



Chương 3

TỐI ƯU HÓA VÀ KHAI PHÁ TRI THỨC

Knowledge Optimization and Mining

- **TS. Lê Đức Như**
- Khoa Công nghệ thông tin - Trường Đại học Hải Phòng
- Phone: (+84)987394900. Email: Nhuongld@dhhp.edu.vn
- Website: www.dhhp.edu.vn/~nhuongld



Nội dung Chương 2

1. Tại sao phải tối ưu hóa Cơ sở tri thức?

2. Loại bỏ luật thừa

3. Loại bỏ luật mâu thuẫn

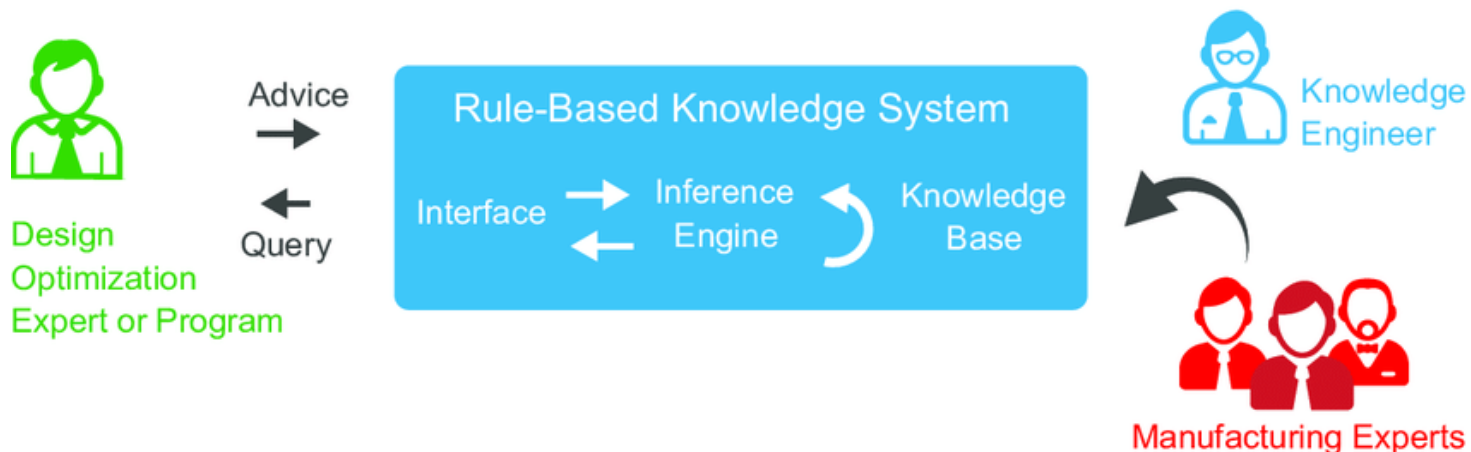
4. Loại bỏ vòng lặp trong suy diễn

5. Khai phá tri thức

1. Tại sao phải tối ưu hóa tri thức?

▪ Tối ưu tri thức

- Tri thức trong một cơ sở tri thức có khả năng **thừa, trùng lặp hoặc mâu thuẫn**. Hệ thống có thể *đổ lỗi* cho người dùng về việc đưa vào những tri thức như vậy.
- Tối ưu một cơ sở tri thức về là **một thao tác khó** (vì giữa các tri thức thường có **quan hệ không tường minh**), nhưng trong giới hạn cơ sở tri thức dưới dạng luật, ta vẫn có một số thuật toán đơn giản để loại bỏ các vấn đề này.





2. Loại bỏ luật thừa

- 2.1 Rút gọn bên phải
- 2.2 Rút gọn bên trái
- 2.3 Phân rã và kết hợp luật
- 2.4 Luật thừa
- 2.5 Thuật toán loại bỏ luật thừa

2.1 Rút gọn bên phải

- Luật sau hiển nhiên đúng :

$$A \wedge B \rightarrow A \quad (1)$$

- Do đó luật

$$A \wedge B \rightarrow A \wedge C \text{ tương đương với } A \wedge B \rightarrow C$$

- *Quy tắc rút gọn*: Có thể loại bỏ những sự kiện bên vế phải nếu những sự kiện đó đã xuất hiện bên vế trái. Nếu sau khi rút gọn mà vế phải trở thành rỗng thì luật đó là luật hiển nhiên. Ta có thể loại bỏ các luật hiển nhiên ra khỏi tri thức.

2.2 Rút gọn bên trái

- Xét các luật sau :

$$(L1) A \wedge B \rightarrow C$$

$$(L2) A \rightarrow X$$

$$(L3) X \rightarrow C$$

- Rõ ràng là luật $A \wedge B \rightarrow C$ có thể được thay thế bằng luật $A \rightarrow C$ mà không làm ảnh hưởng đến các kết luận trong mọi trường hợp.
- Ta nói rằng **sự kiện B trong luật (1) là dư thừa và có thể được loại bỏ khỏi luật** dẫn trên.



2.3 Phân rã và kết hợp luật

- Luật

$$A \vee B \rightarrow C$$

- Tương đương với hai luật

$$A \rightarrow C; B \rightarrow C$$

- Với quy tắc này, ta có thể loại bỏ hoàn toàn các luật có phép nối **HOẶC**. Các luật có phép nối này thường làm cho thao tác xử lý trở nên phức tạp.



2.4 Luật thừa

- Một luật dẫn $A \rightarrow B$ được gọi là thừa nếu có thể suy ra luật này từ những luật còn lại.
- Ví dụ:
 - Trong tập các luật gồm $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ thì luật thứ 3 là luật thừa vì nó có thể được suy ra từ 2 luật còn lại.
 - Loại đi các luật có phép nối **HOẶC**, các luật hiển nhiên hoặc các luật thừa.

2.5 Thuật toán loại bỏ luật thừa

▪ Bước 1: Rút gọn về phải

Với mỗi luật r trong R

Với mỗi sự kiện $A \in \text{VếPhải}(r)$

Nếu $A \in \text{VếTrái}(r)$ **thì** Loại A ra khỏi vế phải của R .

Nếu $\text{VếPhải}(r)$ là rỗng **thì** loại bỏ r ra khỏi hệ luật dẫn:

$$R = R - \{r\}$$

▪ Bước 2: Phân rã các luật

Với mỗi luật $r : X_1 \vee X_2 \vee \dots \vee X_n \rightarrow Y$ trong R

Với mỗi i từ 1 đến n $R := R \cup \{X_i \rightarrow Y\}$

$$R = R \setminus \{r\}$$

2.5 Thuật toán loại bỏ luật thừa

▪ Bước 3: Loại bỏ luật thừa

Với mỗi luật r thuộc R

Nếu $VếPhải(r) \in BaoĐóng(VếTrái(r), R \setminus \{r\})$ **thì** $R := R \setminus \{r\}$

▪ Bước 4 : Rút gọn về trái

Với mỗi luật dẫn $r : X : A_1 \wedge A_2, \dots, A_n \rightarrow Y$ thuộc R

Với mỗi sự kiện A_i thuộc r

Gọi luật $r_1 : X \setminus \{A_i\} \rightarrow Y$ và $S = (R \setminus \{r\}) \cup \{r_1\}$

Nếu $BaoĐóng(X \setminus \{A_i\}, S) \equiv BaoĐóng(X, R)$ thì

$X := X \setminus \{A_i\}$



3. Loại bỏ luật mâu thuẫn

- 3.1 Luật mâu thuẫn
- 3.2 Thuật toán loại bỏ luật mâu thuẫn
- 3.3 Cơ chế xử lý mâu thuẫn luật

3.1 Luật mâu thuẫn

- Cho tập R như sau:

$$r1: A, M \rightarrow N$$

$$r2: B, N \rightarrow C$$

$$r3: A, M \rightarrow B$$

$$r4: A \rightarrow P$$

$$r5: D \rightarrow M$$

$$r6: B, N \rightarrow M$$

$$r7: P, C \rightarrow A$$

$$r8: D, O \rightarrow A$$

- Xét luật **$r: D, O \rightarrow \neg N$** là một luật mâu thuẫn vì

$$D, O \rightarrow A, M, P \text{ (do } r8, r5, r4)$$

$$D, O \rightarrow A, M, P, N \text{ (do } r1) \Rightarrow \mathbf{D, O \rightarrow N}$$

3.1 Luật mâu thuẫn

Định nghĩa luật mâu thuẫn

- Gọi R : tập luật của cơ sở tri thức

$$r \in R: X \rightarrow Y$$

$(X)_{R - \{r\}}$: Tập các mệnh đề suy được từ mệnh đề X
bằng các luật thuộc $R - \{r\}$

$\neg Y$: phủ định của Y

- Luật r được gọi là mâu thuẫn nếu: $\neg Y \in (X)_{R - \{r\}}$

A. Kiểm tra luật mâu thuẫn

- Đặt $R' = R - \{r\}$
- Xác định $(X)_{R'} = \{A_j / A_j \text{ các mệnh đề có thể suy diễn từ } X \text{ dựa trên tập luật } R'\}$ (bao đóng của X trên R')
- Kiểm tra nếu $\exists Y \in (X)_{R'}$ không?
 - Nếu đúng: thì luật r mâu thuẫn đối với tập luật R'
 - Ngược lại r không mâu thuẫn

B. Loại bỏ luật mâu thuẫn trong cơ sở tri thức

- Bước 1: Xét luật r trong R của cơ sở tri thức

Kiểm tra r có mâu thuẫn với tập $R - \{r\}$ không?

- Bước 2: Nếu mâu thuẫn thì $R = R - \{r\}$
- Bước 3: Quay lại B1 với luật khác

3.3 Cơ chế xử lý mâu thuẫn luật

▪ Nguyên tắc 1: Dựa theo trọng số luật

Nếu Luật r và r' mâu thuẫn nhau
Và Luật r' có trọng số xuất hiện nhiều hơn r
Thì Loại bỏ luật r và giữ lại r'

▪ Nguyên tắc 2: Dựa theo tần số xuất hiện

Nếu Luật r và r' mâu thuẫn nhau
Và Luật r' có tần xuất xuất hiện nhiều hơn r
Thì Loại bỏ luật r và giữ lại luật r'

3.3 Cơ chế xử lý mâu thuẫn luật

▪ Nguyên tắc 3: Dựa theo lĩnh vực đang xét

Nếu	Luật r và r' mâu thuẫn nhau
Và	Luật r' thuộc lĩnh vực A
Và	Luật r thuộc lĩnh vực B
Và	Đang xét lĩnh vực A
Thì	Loại bỏ luật r và giữ lại luật r'

3.3 Cơ chế xử lý mâu thuẫn luật

▪ Nguyên tắc 4: Trường hợp chung riêng

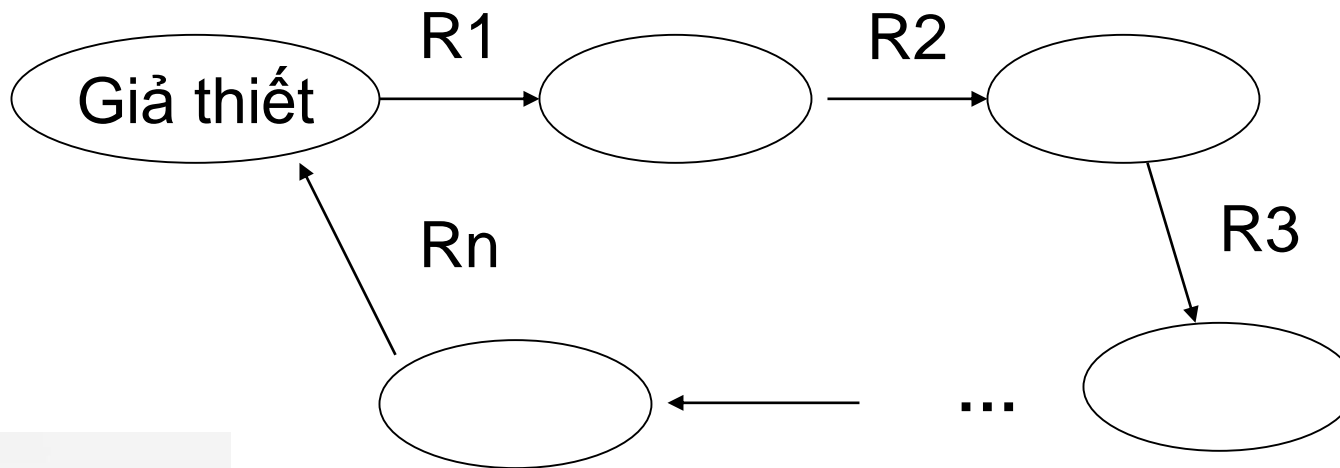
- Nếu** Luật r và r' mâu thuẫn nhau
- Và** Luật r biểu thị trường hợp chung
- Và** Luật r' biểu thị trường hợp riêng
- Thì** Không áp dụng luật r và áp dụng luật r'

▪ Nguyên tắc 5: Dựa vào chuyên gia

- Nếu** Luật r và r' mâu thuẫn nhau
- Và** Chuyên gia chấp nhận luật r'
- Thì** Loại bỏ luật r và giữ lại r'

4. Loại bỏ vòng lặp trong suy diễn

▪ Vòng lặp trong suy diễn



4. Loại bỏ vòng lặp trong suy diễn

Thuật toán phát hiện vòng lặp

- Gọi R : tập luật sẵn có của cơ sở tri thức
 $r: X \rightarrow Y$ là luật mới cần thêm vào
- **Bước 1.** Xác định $(Y)_R = \{A_j / A_j \text{ các mệnh đề có thể suy diễn từ } Y \text{ dựa trên tập luật } R\}$ (bao đóng của Y trên R)
- **Bước 2.** Nếu $X \in (Y)_R$ thì luật r gây ra vòng lặp

4. Loại bỏ vòng lặp trong suy diễn

- Ví dụ: Cho tập luật R như sau:

r1: $A, M \rightarrow N$

r3: $A, M \rightarrow B$

r5: $D \rightarrow M$

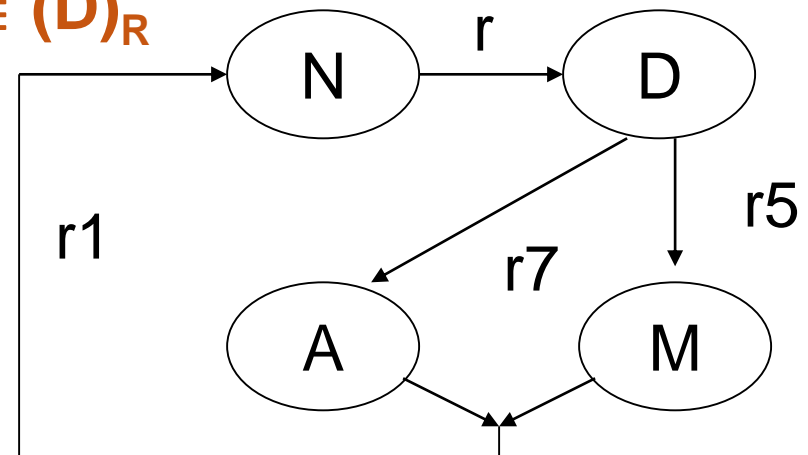
r2: $B, N \rightarrow C$

r4: $A \rightarrow P$

r6: $B, N \rightarrow M$

r7: $D \rightarrow A$

- Giả sử ta muốn thêm luật mới $r: N \rightarrow D$, luật này sẽ làm xuất hiện vòng lặp trong CSTT vì $N \in (D)_R$



5. Khai phá tri thức

Data Mining Process



Data mining techniques

Classification

Clustering

Regression

Outer

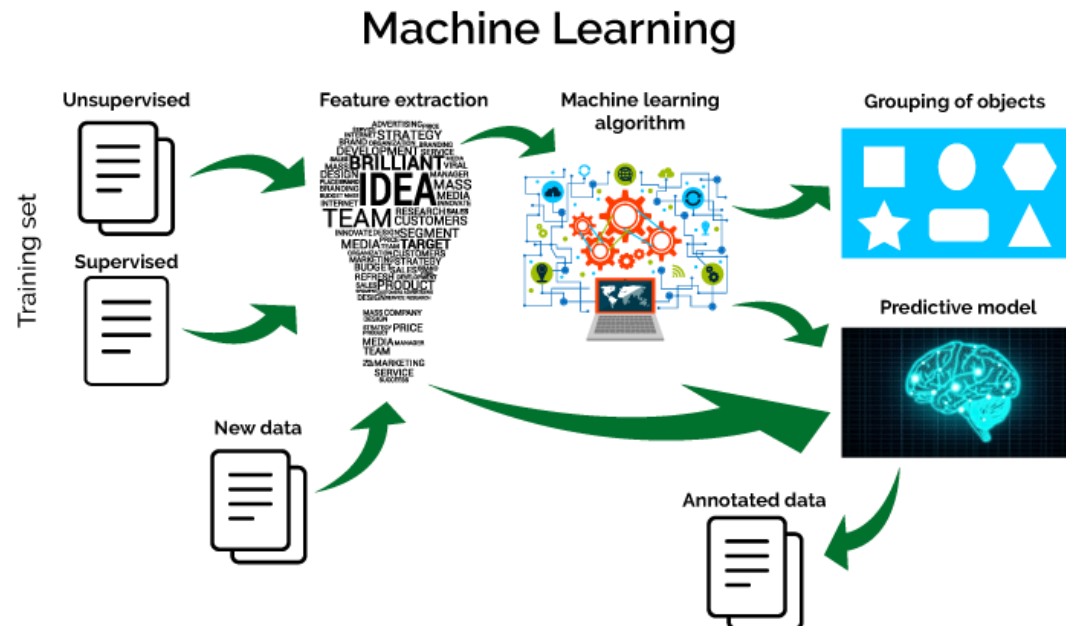
Sequential
Patterns

Prediction

Association
Rules

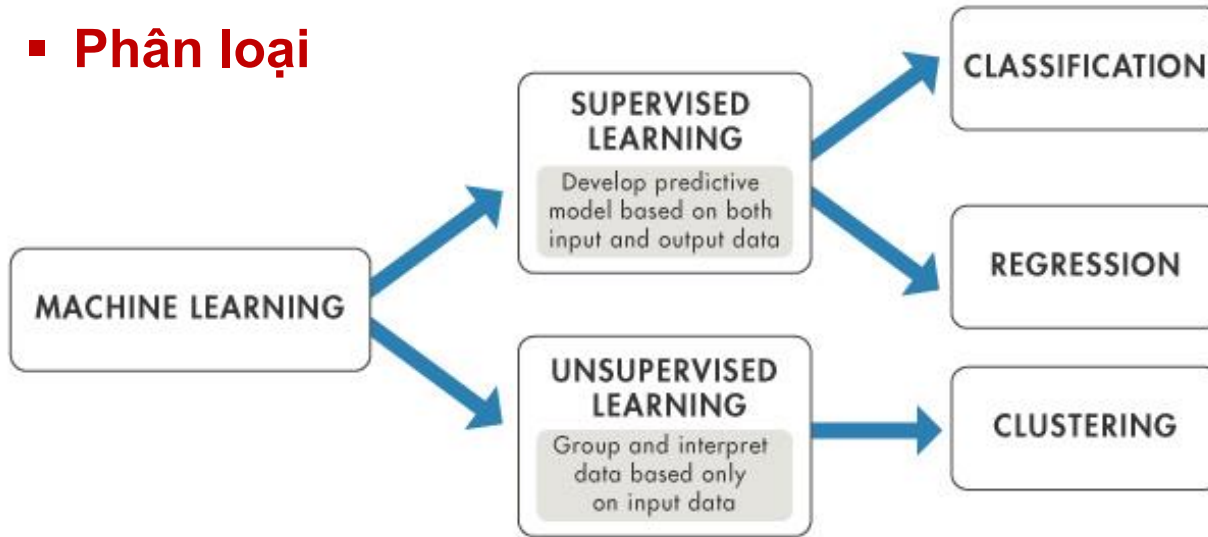
5.1 Máy học

- **Máy học** (**Learning Machine**): Máy tính hay chương trình máy tính có khả năng tự hoàn thiện từ “kinh nghiệm”.
- Máy học còn có nghĩa là việc mô hình hóa môi trường xung quanh hay khả năng một chương trình máy tính sinh ra một cấu trúc dữ liệu mới khác với cấu trúc hiện có. Chẳng hạn **tìm ra những luật If...then... từ tập dữ liệu đầu vào**.



5.1 Máy học

▪ Phân loại



Cách tiếp cận:

- Tiếp cận thống kê
- Tiếp cận toán tử logic
- Tiếp cận hình học (phân hoạch không gian, xây dựng cây định danh, ...)
- Tiếp cận mạng Neural
- Tiếp cận khai mở dữ liệu

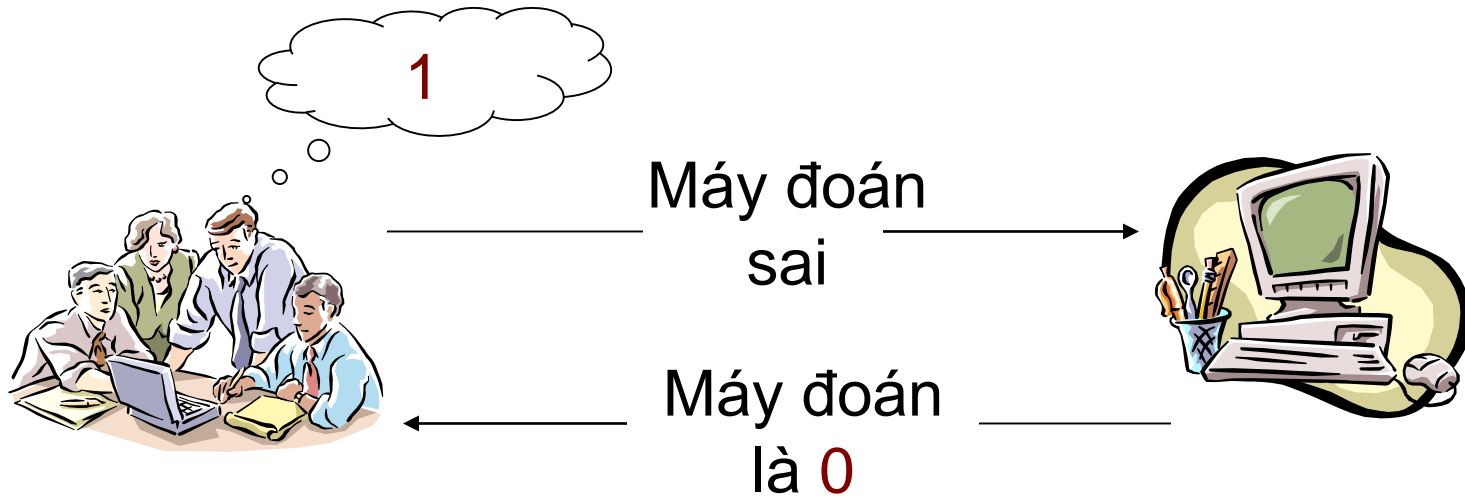
Cấp độ học:

- Học vẹt (Rote learning)
- Học theo giải thích (by explanation)
- Học theo ví dụ, trường hợp (by examples, cases)
- Học khám phá (by discovering)

5.1 Máy học

A. Tiếp cận thống kê:

Ví dụ: Chương trình đoán ý nghĩ con người. Máy sẽ đoán người chơi nghĩ số 0 hay 1 trong đầu, người chơi sẽ phải trả lời cho máy biết là máy đã đoán đúng hay sai. Để từ đó máy tính sẽ học qui luật suy nghĩ của người chơi.



5.1 Máy học

A. Tiếp cận thống kê: Ý tưởng cài đặt: hết sức đơn giản

- Lưu trữ toàn bộ dãy số 0, 1 mà người chơi đã nghĩ ra.
- Lấy 7 con số trước đó (người chơi đưa ra), tính xác suất xuất hiện của số 1 và số 0 sau dãy 7 con số này. Máy sẽ đoán số có xác suất xuất hiện cao hơn.
- Giả sử ở lần đoán thứ i, dãy số mà người dùng đã đoán như sau:

▪ ... 1 1 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 ?

- Từ dữ liệu lưu trữ ở những lần đoán trước, giả sử số lần xuất hiện của 1 sau dãy 0 0 0 0 1 0 0 là 28 và số lần xuất hiện của số 0 là 90
- Xác suất xuất hiện của số 1 sau dãy này là: $28/(28+90) = 23.7\%$
- Xác suất xuất hiện của số 0 sau dãy này là: $90/(28+90) = 76.3\%$

⇒ Máy sẽ đoán số 0



5.1 Máy học

A. Tiếp cận thống kê: Ý tưởng cài đặt: hết sức đơn giản

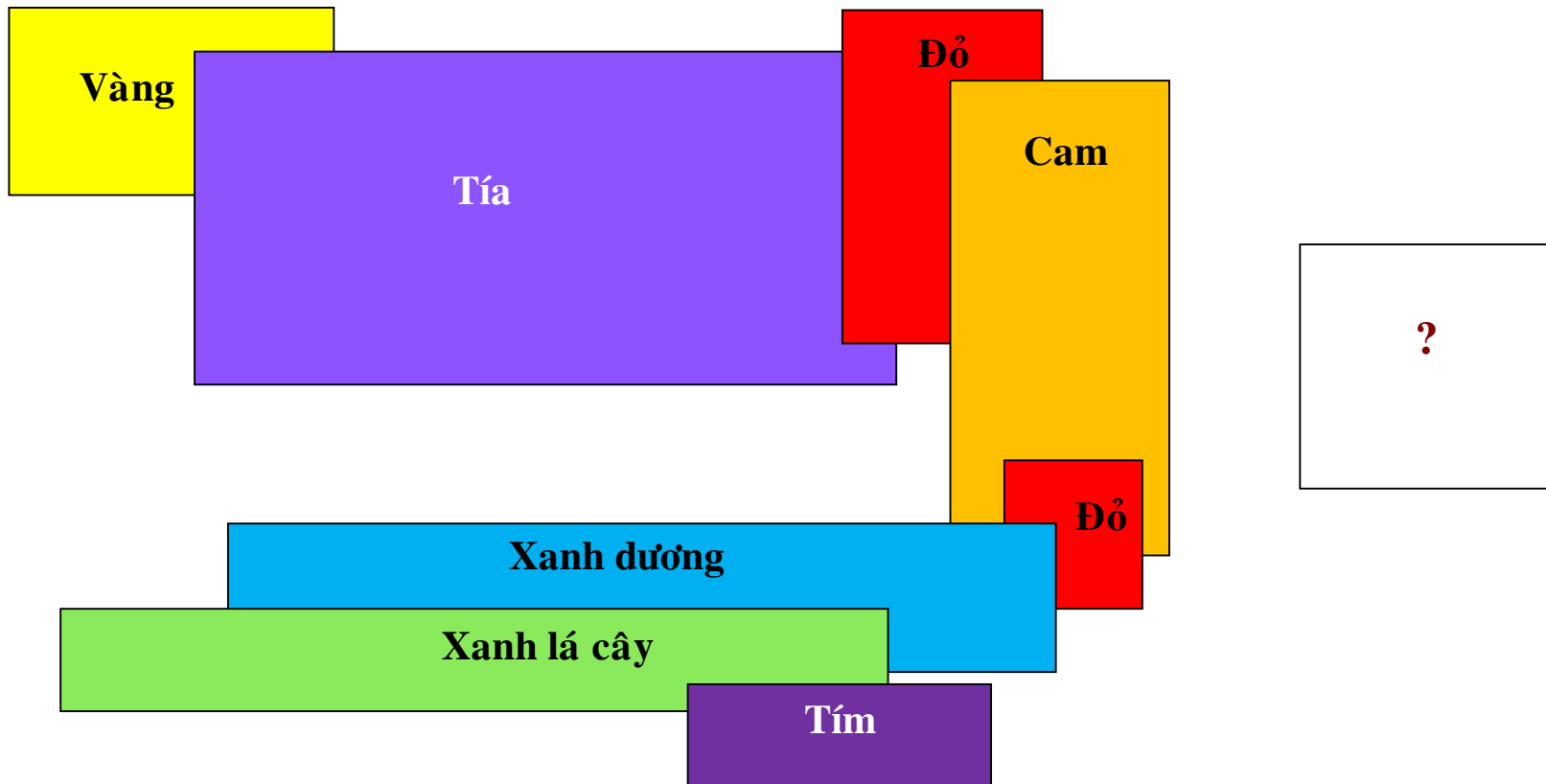
■ Nhận xét:

- Ví dụ đã đưa ra là thuộc cấp độ học vẹt sử dụng cách tiếp cận thống kê.
- Máy không thể đoán đúng ngay được, nhưng càng về sau (vài trăm lần đoán) máy càng trở nên chính xác một cách kinh ngạc (trung bình có thể lên đến 90%).
- Trên thực tế khi cài đặt chương trình này tác giả không chỉ đoán qui luật từ dãy số của người chơi, máy còn sử dụng cả dãy số mà máy đã đoán.

5.1 Máy học

B. Tiếp cận hình học

- Cho tập các hình chữ nhật với kích thước (ngang & rộng) và màu sắc khác nhau (hình vẽ). Cho biết hình bên phải có màu gì?

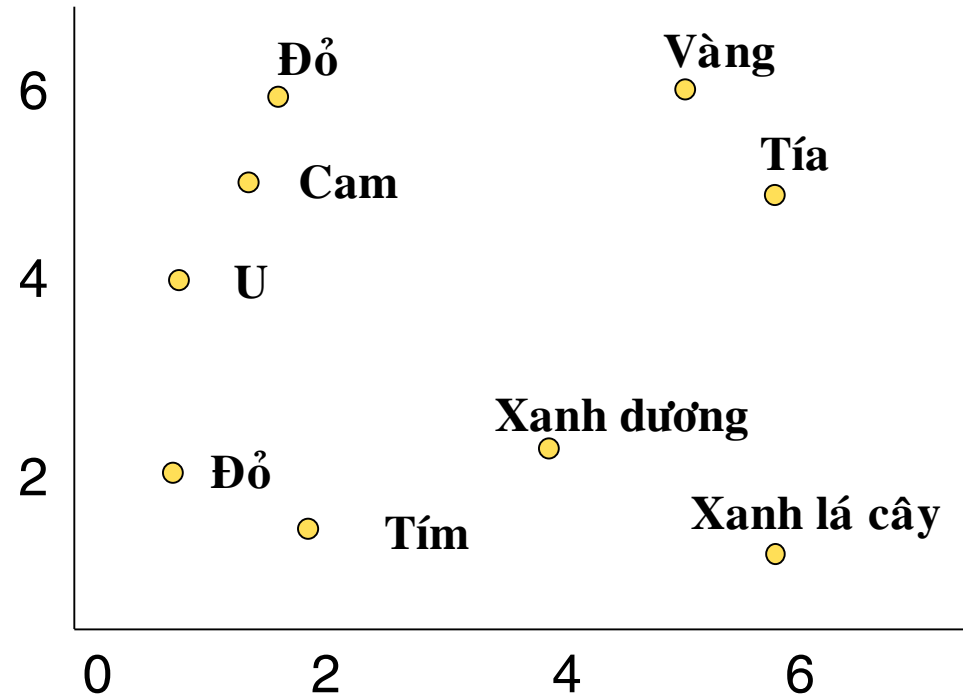


5.1 Máy học

B. Tiếp cận hình học

■ Cách 1:

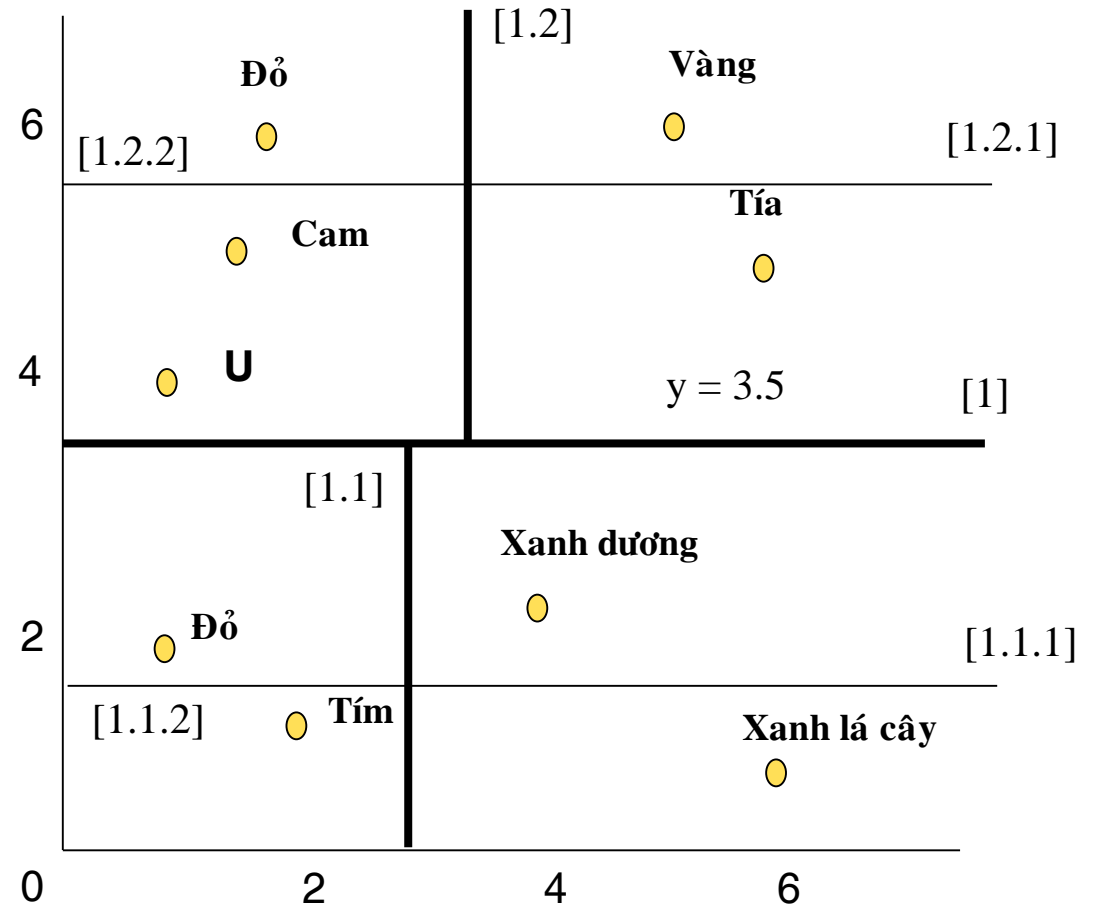
- Phản ứng tự nhiên của con người: **Tìm khối có sẵn gần giống để đoán màu cho khối chưa biết**. Như thế nào là gần giống?
- Biểu diễn 2 thuộc tính chiều rộng & chiều cao dưới dạng 1 điểm trên mặt phẳng 2 chiều.
- Tính khoảng cách từ khối cần tìm đến tất cả các khối còn lại. (bài toán người láng giềng gần nhất với độ phức tạp $O(n)$).



B. Tiếp cận hình học

■ Cách 2:

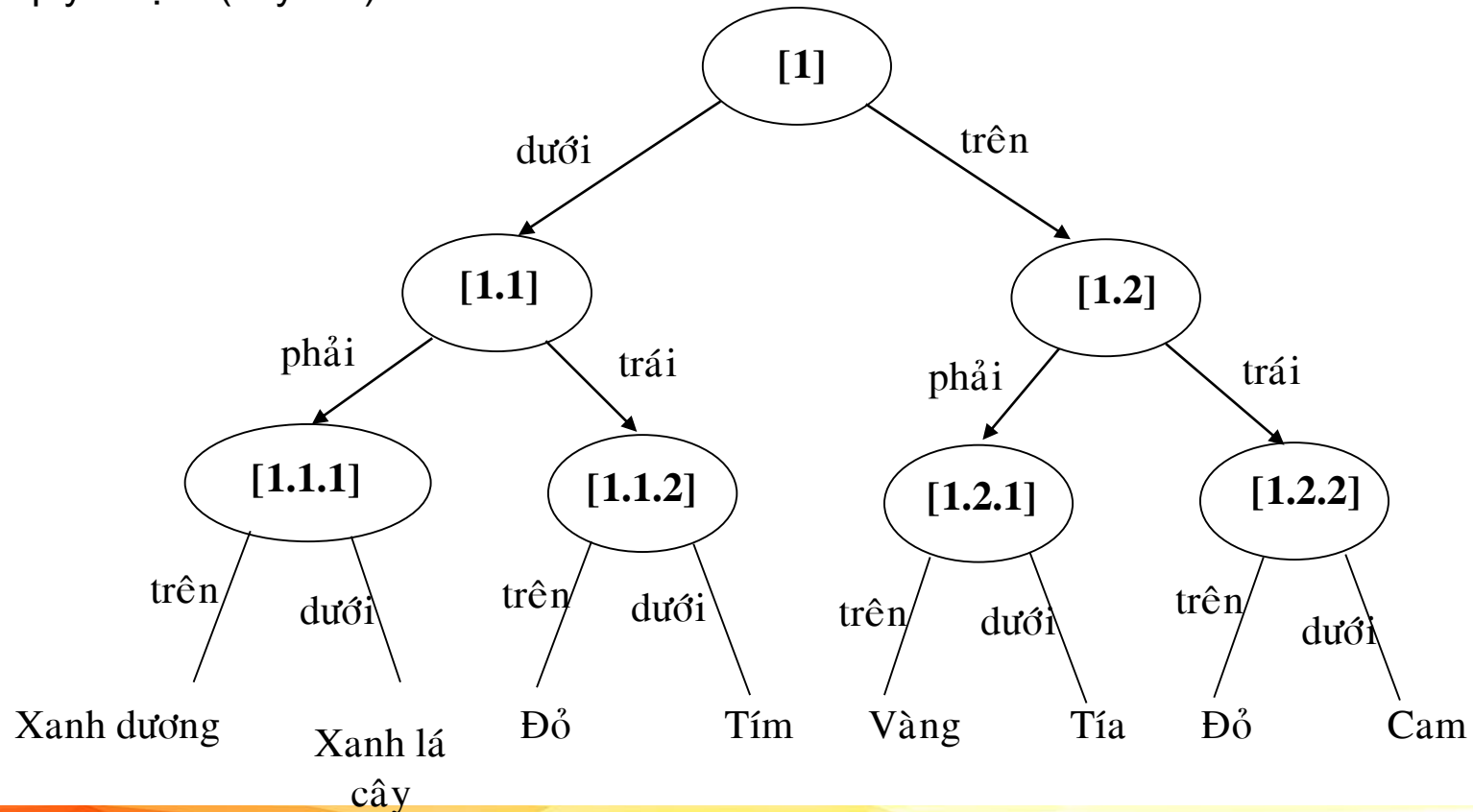
- Cách làm hiệu quả hơn là **tìm cách chia không gian các hình chữ nhật mẫu thành các khu vực riêng biệt theo kiểu phân cấp không gian.**
- 8 không gian riêng biệt ứng với 8 hình chữ nhật đã cho ban đầu.
- Lần lượt xác định vị trí tương đối của U so với các đường chia. Cuối cùng U xếp cùng không gian với hình chữ nhật có màu cam → **U có màu cam**



B. Tiếp cận hình học

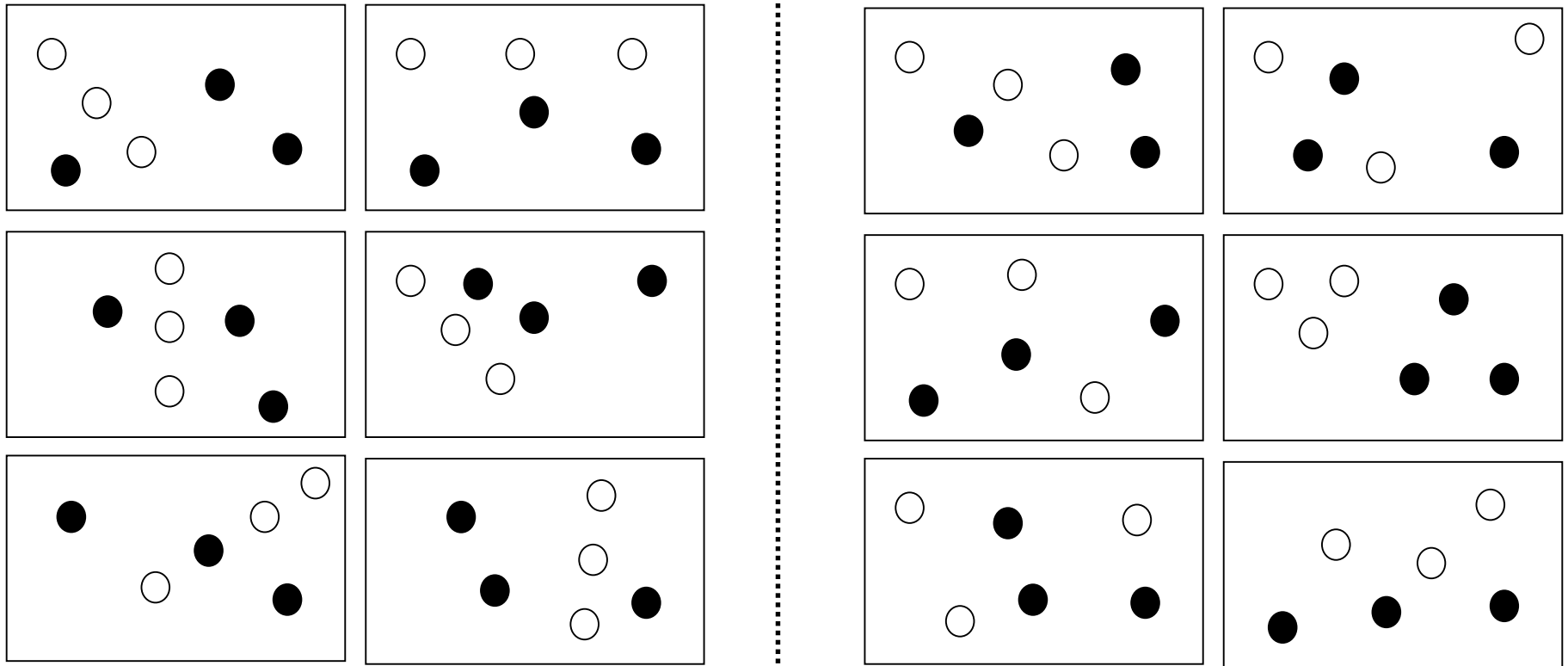
■ Nhận xét bài toán:

- Về mặt thuật toán, phân chia không gian theo cách làm trên là phân chia theo cây k-d.
- Cây quyết định (cây k-2) của bài toán có thể biểu diễn như sau:



C. Tiếp cận logic

Ví dụ 1: Hãy thử tìm đặc tính để phân biệt hai nhóm hình ảnh A và B dưới đây.





5.1 Máy học

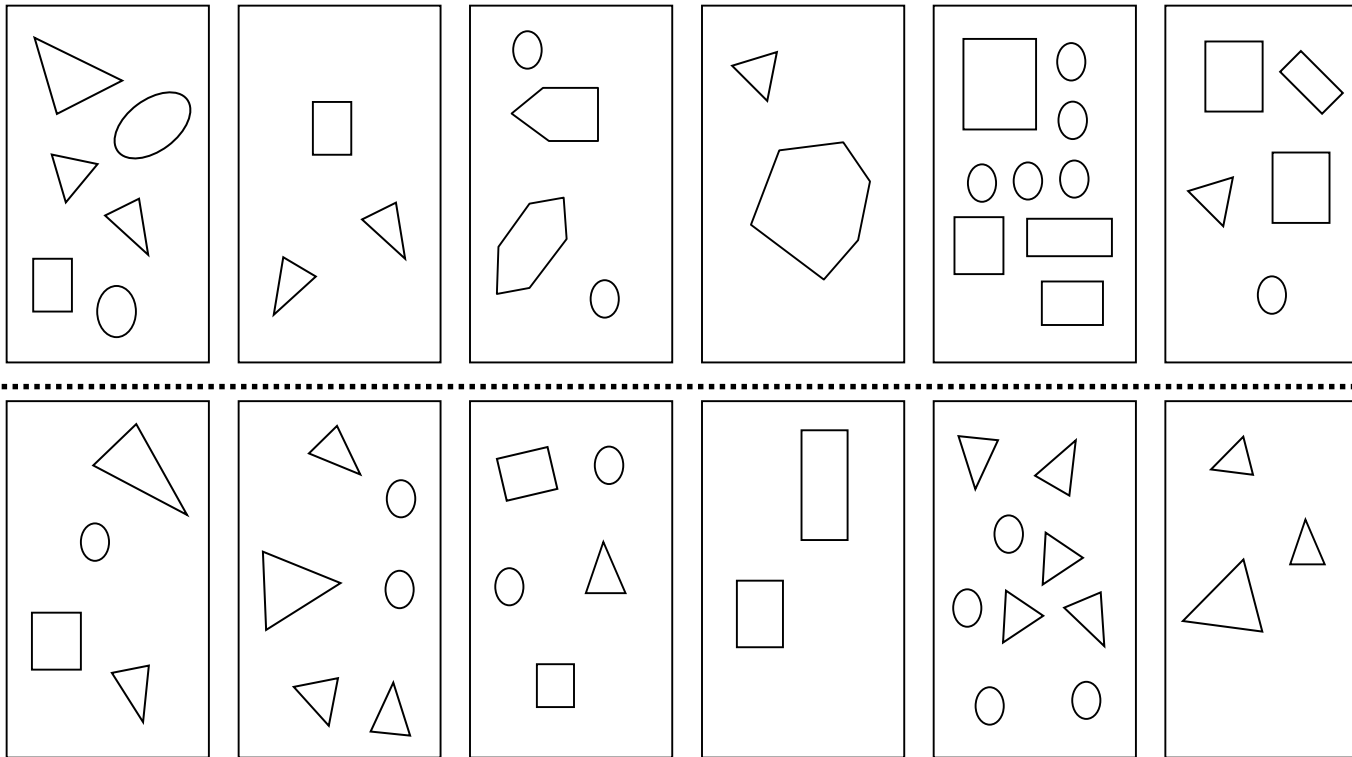
C. Tiếp cận logic

▪ Nhận xét ví dụ 1:

- Nếu tinh ý bạn sẽ nhận thấy các điểm trắng trong nhóm A luôn thẳng hàng.
- Thật khó để phát hiện ra đặc tính vừa nêu trên (ngay cả đối với con người) nhất là đối với các đối tượng hình học.
- Nhà bác học Bongard đã đề ra một phương án xác định mối liên hệ bằng cách xây dựng các mệnh đề logic. (xem ví dụ 2)

C. Tiếp cận logic

Ví dụ 2: Xác định đặc điểm của các nhóm hình A, B.



Nhóm A

Nhóm B



5.1 Máy học

C. Tiếp cận logic

- **Nhóm A:** Tổng số đỉnh trừ tổng số đối tượng là 7. (Chẳng hạn như hình 2 trong nhóm A có 3 hình gồm 2 tam giác và một hình chữ nhật, tổng cộng có 10 đỉnh).
- **Nhóm B:** Tổng số đỉnh trừ tổng số đối tượng là 6.
- Hình ellipse và hình tròn được xem là không có đỉnh nào
- Không được gợi ý thì quan hệ trên là một loại quan hệ rất khó được phát hiện.
- Với phương án của Bongard, ta vẫn có thể tìm ra được mối liên hệ đủ để phân biệt hai nhóm hình này.



5.1 Máy học

C. Tiếp cận logic

- Định ra một số các mệnh đề logic đơn giản như:

P1 : “tồn tại tam giác”

P2 : “tồn tại vòng tròn”

P3 : “tồn tại hình oval”

P4 : “tồn tại hình chữ nhật”

P5 : “tồn tại hình đa giác nhiều hơn 4 cạnh”.

C. Tiếp cận logic

Hình	Tam giác P_1	Vòng tròn P_2	Oval P_3	Chữ nhật P_4	Đa giác P_5	Nhóm
1	1	1	1	1	0	A
2	1	0	0	1	0	A
3	0	1	0	0	1	A
4	1	0	0	0	1	A
5	0	1	0	1	0	A
6	1	1	0	1	0	A
7	1	1	0	0	0	B
8	1	1	0	1	0	B
9	0	0	0	1	0	B
10	1	0	1	0	0	B
11	1	1	0	0	0	B
12	1	0	0	0	0	B

5.1 Máy học

C. Tiếp cận logic

- Sử dụng các mệnh đề logic khá đơn giản, ta đã xây dựng được một liên hệ “đặc trưng” cho nhóm hình A như sau:

$$\varphi = P_1 P_2 P_3 P_4 \neg P_5 \vee P_1 \neg P_2 \neg P_3 P_4 \neg P_5 \vee \neg P_1 P_2 \neg P_3 \neg P_4 P_5 \vee \\ P_1 P_2 \neg P_3 \neg P_4 \neg P_5 \vee \neg P_1 P_2 \neg P_3 P_4 P_5 \vee P_1 P_2 \neg P_3 P_4 \neg P_5$$

- Bằng các phép biến đổi logic toán học, ta có thể thu gọn mệnh đề trên thành:

$$\varphi = \neg P_1 P_2 \vee P_1 (P_2 P_3 \vee \neg P_2 \neg P_3)$$

- Như vậy 1 hình x nào đó để được xếp vào nhóm hình A thì giá trị các mệnh đề P_1 đến P_6 của hình x phải thỏa mệnh đề φ ở trên.



5.1 Máy học

C. Tiếp cận logic

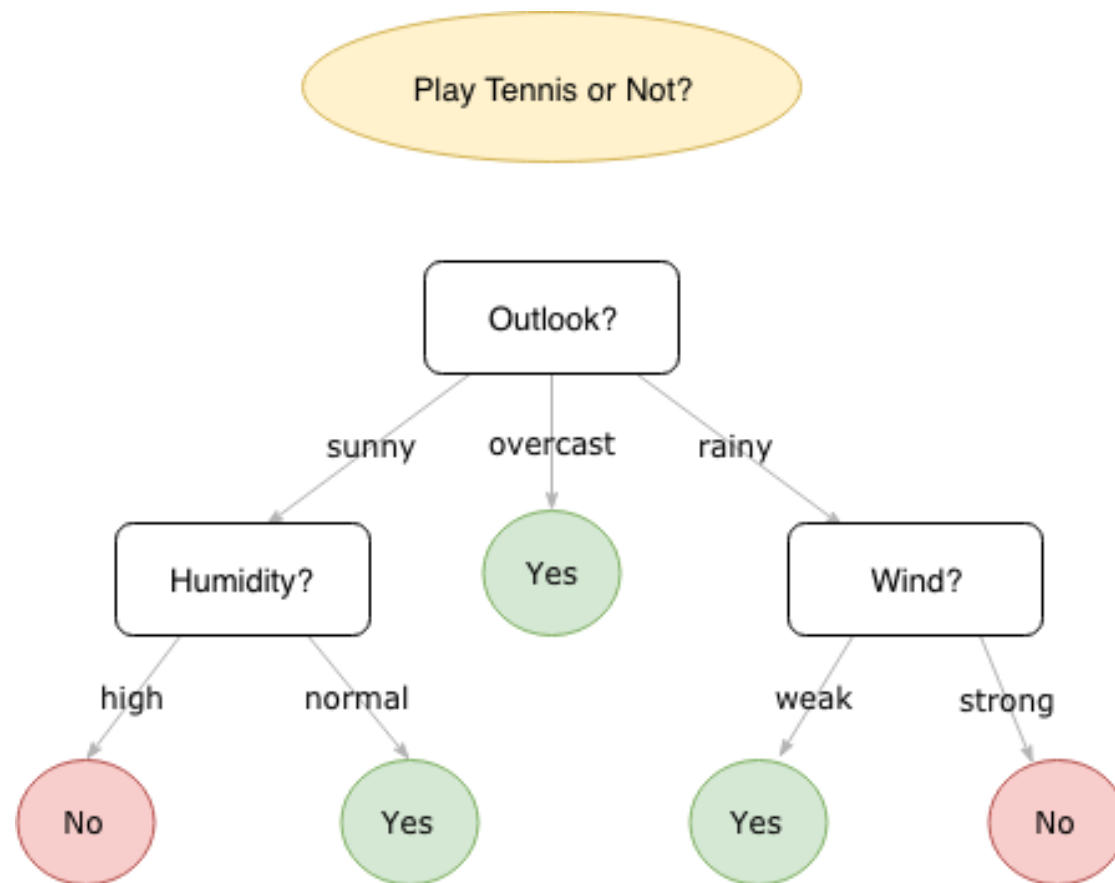
Nhận xét:

- Trong trường hợp tổng quát, phải chọn các mệnh đề cơ sở (như các mệnh đề P_1, P_2, \dots, P_6 trong ví dụ trên) như thế nào để mệnh đề đặc trưng của tất cả các hình trong tập mẫu là khác nhau và mệnh đề đặc trưng của nhóm hình cũng phải khác nhau.
- Làm sao xây dựng thủ tục để kiểm tra giá trị các mệnh đề cơ sở. Mắt người có thể dễ dàng nhận biết sự tồn tại một hình tròn, hình tam giác, ... trong một hình ảnh có nhiều đối tượng khác nhau nhưng làm điều bằng chương trình máy tính hoàn toàn không đơn giản.
- Chính vì lý do đó, phương pháp học này rất cần đến sự hỗ trợ của con người trong việc đưa ra quyết định tính đúng đắn của các mệnh đề thành viên trong mệnh đề đặc trưng.

5.2 Cây quyết định

<https://www.cs.waikato.ac.nz › weka>

- **Cây quyết định** (Decision Tree): Được áp dụng vào 2 bài toán phân loại (Classification) và hồi quy (Regression).

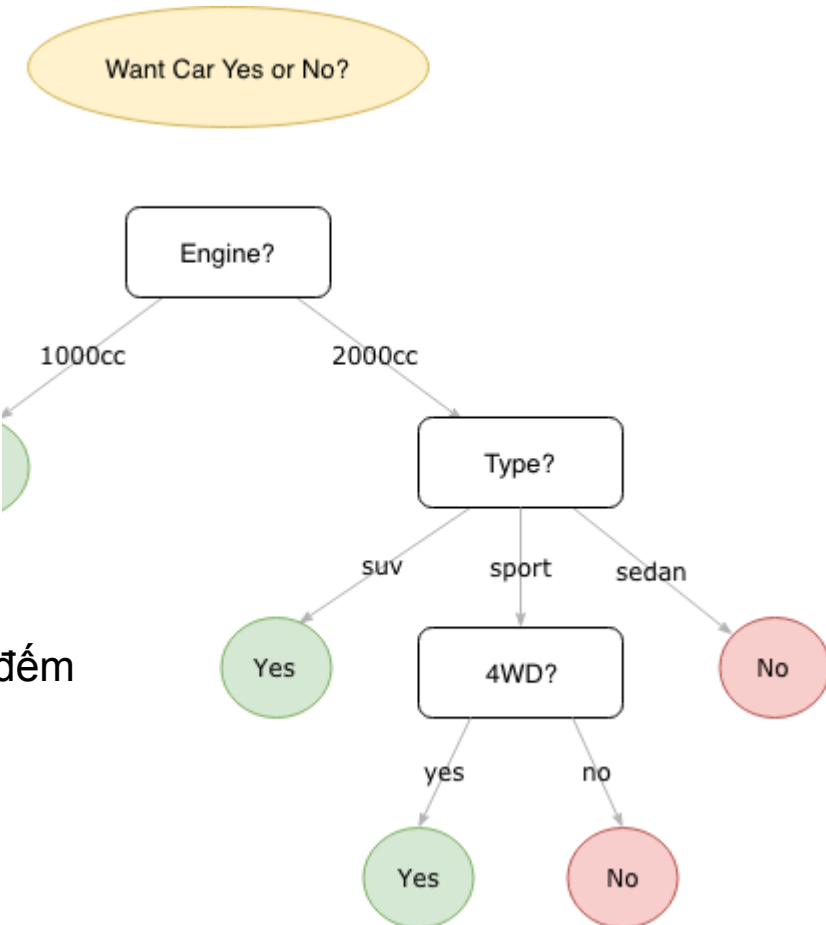


▪ Thuật toán ID3 (Iterative Dichotomiser 3)

ID	Engine	Type	Color	4WD	Want?
1	2000cc	SUV	Silver	Yes	Yes
2	1000cc	Sedan	Silver	Yes	Yes
3	2000cc	Sport	Blue	No	No
4	1000cc	SUV	Blue	No	Yes
5	2000cc	Sedan	Silver	Yes	No
6	2000cc	Sport	Blue	Yes	Yes
7	1000cc	Sedan	Blue	No	Yes
8	1000cc	SUV	Silver	No	Yes

Tập Training Data

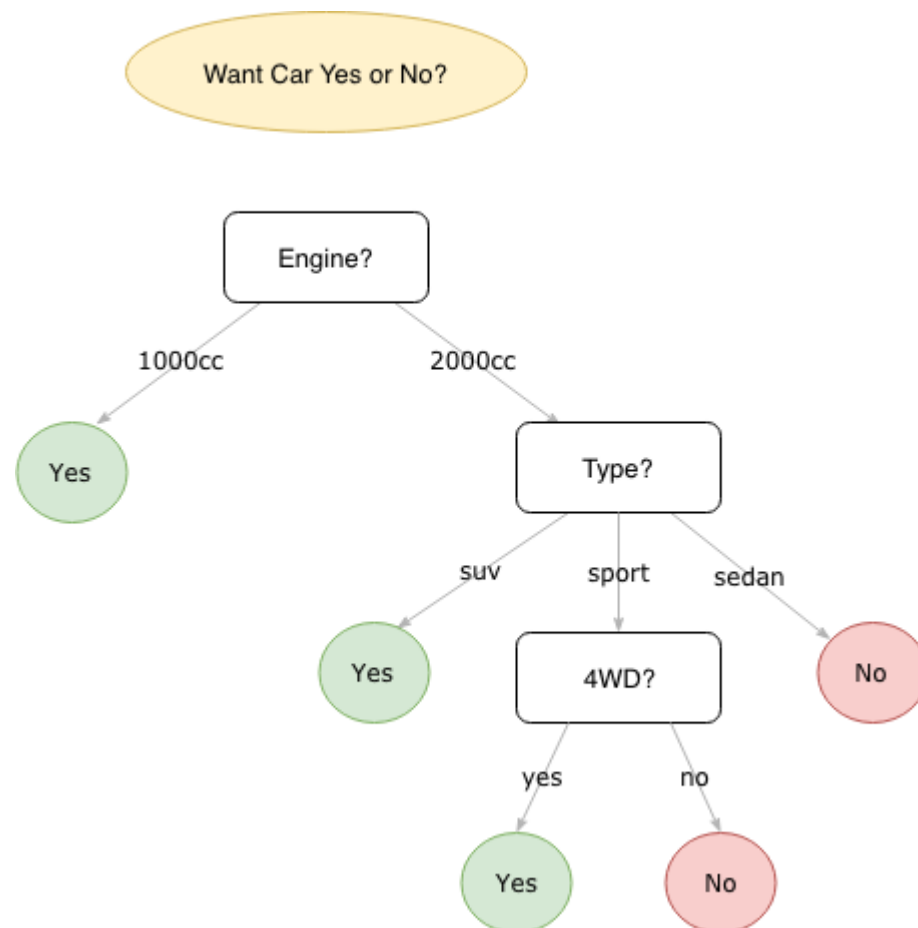
- Lấy tất cả các thuộc tính chưa được sử dụng và đếm entropy liên quan đến mẫu thử của các thuộc tính đó
- Chọn thuộc tính có entropy lớn nhất
- Nối node với thuộc tính đó



Thuật toán J48 (Iterative Dichotomiser 3)

ID	Engine	Type	Color	4WD	Want?
1	2000cc	SUV	Silver	Yes	Yes
2	1000cc	Sedan	Silver	Yes	Yes
3	2000cc	Sport	Blue	No	No
4	1000cc	SUV	Blue	No	Yes
5	2000cc	Sedan	Silver	Yes	No
6	2000cc	Sport	Blue	Yes	Yes
7	1000cc	Sedan	Blue	No	Yes
8	1000cc	SUV	Silver	No	Yes

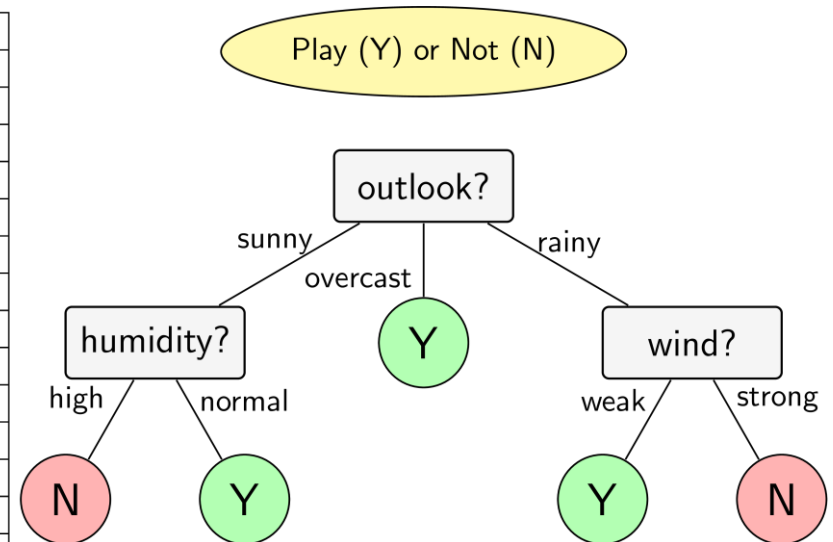
Tập Training Data



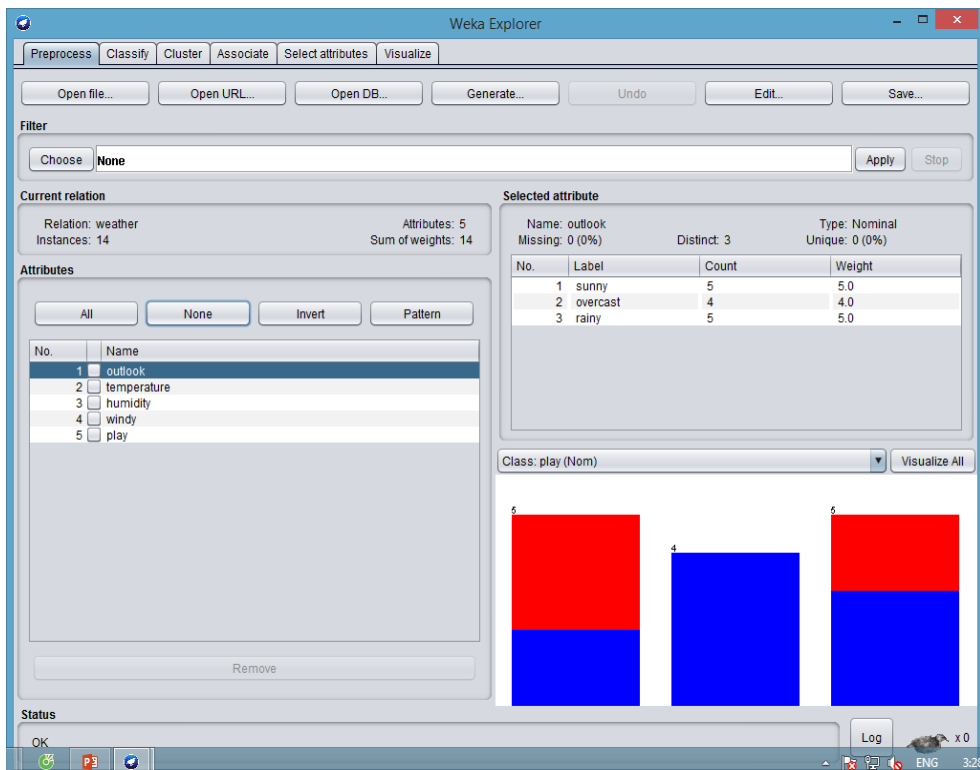
Thuật toán C4.5

- Thuật toán phân lớp dữ liệu dựa trên cây quyết định hiệu quả và phổ biến trong những ứng dụng khai phá cơ sở dữ liệu có kích thước nhỏ. C4.5 xây dựng cây quyết định từ tập training data

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no



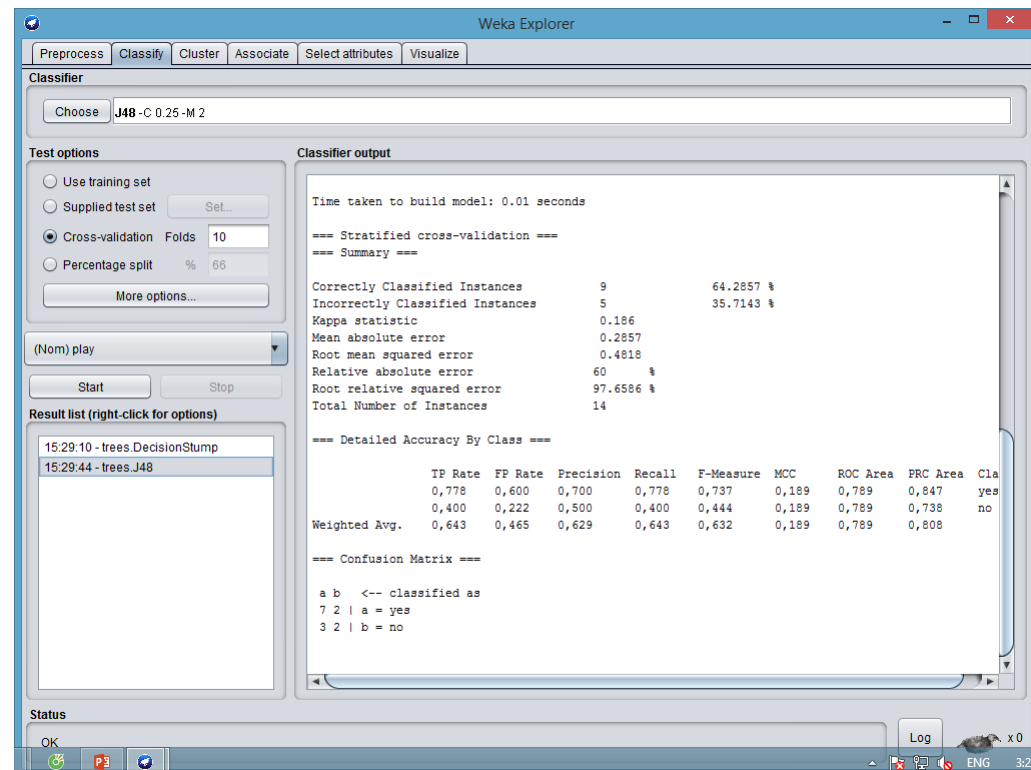
■ Weka



Weka Explorer interface showing the 'weather' dataset. The 'Attributes' list includes outlook, temperature, humidity, windy, and play. The 'Selected attribute' is 'outlook' with a bar chart showing counts for sunny (5), overcast (4), and rainy (5). The 'Class' is 'play (Nom)'.

No.	Name
1	outlook
2	temperature
3	humidity
4	windy
5	play

No.	Label	Count	Weight
1	sunny	5	5.0
2	overcast	4	4.0
3	rainy	5	5.0



Weka Explorer Classifier output for the 'J48' classifier. The output shows stratified cross-validation results and a detailed accuracy by class table.

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
 === Summary ===

Metric	Value	Percentage
Correctly Classified Instances	9	64.2857 %
Incorrectly Classified Instances	5	35.7143 %
Kappa statistic	0.186	
Mean absolute error	0.2857	
Root mean squared error	0.4818	
Relative absolute error	60	%
Root relative squared error	97.6586	%
Total Number of Instances	14	

=== Detailed Accuracy By Class ===

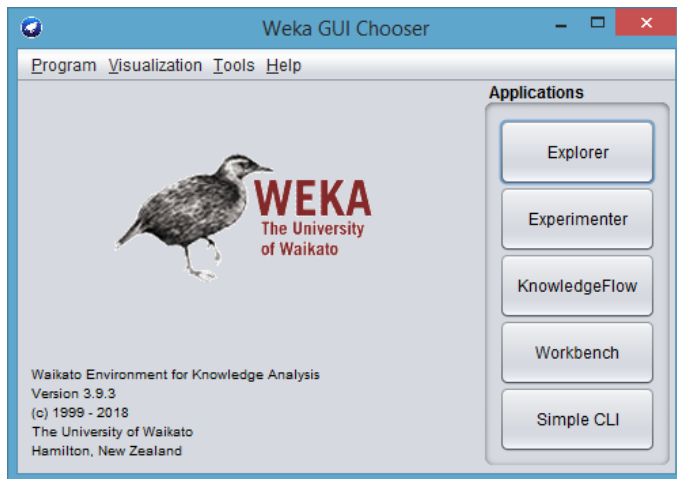
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
Weighted Avg.	0,643	0,465	0,629	0,643	0,632	0,189	0,789	0,808	no

=== Confusion Matrix ===

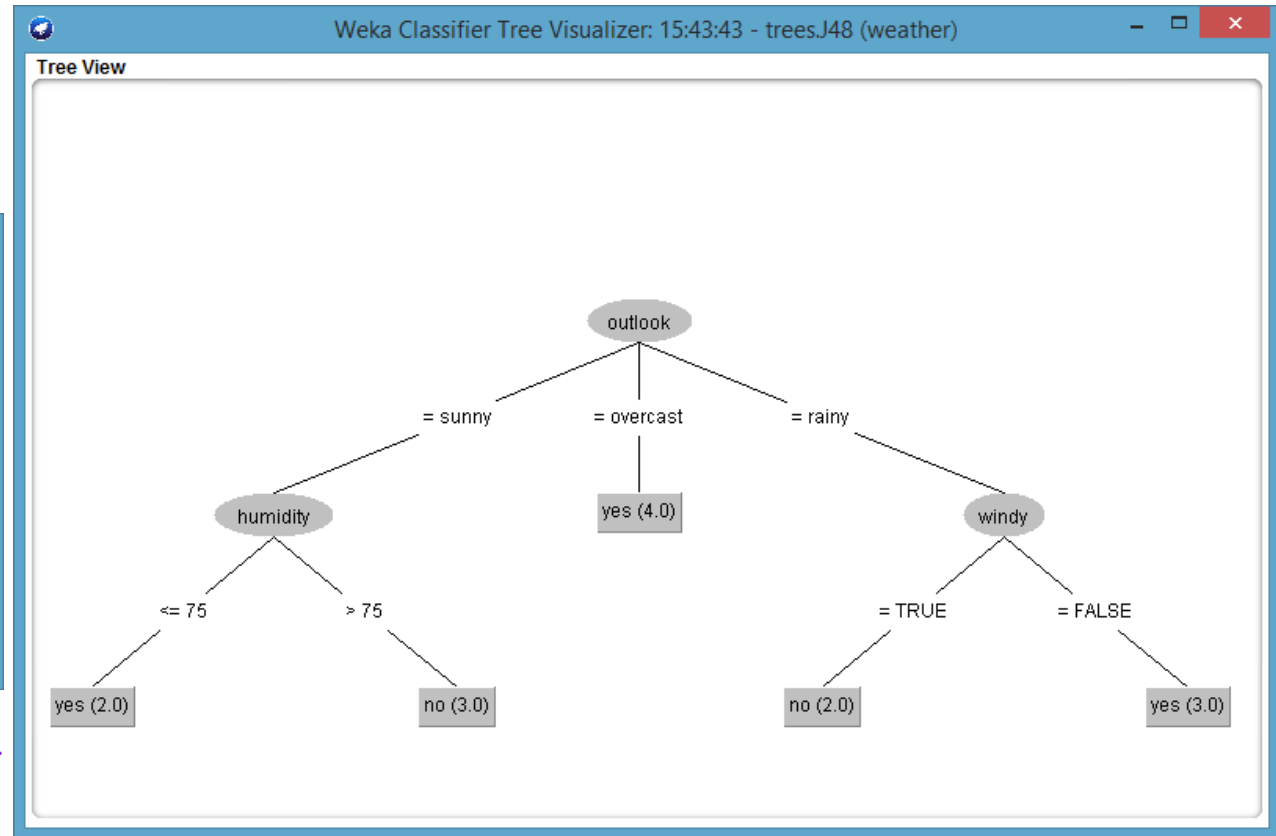
```

a b <-- Classified as
7 2 | a = yes
3 2 | b = no
  
```

■ Công cụ: Weka

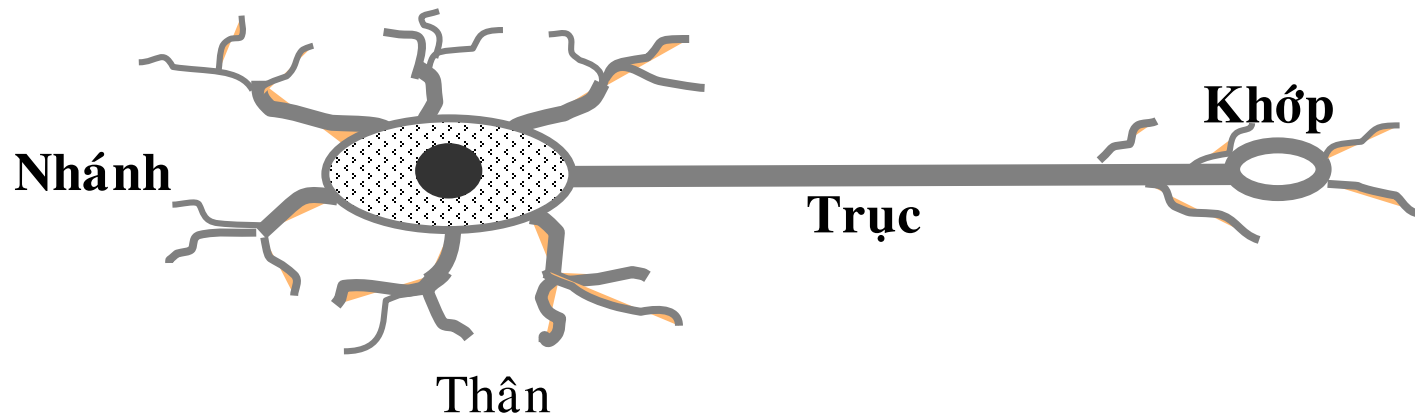


<https://www.cs.waikato.ac.nz › weka>



5.3 Mạng Neural

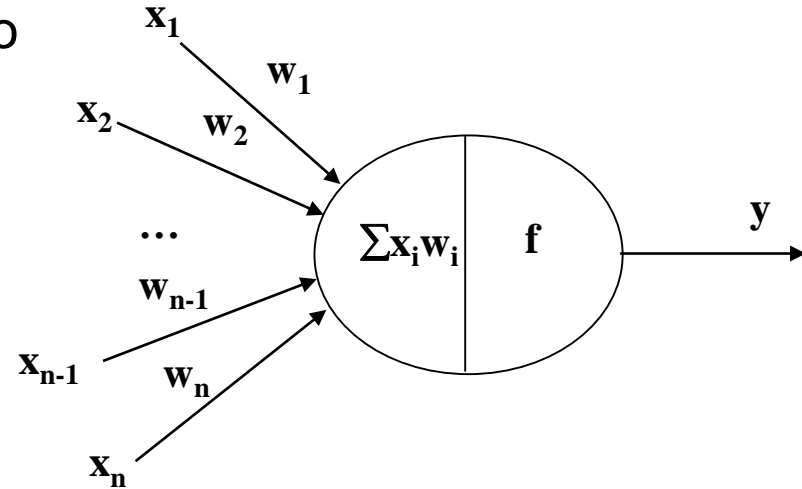
- Mạng neural là phương pháp giải quyết vấn đề – bài toán trên máy tính **mô phỏng hoạt động của các tế bào thần kinh trong não bộ**.
- Mạng neural nhân tạo là sự mô phỏng cấu trúc của mạng neural sinh học. Mạng neural nhân tạo được tạo thành bởi sự nối kết giữa rất nhiều đơn vị thần kinh gọi là **perceptron**.



Cấu trúc của một tế bào thần kinh sinh học

5.3 Mạng Neural

- Cấu tạo một đơn vị thần kinh nhân tạo



- Giá trị đầu ra y của một perceptron được tính bằng công thức sau:

$$y = f((x_n w_n + x_{n-1} w_{n-1} + \dots + w_2 n_2 + w_1 n_1 + w_0) - \phi)$$

(ϕ được gọi là ngưỡng kích hoạt của neural)

- Hàm f được gọi là hàm truyền. Một hàm truyền cần phải có tính chất sau: bị chặn, đơn điệu tăng và là hàm liên tục tăng



5.3 Mạng Neural

Các hàm truyền thường được sử dụng:

- Hàm logistic (hay hàm Sigma)

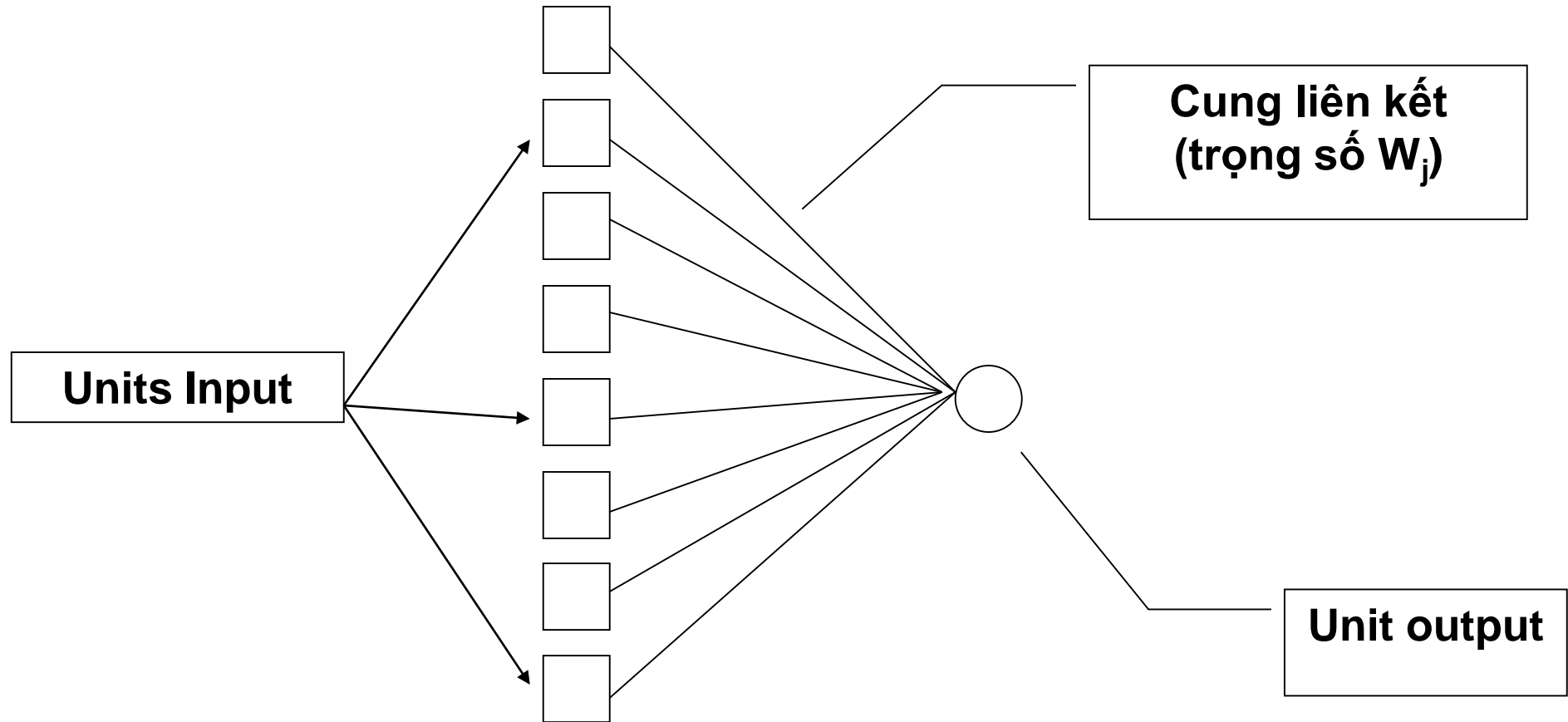
$$f(x) = \frac{1}{1 + e^{-x}}$$

- Hàm hyperbol $h(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$

- Hàm tang-hyperbol $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

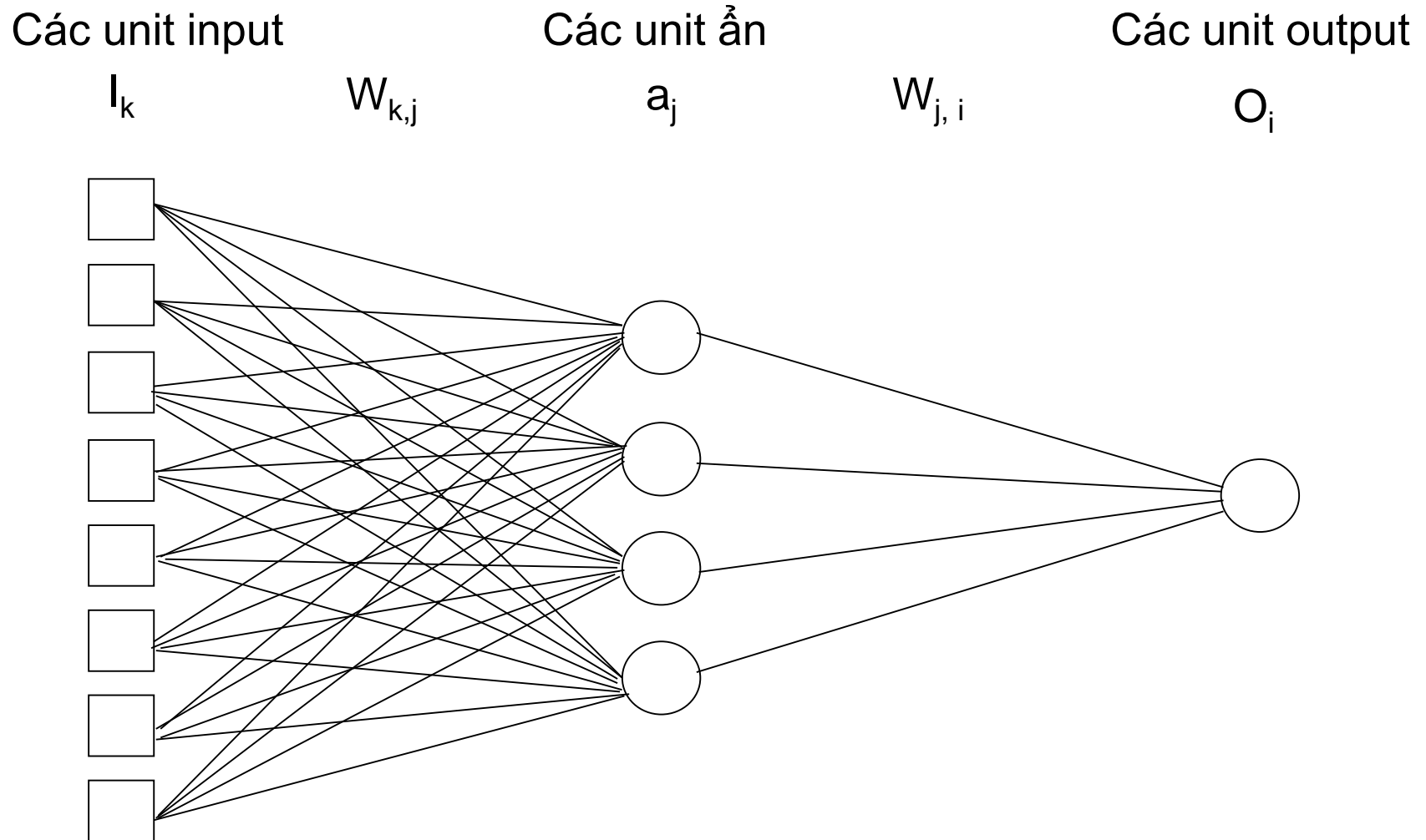
5.3 Mạng Neural

▪ Mô hình minh họa mạng neural 1 lớp



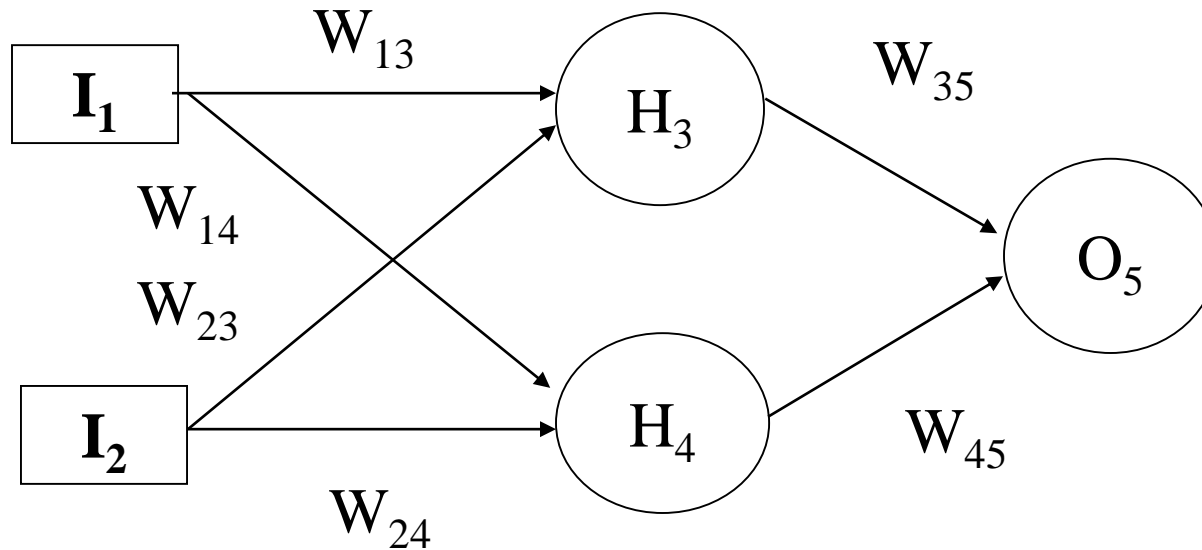
5.3 Mạng Neural

■ Mô hình minh họa mạng neural tổng quát



5.3 Mạng Neural

▪ Mạng lan truyền (Feed Forward)

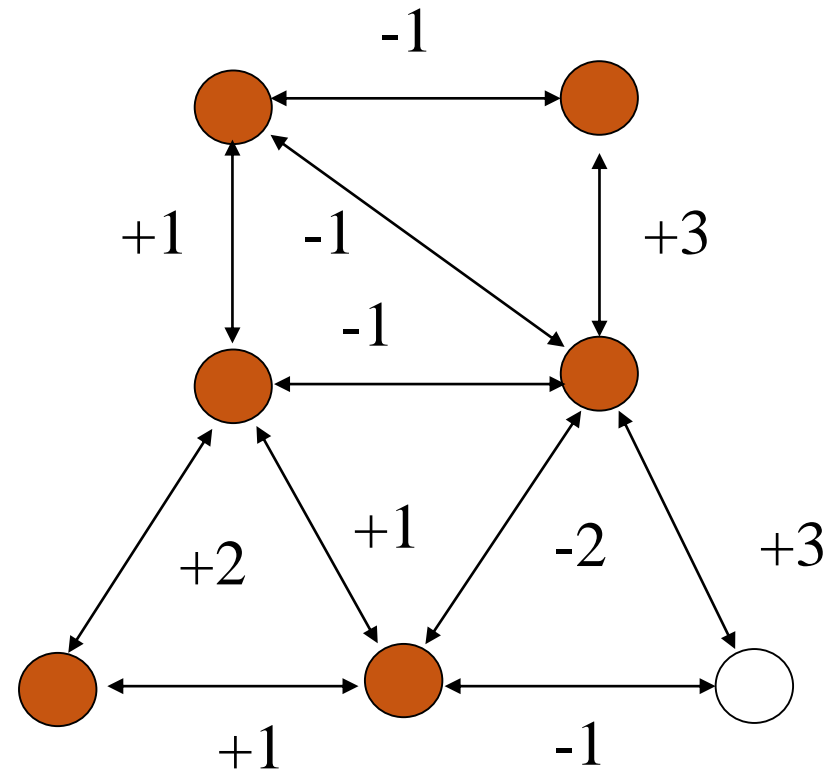
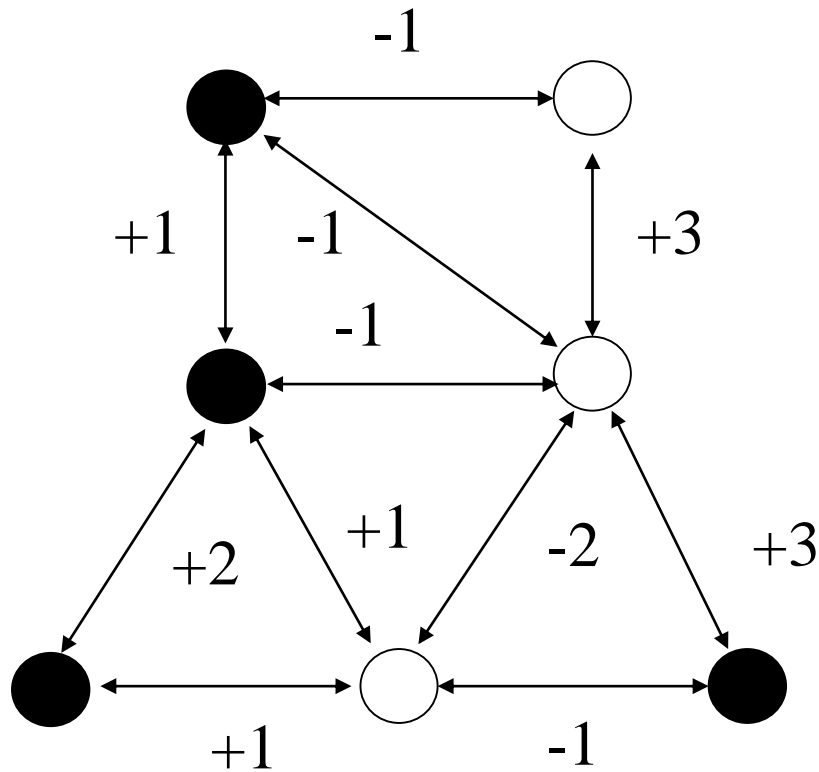


Nguyên tắc xác định giá trị Output của node 5:

$$\begin{aligned} a_5 &= f(W_{3,5}a_3 + W_{4,5}a_4) \\ &= f(W_{3,5} f(W_{1,3}a_1 + W_{2,3}a_2) + W_{4,5} f(W_{1,4}a_1 + W_{2,4}a_2)) \end{aligned}$$

5.3 Mạng Neural

▪ Mạng Hopfield



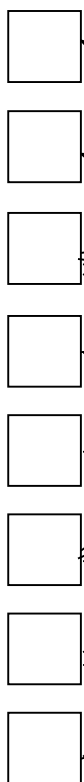
Mạng Hopfield hoạt động tương tự hoạt động một bộ nhớ kết hợp có:

$$W_{i,j} = W_{j,i}$$

▪ Mạng Perceptron

Các unit input

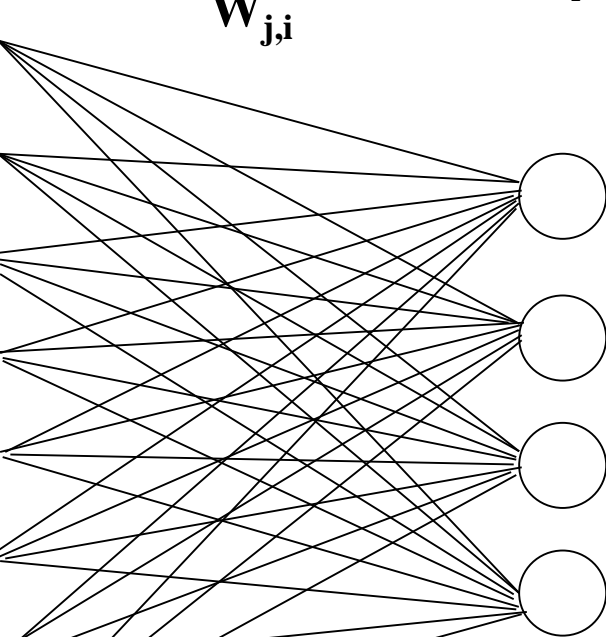
I_j



Các unit output

O_i

$W_{j,i}$



Các unit input

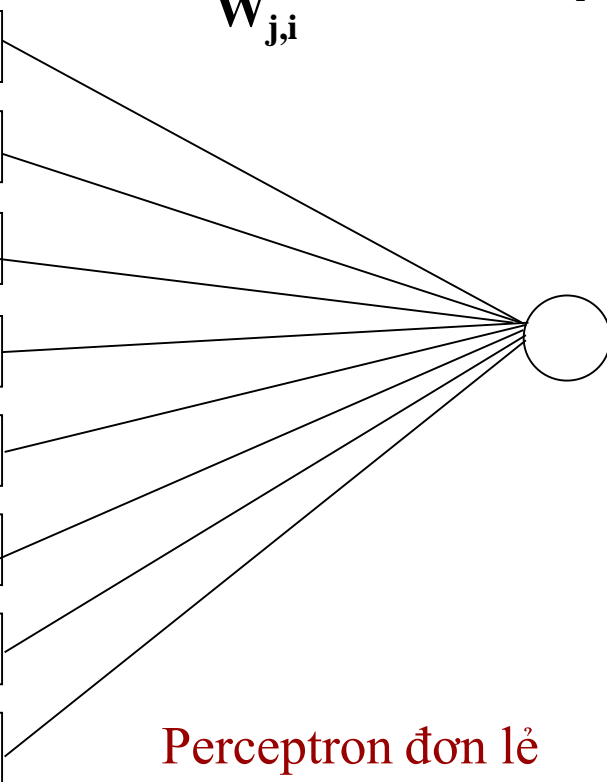
I_j



Các unit output

O_i

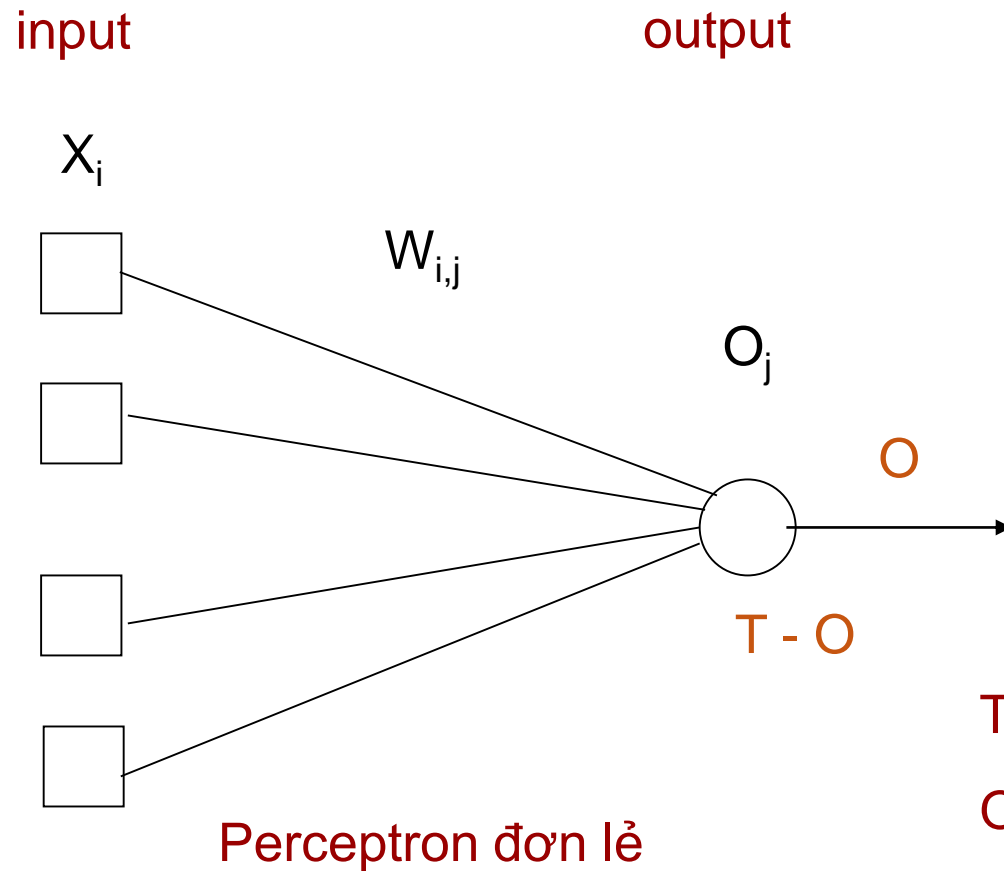
$W_{j,i}$



Perceptron đơn lẻ

5.3 Mạng Neural

▪ Mạng Perceptron



$$W_i = W_i + \alpha * X_i$$

$$W_i = W_i - \alpha * X_i$$

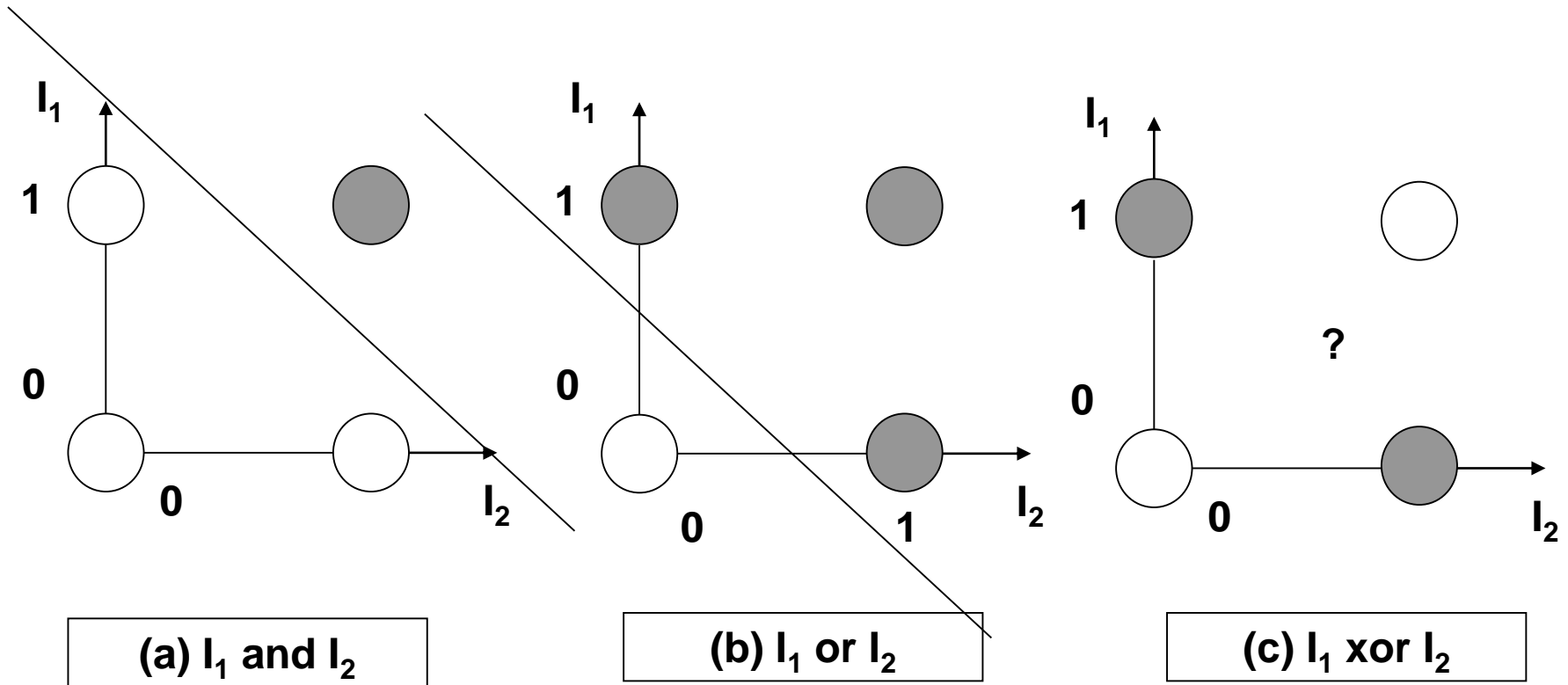
α : hệ số học

T : đầu ra mong muốn

O: đầu ra của mạng

5.3 Mạng Neural

▪ Khả năng phân chia tuyến tính trong mạng Perceptron



$$Y(I_1, I_2) = \begin{cases} 1 & \text{nếu } W_1 I_1 + W_2 I_2 > \theta \\ 0 & \text{ngược lại} \end{cases}$$

5.3 Mạng Neural

▪ Hạn chế của mạng Perceptron

- Năm 1969, Minsky và Papert đã phân tích và chứng minh sự thất bại của mạng Perceptron với bài toán XOR.

I_1	I_2	$y(I_1, I_2) = I_1 \text{ XOR } I_2$
1	1	0
1	0	1
0	1	1
0	0	0

Xét từng trường hợp:

Nếu $[I_1, I_2]=[0,0]$ thì $y(I_1, I_2)=0 \Rightarrow W_1*0 + W_2*0 < \theta \Rightarrow \theta > 0$

Nếu $[I_1, I_2]=[0,1]$ thì $y(I_1, I_2)=1 \Rightarrow W_1*0 + W_2*1 \geq \theta \Rightarrow \theta \leq W_1$

Nếu $[I_1, I_2]=[1,0]$ thì $y(I_1, I_2)=1 \Rightarrow W_1*1 + W_2*0 \geq \theta \Rightarrow \theta \leq W_2$

Nếu $[I_1, I_2]=[1,1]$ thì $y(I_1, I_2)=0 \Rightarrow W_1*1 + W_2*1 < \theta \Rightarrow \theta > W_1 + W_2$

\Rightarrow Rõ ràng không thể tìm được W_1, W_2, θ thỏa mãn 4 bất đẳng thức trên.

\Rightarrow Mô hình Perceptron không thể tìm ra lời giải trong trường hợp này vì hàm XOR không phải là hàm ngưỡng tuyến tính.

5.3 Mạng Neural

■ Ví dụ minh họa việc học của mạng Perceptron

	Cột thêm vào	Độ dài đài hoa	Độ rộng đài hoa	Độ dài cánh hoa	Độ rộng cánh hoa	Loài
Cây Iris A	1	4.7	3.2	1.3	0.2	-1
Cây Iris B	1	6.1	2.8	4.7	1.2	1
Cây Iris C	1	5.6	3.0	4.1	1.3	1
Cây Iris D	1	5.8	2.7	5.1	1.9	-1
Cây Iris E	1	6.5	3.2	5.1	2.0	-1
Cây Iris Q	1	5.8	2.7	3.9	1.2	???

Khởi tạo trọng số: $w_1=1$; $w_2=0$; $w_3=0$; $w_4=0$; $w_5=1$

5.3 Mạng Neural

▪ Ví dụ minh họa việc học của mạng Perceptron

- Dự đoán lần thứ nhất đối với mẫu “Cây Iris A”

$$\sum_{i=1}^5 w_i x_i = 1 * 1 + 0 * 4.7 + 0 * 3.2 + 0 * 1.3 + 1 * 0.2 = 1.2$$

Perceptron dự đoán “Loài” có kết quả là 1 so với kết quả đúng là -1

- Cập nhật trọng số mạng $w_i \leftarrow w_i - \alpha * x_i$
- Nếu kết quả “Loài” là 1 so với kết quả đúng là -1 (α : Tỷ lệ học nên chọn nhỏ, ở đây ta chọn $\alpha = 0.05$)

5.3 Mạng Neural

■ Ví dụ minh họa việc học của mạng Perceptron

$$w_1 \leftarrow w_1 - 0.05 * x_1 = 1 - 0.05 * 1 = 0.95$$

$$w_2 \leftarrow w_2 - 0.05 * x_2 = 0 - 0.05 * 4.7 = -0.24$$

$$w_3 \leftarrow w_3 - 0.05 * x_3 = 0 - 0.05 * 3.2 = -0.16$$

$$w_4 \leftarrow w_4 - 0.05 * x_4 = 0 - 0.05 * 1.3 = -0.07$$

$$w_5 \leftarrow w_5 - 0.05 * x_5 = 1 - 0.05 * 0.2 = 0.99$$

Các trọng số này dùng để luyện cho những mẫu kế tiếp

❖ Điều kiện dừng:

Tất cả các mẫu đã được dự đoán đúng.

Dừng sau một số bước lặp do người dùng quyết định.

5.3 Mạng Neural

Nhận xét: Cập nhật trọng số sao cho tối thiểu sai lệch.

(1) $w_i \leftarrow w_i - \alpha * x_i$ Nếu kết quả “Loài” là 1 so với kết quả đúng là -1

(2) $w_i \leftarrow w_i + \alpha * x_i$ Nếu kết quả “Loài” là -1 so với kết quả đúng là 1

(1) $w_i x_i$ góp phần làm cho $\sum_{i=1}^n w_i x_i > 0$.

- Trọng số mới là $w_i \leftarrow w_i - \alpha * x_i$. Vì thế bước kế tiếp cho cùng mẫu dữ liệu ta được $(w_i - \alpha * x_i) * x_i = w_i x_i - \alpha * x_i^2 < w_i x_i$
- Như vậy $\sum_{i=1}^n w_i x_i$ sau khi thay đổi trọng số nhỏ hơn trước đó.
- Vì vậy kết quả dự đoán sẽ gần về kết quả đúng là -1.

5.3 Mạng Neural

- Nguyên lý hoạt động của mạng Perceptron

- Bước 1: Xác định giá trị của Err

$$Err = T - O \text{ (} T \text{: đầu ra mong muốn)}$$

if $Err > 0$ then tăng O

else giảm O

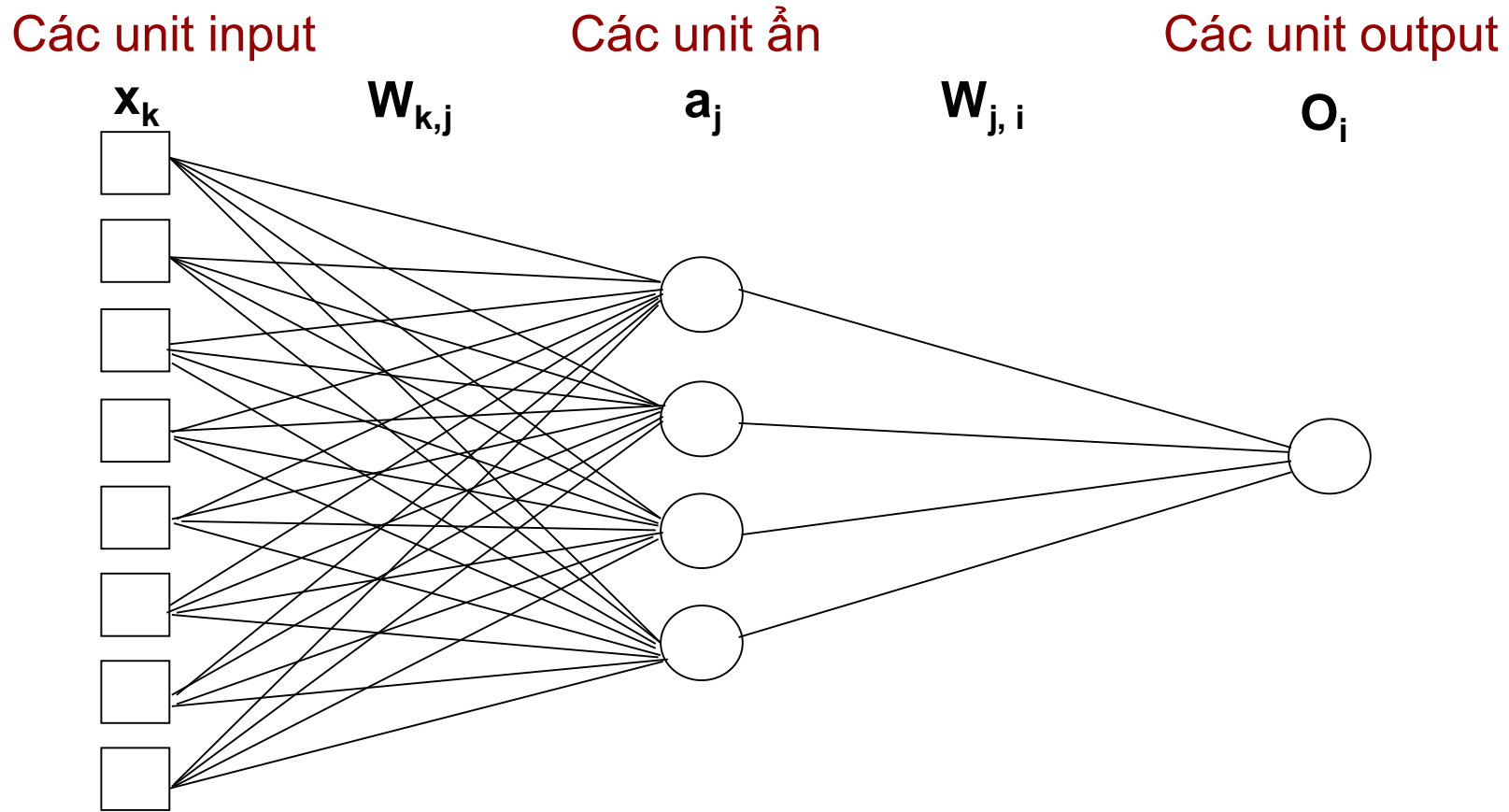
- Bước 2: Cập nhật giá trị ma trận trọng số

$$W_j = W_j + \alpha * x_j * Err. \text{ (}\alpha \text{ là hệ số học)}$$

- Bước 3: Lặp lại bước 1 và 2 cho đến khi $Err < \varepsilon$ thì dừng.

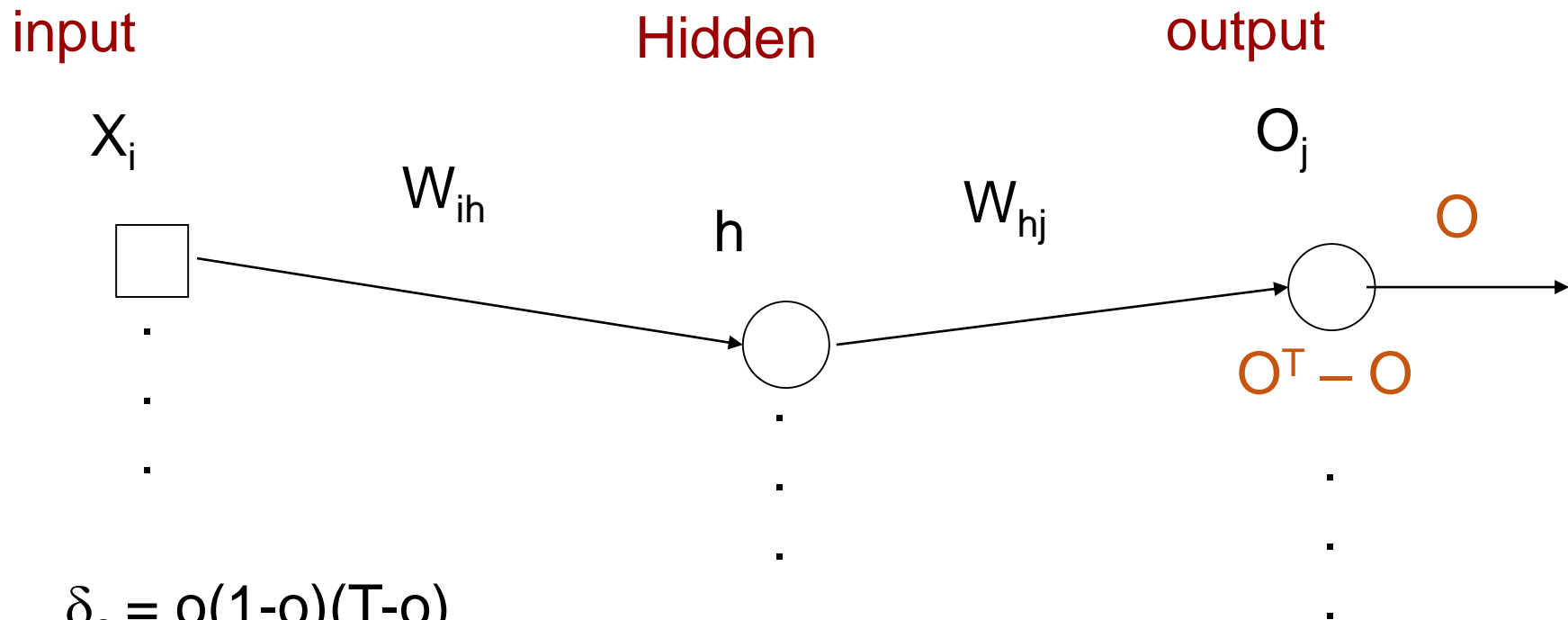
5.3 Mạng Neural

Mạng lan truyền ngược (Back propagation)



5.3 Mạng Neural

Mạng lan truyền ngược (Back propagation)



$$\delta_o = o(1-o)(T-o)$$

$$W_{hj} = W_{hj} + \alpha * \delta_{oj} * o_j$$

$$W_{ih} = W_{ih} + \alpha * \delta_o * W_h * o_h * (1-o_h) * x_i$$

5.3 Mạng Neural

Mạng lan truyền ngược (Back propagation)

- Tập mẫu cho quá trình học

Mẫu	Đầu vào	Đầu ra mong muốn
1	$x^1 = (x_1^1, x_2^1, \dots, x_n^1)$	y^1
2	$x^2 = (x_1^2, x_2^2, \dots, x_n^2)$	y^2
...
M	$x^m = (x_1^m, x_2^m, \dots, x_n^m)$	y^m

- Cho mẫu x^k vào luyện \rightarrow Cho ra O^k sai lệch với kết quả mong muốn y^k . Sai lệch của mẫu thứ k là E_k :

$$E_k = \frac{1}{2} (y^k - O^k)^2 = \frac{1}{2} (y^k - f(w^T x))^2$$

5.3 Mạng Neural

Mạng lan truyền ngược (Back propagation)

- Sai lệch của tất cả các mẫu:

$$E = \sum_{k=1}^m E_k = E_1 + E_2 + \dots + E_m$$

- Cập nhật trọng số: theo hướng cực tiểu hóa sai lệch

$$w = w - \alpha * E'_k(w)$$

α : là hệ số học

$$E'_k(w) = \frac{d}{dw} \left(\frac{1}{2} (y^k - f(w^T x))^2 \right) = -(y^k - f(w^T x)) f'(w^T x)$$

5.3 Mạng Neural

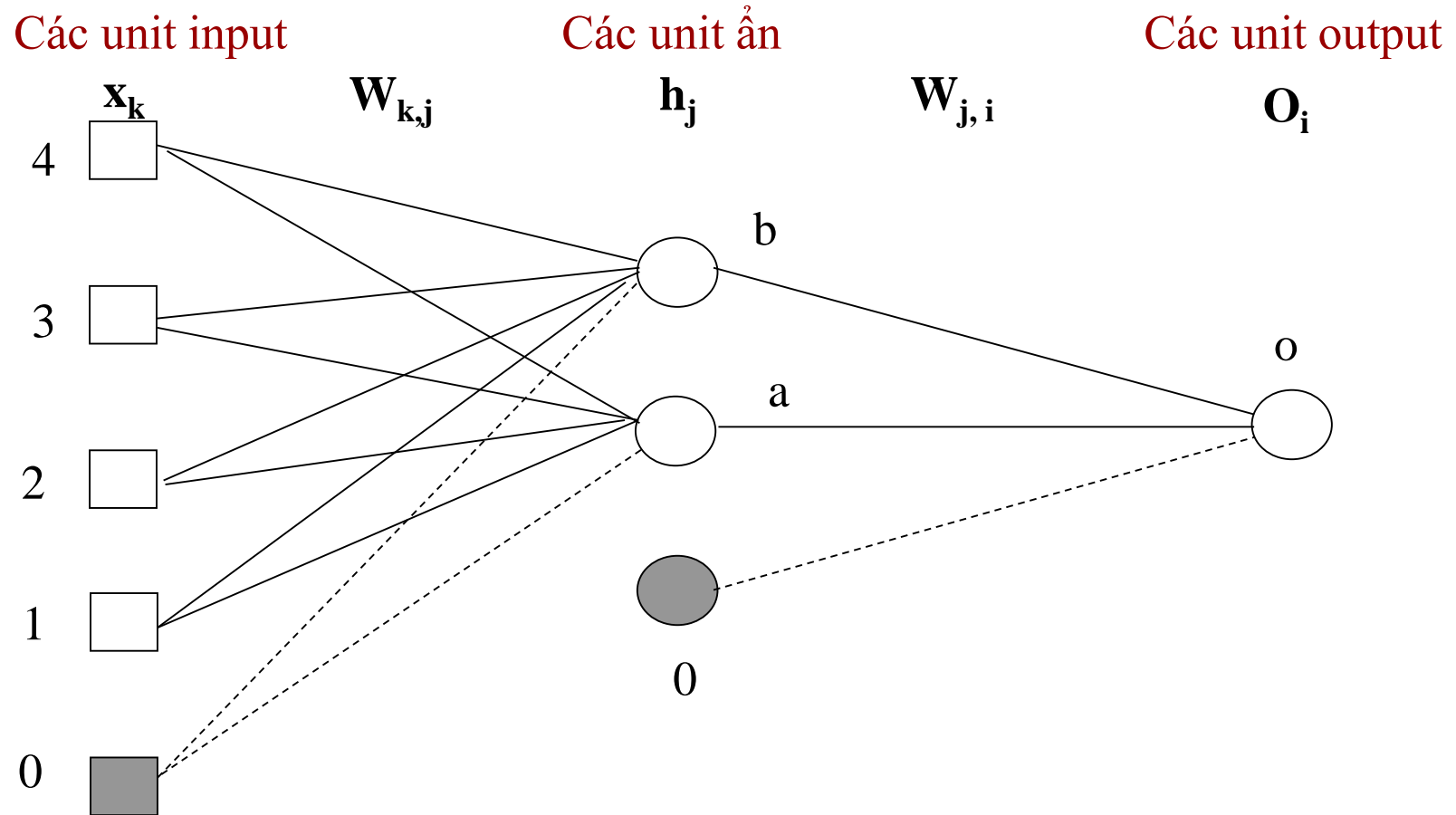
Ví dụ minh họa cho phương pháp lan truyền ngược

	Cột thêm vào	Độ dài đài hoa	Độ rộng đài hoa	Độ dài cánh hoa	Độ rộng cánh hoa	Loài
Cây Iris A	1	4.7	3.2	1.3	0.2	0
Cây Iris B	1	6.1	2.8	4.7	1.2	1
Cây Iris C	1	5.6	3.0	4.1	1.3	1
Cây Iris D	1	5.8	2.7	5.1	1.9	0
Cây Iris E	1	6.5	3.2	5.1	2.0	0
Cây Iris Q	1	5.8	2.7	3.9	1.2	???

Chọn hàm truyền: $f(x) = \frac{1}{1 + e^{-x}}$

5.3 Mạng Neural

Ví dụ minh họa cho phương pháp lan truyền ngược



5.3 Mạng Neural

Ví dụ minh họa cho phương pháp lan truyền ngược

Khởi tạo các trọng số có giá trị ngẫu nhiên nhỏ

$W_{0a} = 0.0$	Trọng số cột giá trị -1 trong đơn vị ẩn a
$W_{1a} = 0.1$	Trọng số trong đơn vị ẩn a từ giá trị nhập thứ 1
$W_{2a} = -0.1$	Trọng số trong đơn vị ẩn a từ giá trị nhập thứ 2
$W_{3a} = -0.1$	Trọng số trong đơn vị ẩn a từ giá trị nhập thứ 3
$W_{4a} = 0.0$	Trọng số trong đơn vị ẩn a từ giá trị nhập thứ 4
$W_{0b} = 0.0$	Trọng số cột giá trị -1 trong đơn vị ẩn b
$W_{1b} = -0.1$	Trọng số trong đơn vị ẩn b từ giá trị nhập thứ 1
$W_{2b} = 0.2$	Trọng số trong đơn vị ẩn b từ giá trị nhập thứ 2
$W_{3b} = 0.1$	Trọng số trong đơn vị ẩn b từ giá trị nhập thứ 3
$W_{4b} = -0.1$	Trọng số trong đơn vị ẩn b từ giá trị nhập thứ 4
$W_{0o} = 0.1$	Trọng số cột giá trị -1 trong đơn vị xuất o
$W_{ao} = 0.2$	Trọng số cột trong đơn vị ẩn b từ đơn vị ẩn a
$W_{bo} = -0.1$	Trọng số cột trong đơn vị ẩn b từ đơn vị ẩn b

5.3 Mạng Neural

Ví dụ minh họa cho phương pháp lan truyền ngược

Bước 1: Dự đoán

- Đối với “cây Iris A” đầu tiên ta tính giá trị xuất của nút ẩn a

$$\begin{aligned}O_a &= f(w_{0a} + w_{1a}x_1 + w_{2a}x_2 + w_{3a}x_3 + w_{4a}x_4) \\ &= f(0.0 + 0.1*4.7 + (-0.1)*3.2 + (-0.1)*1.3 + 0.0*0.2) = f(0.02) = 1/(1 + e^{-0.02}) = 0.5050\end{aligned}$$

- Tương tự với nút ẩn b

$$\begin{aligned}O_b &= f(w_{0b} + w_{1b}x_1 + w_{2b}x_2 + w_{3b}x_3 + w_{4b}x_4) \\ &= f(0.0 + (-0.1)*4.7 + 0.2*3.2 + 0.1*1.3 + (-0.1)*0.2) = f(0.28) = 1/(1 + e^{-0.28}) = 0.5695\end{aligned}$$

- Cuối cùng ta tính giá trị xuất cho nút xuất o

$$\begin{aligned}O_o &= f(w_{0o} + w_{ao}O_a + w_{bo}O_b) \\ &= f(0.1 + 0.2*0.5050 + (-0.1)*0.5695) = f(0.1440) = 1/(1 + e^{-0.1440}) = 0.5359\end{aligned}$$

⇒ Giá trị dự đoán cho “cây Iris A” là 0.5359

5.3 Mạng Neural

Ví dụ minh họa cho phương pháp lan truyền ngược

Bước 2: Xác định sai lệch

- Dùng kỹ thuật lan truyền ngược (back propagation)

Với t_o là giá trị xuất mong muốn và o_o là giá trị xuất thật

Sai lệch cần tính cho nút xuất o : $\delta_o = o_o(1 - o_o)(t_o - o_o)$

Sai lệch cần tính cho nút ẩn h từ o là: $o_h(1-o_h)w_{ho} \delta_o$

- Trong ví dụ đang xét “cây Iris A” có $t_o = 0$.

Sai lệch của nút xuất o : $\delta_o = o_o(1-o_o)(t_o-o_o) = 0.5359*(1-0.5359)*(0-0.5359) = -0.1333$

Sai lệch của **nút ẩn a** :

$$\delta_a = o_a(1-o_a)w_{ao} \delta_o = 0.5050*(1-0.5050)*0.2*(-0.1333) = -0.0067$$

Sai lệch của **nút ẩn b** :

$$\delta_b = o_b(1-o_b)w_{bo} \delta_o = 0.5695*(1-0.5695)*(-0.1)*(-0.1333) = 0.0032$$

5.3 Mạng Neural

Ví dụ minh họa cho phương pháp lan truyền ngược

Bước 3: Bước cập nhật trọng số. Chọn tỉ lệ học $\alpha = 0.1$

$$w_{0a} = w_{0a} + \alpha \cdot \delta_a \cdot 1 = 0.0 + 0.1 \cdot (-0.0067) \cdot 1 = -0.0007$$

$$w_{1a} = w_{1a} + \alpha \cdot \delta_a \cdot x_1 = 0.1 + 0.1 \cdot (-0.0067) \cdot 4.7 = 0.0969$$

$$w_{2a} = w_{2a} + \alpha \cdot \delta_a \cdot x_2 = -0.1 + 0.1 \cdot (-0.0067) \cdot 3.2 = -0.1021$$

$$w_{3a} = w_{3a} + \alpha \cdot \delta_a \cdot x_3 = -0.1 + 0.1 \cdot (-0.0067) \cdot 1.2 = -0.1009$$

$$w_{4a} = w_{4a} + \alpha \cdot \delta_a \cdot x_4 = 0.0 + 0.1 \cdot (-0.0067) \cdot 0.2 = -0.0001$$

$$w_{0b} = w_{0b} + \alpha \cdot \delta_b \cdot 1 = 0.0 + 0.1 \cdot (0.0032) \cdot 1 = 0.0003$$

$$w_{1b} = w_{1b} + \alpha \cdot \delta_b \cdot x_1 = -0.1 + 0.1 \cdot (0.0032) \cdot 4.7 = -0.0985$$

$$w_{2b} = w_{2b} + \alpha \cdot \delta_b \cdot x_2 = 0.2 + 0.1 \cdot (0.0032) \cdot 3.2 = 0.2010$$

$$w_{3b} = w_{3b} + \alpha \cdot \delta_b \cdot x_3 = 0.1 + 0.1 \cdot (0.0032) \cdot 1.2 = 0.1004$$

$$w_{4b} = w_{4b} + \alpha \cdot \delta_b \cdot x_4 = -0.1 + 0.1 \cdot (0.0032) \cdot 0.2 = -0.0999$$

$$w_{0o} = w_{0o} + \alpha \cdot \delta_o \cdot 1 = 0.1 + 0.1 \cdot (-0.1333) \cdot 1 = 0.0867$$

$$w_{ao} = w_{ao} + \alpha \cdot \delta_o \cdot o_a = 0.2 + 0.1 \cdot (-0.1333) \cdot 0.5050 = 0.1933$$

$$w_{bo} = w_{bo} + \alpha \cdot \delta_o \cdot o_b = -0.1 + 0.1 \cdot (-0.1333) \cdot 0.5695 = -0.1076$$

Tất cả các trọng số này sẽ dùng cho các mẫu kế tiếp.

5.3 Mạng Neural

Ví dụ minh họa cho phương pháp lan truyền ngược

- Thử tính cho “cây Iris A” thêm một bước kế tiếp, có kết quả:

$$O_a = f (-0.0007 + 0.0969*4.7 + (-0.1021)*3.2 + (-0.1009)*1.3 + (-0.0001)*0.2) = f (-0.0032) = 0.4992$$

$$O_b = f (0.0003 + -(0.0985)*4.7 + 0.2010*3.2 + 0.1004*1.3 + (-0.0999)*0.2) = f (0.2911) = 0.5724$$

$$O_o = f (0.0867 + 0.1933*0.4992 + (-0.1076)*0.5724) = f (0.1440) = 0.5304$$

- Kết quả, ta thấy 0.5304 gần về 0 (kết quả mong muốn) so với kết quả của bước trước đó là 0.5359.

5.3 Mạng Neural

Tóm tắt giải thuật

Cho trước K mẫu dữ liệu : $\{(x^1, y^1), \dots, (x^K, y^K)\}$ với $x^k = (x_1^k, x_2^k, \dots, x_n^k)$ và $y^k \in \mathbb{R}$, $k = 1, \dots, K$.

Bước 1: Chọn trước giá trị $\alpha > 0$ và $E_{\max} > 0$.

Bước 2: Khởi tạo ngẫu nhiên w , bắt đầu với mẫu thứ nhất $k = 1$ và gán sai lệch $E = 0$.

Bước 3: Bắt đầu quá trình học, gán $x = x^k$, $y = y^k$. Đầu ra của mạng neuron tính theo:

$$O^k = \frac{1}{1 + \exp(-W^T o^k)}$$

trong đó o^k là giá trị ra của lớp ẩn, được tính bởi:
$$o_l^k = \frac{1}{1 + \exp(-w_l^T x_k)}$$

Bước 4: Cập nhật trọng số neuron đầu ra: $W = W + \alpha \delta_o$ với $\delta = (y - O)O(1 - O)$

Bước 5: Cập nhật trọng số của neuron ẩn $w_l = w_l + \alpha \delta W_l o_l (1 - o_l) x$ (cho $l = 1, \dots, L$)

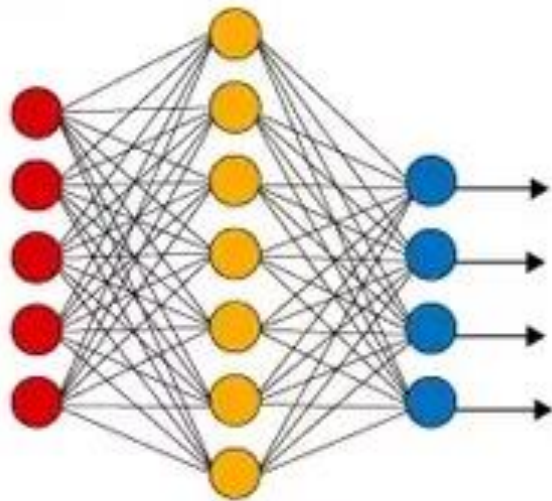
Bước 6: Tính sai lệch bằng cách cộng thêm sai lệch hiện tại $E = E + \frac{1}{2}(y - O)^2$

Bước 7: Nếu $k < K$ thì $k = k + 1$ và trở lại Bước 3. Nếu không thì qua Bước 8.

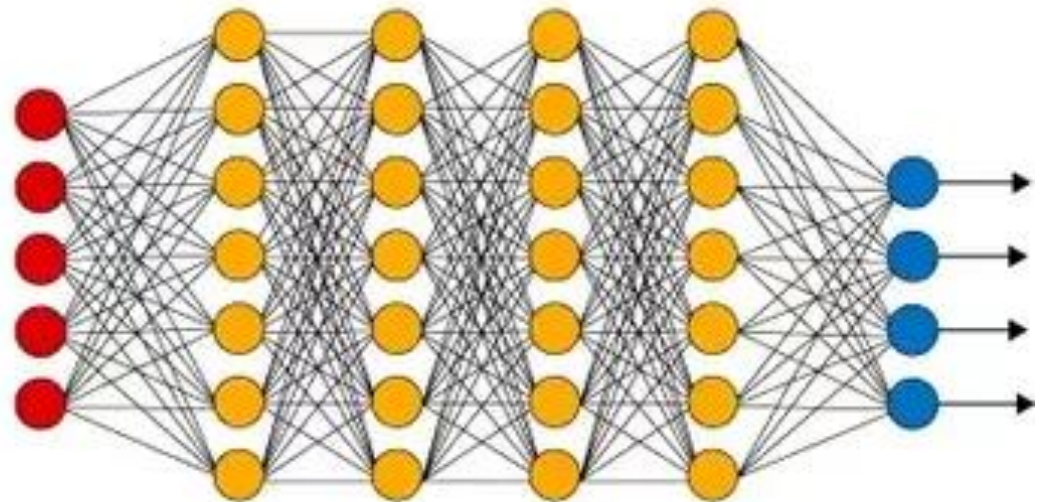
Bước 8: Nếu $E \leq E_{\max}$ thì kết thúc quá trình học. Còn nếu $E > E_{\max}$ gán $E = 0$ và bắt đầu một chu kỳ học mới bằng cách trở lại Bước 3.

5.4 Deep Learning

Simple Neural Network



Deep Learning Neural Network



● Input Layer

● Hidden Layer

● Output Layer

5.5 Random Forest

Random Forest Simplified

