Trí Tuệ Nhân Tạo

(Artificial Intelligence)

Lê Thanh Hương

Viện Công nghệ thông tin và Truyền thông
Trường Đại Học Bách Khoa Hà Nội

Nội dung môn học

- Chương 1. Tổng quan
- Chương 2. Tác tử thông minh
- Chương 3. Giải quyết vấn đề
- Chương 4. Tri thức và suy diễn
- Chương 5. Biểu diễn tri thức
- Chương 6. Học máy
  - Giới thiệu về học máy
  - K láng giềng gần
  - Phân lớp Naïve Bayes
  - Học cây quyết định
  - Mạng nơron

Học cây quyết định

Bài toán: quyết định có đợi 1 bàn ở quán ăn không, dựa trên các thông tin sau:

1. Lựa chọn khác: có quán ăn nào khác gần đó không?
2. Quán rượu: có khu vực phục vụ đồ uống gần đó không?
3. Fri/Sat: hôm nay là thứ sáu hay thứ bảy?
4. Đói: chúng ta đã đói chưa?
5. Khách hàng: số khách trong quán (không có, vài người, đầy)
6. Giá cả: khoảng giá ($,$$,$$$)
7. Mưa: ngoài trời có mưa không?
8. Đặt chỗ: chúng ta đã đặt trước chưa?
9. Loại: loại quán ăn (Pháp, Ý, Thái, quán ăn nhanh)
10. Thời gian đợi: 0-10, 10-30, 30-60, >60

Phép biểu diễn dựa trên thuộc tính

- Các mẫu được miêu tả dưới dạng các giá trị thuộc tính (logic, rời rạc, liên tục)
  - Ví dụ, tình huống khi đợi 1 bàn ăn

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

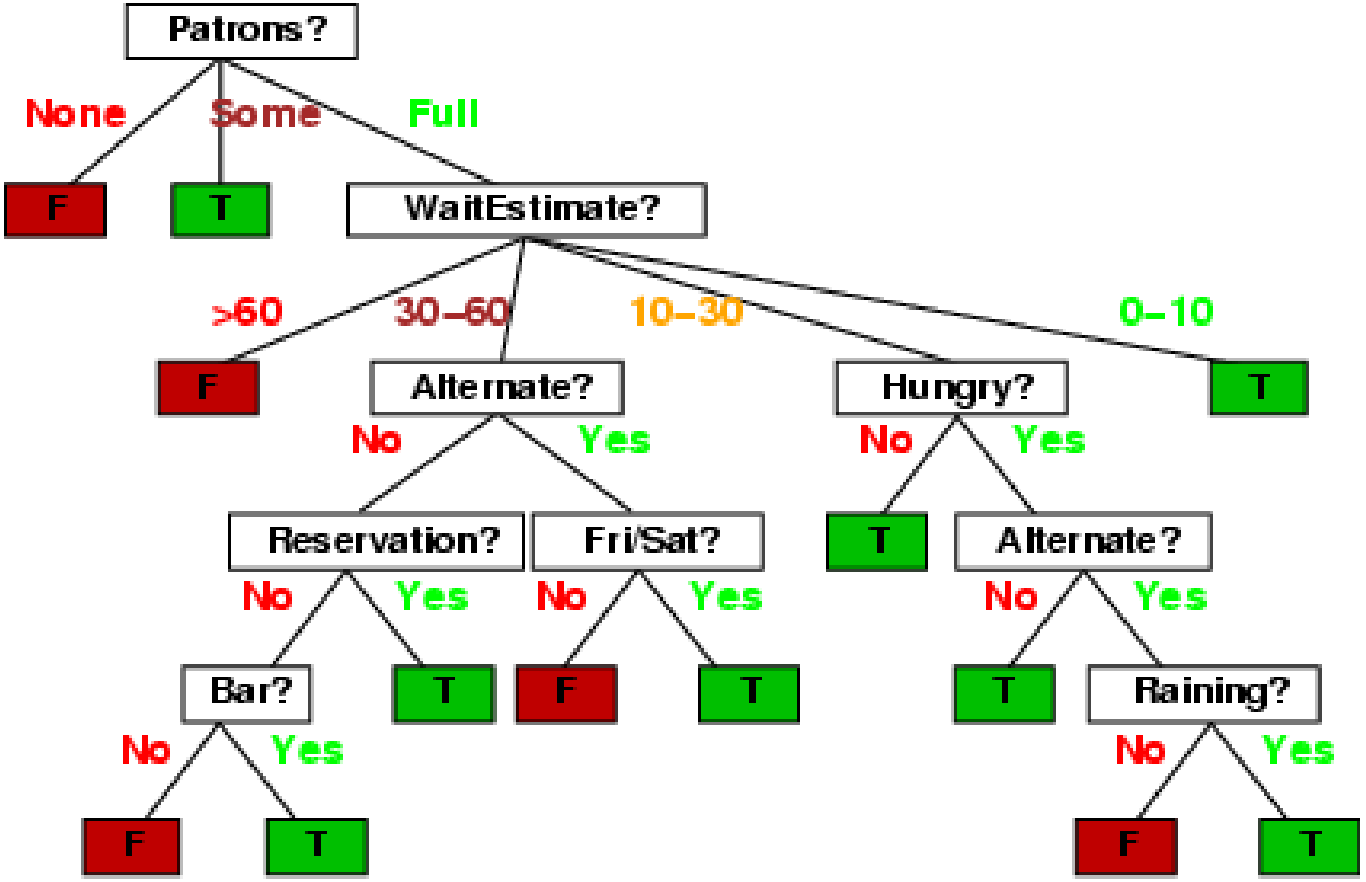
- Các loại (lớp) của mẫu là khẳng định (T) hoặc phủ định (F)

Attributes										Target
Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
T	F	F	T	Some	$$$	F	T	French	0–10	T
T	F	F	T	Full	$	F	F	Thai	30–60	F
F	T	F	F	Some	$	F	F	Burger	0–10	T
T	F	T	T	Full	$	F	F	Thai	10–30	T
T	F	T	F	Full	$$$	F	T	French	>60	F
F	T	F	T	Some	$$	T	T	Italian	0–10	T
F	T	F	F	None	$	T	F	Burger	0–10	F
F	F	F	T	Some	$$	T	T	Thai	0–10	T
F	T	T	F	Full	$	T	F	Burger	>60	F
T	T	T	T	Full	$$$	F	T	Italian	10–30	F
F	F	F	F	None	$	F	F	Thai	0–10	F
T	T	T	T	Full	$	F	F	Burger	30–60	T

Patrons, WaitEstimates, Alternative, Hungry, Rain

Cây quyết định

… là cách biểu diễn các giả thiết.



Không gian giả thiết

Khi có n thuộc tính Boolean, số lượng các cây quyết định là?

= số các hàm Boolean

= số các giá trị khác nhau trong bảng ví dụ mẫu với 2^n hàng

= 2^{2^n}

Ví dụ, với 6 thuộc tính Boolean, có
18,446,744,073,709,551,616 cây

Thuật toán ID3

Mục đích: tìm cây thoả mãn tập mẫu

Ý tưởng: (lặp) chọn thuộc tính quan trọng nhất làm gốc của cây/cây con

ID3(Examples, Target_attribute, Attributes)

/ *Examples*: các mẫu luyện

Target_attribute: thuộc tính cần đoán giá trị

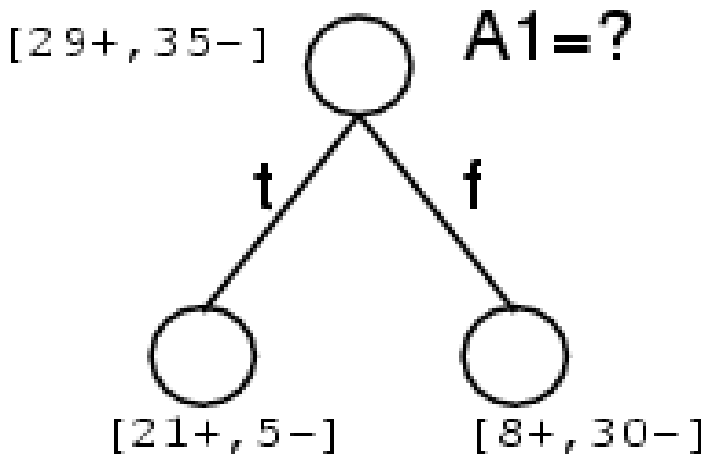
Attributes: các thuộc tính có thể được kiểm tra qua phép học cây quyết định. */

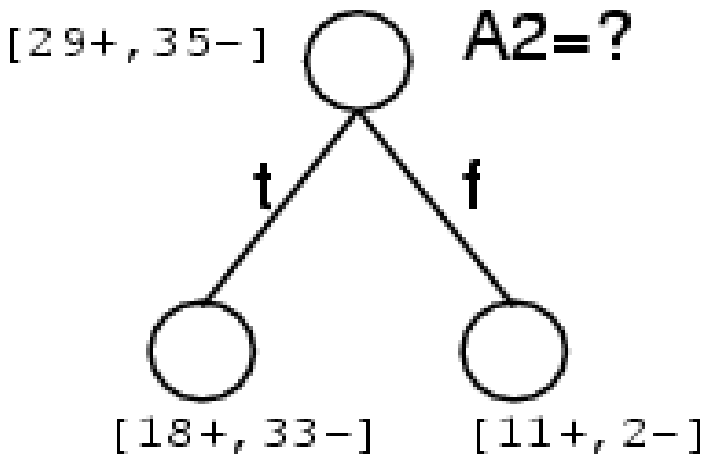
- Tạo 1 nút gốc *Root* cho cây
- If ∀ *Examples* +, trả về cây chỉ có 1 nút Root, với nhãn +
- If ∀ *Examples* -, trả về cây chỉ có 1 nút Root, với nhãn –
- If *Attributes* rỗng, trả về cây chỉ có 1 nút *Root*, với nhãn = giá trị thường xuất hiện nhất của *Target_attribute* trong *Examples*

Thuật toán ID3

- Otherwise Begin:
  - A ← thuộc tính trong *Attributes* cho phép phân loại tốt nhất *Examples*
  - Thuộc tính quyết định của nút gốc ← A
  - Với các giá trị v_i có thể có của A,
    - Thêm 1 nhánh mới dưới gốc, ứng với phép kiểm tra A = v_i
    - Đặt $Examples_{vi}$ = tập con của Examples với giá trị thuộc tính A = v_i
    - If $Examples_{vi}$ rỗng
      - Then, dưới nhánh mới này, thêm 1 lá với nhãn = giá trị thường xuất hiện nhất của *Target_attribute* trong *Examples*
      - Else, dưới nhánh mới này thêm cây con ID3($Examples_{vi}$, Target_attribute, Attributes - {A}))
- End
- Return *Root*

Thuộc tính nào tốt nhất?

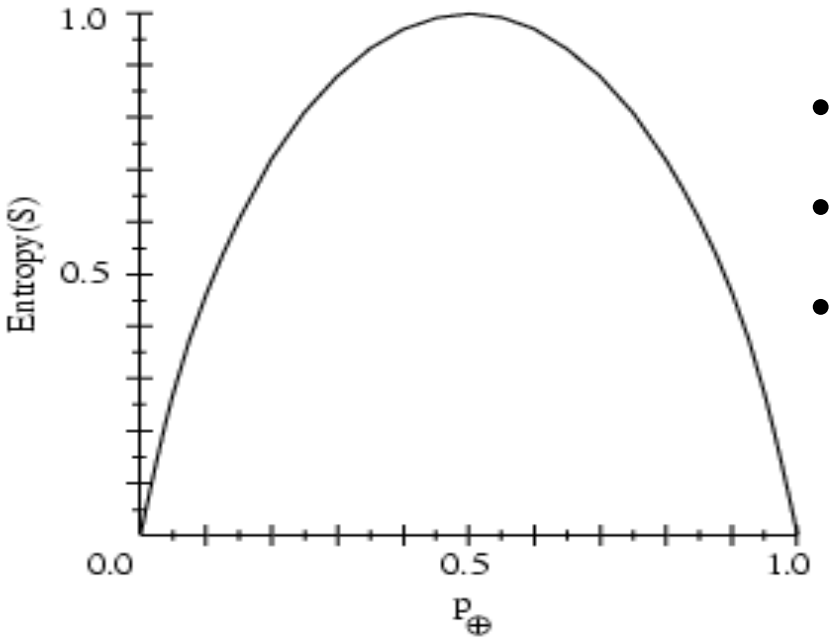




Sử dụng lượng thông tin đạt được Information Gain

⇨ xác định thông qua độ đo Entropy

Entropy của một tập mẫu



- S là một tập mẫu của tập luyện
- p_+ là tỷ lệ các mẫu dương trong S
- p_- là tỷ lệ các mẫu âm trong S

- Entropy đo độ nhiễu của S = số các bit cần thiết để mã hoá lớp + hoặc - của các thành viên ngẫu nhiên của S
- Entropy(S) = $- p_+ * \log_2 p_+ \; - p_- * \log_2 p_-$

Entropy

Entropy H(X) của biến ngẫu nhiên X:

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

Ví dụ, với S gồm 9 mẫu dương và 5 mẫu âm, kí hiệu S([9+,5-]).

Entropy([9+,5-])

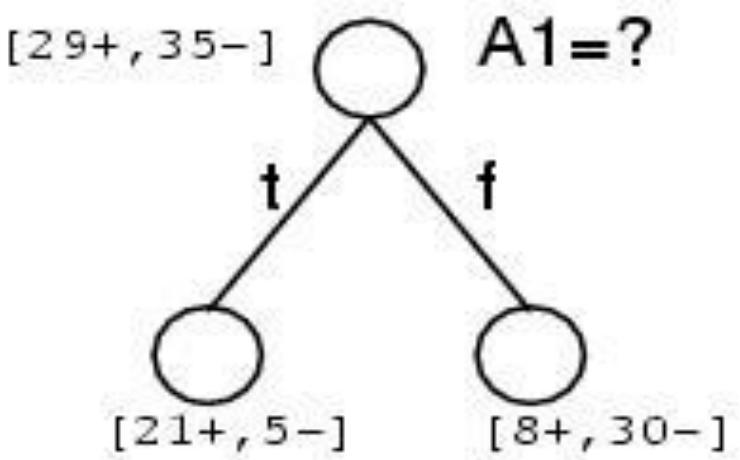
= - (9/14)$\log_2$(9/14) – (5/14)$\log_2$(5/14)

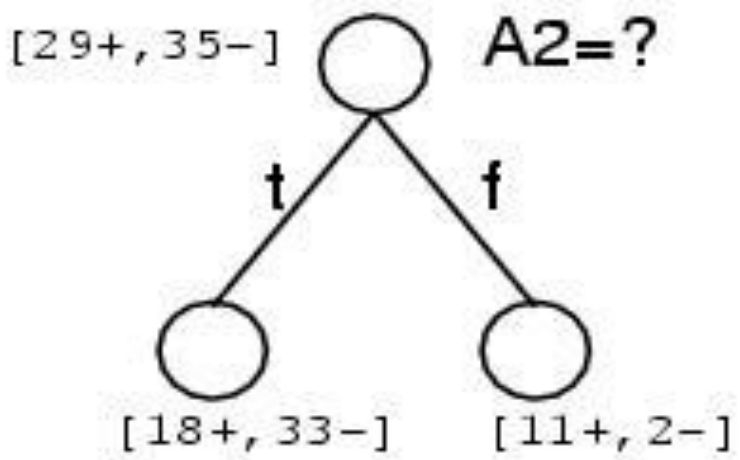
= 0.940

Information Gain

Gain(S, A) = độ giảm entropy do việc phân loại trong A

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|Sv|}{|S|} Entropy(Sv)$$





Ví dụ: tập luyện

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

S = [9+,5-]

Humidity ={High,Normal}:

S_{high}=[3+,4-];

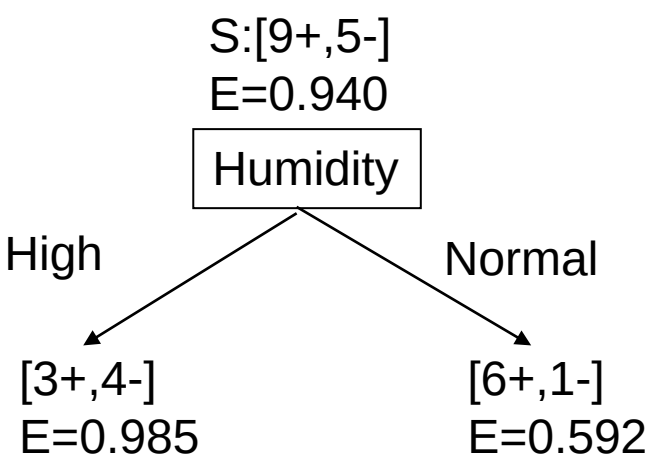
S_{normal}=[6+,1-]

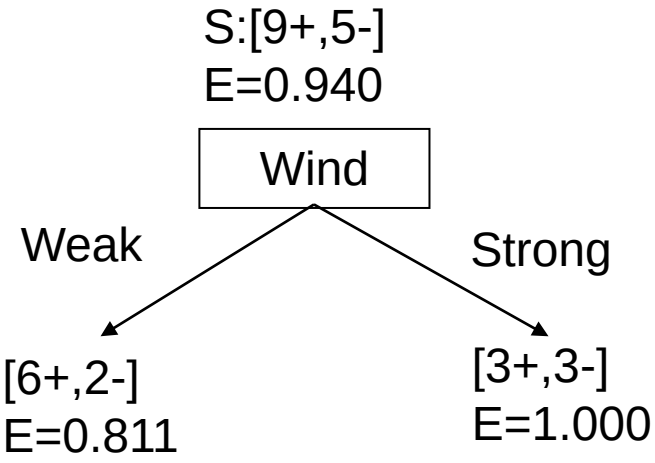
Wind ={Weak,Strong}:

S_{weak} = [6+,2-];

S_{strong} = [3+,3-]

Thuộc tính nào phân loại tốt nhất?





Gain(S,Wind) = Entropy(S) – $\sum_{v \in Values(A)} \frac{|Sv|}{|S|} Entropy(Sv)$

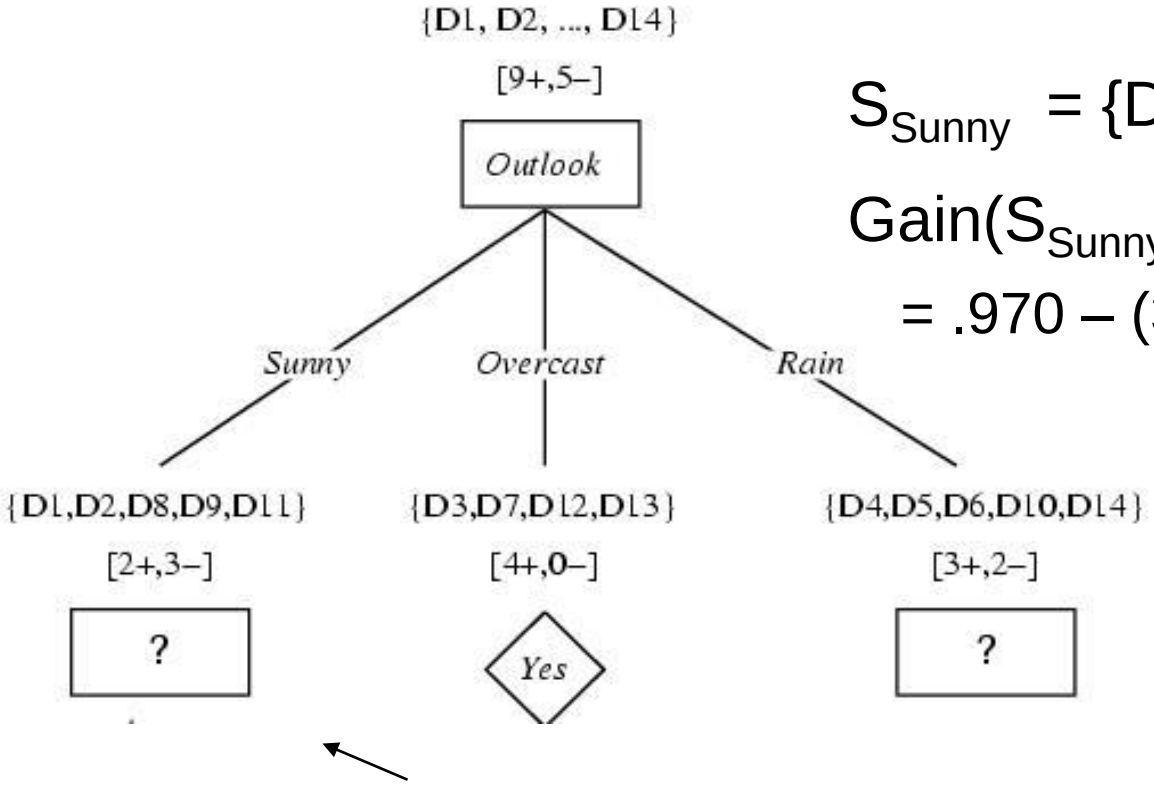
= Entropy(S) – (8/14)Entropy(S_{Weak}) – (6/14)Entropy(S_{Strong})

= 0.940 – (8/14)*0.811 – (6/14)*1.00 = 0.048

Gain(S,Humidity) = 0.940 – (7/14)*0.985 – (7/14)*0.592 = 0.151

Gain(S,Outlook)=0.246; Gain(S,Humidity)=0.151

Gain(S,Wind)=0.048; Gain(S,Temperature)=0.029



S_{Sunny} = {D1,D2,D8,D9,D11}

Gain(S_{Sunny}, Humidity)

= .970 – (3/5)*0.0 – (2/5)*0.0 = .970

Thuộc tính nào tiếp?

Gain(S_{Sunny}, Temperature)

= .970 –(2/5)*0.0– (2/5)*1.0– (1/5)*0.0=.570

Gain(S_{Sunny}, Wind)

= 0.970 – (2/5)*1.0 – (3/5)*0.918 = 0.019

Cây quyết định sử dụng khi nào?

Các bài toán với các đặc tính sau thích hợp với học cây quyết định:

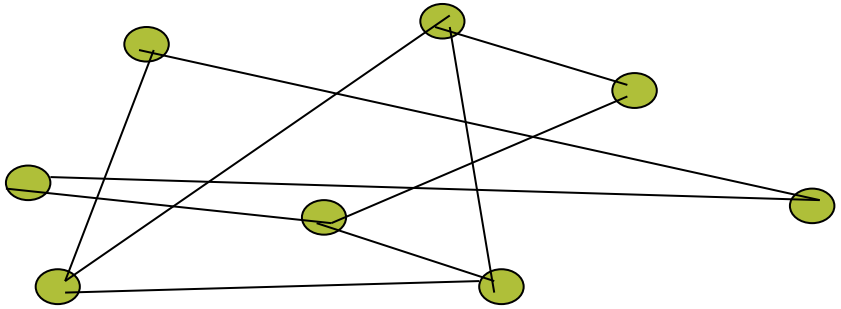
- Các mẫu mô tả được bởi các cặp thuộc tính-giá trị
- Hàm đích có giá trị rời rạc
- Cần có các giả thiết rời rạc
- Các dữ liệu luyện có thể có nhiễu
- Dữ liệu luyện có thể thiếu giá trị thuộc tính

Ví dụ:

- Chẩn đoán y tế
- Phân tích các nguy cơ về tín dụng
- Mô hình hoá việc lập lịch

Mạng nơron nhân tạo

nghiên cứu và mô phỏng các tiến trình xử lý song song và phân tán khổng lồ diễn ra trong bộ não con người



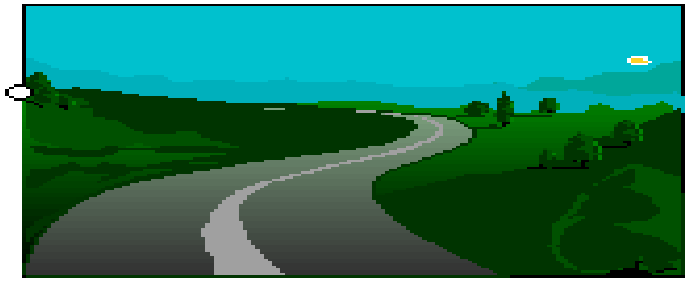
Các vấn đề:

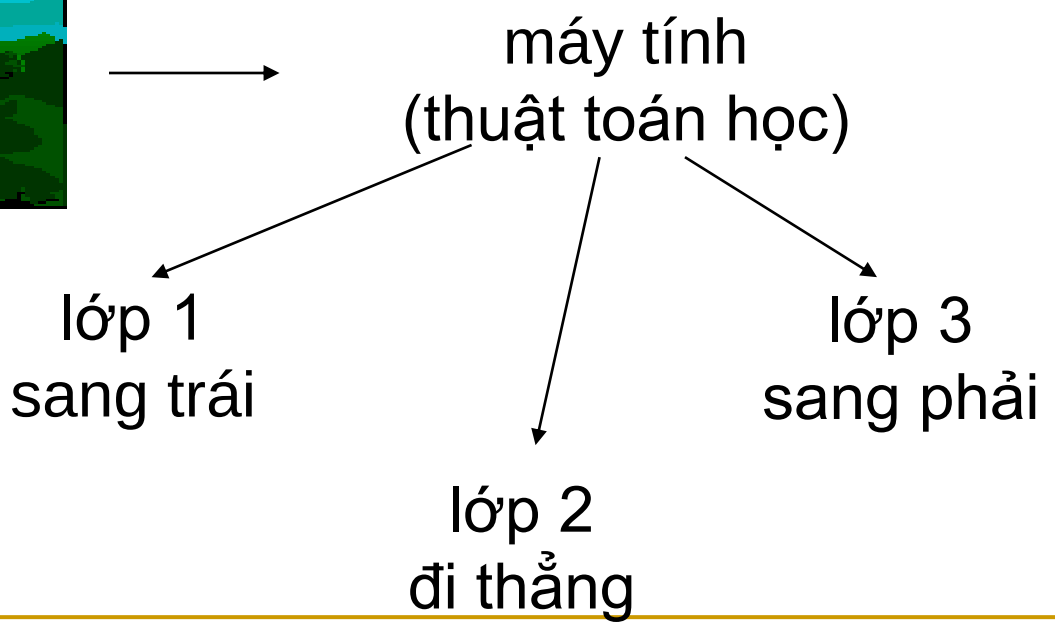
- Tốc độ bộ não nhận dạng hình ảnh
- Rất nhiều nơron trong một bộ não
- Tốc độ một nơron truyền dữ liệu

Ví dụ

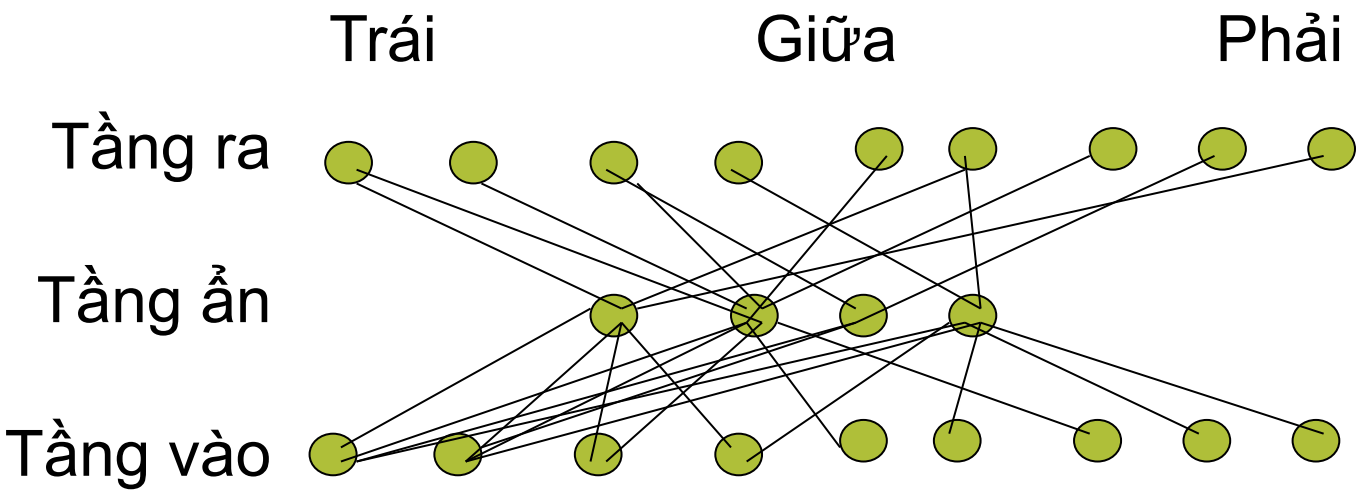
Lái xe

Luyện bộ phận điền khiển xe lái xe chính xác trên nhiều địa hình khác nhau





Biểu diễn mạng nơron



Định nghĩa

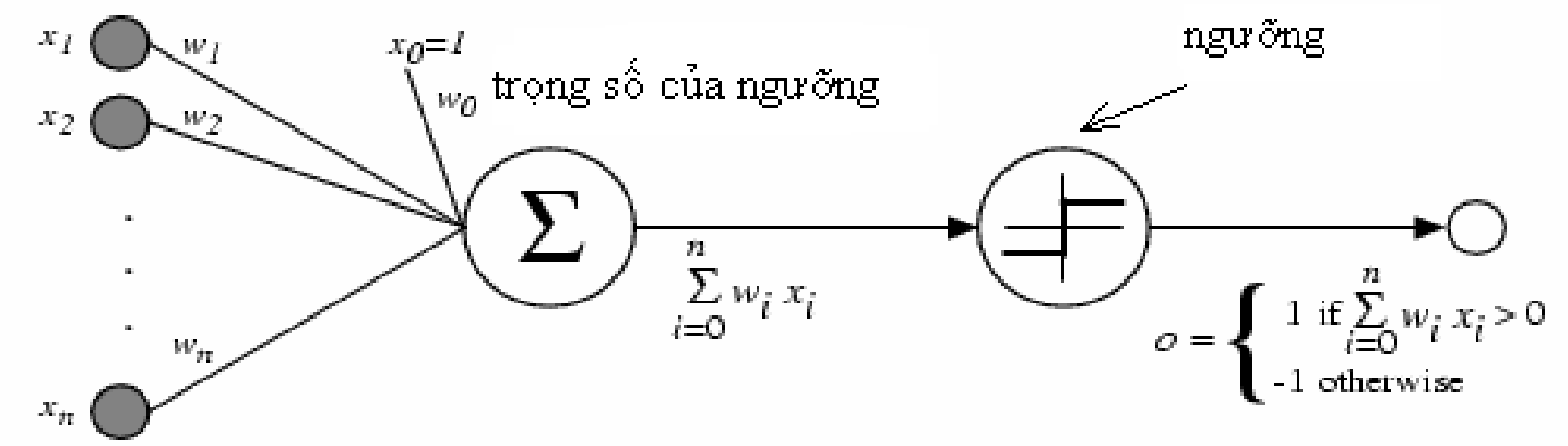
- Là một hệ thống gồm rất nhiều phần tử xử lý đơn giản hoạt động song song.
- Tính năng: phụ thuộc vào
  - cấu trúc hệ thống
  - mức độ liên kết giữa các phần tử
  - quá trình xử lý bên trong các phần tử
- Có thể học từ số liệu và tổng quát hoá từ các số liệu đó.

Khi nào sử dụng mạng nơron?

Mạng nơron thích hợp với những bài toán có đặc điểm sau:

- Các mẫu luyện được thể hiện bởi nhiều cặp giá trị-thuộc tính (ví dụ, điểm ảnh)
- Các mẫu luyện có thể có lỗi
- Chấp nhận thời gian huấn luyện dài
- Cần đánh giá nhanh hàm mục tiêu được học
- Không cần hiểu giả thiết cuối cùng vì NN được coi là hộp đen

Perceptron



$o(x_1,...,x_n) = \begin{cases} 1 \text{ nếu } w_0 + w_1x_1 + w_2x_2 + \ldots\ldots + w_nx_n > 0 \\ -1 \text{ trong trường hợp ngược lại} \end{cases}$

hay:

$o(\vec{x}) = \text{sgn}(\vec{w}.\vec{x})$

trong đó

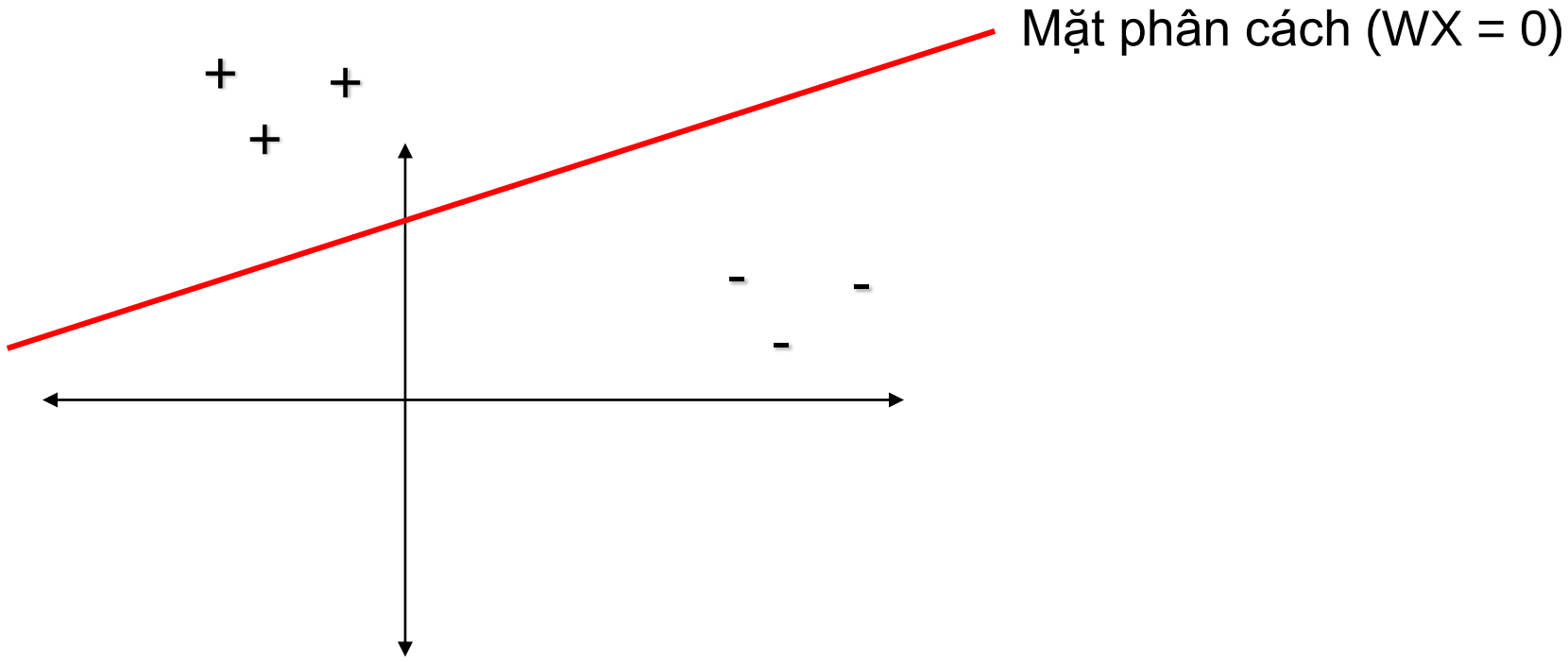
$\text{sgn}(y) = \begin{cases} 1 \text{ nếu } y > 0 \\ -1 \text{ nếu ngược lại} \end{cases}$

Nhiệm vụ học: tìm các giá trị của w

Không gian giả thiết: không gian các vectơ trọng số

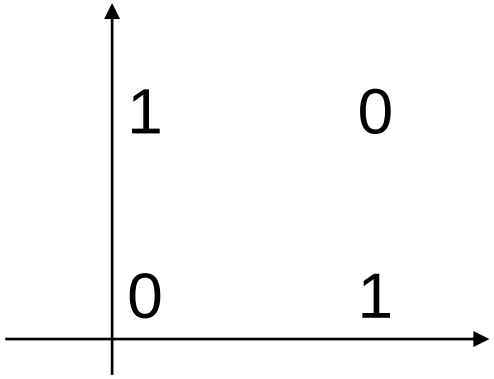
Khả năng của Perceptron

Mỗi perceptron tạo ra 1 mặt phân cách siêu phẳng trên không gian đầu vào n chiều



Khả năng của Perceptron

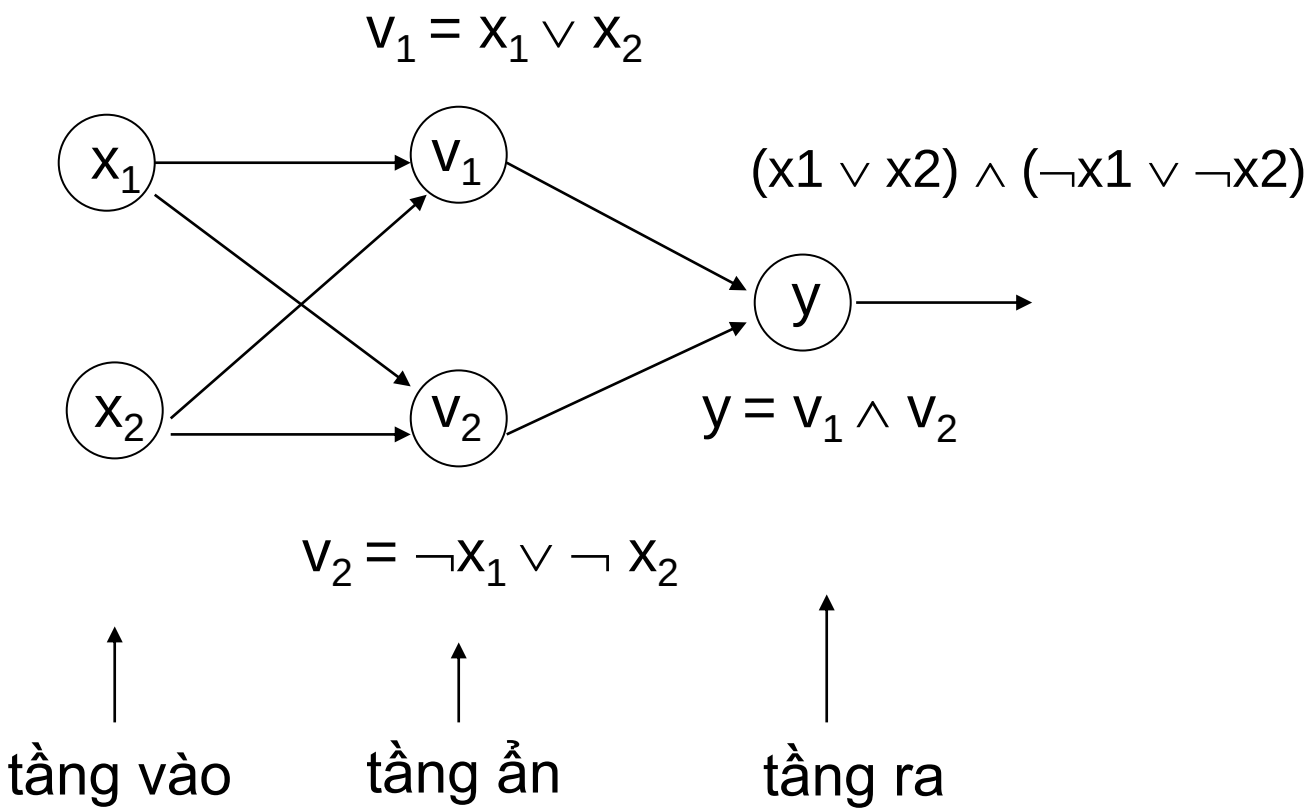
- có thể học các hàm $\neg$, $\wedge$, $\vee$, NAND, NOR
- không biểu diễn được các hàm không phân tách được bằng đường tuyến tính, vd XOR



$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$

➢ Mọi hàm logic đều có thể biểu diễn bằng 1 mạng perceptron có ít nhất 2 tầng

Mạng nơron biểu diễn hàm XOR



Học các trọng số mạng

- Luật perceptron:

  dùng khi tập luyện
  - phân tách được bằng 1 đường tuyến tính
  - đủ nhỏ

- Luật delta:

  dùng khi tập luyện không phân thể tách tuyến tính

Luật huấn luyện Perceptron

- Khởi tạo một vector có các trọng số ngẫu nhiên
- Lặp lại hàm perceptron cho mỗi mẫu luyện đến khi hàm perceptron phân loại đúng tất cả các mẫu luyện:
  - Các trọng số được sửa đổi tại mỗi bước dựa vào luật huấn luyện perceptron:

$$w_i \longleftarrow w_i + \Delta w_i$$

trong đó

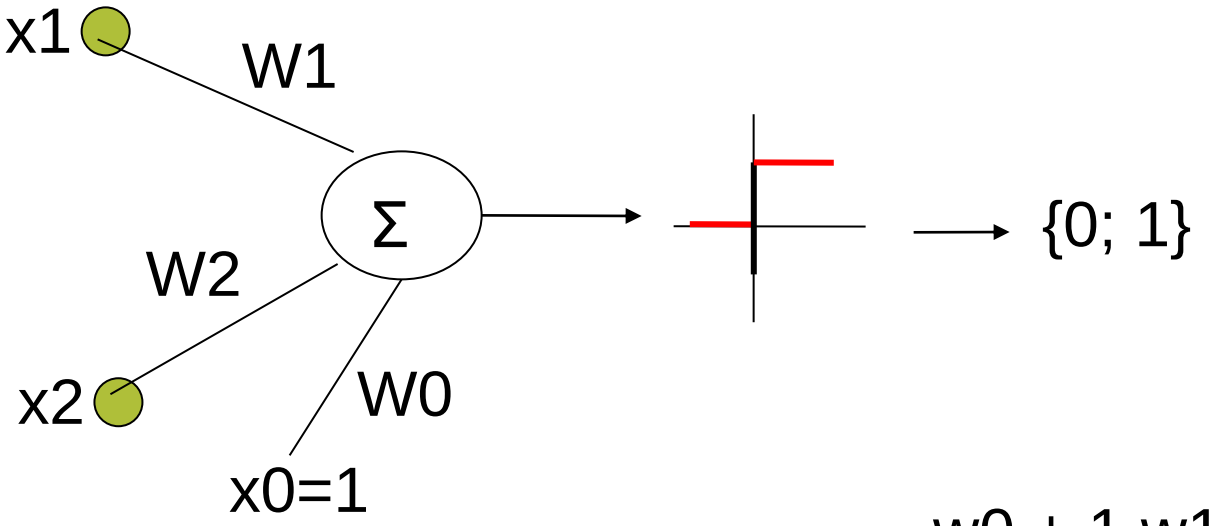
$$\Delta w_i \longleftarrow \Delta w_i + \eta(t - o)x_i$$

với

- t = c($\vec{x}$) là hàm đích
- o là đầu ra perceptron
- η = tốc độ học, là hằng số nhỏ

Ví dụ...

Biểu diễn g(x1,x2) = AND(x1,x2)



x1	x2	g
0	0	0
0	1	0
1	0	0
1	1	1

$$o(x) = \begin{cases} 1 \text{ nếu } \vec{w}.\vec{x} > 0 \\ 0 \text{ nếu ngược lại} \end{cases}$$

w0 + 1.w1 + 1.w2 >0
w0 + 1.w1 + 0.w2 < 0
w0 + 0.w1 + 1.w2 < 0
w0 + 0.w1 + 0.w2 < 0
⇨w0 = -0.8; w1= 0.5; w2 = 0.5

Ví dụ 1...

$$\vec{w}.\vec{x} = \sum_{i=0}^{2} w_i * x_i \qquad w_i \longleftarrow w_i + \Delta w_i \qquad \Delta w_i \longleftarrow \Delta w_i + \eta(t-o)x_i$$

Khởi tạo các giá trị đầu: $\Delta w0 = 0$, $\Delta w1 = 0$, $\Delta w2 = 0$

$w0 = -1.5$, $w1 = -0.5$, $w2 = 0.5$, $\eta = 0.1$

$\sum = x0.w0+x1.w1+x2.w2 = 1.w0+0.w1+0.w2 = w0 = -1.5$

x0	x1	x2	t	$\sum$	o	$\Delta w0$	$\Delta w1$	$\Delta w2$	w0	w1	w2
1	0	0	0	-1.5	0	0	0	0	-1.5	-0.5	0.5
1	0	1	0	-1	0	0	0	0	-1.5	-0.5	0.5
1	1	0	0	-2	0	0	0	0	-1.5	-0.5	0.5
1	1	1	1	-1.5	0	0.1	0.1	0.1			

Ví dụ 1...

$$\vec{w}.\vec{x} = \sum_{i=0}^{2} w_i * x_i \qquad w_i \longleftarrow w_i + \Delta w_i \qquad \Delta w_i \longleftarrow \Delta w_i + \eta(t-o)x_i$$

Khởi tạo các giá trị đầu: $\Delta w0 = 0$, $\Delta w1 = 0$, $\Delta w2 = 0$

$w0 = -1.5$, $w1 = -0.5$, $w2 = 0.5$, $\eta = 0.1$

$\Delta w0 = \Delta w0 + \eta*(t-o)*x0 = 0 + 0.1*(0-0)*1 = 0$

$w0 = w0 + \Delta w0 = -1.5 + 0 = -1.5$, $w1 = -0.5$, $w2 = 0.5$

x0	x1	x2	t	∑	o	Δw0	Δw1	Δw2	w0	w1	w2
1	0	0	0	-1.5	0	0	0	0	-1.5	-0.5	0.5
1	0	1	0	-1	0	0	0	0	-1.5	-0.5	0.5
1	1	0	0	-2	0	0	0	0	-1.5	-0.5	0.5
1	1	1	1	-1.5	0	0.1	0.1	0.1			

Ví dụ 1...

$$\vec{w}.\vec{x} = \sum_{i=0}^{2} w_i * x_i$$

$$w_i \longleftarrow w_i + \Delta w_i \qquad \Delta w_i \longleftarrow \Delta w_i + \eta(t-o)x_i$$

Khởi tạo các giá trị đầu: $\Delta w0 = 0$, $\Delta w1 = 0$, $\Delta w2 = 0$

$w0 = -1.5$, $w1 = -0.5$, $w2 = 0.5$, $\eta = 0.1$

x0	x1	x2	t	$\sum$	o	$\Delta w0$	$\Delta w1$	$\Delta w2$	w0	w1	w2
1	0	0	0	-1.5	0	0	0	0			
1	0	1	0	-1	0	0	0	0			
1	1	0	0	-2	0	0	0	0			
1	1	1	1	-1.5	0	0.1	0.1	0.1			

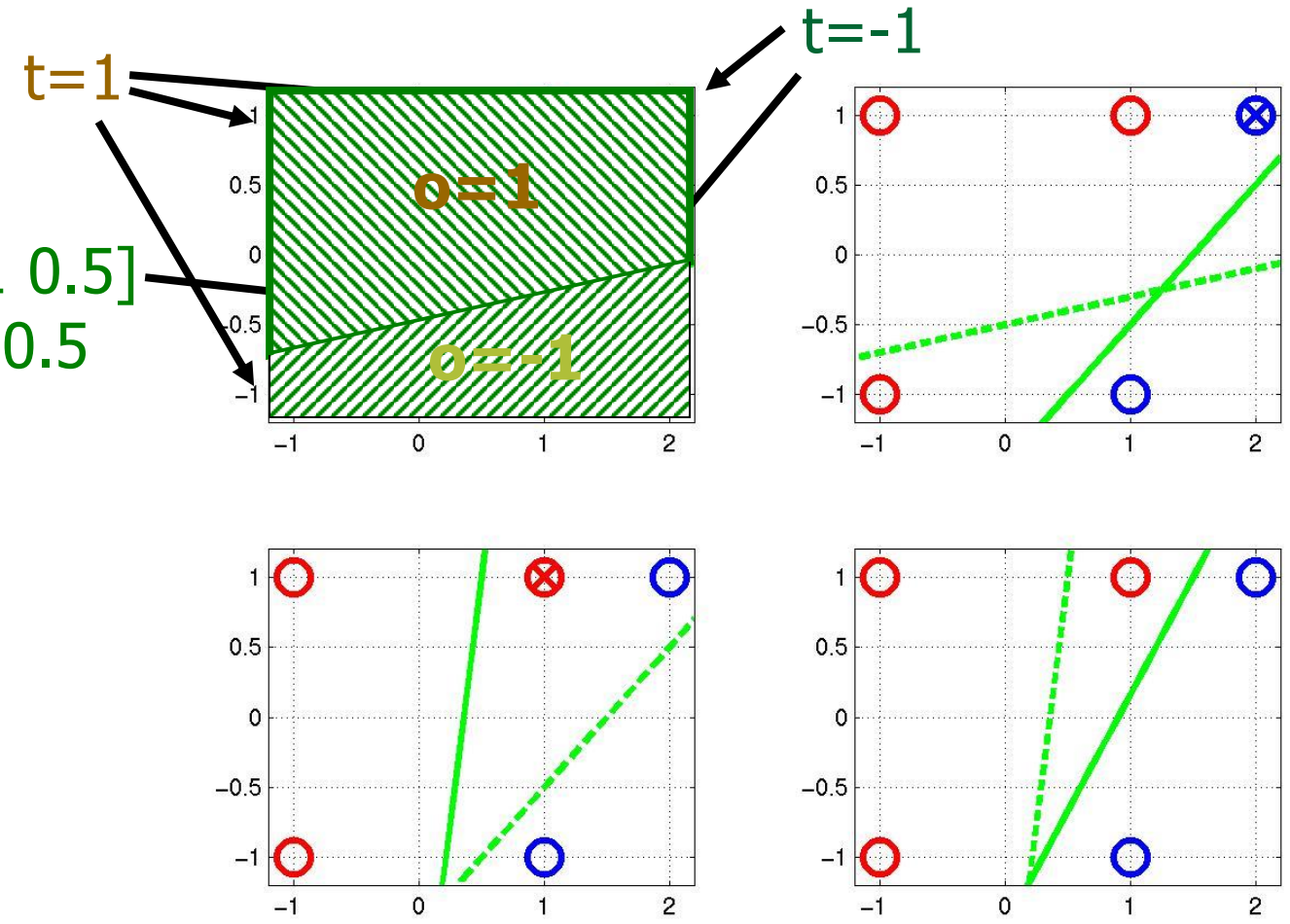
$\Delta w0 = \Delta w0 + \eta*(t-o)*x0 = 0 + 0.1*(1-0)*1 = 0.1$

$w0 = w0 + \Delta w0 = -1.5 + 0.1 = -1.4$, $w1 = -0.4$, $w2 = 0.6$

Ví dụ 2

w=[0.25 –0.1 0.5]

$x_2 = 0.2\ x_1 - 0.5$



Luật học Gradient Descent

- Giả sử một nơ ron có đầu ra tuyến tính và liên tục
  - $o=w_0 + w_1 x_1 + \ldots + w_n x_n$
- Nhiệm vụ học là xác định tập trọng số để tối thiểu hàm lỗi
  - $E[w_1,\ldots,w_n] = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$

  D là tập mẫu học

Gradient descent

- **Gradient của *E (ký hiệu là ∇E) là một vectơ***
  - Có hướng chỉ đi lên (dốc)
  - Có độ dài tỷ lệ thuận với độ dốc
- Gradient *∇E xác định hướng gây ra việc* ***tăng nhanh nhất (steepest increase) đối với giá trị lỗi E***

$$\nabla E(w) = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_N} \right)$$

trong đó *N là tổng số các trọng số (các liên kết) trong mạng*

- Vì vậy, hướng gây ra việc **giảm nhanh nhất (steepest decrease) là** giá trị phủ định của gradient của *E*

$$\Delta w = -\eta . \nabla E(w); \quad \Delta w_i = -\eta . \nabla E(w_i), \forall i = 1..N$$

- Yêu cầu: Các hàm tác động được sử dụng trong mạng phải là các hàm liên tục đối với các trọng số và có đạo hàm liên tục

Luật học Gradient Descent

D={<(1,1),1>,<(-1,-1),1>,
<(1,-1),-1>,<(-1,1),-1>}

Gradient:

$\nabla E[w]=[\partial E/\partial w_0,\ldots \partial E/\partial w_n]$

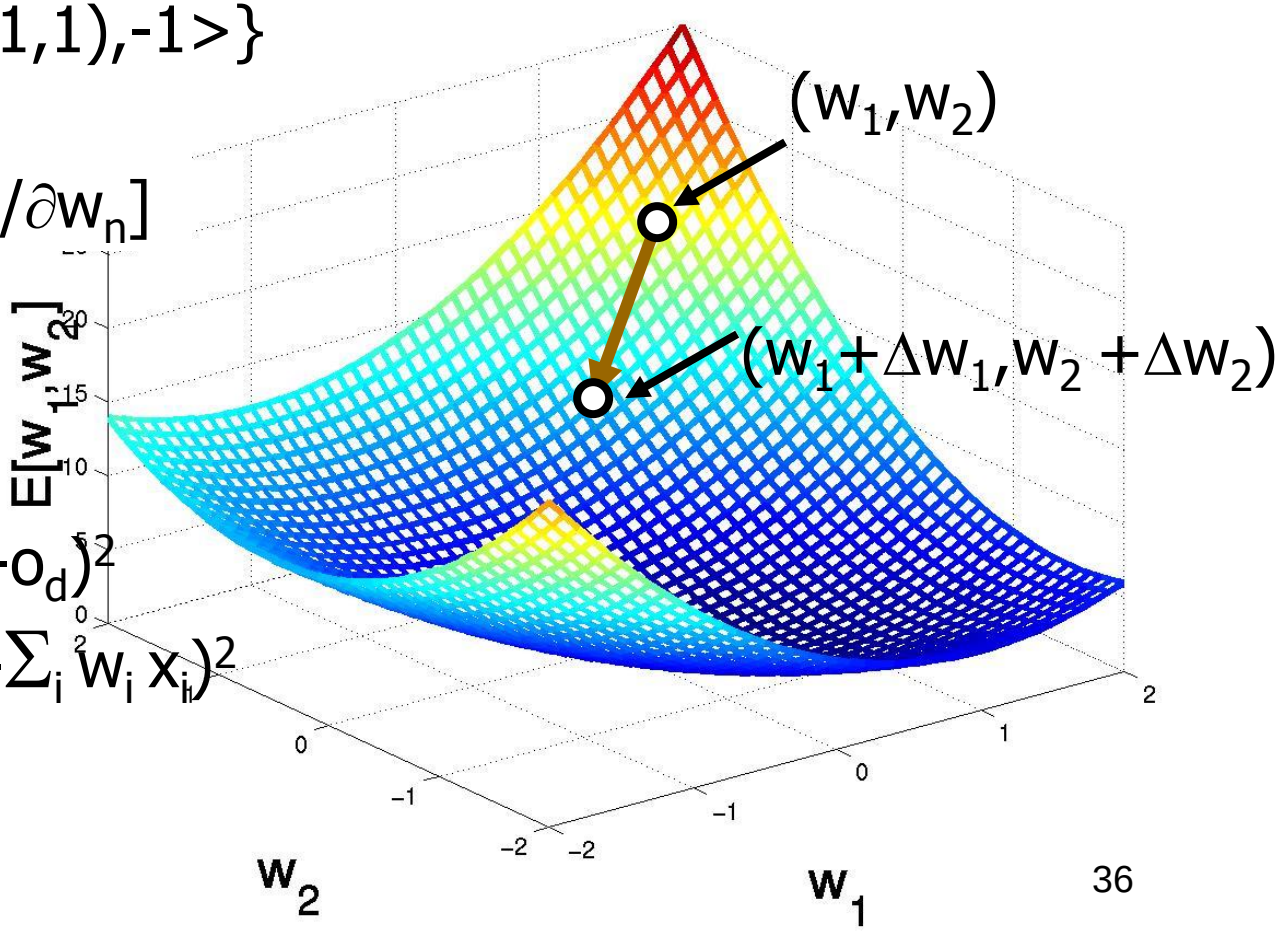
$\Delta w=-\eta \nabla E[w]$

$\Delta w_i=-\eta \partial E/\partial w_i$

$=-\eta \partial/\partial w_i \ 1/2\Sigma_d(t_d-o_d)^2$

$=-\eta \partial/\partial w_i \ 1/2\Sigma_d(t_d-\Sigma_i w_i x_i)^2$

$=-\eta \Sigma_d(t_d- o_d)(-x_i)$



Luật học Gradient Descent

Gradient-Descent(*tập mẫu*, η)

Mỗi mẫu là một bộ d=<(x_1,…x_n),t> với (x_1,…,x_n) là véc tơ đầu vào và t là giá trị đích, η là hệ số học

- Khởi tạo ngẫu nghiên w_i
- Lặp cho tới khi gặp điều kiện dừng
  - Khởi tạo Δw_i bằng 0
  - Với mỗi mẫu d trong *tập mẫu D*
    - Đưa (x_1,…,x_n) vào mạng để tính toán giá trị đầu ra o
    - Với mỗi trọng số w_i
      - $\Delta w_i = -\eta \, \partial E_d / \partial w_i$
    - Với mỗi trọng số w_i
      - $w_i = w_i + \Delta w_i$

Luật học Gradient Descent

- Chế độ 1: cập nhật một lần

  $w=w - \eta \nabla E_D[w]$

  $E_D[w]=1/2\Sigma_d(t_d-o_d)^2$

- Chế độ 2 : cập nhật với từng mẫu

  $w=w - \eta \nabla E_d[w]$

  $E_d[w]=1/2 (t_d-o_d)^2$

So sánh luật học của Perceptron và Gradient Descent

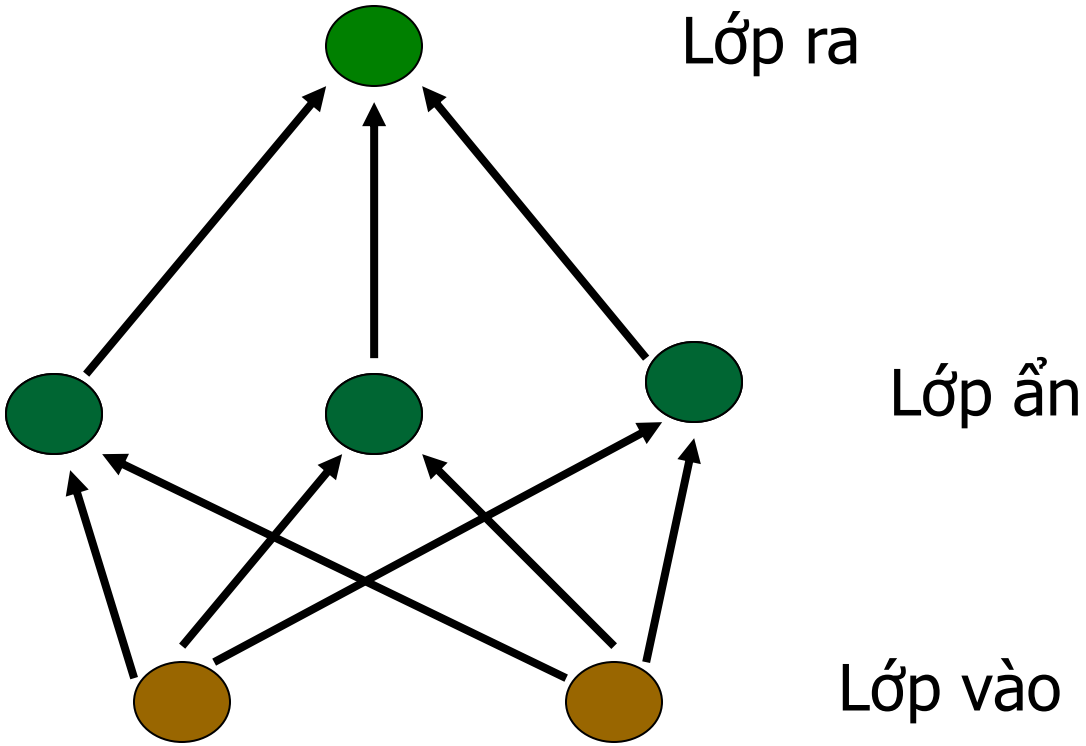
Perceptron đảm bảo thành công nếu

- Tập mẫu khả tách tuyến tính
- Hằng số học η đủ nhỏ

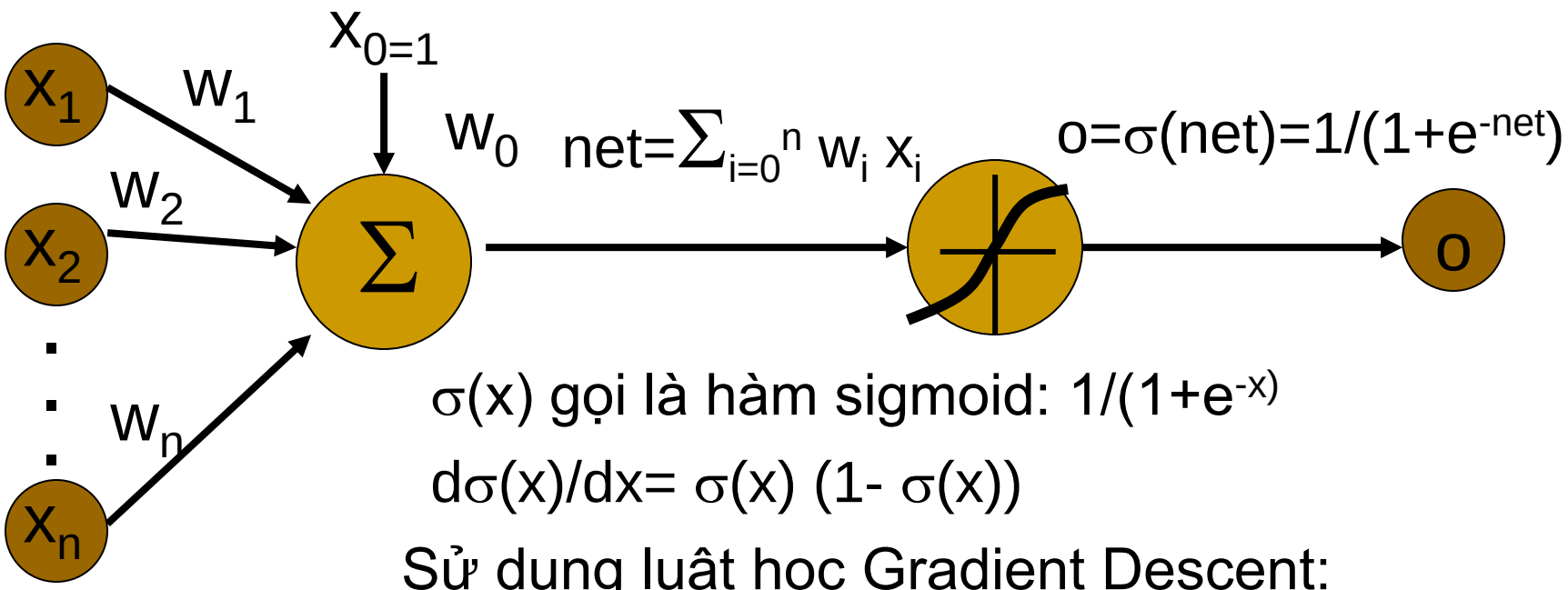
Huấn luyện với Gradient Descent

- Đảm bảo hội tụ với lỗi nhỏ nhất
- Hằng số học η nhỏ
- Có thể sử dụng mẫu có nhiễu
- Dữ liệu học thậm chí không khả tách tuyến tính

Mạng nơ ron nhiều lớp



Hàm Sigmoid



$\sigma(x)$ gọi là hàm sigmoid: $1/(1+e^{-x})$

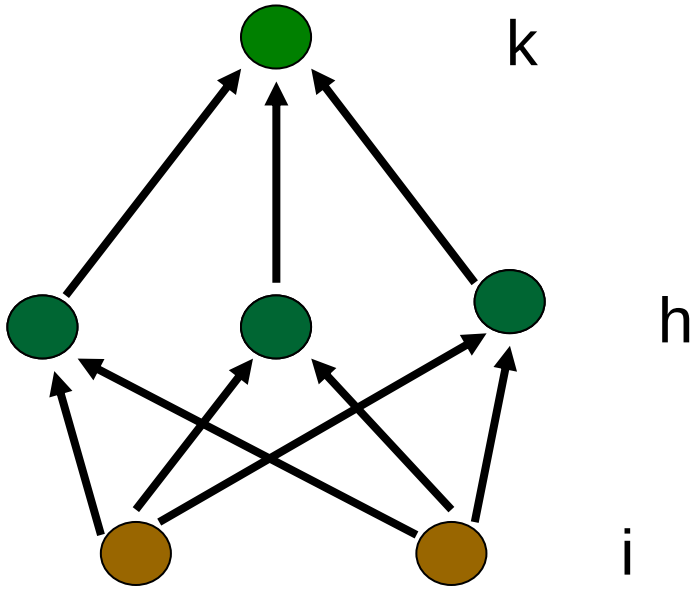
$d\sigma(x)/dx= \sigma(x)\ (1- \sigma(x))$

Sử dụng luật học Gradient Descent:

- $\partial E/\partial w_i = -\sum_d (t_d - o_d)\ o_d\ (1-o_d)$
- Mạng nhiều lớp thường sử dụng hàm Sigmoid làm hàm truyền

Giải thuật lan truyền ngược sai số

- Khởi tạo w_i với các giá trị ngẫu nhiên nhỏ
- Lặp cho tới khi đạt điều kiện dừng
  - Với mỗi mẫu huấn luyện $<(x_1,\ldots x_n),t>$
    - Đưa $(x_1,\ldots,x_n)$ vào mạng để tính toán giá trị ra o_k
    - Với mỗi đầu ra k
      - $\delta_k=o_k(1-o_k)(t_k-o_k)$
    - Với mỗi nút ẩn h
      - $\delta_h=o_h(1-o_h)\sum_k w_{h,k}\,\delta_k$
    - Với mỗi trọng số $w_{i,j}$
      - $w_{i,j}=w_{i,j}+\Delta w_{i,j}$ với $\Delta w_{i,j}=\eta\,\delta_j\,x_i$



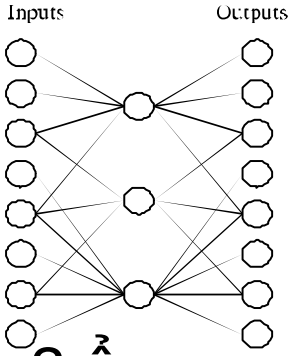
Giải thuật lan truyền ngược sai số

- Sử dụng Gradient Descent trên toàn bộ các trọng số
- Chỉ tìm được tối ưu cục bộ
- Thường bổ sung hệ số quán tính :

$$\Delta w_{i,j}(n) = \eta\, \delta_j\, x_i + \alpha\, \Delta w_{i,j}(n-1)$$

- Tối ưu được hàm lỗi trên tập mẫu nhưng chưa chắc trên dữ liệu thực tế (hiệu ứng overfitting)
- Thời gian học lâu (1000-10000 vòng)
- Sử dụng mô hình nhanh

8-3-8 mã hoá và giải mã



8 đầu vào 3 ẩn 8 đầu ra

A target function:

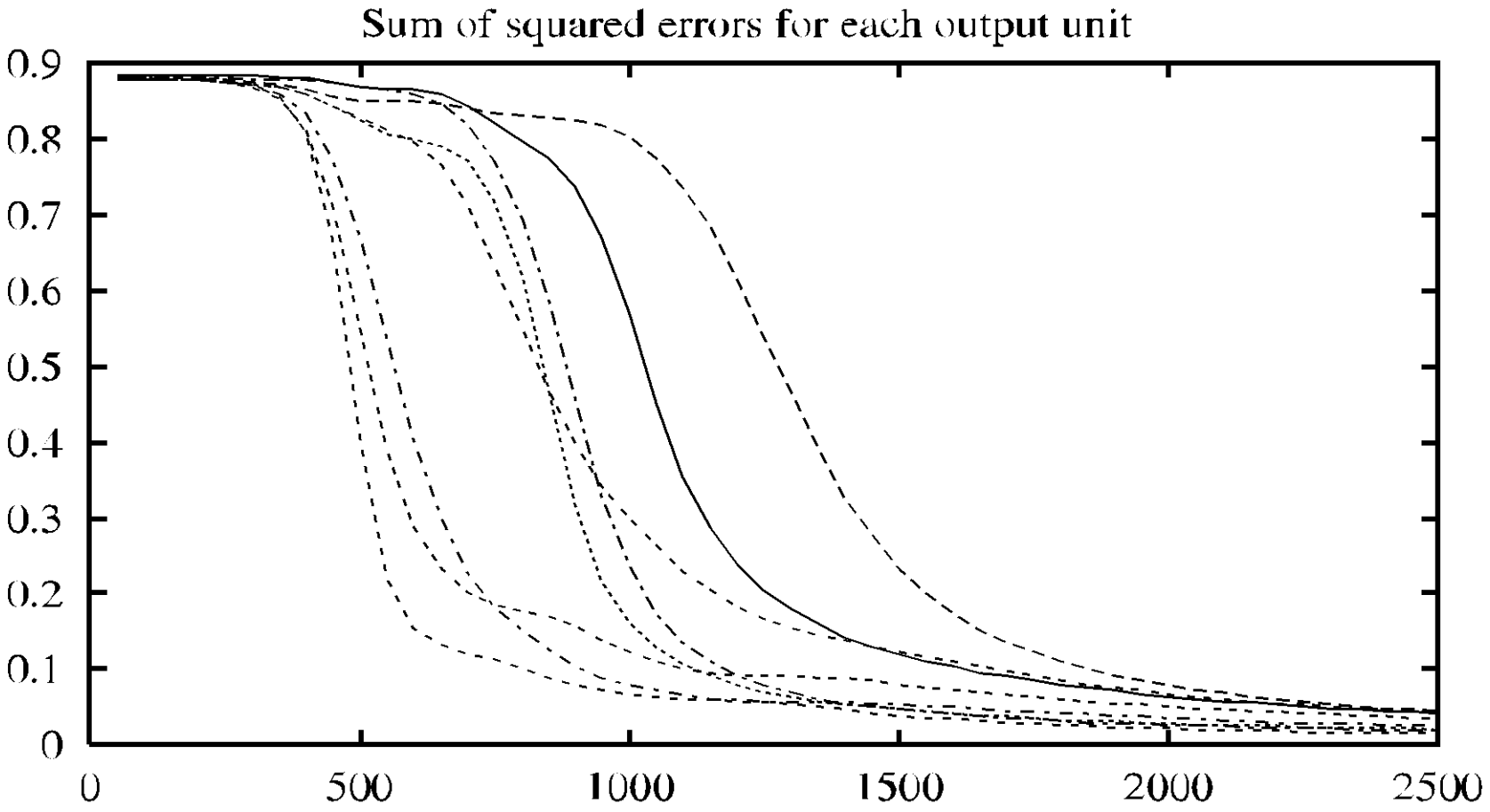
Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

Giá trị nút ẩn

.89 .04 .08
.01 .11 .88
.01 .97 .27
.99 .97 .71
.03 .05 .02
.22 .99 .99
.80 .01 .98
.60 .94 .01

Can this be learned??

Thống kê lỗi cho các nút ra



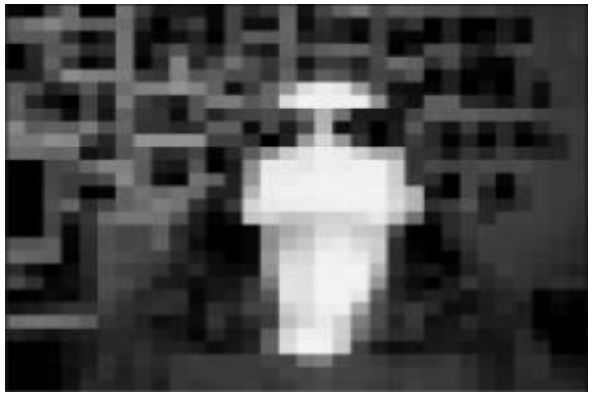
Sự hội tụ của giải thuật lan truyền ngược sai số

- Có thể không phải cực trị toàn cục
- Cải thiện bằng hệ số quán tính
- Kết hợp với một giải thuật tối ưu toàn cục khác

Điều kiện hội tụ

- Khởi tạo trọng số gần 0
- Mạng khởi tạo gần như mang tính chất tuyến tính
- Tính chất phi tuyến của mạng sẽ dần dần xuất hiện trong quá trình học

Ứng dụng của mạng nơron - Nhận dạng mặt



Ứng dụng của mạng nơron - Nhận dạng mặt

- Có nhiều hàm đích có thể học trong việc nhận dạng ảnh:
  - xác định người
  - hướng quay (trái, phải, thẳng, ...)
  - giới tính
  - có đeo kính hay không

Ứng dụng của mạng nơron - Nhận dạng mặt

- Nhiệm vụ học: phân loại các hình ảnh camera về mặt người với nhiều góc độ khác nhau
- CSDL hình ảnh
- Các hình ảnh với 624 grayscale: 20 người, mỗi người khoảng 32 ảnh
- Nhiều cách biểu cảm (vui, buồn, giận, bình thường)
- Các hướng khác nhau (trái, phải, thẳng, hướng lên)
- Độ phân giải 120x128

- Học hướng quay của mặt người:
  - không cần các lựa chọn tối ưu, phương pháp này cho kết quả tốt
  - sau khi luyện trên 260 hình ảnh, việc phân loại đạt độ chính xác trên tập thử là 90%

Các lựa chọn

1. Mã hoá đầu vào: hình ảnh hay các đặc tính
2. Mã hoá đầu ra: số lượng đầu ra, các hàm đích cho đầu ra
3. Cấu trúc mạng: số lượng nút mạng và liên kết giữa chúng
4. Các tham số thuật toán học
   - Tốc độ học
   - giá trị momentum

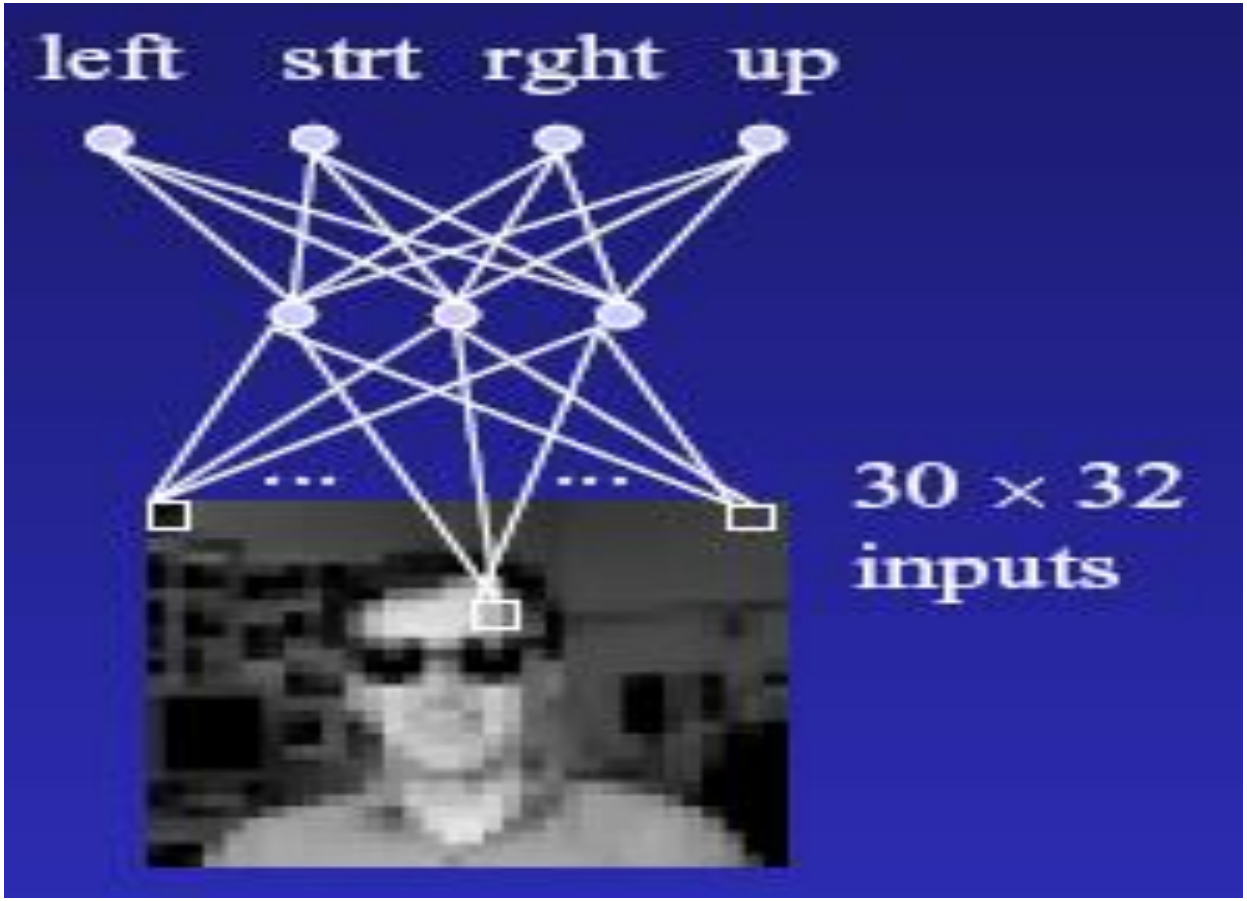
Mã hoá đầu vào

- Thiết kế các lựa chọn
- Tiền xử lý hình ảnh để rút ra các hướng, các vùng có mật độ giống nhau, hoặc các đặc tính hình ảnh cục bộ khác
- Khó khăn: số cạnh có thể thay đổi, trong khi NN có số lượng cố định các đầu vào
- Các hình đã mã hoá là 1 tập cố định các giá trị mật độ 30x32 điểm ảnh (tóm tắt về độ phân giải của ảnh ban đầu), từ 0 đến 255

Mã hoá đầu ra

- Mỗi đầu ra: 4 giá trị xác định hướng mà người nhìn (trái, phải, thẳng, hướng lên)
- Mỗi đơn vị: phân loại sử dụng 1 đầu ra, gán 0.2, 0.4, 0.6 và 0.8 cho 4 giá trị
- Chọn 1 trong n đầu ra mã hoá:
  - cung cấp nhiều mức độ tự do để biểu diễn hàm đích (n lần số trọng số ở tầng ra)
  - độ khác nhau giữa giá trị cao nhất và nhì dùng để đo độ tin cậy

Cấu trúc mạng



mạng 960 x 3 x 4