ liệu

Thực hành – Bài 1

- **Các bước cần thực hiện**
  - Định nghĩa dữ liệu vào và ra
  - Thiết kế và huấn luyện mạng theo dữ liệu
  - Vẽ bao đóng phân tách dữ liệu

Thực hành – Bài 2

- **Bài toán:** Dữ liệu 2 chiều được phân tách thành 4 cụm (tương ứng với 4 lớp)
- **Nhiệm vụ:** tạo một mạng perceptron để phân 4 lớp dữ liệu với đầu vào 2 chiều và đầu ra 2 chiều

Thực hành – Bài 2

- **Các bước cần thực hiện**
  - Định nghĩa dữ liệu vào và ra
  - Thiết kế và huấn luyện mạng theo dữ liệu
  - Vẽ bao đóng phân tách dữ liệu