Chương 5 - Học máy

Lê Thanh Hương
Viện CNTT & TT - ĐHBK HN

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. Học máy

- Tổng quan
- Học cây quyết định
- K láng giềng gần
- Mạng nơron

Giới thiệu về Học máy

- Định nghĩa

"Học đề cập đến các thay đổi của hệ thống theo hướng thích nghi: chúng cho phép hệ thống thực hiện các công việc trong cùng một môi trường hiệu quả hơn từ lần thực hiện thứ 2"

Biểu diễn một bài toán học máy [Mitchell, 1997]

Học máy = Cải thiện hiệu quả một công việc thông qua kinh nghiệm

- Một công việc (nhiệm vụ) **T**
- Đối với các tiêu chí đánh giá hiệu suất **P**
- Thông qua (sử dụng) kinh nghiệm **E**

Các ví dụ về học máy (1)

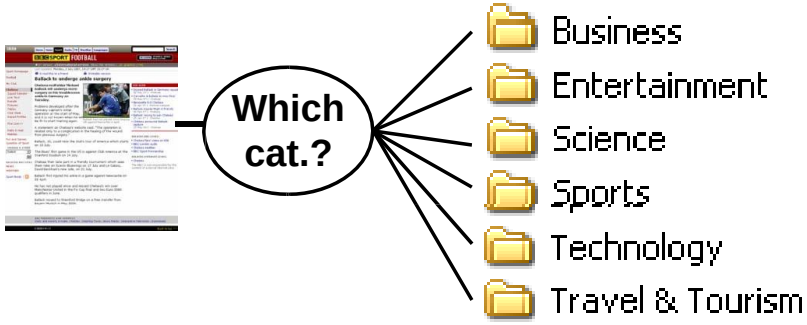
Bài toán lọc các trang Web theo sở thích của một người dùng

- **T**: Dự đoán (để lọc) xem những trang Web nào mà một người dùng cụ thể thích đọc
- **P**: Tỷ lệ (%) các trang Web được dự đoán đúng
- **E**: Một tập các trang Web mà người dùng đã chỉ định là thích đọc và một tập các trang Web mà anh ta đã chỉ định là không thích đọc

Các ví dụ về học máy (2)

Bài toán phân loại các trang Web theo các chủ đề

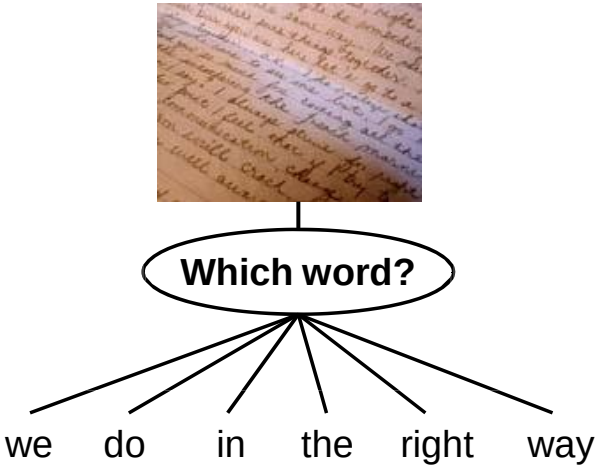
- **T**: Phân loại các trang Web theo các chủ đề đã định trước
- **P**: Tỷ lệ (%) các trang Web được phân loại chính xác
- **E**: Một tập các trang Web, trong đó mỗi trang Web gắn với một chủ đề



Các ví dụ về học máy (3)

Bài toán nhận dạng chữ viết tay

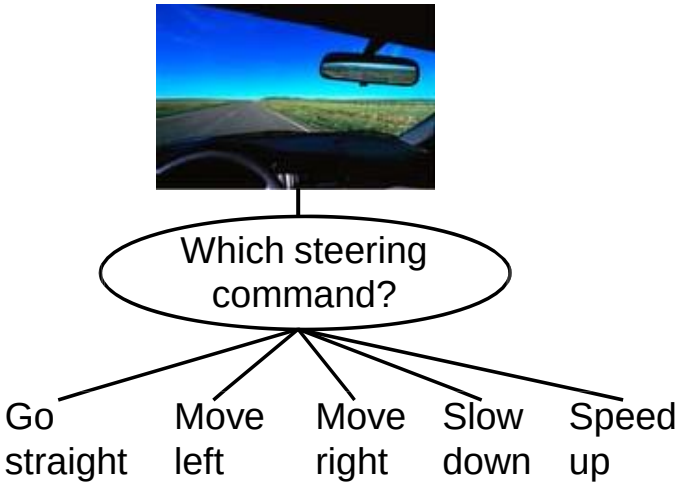
- **T**: Nhận dạng và phân loại các từ trong các ảnh chữ viết tay
- **P**: Tỷ lệ (%) các từ được nhận dạng và phân loại đúng
- **E**: Một tập các ảnh chữ viết tay, trong đó mỗi ảnh được gắn với một định danh của một từ



Các ví dụ về học máy (4)

Bài toán robot lái xe tự động

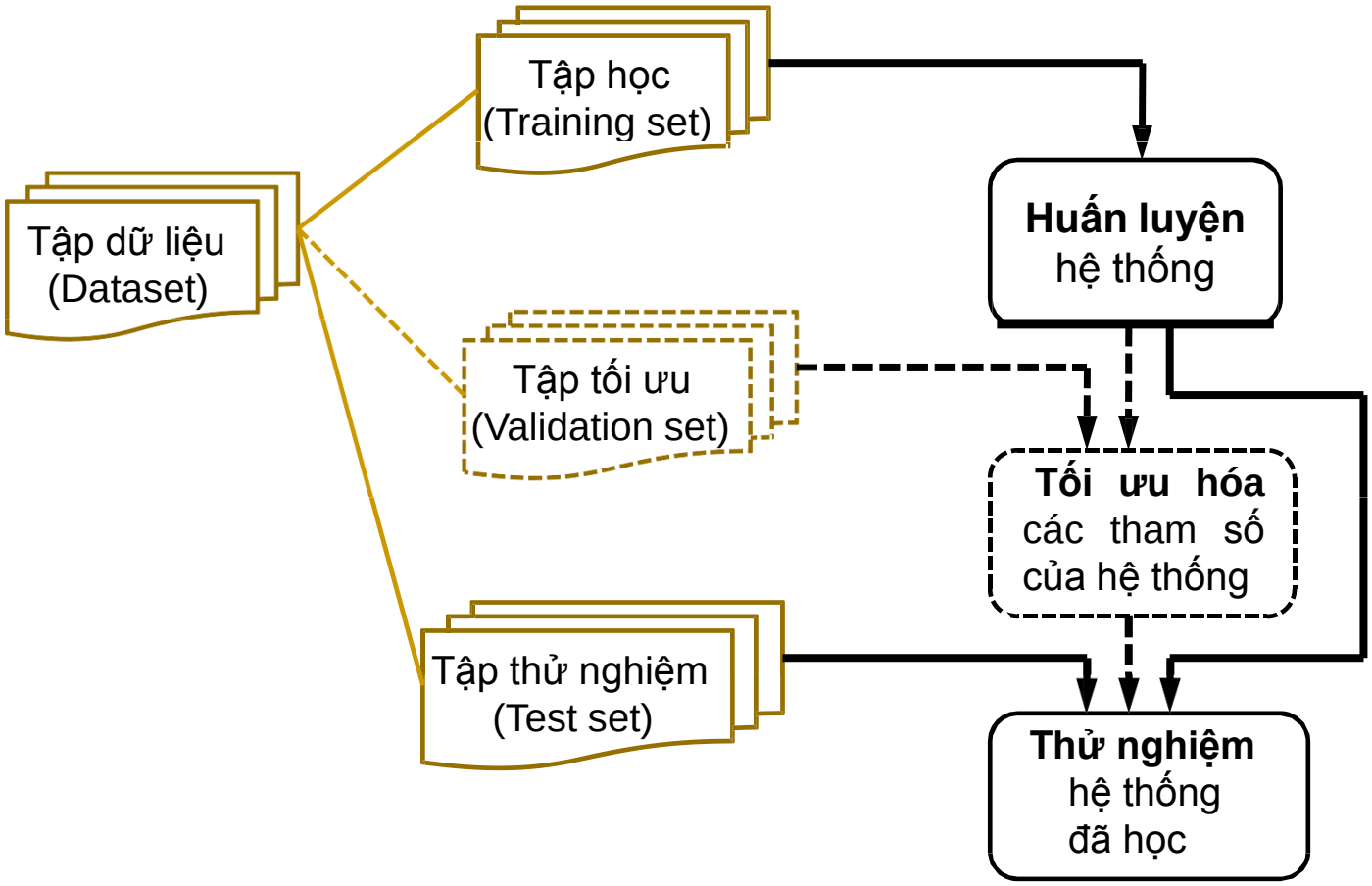
- **T**: Robot (được trang bị các camera quan sát) lái xe tự động trên đường cao tốc
- **P**: Khoảng cách trung bình mà robot có thể lái xe tự động trước khi xảy ra lỗi (tai nạn)
- **E**: Một tập các ví dụ được ghi lại khi quan sát một người lái xe trên đường cao tốc, trong đó mỗi ví dụ gồm một chuỗi các ảnh và các lệnh điều khiển xe



Các phương pháp học

- **Học có giám sát**: biết trước câu trả lời đúng
- **Học không giám sát**: không biết trước câu trả lời đúng
- **Học tăng cường**: đôi khi có thưởng/phạt cho các hành động

Quá trình học máy



Học có vs. không có giám sát

- Học có giám sát (supervised learning)
  - Mỗi ví dụ học gồm 2 phần: mô tả (biểu diễn) của ví dụ học, và nhãn lớp (hoặc giá trị đầu ra mong muốn) của ví dụ học đó
  - Bài toán học **phân lớp (classification problem)**

D_train = {(<Biểu_diễn_của_x>, <Nhãn_lớp_của_x>)}

  - Bài toán học **dự đoán/hồi quy (prediction/regression problem)**

D_train = {(<Biểu_diễn_của_x>, <Giá_trị_đầu_ra_của_x>)}

- Học không có giám sát (unsupervised learning)
  - Mỗi ví dụ học chỉ chứa mô tả (biểu diễn) của ví dụ học đó - mà không có bất kỳ thông tin nào về nhãn lớp hay giá trị đầu ra mong muốn của ví dụ học đó
  - Bài toán học **phân cụm (Clustering problem)**

Tập học D_train = {(<Biểu_diễn_của_x>)}

Các thành phần chính (1)

- Lựa chọn các ví dụ học (training/learning examples)
  - Các thông tin hướng dẫn quá trình học (training feedback) được chứa ngay trong các ví dụ học, hay là được cung cấp gián tiếp (vd: từ môi trường hoạt động)
  - Các ví dụ học theo kiểu có giám sát (supervised) hay không có giám sát (unsupervised)
  - Các ví dụ học phải tương thích với các ví dụ sẽ được sử dụng bởi hệ thống trong tương lai (future test examples)
- Xác định hàm mục tiêu (giả thiết, khái niệm) cần học
  - F: X → {0,1}
  - F: X → {Một tập các nhãn lớp}
  - F: X → R^+ (miền các giá tri số thực dương)
  - …

Các thành phần chính (2)

- Lựa chọn cách biểu diễn cho hàm mục tiêu cần học
  - Hàm đa thức (a polynomial function)
  - Một tập các luật (a set of rules)
  - Một cây quyết định (a decision tree)
  - Một mạng nơ-ron nhân tạo (an artificial neural network)
  - …
- Lựa chọn một giải thuật học máy có thể học (xấp xỉ) được hàm mục tiêu
  - Phương pháp học hồi quy (Regression-based)
  - Phương pháp học quy nạp luật (Rule induction)
  - Phương pháp học cây quyết định (ID3 hoặc C4.5)
  - Phương pháp học lan truyền ngược (Back-propagation)
  - …

Các vấn đề trong Học máy (1)

- Giải thuật học máy (Learning algorithm)
  - Những giải thuật học máy nào có thể học (xấp xỉ) một hàm mục tiêu cần học?
  - Với những điều kiện nào, một giải thuật học máy đã chọn sẽ hội tụ (tiệm cận) hàm mục tiêu cần học?
  - Đối với một lĩnh vực bài toán cụ thể và đối với một cách biểu diễn các ví dụ (đối tượng) cụ thể, giải thuật học máy nào thực hiện tốt nhất?

Các vấn đề trong Học máy (2)

- Các ví dụ học (Training examples)
  - Bao nhiêu ví dụ học là đủ?
  - Kích thước của tập học (tập huấn luyện) ảnh hưởng thế nào đối với độ chính xác của hàm mục tiêu học được?
  - Các ví dụ lỗi (nhiễu) và/hoặc các ví dụ thiếu giá trị thuộc tính (missing-value) ảnh hưởng thế nào đối với độ chính xác?

Các vấn đề trong Học máy (3)

- Quá trình học (Learning process)
  - Chiến lược tối ưu cho việc lựa chọn thứ tự sử dụng (khai thác) các ví dụ học?
  - Các chiến lược lựa chọn này làm thay đổi mức độ phức tạp của bài toán học máy như thế nào?
  - Các tri thức cụ thể của bài toán (ngoài các ví dụ học) có thể đóng góp thế nào đối với quá trình học?

Các vấn đề trong Học máy (4)

- Khả năng/giới hạn học
  - Hàm mục tiêu nào mà hệ thống cần học?
    - Biểu diễn hàm mục tiêu: Khả năng biểu diễn (vd: hàm tuyến tính / hàm phi tuyến) vs. Độ phức tạp của giải thuật và quá trình học
  - Các giới hạn đối với khả năng học của các giải thuật học máy?
  - Khả năng khái quát hóa của hệ thống từ các ví dụ học?
    - Để tránh vấn đề "over-fitting" (đạt độ chính xác cao trên tập học, nhưng đạt độ chính xác thấp trên tập thử nghiệm)
- Khả năng hệ thống tự động thay đổi (thích nghi) biểu diễn cấu trúc bên trong của nó?

Vấn đề over-fitting (1)

- Một hàm mục tiêu học được *h* sẽ được gọi là **quá khớp (over-fit)** với một tập học nếu tồn tại một hàm mục tiêu khác *h'* sao cho:
  - *h'* kém phù hợp hơn (đạt độ chính xác kém hơn) *h* đối với tập học, nhưng
  - *h'* đạt độ chính xác cao hơn *h* đối với toàn bộ tập dữ liệu (bao gồm cả những ví dụ được sử dụng sau quá trình huấn luyện)
- Vấn đề over-fitting thường do các nguyên nhân:
  - Lỗi trong tập huấn luyện
  - Số lượng các ví dụ học quá nhỏ, không đại diện cho toàn bộ tập của các ví dụ của bài toán học

Vấn đề over-fitting (2)

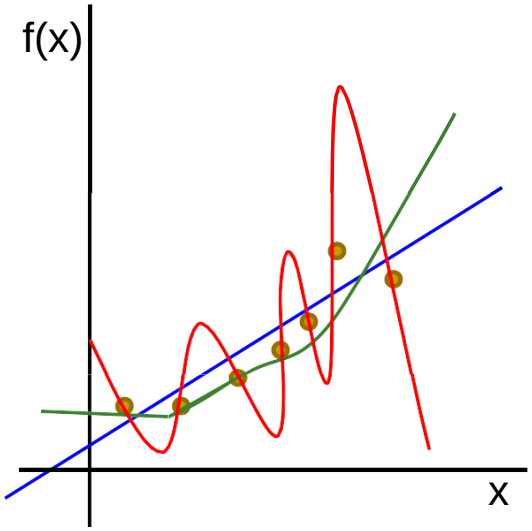
- Giả sử gọi D là tập toàn bộ các ví dụ, và D_train là tập các ví dụ học
- Giả sử gọi $Err_D(h)$ là mức lỗi mà giả thiết h sinh ra đối với tập D, và $Err_{D\_train}(h)$ là mức lỗi mà giả thiết h sinh ra đối với tập D_train
- Giả thiết h quá khớp (quá phù hợp) tập học D_train nếu tồn tại một giả thiết khác h':
  - $Err_{D\_train}(h) < Err_{D\_train}(h')$, và
  - $Err_D(h) > Err_D(h')$

Vấn đề over-fitting (3)

Trong số các hàm mục tiêu học được, hàm mục tiêu nào khái quát hóa tốt nhất từ các ví dụ học?

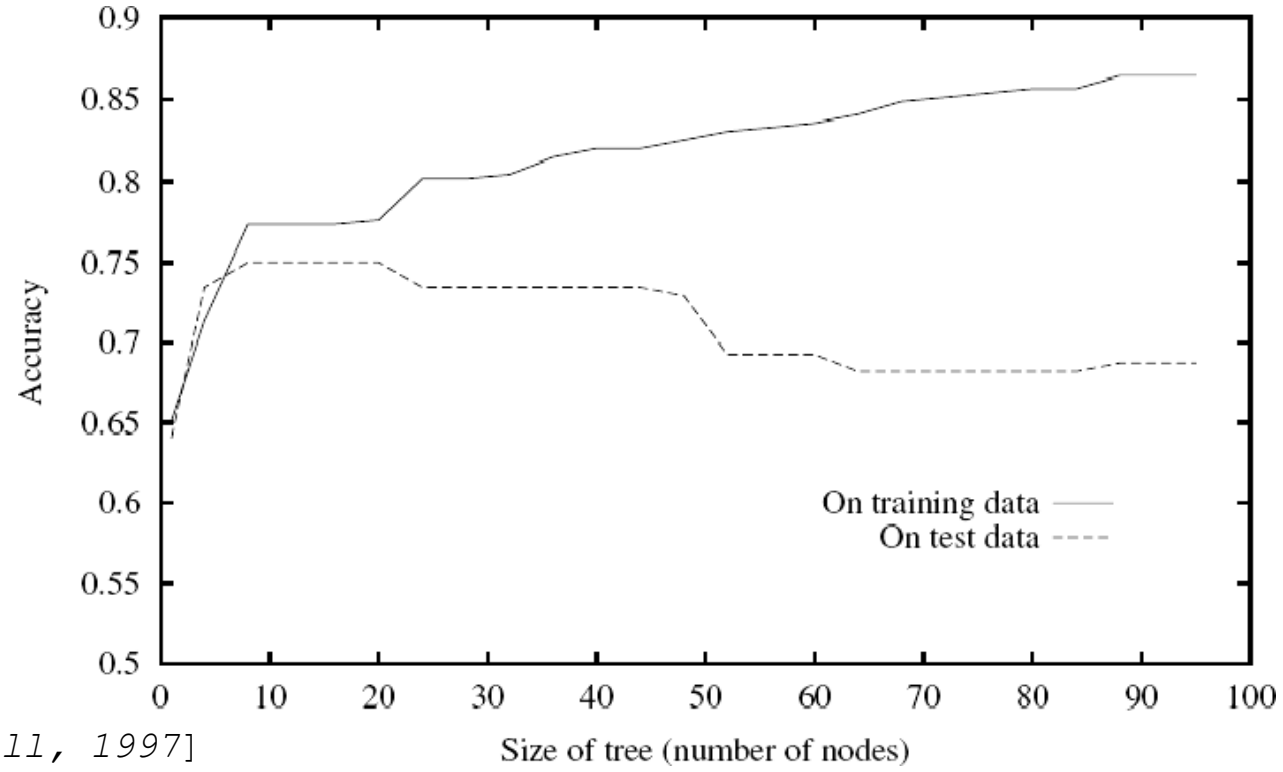
Lưu ý: Mục tiêu của học máy là để đạt được độ chính xác cao trong dự đoán đối với các ví dụ sau này, không phải đối với các ví dụ học

- **Occam's razor**: Ưu tiên chọn hàm mục tiêu đơn giản nhất phù hợp (không nhất thiết hoàn hảo) với các ví dụ học
  - → Khái quát hóa tốt hơn
  - → Dễ giải thích/diễn giải hơn
  - → Độ phức tạp tính toán ít hơn



Vấn đề over-fitting – Ví dụ

Tiếp tục quá trình học cây quyết định sẽ làm giảm độ chính xác đối với tập thử nghiệm mặc dù tăng độ chính xác đối với tập học



[*Mitchell, 1997*]

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. Học máy

- Tổng quan
- Học cây quyết định
- K láng giềng gần
- 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

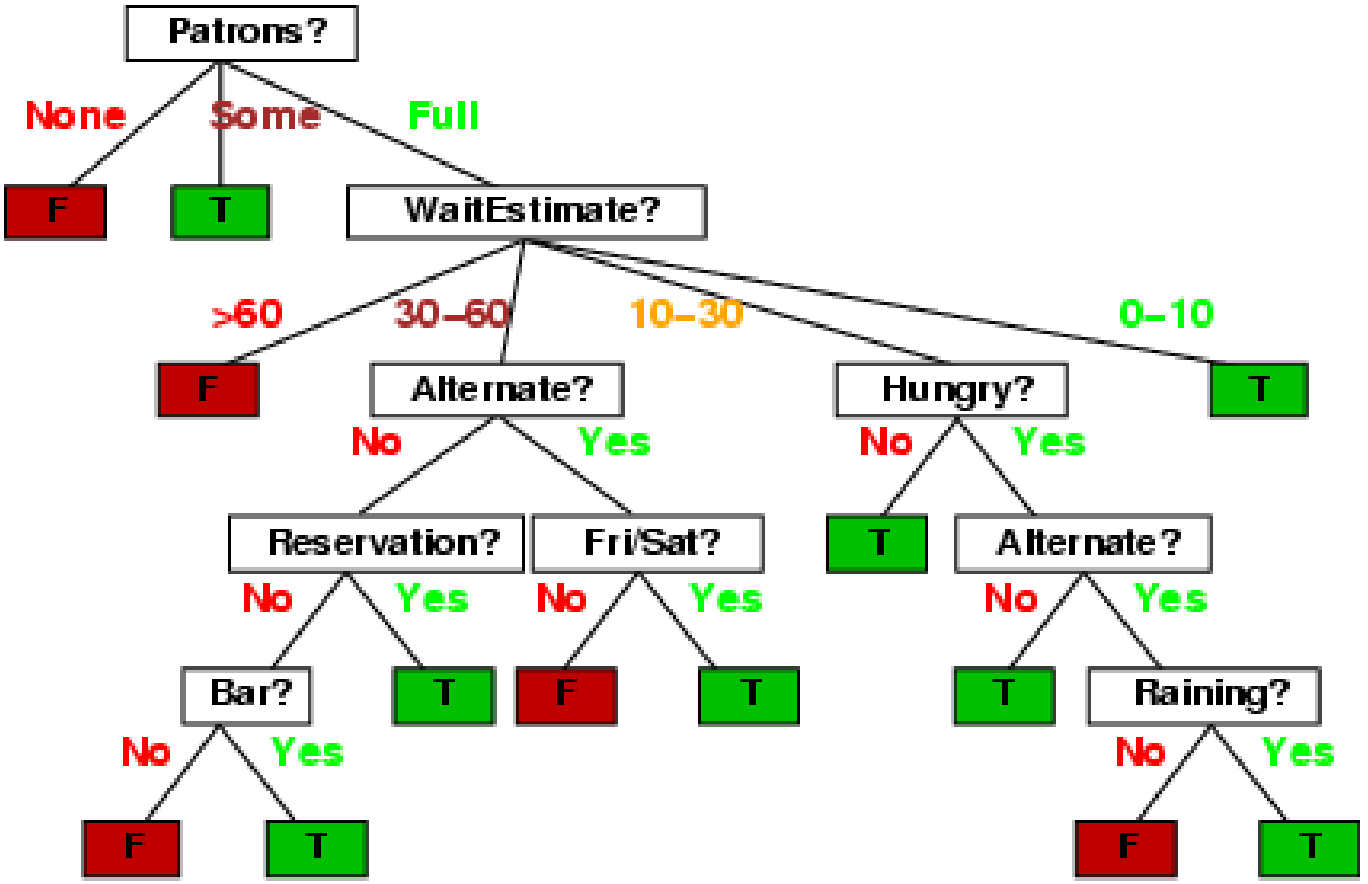
- Các loại (lớp) của mẫu là khẳng định (T) hoặc phủ định (F)

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

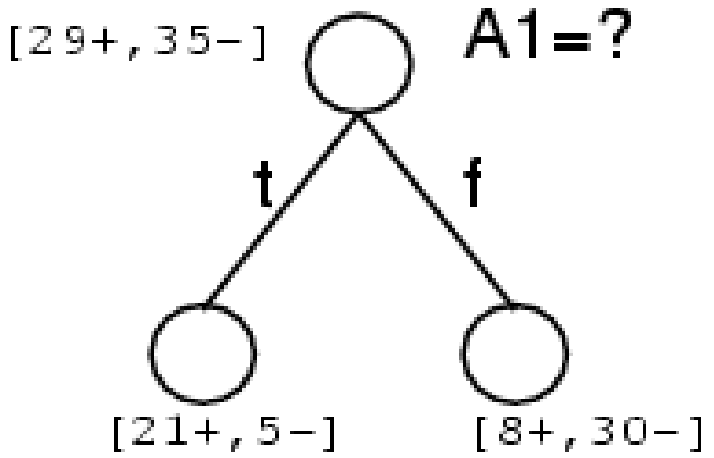
Attributes: các thuộc tính có thể được kiểm tra qua phép học cây quyết định. */

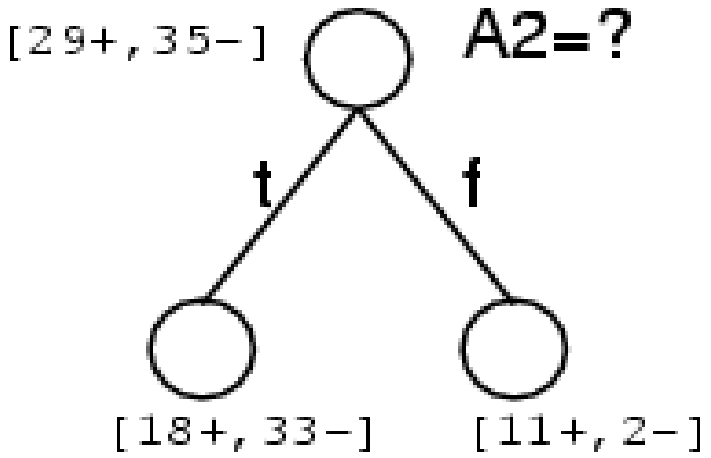
- Tạo 1 nút gốc *Root* cho cây
- If ∀ *Examples* +, trả về cây chỉ có 1 nút Root, với nhãn +
- If ∀ *Examples* -, trả về cây chỉ có 1 nút Root, với nhãn –
- If *Attributes* rỗng, trả về cây chỉ có 1 nút *Root*, với nhãn = giá trị thường xuất hiện nhất của *Target_attribute* trong *Examples*

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?

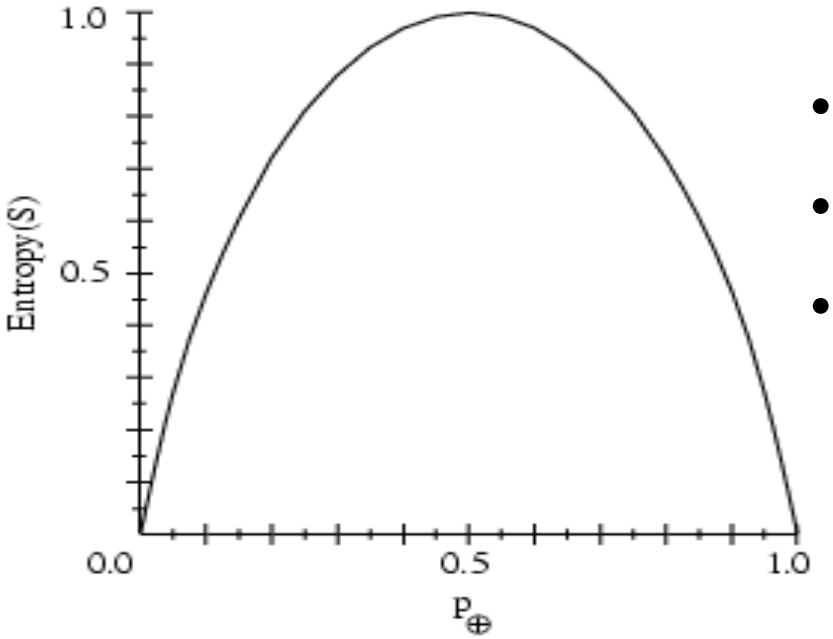




Sử dụng lượng thông tin đạt được Information Gain

⇨ 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-])

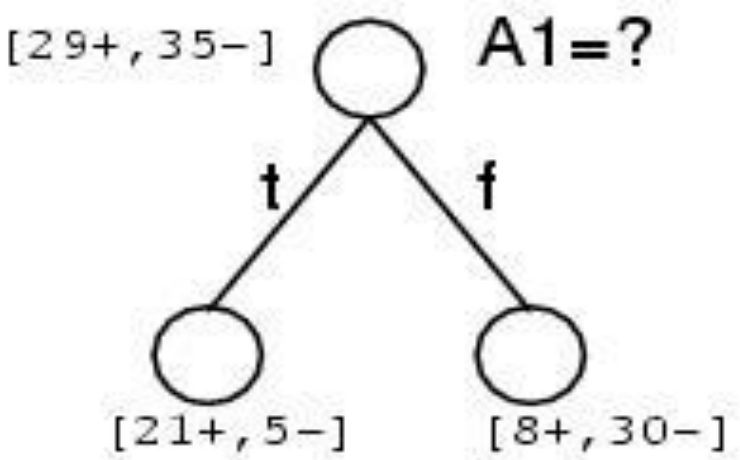
= - (9/14)$\log_2$(9/14) – (5/14)$\log_2$(5/14)

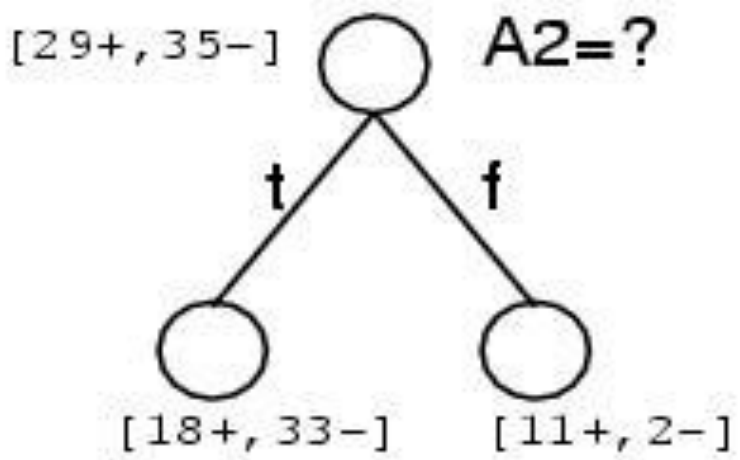
= 0.940

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-];

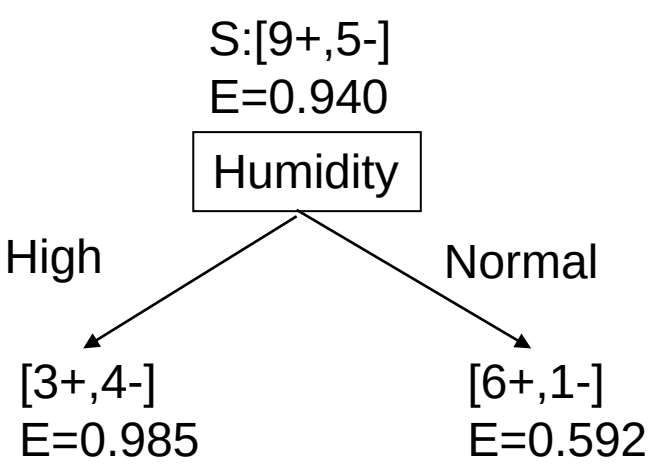
S_{normal}=[6+,1-]

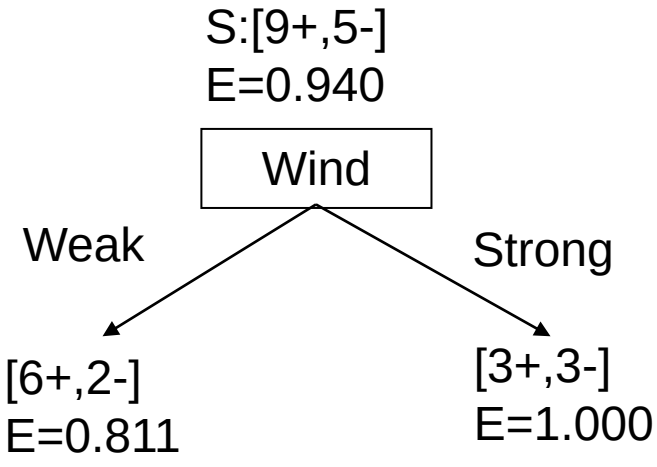
Wind ={Weak,Strong}:

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

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

Thuộc tính nào phân loại tốt nhất?





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

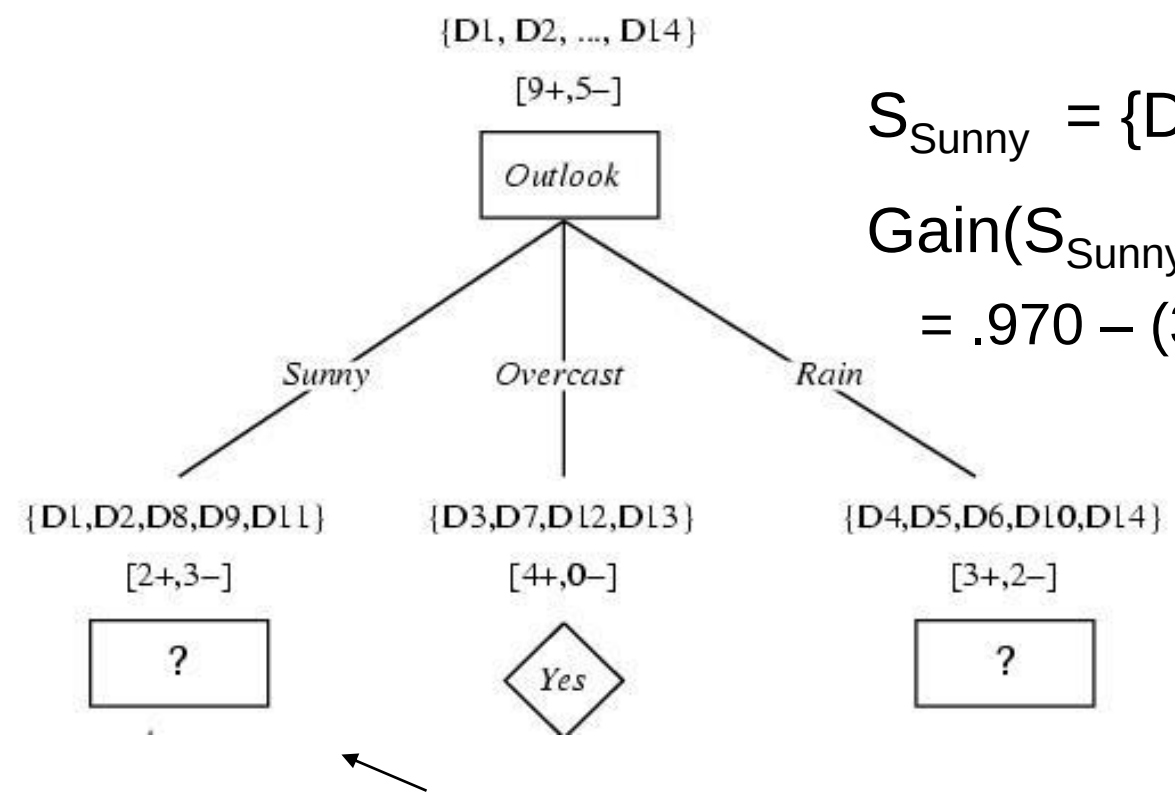
= Entropy(S) – (8/14)Entropy(S_{Weak}) – (6/14)Entropy(S_{Strong})

= 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

Đánh giá và lựa chọn mô hình

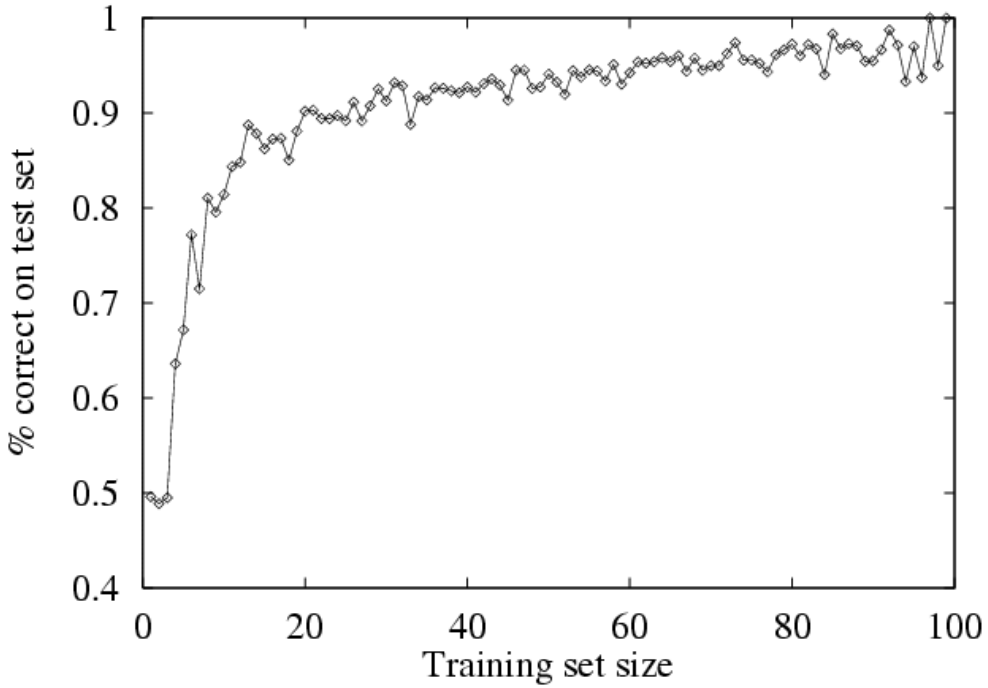
- Cho trước tập quan sát D, ta cần lựa chọn tham số λ (model selection) cho phương pháp học A và đánh giá (assessment) chất lượng tổng thể của A.
  - Chọn tập hữu hạn S mà chứa các giá trị tiềm năng cho λ.
  - Chọn độ đo P để đánh giá hiệu năng.
  - Chia tập D thành 3 tập rời nhau: D_{train}, $T_{validation}$, và T_{test}
  - Với mỗi giá trị $\lambda \in S$:
    - Học A từ tập học D_{train} với tham số đầu vào λ. Đo hiệu năng trên tập $T_{validation}$ → thu được P_λ
  - Chọn λ^* mà có P_λ tốt nhất.
  - Huấn luyện A trên tập $D_{train} \cup T_{validation}$, với tham số đầu vào λ^*.
  - Đo hiệu năng của hệ thống trên tập T_{test}.
- Có thể thay Hold-out bằng kỹ thuật khác (cross-validation).

Đo độ chính xác

- Làm sao để biết $h \approx f$?
- Sử dụng lý thuyết tính toán

1. Thử giả thiết h trên 1 tập các ví dụ mới (tập thử) (sử dụng cùng 1 mức độ phân bố các mẫu như tập luyện)

Learning curve = % chính xác trên tập thử, sử dụng hàm xây dựng trên tập luyện



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. Học máy

- Tổng quan
- Học cây quyết định
- K láng giềng gần
- Mạng nơron

Học dựa trên các láng giềng gần nhất

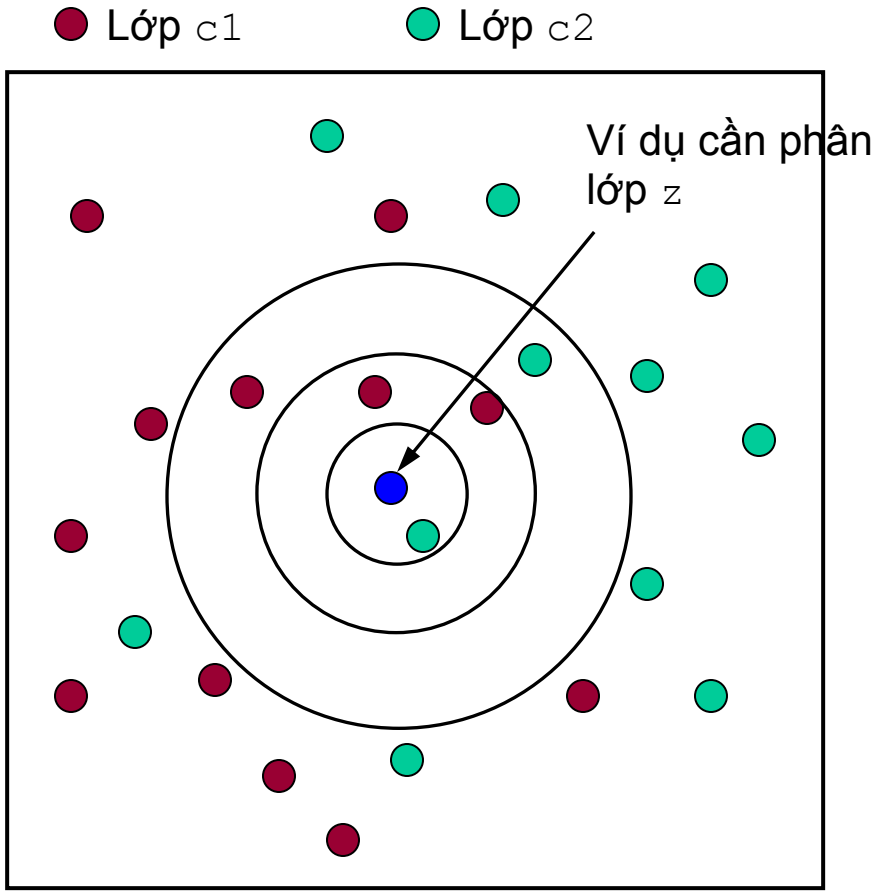
- Một số tên gọi khác của phương pháp **học dựa trên các láng giềng gần nhất (Nearest neighbor learning)**
  - Instance-based learning
  - Lazy learning
  - Memory-based learning
- Ý tưởng của phương pháp học dựa trên các láng giềng gần nhất
  - Với một tập các ví dụ học
    - (Đơn giản là) lưu lại các ví dụ học
    - Không cần xây dựng một mô hình (mô tả) rõ ràng và tổng quát của hàm mục tiêu cần học
  - Đối với một ví dụ cần phân loại/dự đoán
    - Xét quan hệ giữa ví dụ đó với các ví dụ học để gán giá trị của hàm mục tiêu (một nhãn lớp, hoặc một giá trị thực)

Học dựa trên các láng giềng gần nhất

- Biểu diễn đầu vào của bài toán
  - Mỗi ví dụ x được biểu diễn là một vectơ n chiều trong không gian các vectơ $X \in R^n$
  - $x = (x_1, x_2, \ldots, x_n)$, trong đó x_i ($\in R$) là một số thực
- Có thể áp dụng được với cả 2 kiểu bài toán học
  - Bài toán *phân lớp (classification)*
    - Hàm mục tiêu có giá trị rời rạc (a discrete-valued target function)
    - Đầu ra của hệ thống là một trong số các giá trị rời rạc đã xác định trước (một trong các nhãn lớp)
  - Bài toán *dự đoán/hồi quy (prediction/regression)*
    - Hàm mục tiêu có giá trị liên tục (a continuous-valued target function)
    - Đầu ra của hệ thống là một giá trị số thực

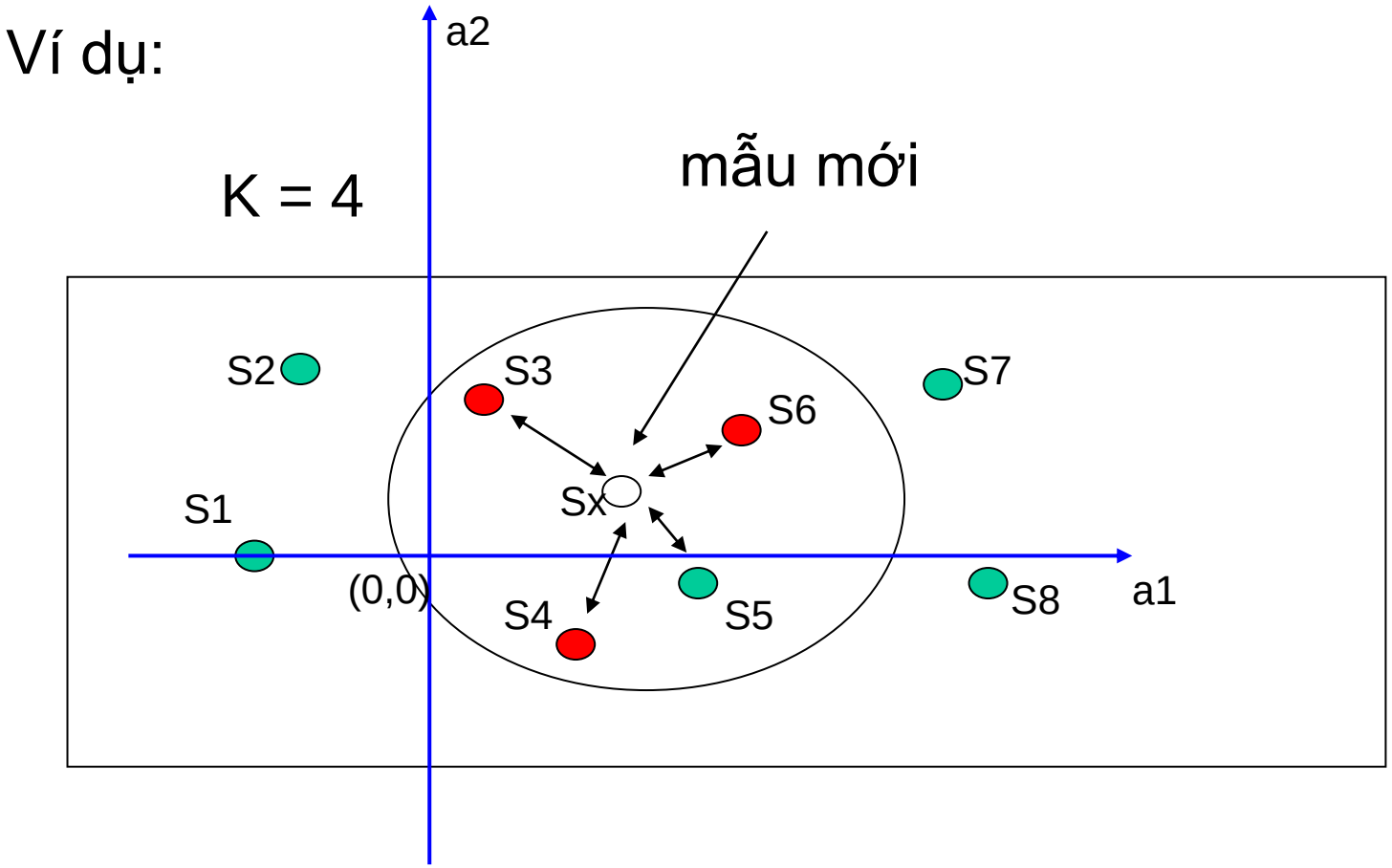
Ví dụ bài toán phân lớp

- Xét 1 láng giềng gần nhất
  → Gán z vào lớp c2
- Xét 3 láng giềng gần nhất
  → Gán z vào lớp c1
- Xét 5 láng giềng gần nhất
  → Gán z vào lớp c1



kNN

Ví dụ:



	a1	a2	T
S1	-2	0	B
S2	-1.5	2	B
S3	0.4	1.8	R
S4	1.5	-1	R
S5	3	-0.2	B
S6	3.2	1.5	R
S7	4.5	2	B
S8	4.7	-0.2	B
Sx	2	0.5	X

kNN

- Để định nghĩa độ tương tự giữa 2 TH, ta dùng ma trận.
- Giả sử các mẫu là các điểm trong không gian n chiều R^n và dùng khoảng cách Euclidean
- Cho X_i và X_j là 2 ví dụ. Khoảng cách của chúng là

$$d^2(X_i, X_j) = \sum_k \left[x_{ik} - x_{jk}\right]^2$$

trong đó x_{ik} là giá trị của thuộc tính k trên ví dụ X_i.

Giải thuật phân lớp k-NN

- Mỗi ví dụ học x được biểu diễn bởi 2 thành phần:
  - Mô tả của ví dụ: $x=(x_1,x_2,\ldots,x_n)$, trong đó $x_i \in R$
  - Nhãn lớp : c ($\in C$, với C là tập các nhãn lớp được xác định trước)
- Giai đoạn học
  - Đơn giản là lưu lại các ví dụ học trong tập học: $D = \{x\}$
- Giai đoạn phân lớp: Để phân lớp cho một ví dụ (mới) z
  - Với mỗi ví dụ học $x \in D$, tính khoảng cách giữa x và z
  - Xác định tập $NB(z)$ – các láng giềng gần nhất của z

    → Gồm k ví dụ học trong D gần nhất với z tính theo một hàm khoảng cách d
  - Phân z vào lớp chiếm số đông (the majority class) trong số các lớp của các ví dụ học trong $NB(z)$

Giải thuật kNN cho các giá trị rời rạc

Thuật toán (tham số k)

1. Với mỗi mẫu luyện (x, y_x), bổ sung mẫu vào tập luyện
2. Khi có mẫu mới z, gán lớp:

   y_z = lớp của đa số các thành viên trong k láng giềng gần nhất của z

$$\hat{f}(Xq) = \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(Xi))$$

với δ(a,b) = 1 nếu a = b và 0 nếu ngược lại

Giải thuật dự đoán k-NN

- Mỗi ví dụ học x được biểu diễn bởi 2 thành phần:
  - Mô tả của ví dụ: $x=(x_1,x_2,\ldots,x_n)$, trong đó $x_i \in R$
  - Giá trị đầu ra mong muốn: $y_x \in R$ (là một số thực)
- Giai đoạn học
  - Đơn giản là lưu lại các ví dụ học trong tập học D
- Giai đoạn dự đoán: Để dự đoán giá trị đầu ra cho ví dụ z
  - Đối với mỗi ví dụ học $x \in D$, tính khoảng cách giữa x và z
  - Xác định tập $NB(z)$ – các láng giềng gần nhất của z

    → Gồm k ví dụ học trong D gần nhất với z tính theo một hàm khoảng cách d
  - Dự đoán giá trị đầu ra đối với z: $y_z = \frac{1}{k}\sum_{x \in NB(z)} y_x$

Một hay nhiều láng giềng gần nhất?

- Việc phân lớp (hay dự đoán) chỉ dựa trên duy nhất một láng giềng gần nhất (là ví dụ học gần nhất với ví dụ cần phân lớp/dự đoán) thường *không* chính xác
  - Nếu ví dụ học này là một ví dụ bất thường, không điển hình (an outlier) – rất khác so với các ví dụ khác
  - Nếu ví dụ học này có nhãn lớp (giá trị đầu ra) sai – do lỗi trong quá trình thu thập (xây dựng) tập dữ liệu
- Thường xét k (>1) các ví dụ học (các láng giềng) gần nhất với ví dụ cần phân lớp/dự đoán
- Đối với bài toán phân lớp có 2 lớp, k thường được chọn là một số lẻ, để tránh cân bằng về tỷ lệ các ví dụ giữa 2 lớp
  - Ví dụ: k= 3, 5, 7,…

Hàm tính khoảng cách (1)

- Hàm tính khoảng cách ***d***
  - Đóng vai trò rất quan trọng trong phương pháp học dựa trên các láng giềng gần nhất
  - Thường được xác định trước, và không thay đổi trong suốt quá trình học và phân loại/dự đoán
- Lựa chọn hàm khoảng cách ***d***
  - *Các hàm khoảng cách hình học*: Dành cho các bài toán có các thuộc tính đầu vào là kiểu số thực ($x_i \in \mathbb{R}$)
  - *Hàm khoảng cách Hamming*: Dành cho các bài toán có các thuộc tính đầu vào là kiểu nhị phân ($x_i \in \{0,1\}$)
  - *Hàm tính độ tương tự Cosine*: Dành cho các bài toán phân lớp văn bản (x_i là giá trị trọng số TF/IDF của từ khóa thứ *i*)

Hàm tính khoảng cách (2)

- Các hàm tính khoảng cách hình học (Geometry distance functions)
  - Hàm Minkowski (p-norm):

$$d(x,z) = \left(\sum_{i=1}^{n} |x_i - z_i|^p \right)^{1/p}$$

  - Hàm Manhattan (p=1):

$$d(x,z) = \sum_{i=1}^{n} |x_i - z_i|$$

  - Hàm Euclid (p=2):

$$d(x,z) = \sqrt{\sum_{i=1}^{n} (x_i - z_i)^2}$$

  - Hàm Chebyshev (p=∞):

$$d(x,z) = \lim_{p \to \infty} \left(\sum_{i=1}^{n} |x_i - z_i|^p \right)^{1/p}$$

$$= \max_i |x_i - z_i|$$

Hàm tính khoảng cách (3)

- Hàm khoảng cách Hamming
  - Đối với các thuộc tính đầu vào là kiểu nhị phân ({0,1})
  - Ví dụ: x=(0,1,0,1,1)

$$d(x,z) = \sum_{i=1}^{n} Difference(x_i, z_i)$$

$$Difference(a,b) = \begin{cases} 1, if\ (a \neq b) \\ 0, if\ (a = b) \end{cases}$$

- Hàm tính độ tương tự Cosine
  - Đối với đầu vào là một vectơ các giá trị trọng số (TF/IDF) của các từ khóa

$$d(x,z) = \frac{x.z}{\|x\|\|z\|} = \frac{\sum_{i=1}^{n} x_i z_i}{\sqrt{\sum_{i=1}^{n} x_i^{\,2}}\sqrt{\sum_{i=1}^{n} z_i^{\,2}}}$$

Chuẩn hóa miền giá trị thuộc tính

- Hàm tính khoảng cách Euclid: $$d(x,z) = \sqrt{\sum_{i=1}^{n}(x_i - z_i)^2}$$
- Giả sử mỗi ví dụ được biểu diễn bởi 3 thuộc tính: `Age`, `Income` (cho mỗi tháng), và `Height` (đo theo mét)
  - `x` = (`Age`=20, `Income`=12000, `Height`=1.68)
  - `z` = (`Age`=40, `Income`=1300, `Height`=1.75)
- Khoảng cách giữa `x` và `z`
  - `d(x,z)` = $[(20-40)^2 + (12000-1300)^2 + (1.68-1.75)^2]^{1/2}$
  - Giá trị khoảng cách bị quyết định chủ yếu bởi giá trị khoảng cách (sự khác biệt) giữa 2 ví dụ đối với thuộc tính `Income`

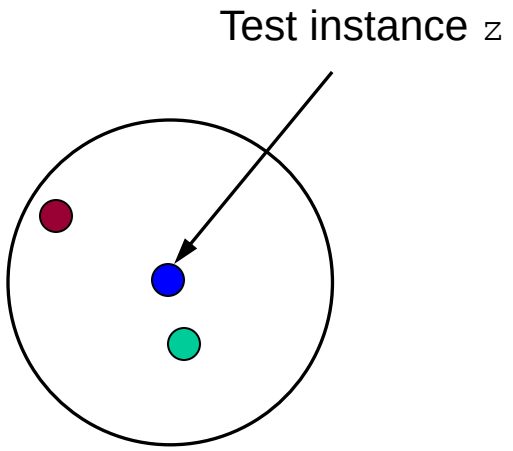
  → Vì: Thuộc tính `Income` có miền giá trị rất lớn so với các thuộc tính khác
- Cần phải chuẩn hóa miền giá trị (đưa về cùng một khoảng giá trị)
  - Khoảng giá trị [0,1] thường được sử dụng
  - Đối với mỗi thuộc tính `i`: $x_i = x_i / \max(f_i)$

Trọng số của các thuộc tính

- Hàm khoảng cách Euclid: $d(x,z) = \sqrt{\sum_{i=1}^{n}(x_i - z_i)^2}$
  - Tất cả các thuộc tính có cùng (như nhau) ảnh hưởng đối với giá trị khoảng cách
- **Các thuộc tính khác nhau** có thể (nên) có **mức độ ảnh hưởng khác nhau** đối với giá trị khoảng cách
- Cần phải tích hợp (đưa vào) các giá trị trọng số của các thuộc tính trong hàm tính khoảng cách
  - w_i là trọng số của thuộc tính i: $d(x,z) = \sqrt{\sum_{i=1}^{n} w_i(x_i - z_i)^2}$
- Làm sao để xác định các giá trị trọng số của các thuộc tính?
  - Dựa trên các tri thức cụ thể của bài toán (vd: được chỉ định bởi các chuyên gia trong lĩnh vực của bài toán đang xét)
  - Bằng một quá trình tối ưu hóa các giá trị trọng số (vd: sử dụng một tập học để học một bộ các giá trị trọng số tối ưu)

Khoảng cách của các láng giềng (1)

- Xét tập `NB(z)` – gồm k ví dụ học gần nhất với ví dụ cần phân lớp/dự đoán `z`
  - Mỗi ví dụ (láng giềng gần nhất) này có khoảng cách khác nhau đến `z`
  - Các láng giềng này có ảnh hưởng như nhau đối với việc phân lớp/dự đoáncho `z`? → KHÔNG!



- Cần gán các mức độ ảnh hưởng (đóng góp) của mỗi láng giềng gần nhất tùy theo khoảng cách của nó đến `z`
  - Mức độ ảnh hưởng cao hơn cho các láng giềng gần hơn!

Khoảng cách của các láng giềng (2)

- Gọi v là hàm xác định trọng số theo khoảng cách
  - Đối với một giá trị d(x,z) – khoảng cách giữa x và z
  - v(x,z) tỷ lệ nghịch với d(x,z)
- Đối với bài toán phân lớp:

$$c(z) = \arg\max_{c_j \in C} \sum_{x \in NB(z)} v(x,z).Identical(c_j, c(x))$$

$$Identical(a,b) = \begin{cases} 1, if\ (a = b) \\ 0, if\ (a \neq b) \end{cases}$$

- Đối với bài toán dự đoán (hồi quy):

$$f(z) = \frac{\sum_{x \in NB(z)} v(x,z).f(x)}{\sum_{x \in NB(z)} v(x,z)}$$

- Lựa chọn một hàm xác định trọng số theo khoảng cách:

$$v(x,z) = \frac{1}{\alpha + d(x,z)} \qquad v(x,z) = \frac{1}{\alpha + [d(x,z)]^2} \qquad v(x,z) = e^{-\frac{d(x,z)^2}{\sigma^2}}$$

Ảnh hưởng của số chiều

Giả thiết các mẫu được mô tả bởi 20 thuộc tính, nhưng chỉ có 2 thuộc tính liên quan đến hàm đích

Ảnh hưởng của số chiều: phương pháp kNN thường bị mất phương hướng khi X nhiều chiều

Một số giải pháp:

- Giãn chiều thứ j bởi trọng số z_j, trong đó $z_1,\ldots,z_n$ được chọn để tối thiểu hoá lỗi dự tính
- Sử dụng crossvalidation để tự động chọn các trọng số $z_1,\ldots,z_n$

Khi nào nên dùng láng giềng gần

- Các mẫu tương ứng với các điểm trong R^n
- Mỗi mẫu có dưới 20 thuộc tính
- Nhiều mẫu luyện

Ưu điểm:

- Luyện rất nhanh
- Học các hàm đích phức tạp
- Không mất thông tin

Nhược điểm:

- Chậm khi truy vấn
- Dễ bị ảnh hưởng bởi các thuộc tính không liên quan

Nội dung môn học

Chương 1. Tổng quan

Chương 2. Tác tử thông minh

Chương 3. Giải quyết vấn đề

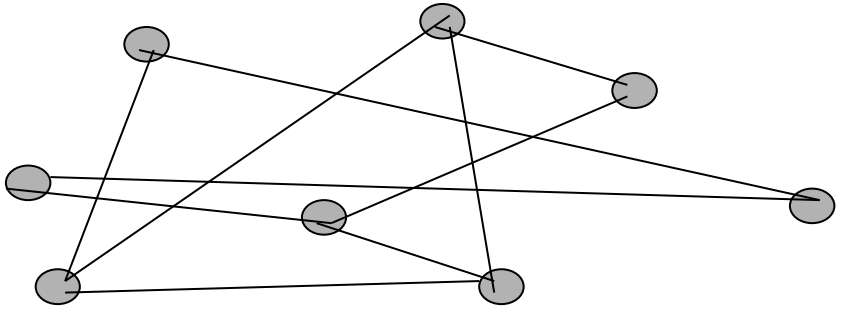
Chương 4. Tri thức và suy diễn

Chương 5. Học máy

- Tổng quan
- Học cây quyết định
- K láng giềng gần
- Mạng nơron

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



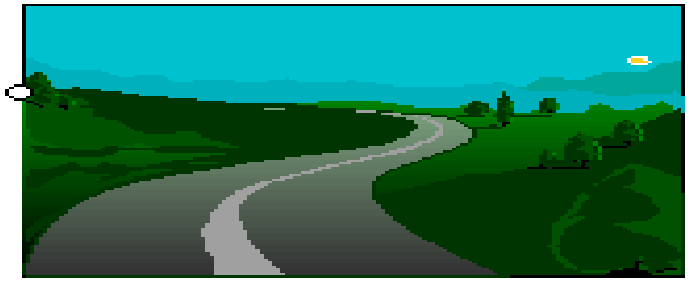
Các vấn đề:

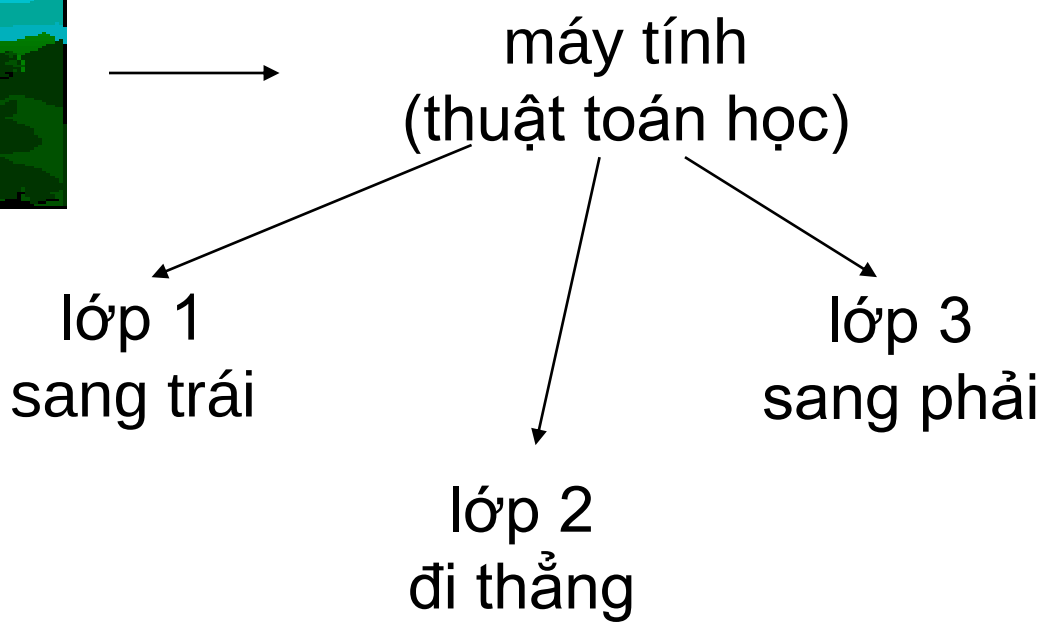
- Tốc độ bộ não nhận dạng hình ảnh
- Rất nhiều nơron trong một bộ não
- Tốc độ một nơron truyền dữ liệu

Ví dụ

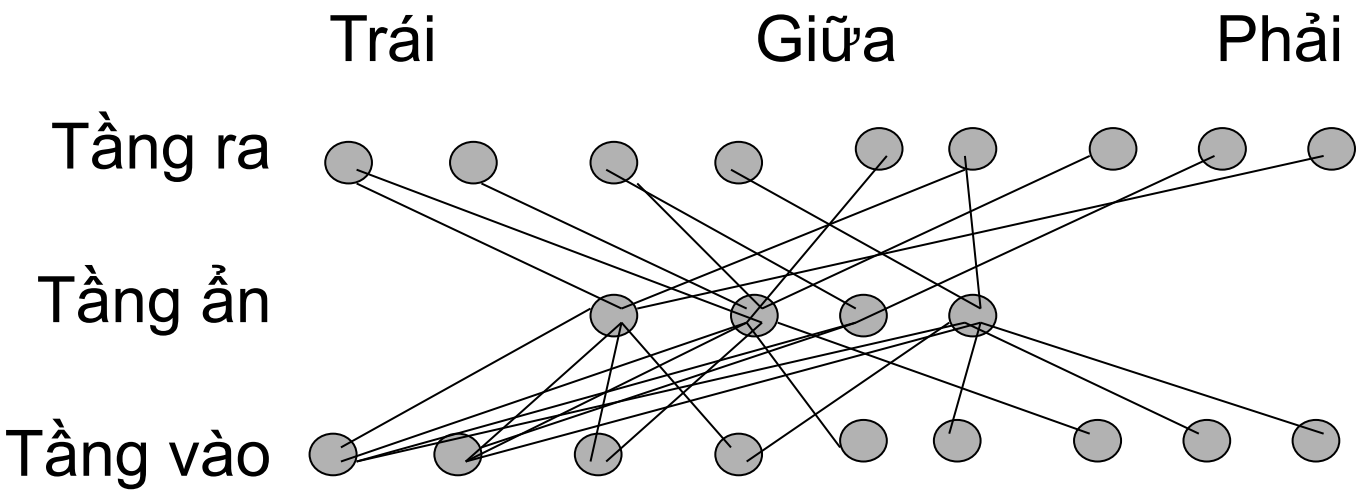
Lái xe

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





Biểu diễn mạng nơron



Định nghĩa

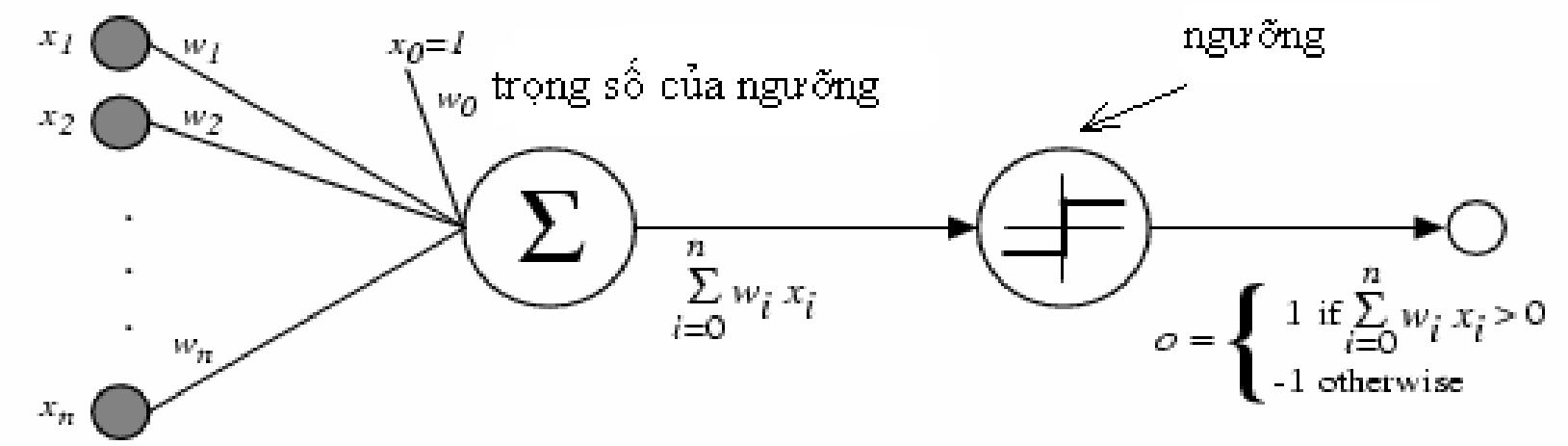
- Là một hệ thống gồm rất nhiều phần tử xử lý đơn giản hoạt động song song.
- Tính năng: phụ thuộc vào
  - cấu trúc hệ thống
  - mức độ liên kết giữa các phần tử
  - quá trình xử lý bên trong các phần tử
- Có thể học từ số liệu và tổng quát hoá từ các số liệu đó.

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

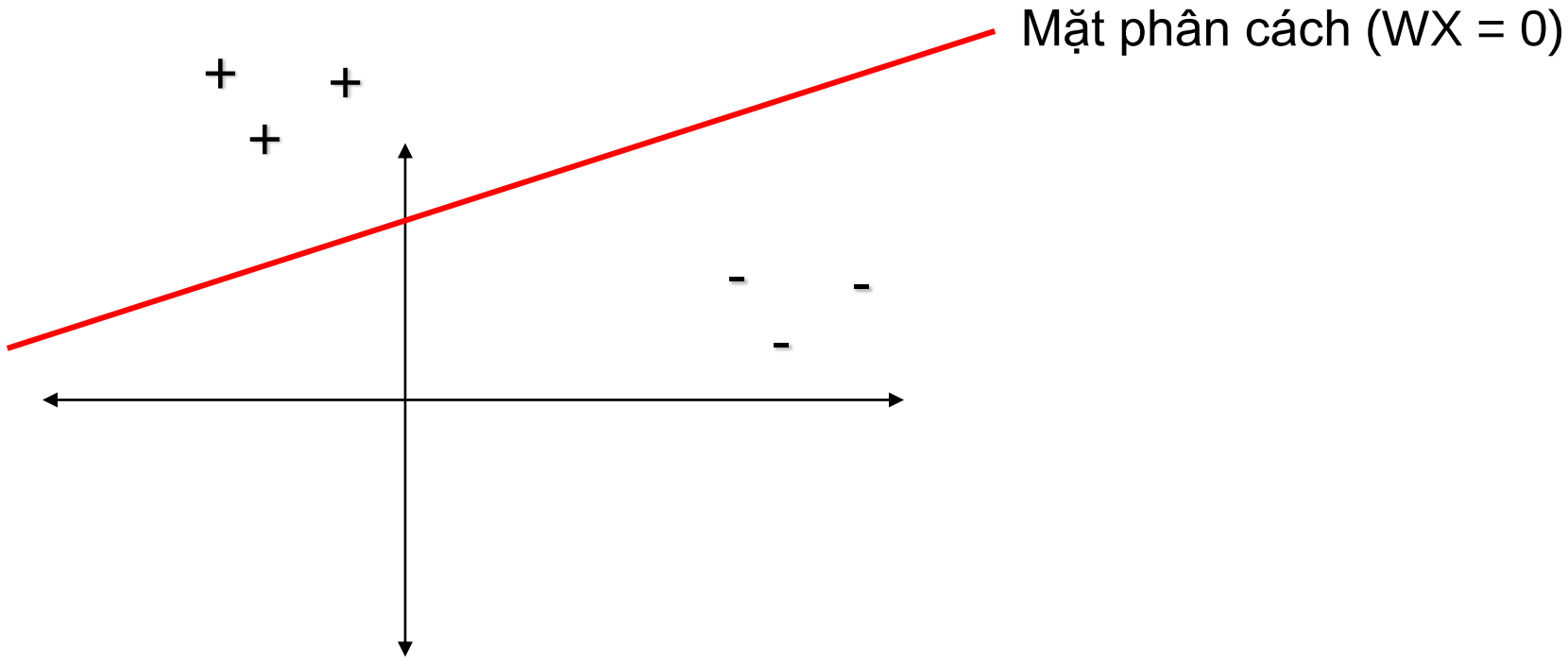
$\text{sgn}(y) = \begin{cases} 1 \text{ nếu } y > 0 \\ -1 \text{ nếu ngược lại} \end{cases}$

Nhiệm vụ học: tìm các giá trị của w

Không gian giả thiết: không gian các vectơ trọng số

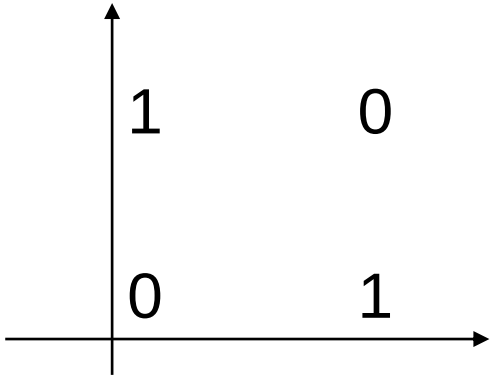
Khả năng của Perceptron

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



Khả năng của Perceptron

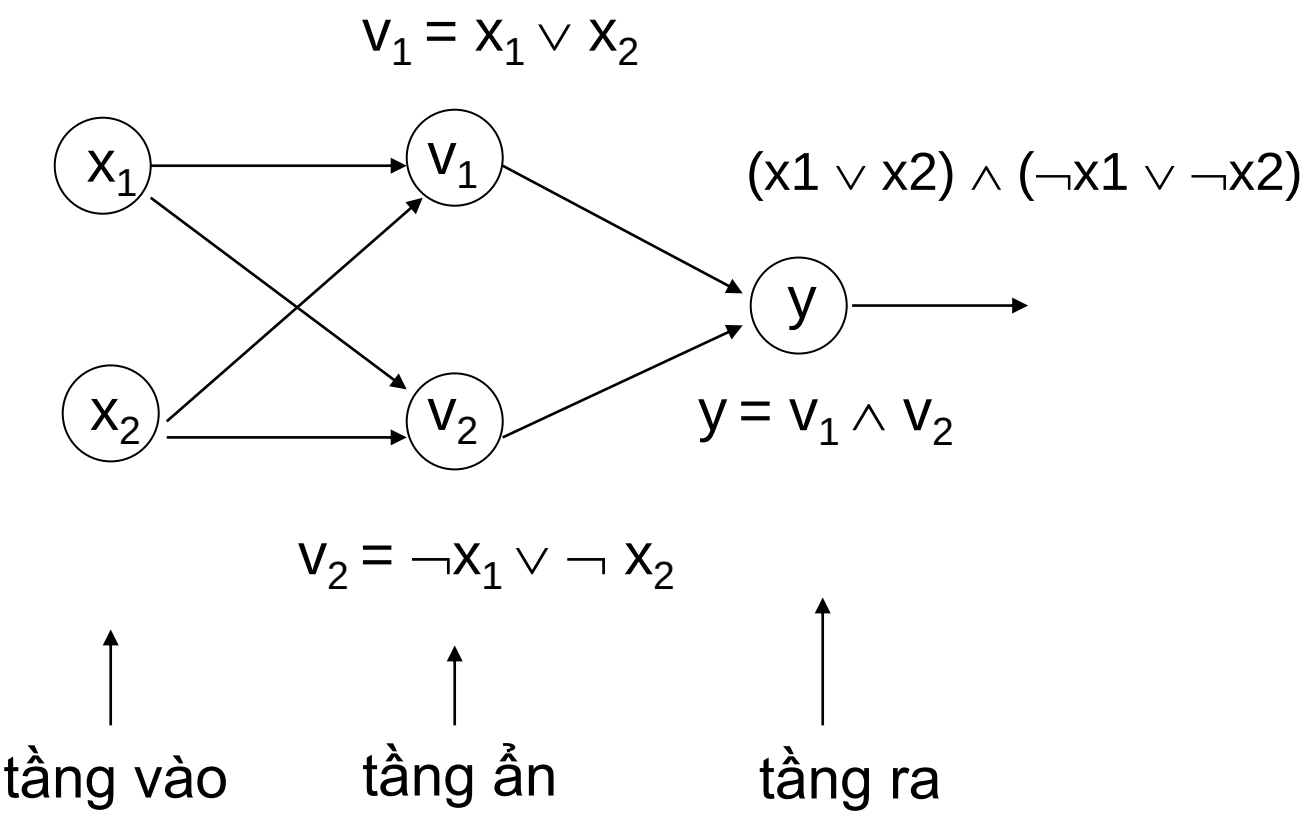
- có thể học các hàm $\neg$, $\wedge$, $\vee$, NAND, NOR
- không biểu diễn được các hàm không phân tách được bằng đường tuyến tính, vd XOR



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

➢ 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

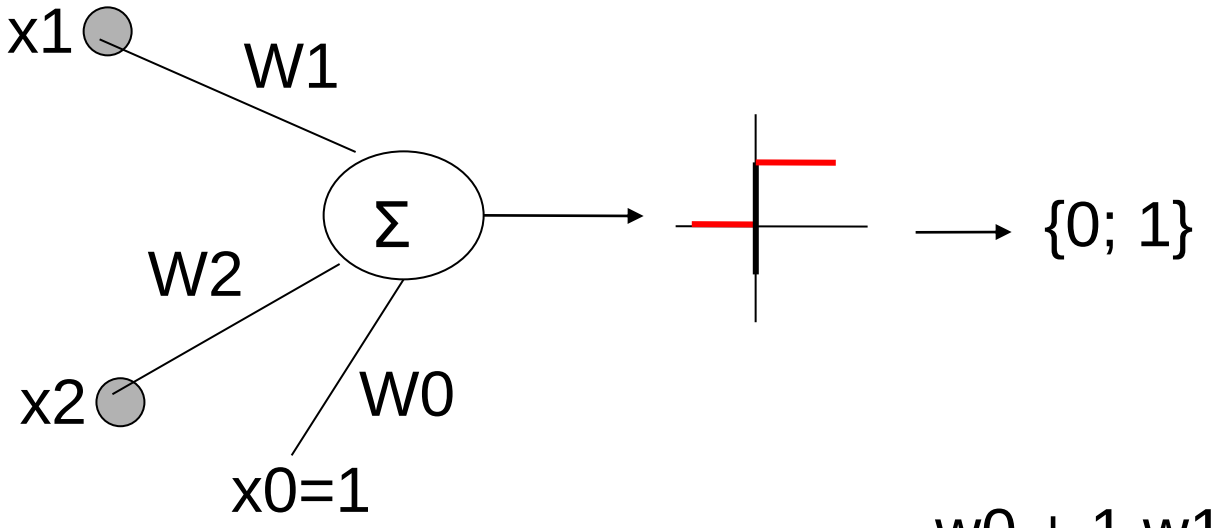
$$\Delta w_i \longleftarrow \Delta w_i + \eta(t - o)x_i$$

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

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

Ví dụ 1...

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

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	Δw0	Δw1	Δw2
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

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

Ví dụ 1...

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

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			

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

Ví dụ 1...

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

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			

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

Ví dụ 1...

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

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			

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

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

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

Ví dụ 1...

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

Δw0 = 0.1, Δw1 = 0.1, Δw2 = 0.1

w0 = -1.4, w1 = -0.4, w2 = 0.6, η = 0.1

$\sum$ = x0.w0+x1.w1+x2.w2 = 1.w0+0.w1+0.w2 = w0 = -1.4

x0	x1	x2	t	$\sum$	o	Δw0	Δw1	Δw2	w0	w1	w2
1	0	0	0	-1.4	0	0.1	0.1	0.1			
1	0	1	0	-0.6	0						
1	1	0	0	-1.4	0						
1	1	1	1								

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

Ví dụ 1...

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

Δw0 = 0.1, Δw1 = 0.1, Δw2 = 0.1

w0 = -1.4, w1 = -0.4, w2 = 0.6, η = 0.1

Δw0 = Δw0 + η*(t-o)*x0 = 0.1 + 0.1*(0-0)*1 = 0.1

x0	x1	x2	t	$\sum$	o	Δw0	Δw1	Δw2	w0	w1	w2
1	0	0	0	-1.4	0	0.1	0.1	0.1			
1	0	1	0	-0.6	0	0.1	0.1	0.1			
1	1	0	0								
1	1	1	1								

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

Ví dụ 1...

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

Δw0 = 0.1, Δw1 = 0.1, Δw2 = 0.1

w0 = -1.4, w1 = -0.4, w2 = 0.6, η = 0.1

Δw0 = Δw0 + η*(t-o)*x0 = 0.1 + 0.1*(0-0)*1 = 0.1

w0 = w0 + Δw0 = -1.4 + 0.1 = -1.3, w1 = -0.3, w2 = 0.7

x0	x1	x2	t	$\sum$	o	Δw0	Δw1	Δw2	w0	w1	w2
1	0	0	0	-1.4	0	0.1	0.1	0.1	-1.3	-0.3	0.7
1	0	1	0	-0.6	0	0.1	0.1	0.1	-1.2	-0.2	0.8
1	1	0	0								
1	1	1	1								

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

Ví dụ...

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

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

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

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

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

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

x0	x1	x2	t	$\sum$	o	Δw0	Δw1	Δw2
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

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

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

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

Ví dụ...

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

Δw0 = 0.1, Δw1 = 0.1, Δw2 = 0.1

w0 = -1.4, w1 = -0.4, w2 = 0.6, η = 0.1

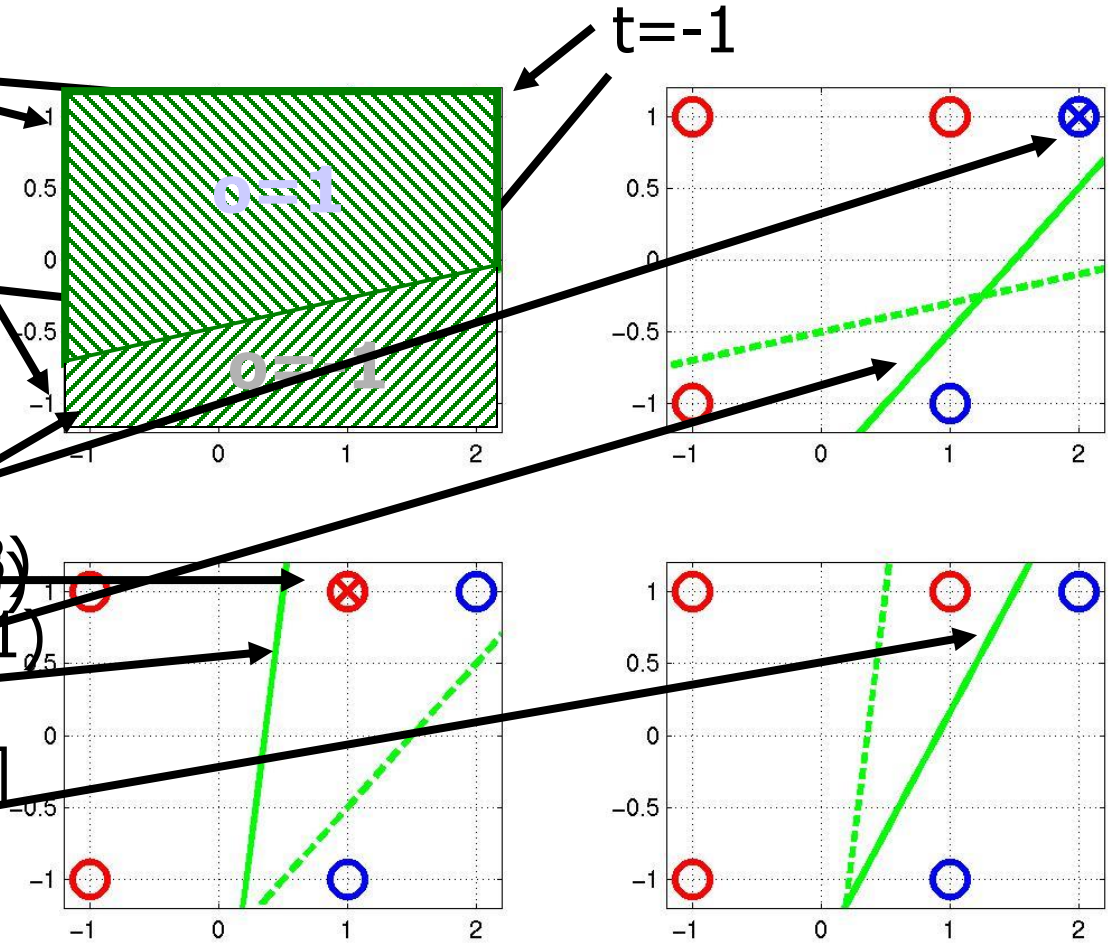
$\sum$ = x0.w0+x1.w1+x2.w2 = 1.w0+0.w1+0.w2 = w0 = -1.4

Δw0 = Δw0 + η*(t-o)*x0 = 0.1 + 0.1*(0-0)*1 = 0.1

w0 = w0 + Δw0 = -1.4 + 0.1 = -1.3, w1 = -0.3, w2 = 0.7

x0	x1	x2	t	$\sum$	o	Δw0	Δw1	Δw2
1	0	0	0	-1.4	0	0.1	0.1	0.1
1	0	1	0	-0.6	0	0.1	0.1	0.1
1	1	0	0					
1	1	1	1					

Ví dụ 2


t=1
t=-1
o=1
o=-1
w=[0.25 –0.1 0.5]
$x_2 = 0.2\ x_1 - 0.5$
(x,t)=([2,1],-1)
o=sgn(0.45*2-0.6*1+0.3)
=sgn(0.9-0.6+0.3)
=1
Δw=[-0.2 -0.4 -0.2]
(x,t)=([-1,-1],1)
o=sgn(0.25+0.1-0.5)
=sgn(0.25-0.7+0.1)
=-1
Δw=[0.2 0.2 0.2]

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***

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

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:

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

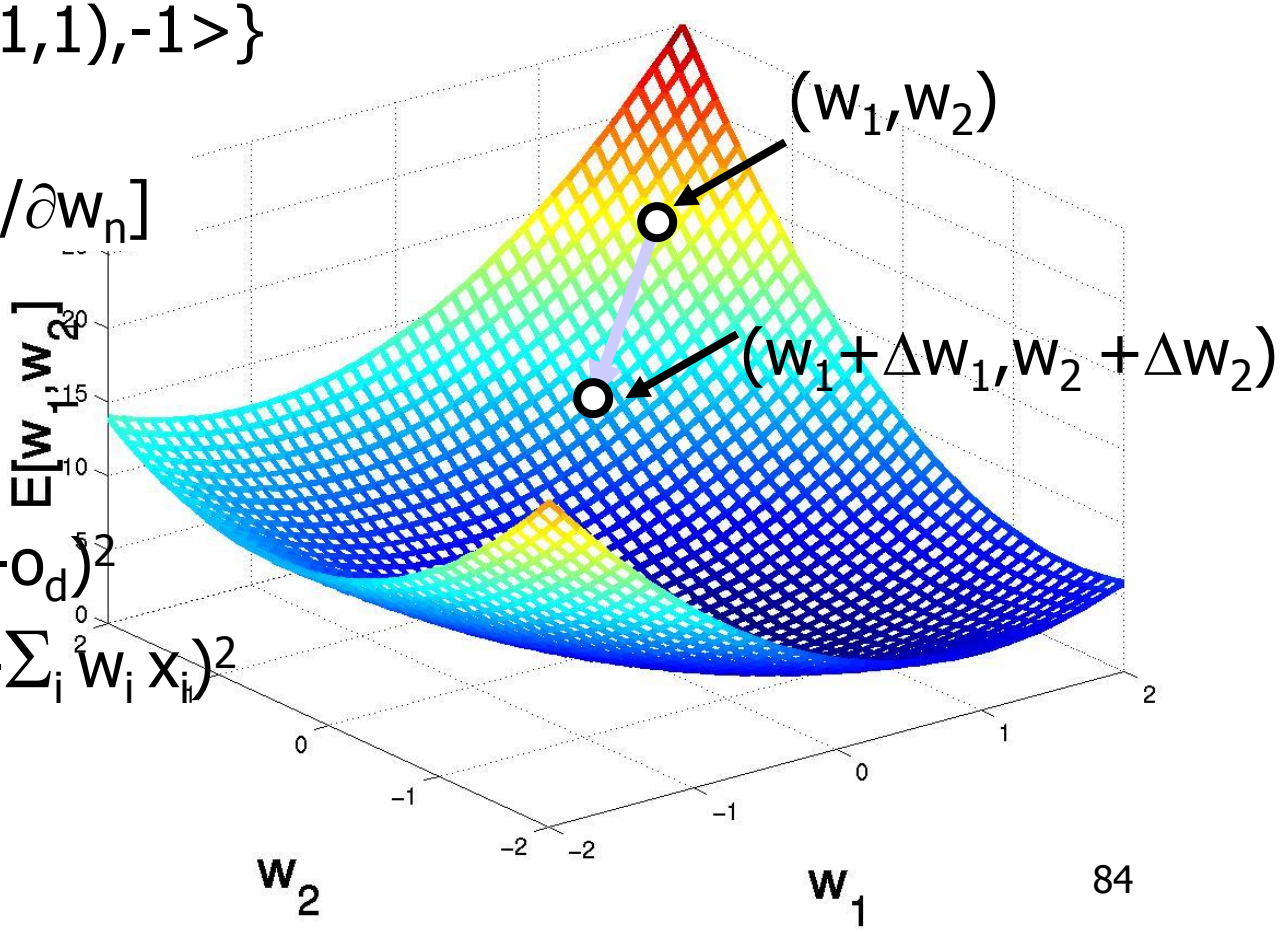
$\Delta w=-\eta \nabla E[w]$

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

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

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

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



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

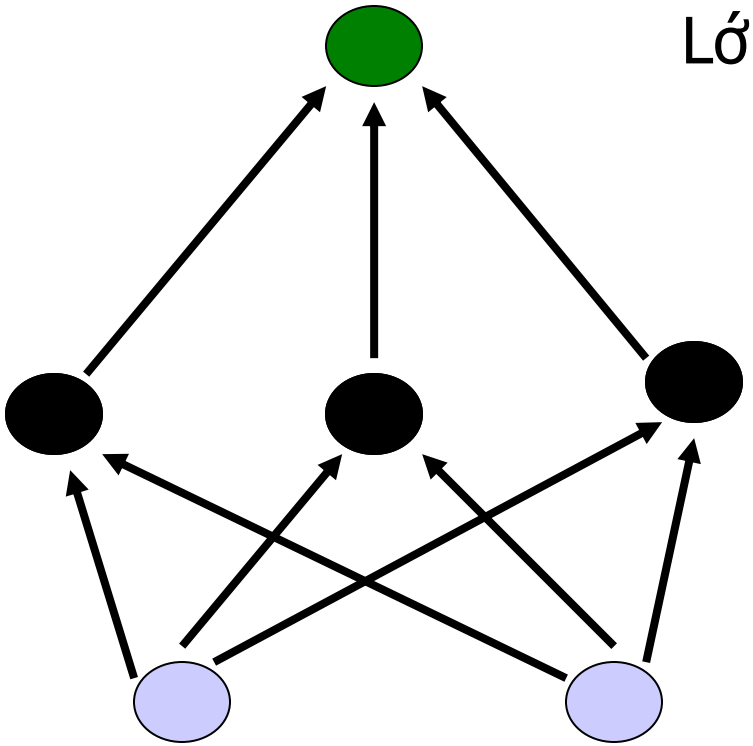
Perceptron đảm bảo thành công nếu

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

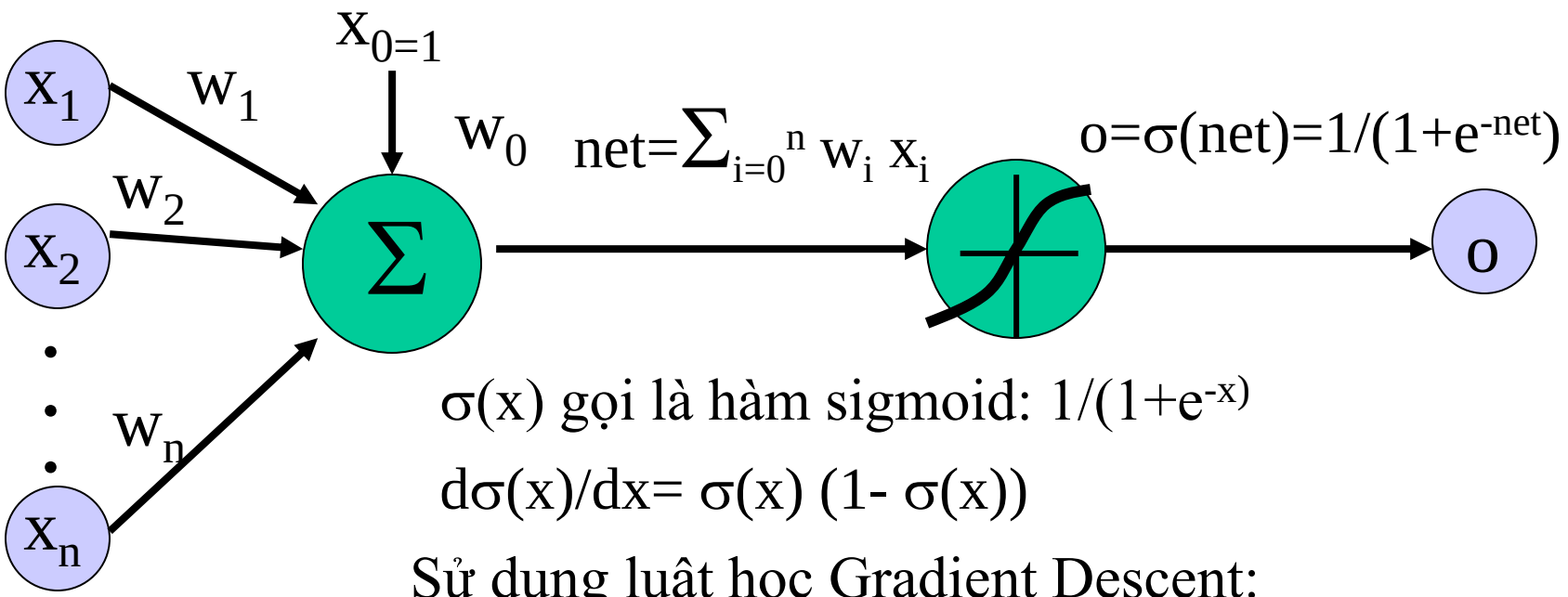
Huấn luyện với Gradient Descent

- 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: $1/(1+e^{-x})$

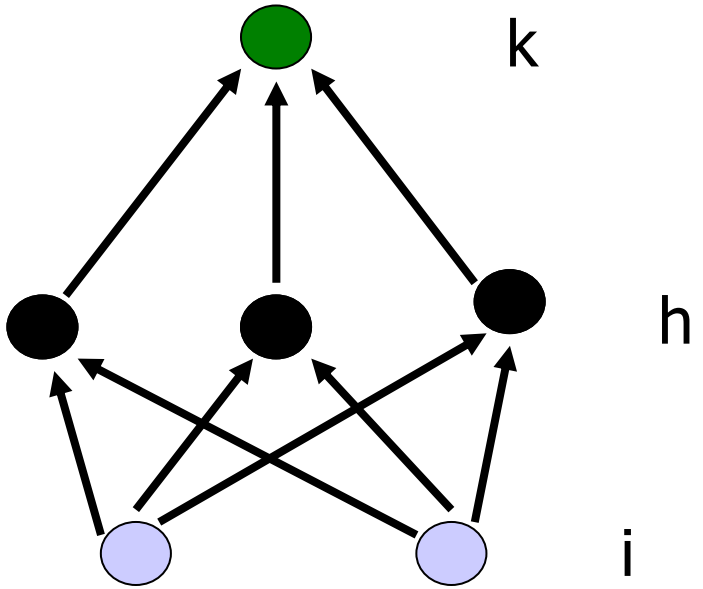
$d\sigma(x)/dx= \sigma(x)\,(1- \sigma(x))$

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
        $\Delta w_{i,j}=\eta\,\delta_j\,x_i$



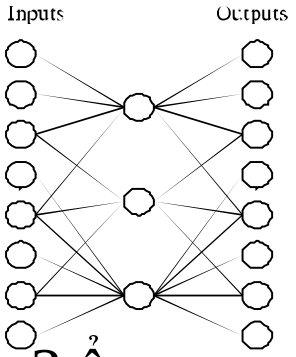
Giải thuật lan truyền ngược sai số

- 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:

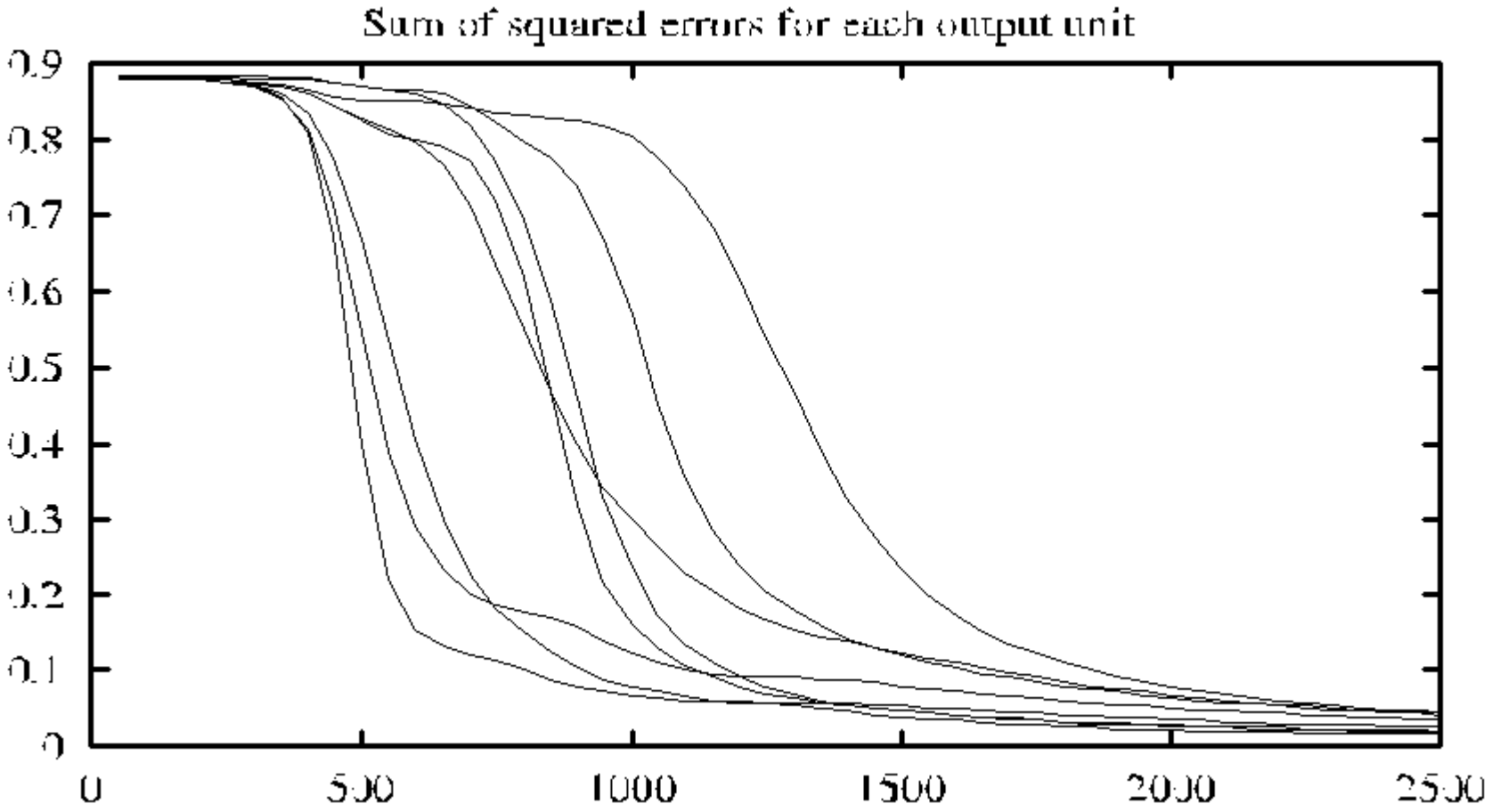
Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

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



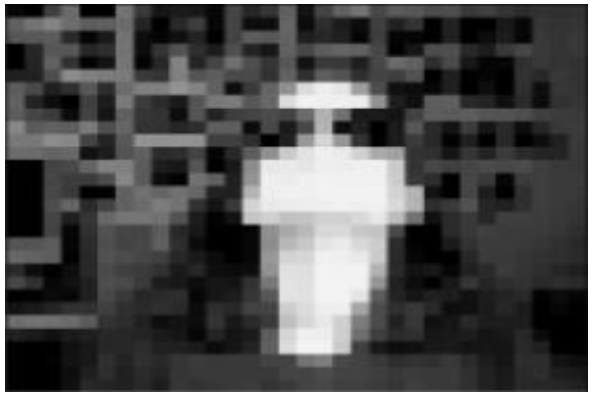
Sự hội tụ của giải thuật lan truyền ngược sai số

- 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

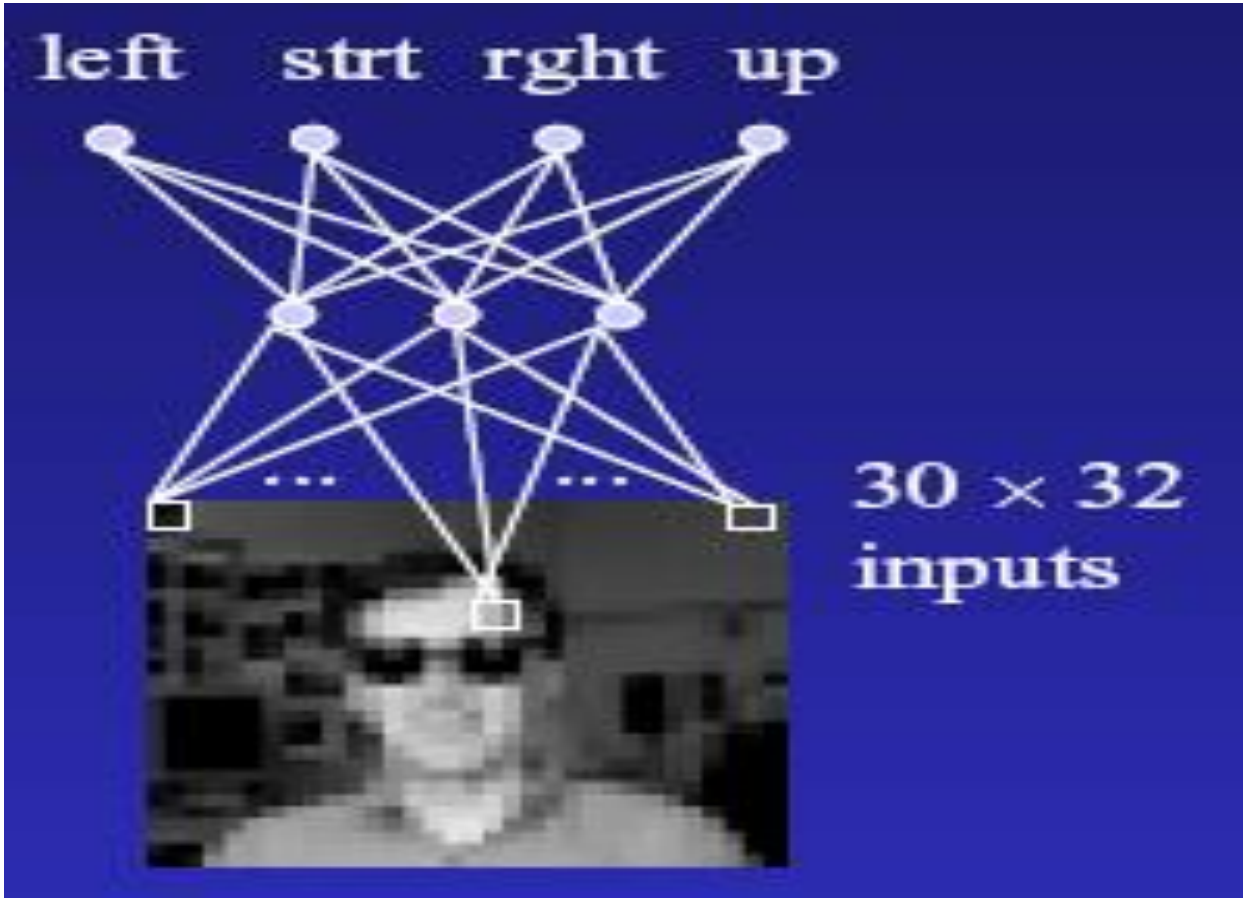
Mã hoá đầu vào

- 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