

ARTIFICIAL INTELLIGENCE

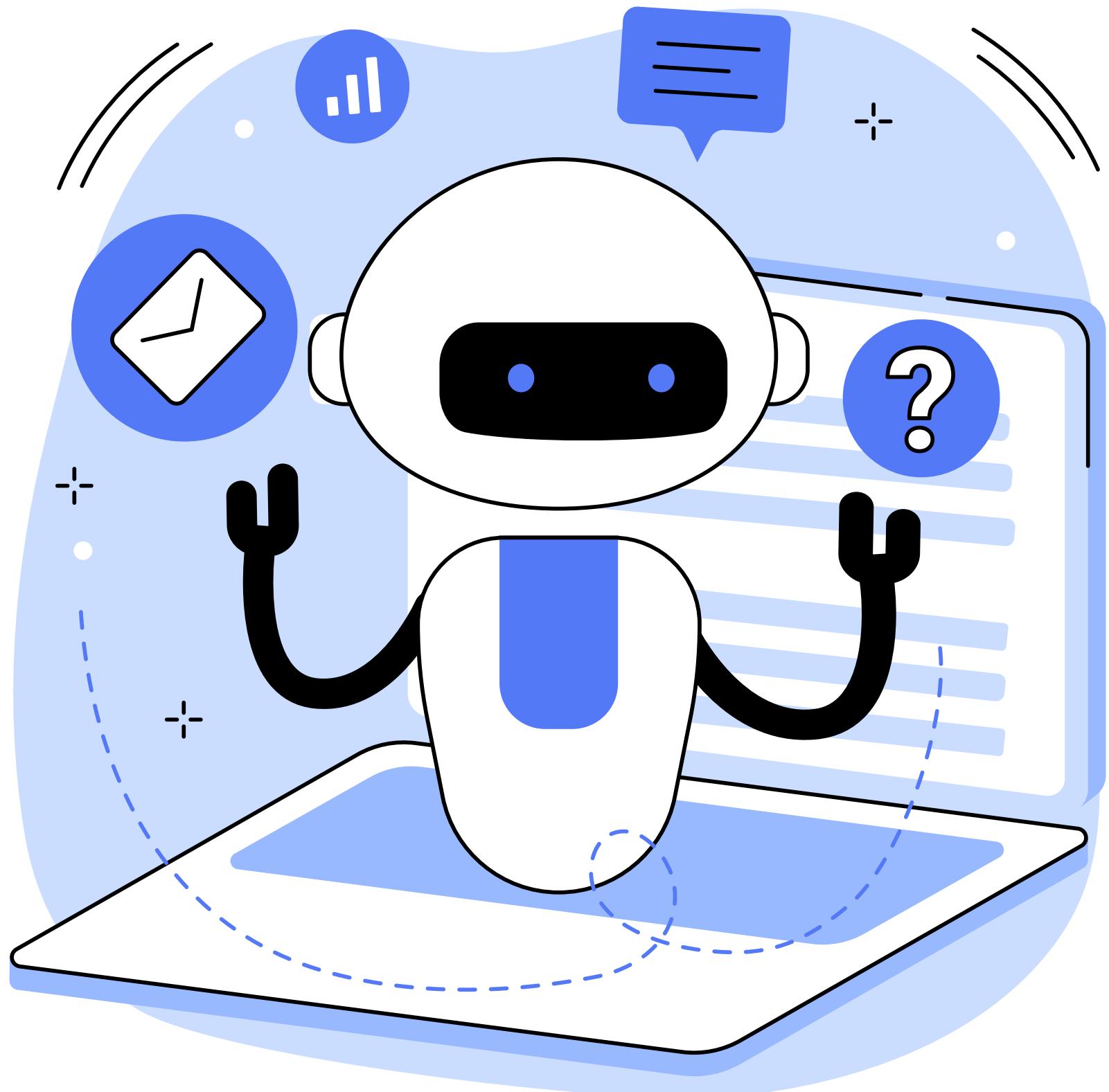
Learning:

- *Learning from Observations.*
- *Learning in Neural and Belief network.*



Học tập là gì?

- Học tập (Learning) là quá trình tiếp thu sự hiểu biết, kiến thức, hành vi, kỹ năng, giá trị, ... Khả năng học hỏi được thấy ở con người, động vật và một số máy móc.
- Trong bối cảnh trí tuệ nhân tạo AI, học tập là quá trình một hệ thống hoặc tác nhân cải thiện hiệu suất hoặc kiến thức của mình dựa trên kinh nghiệm và dữ liệu. Quá trình này giúp điều khiển hành vi hoặc đưa ra các quyết định tốt hơn khi đối mặt với các tình huống được đặt ra.



I. Learning from Observations



1 Thế nào là học tập từ quan sát?

Learning from Observations là một phương pháp học hỏi trong đó các kiến thức được tạo ra thông qua việc quan sát và xử lý dữ liệu thu thập từ thế giới thực. Trong Learning from Observations, không có sự can thiệp trực tiếp từ một người lập trình hoặc từ các quy tắc được chỉ định trước. Thay vào đó, các thuật toán hoặc mô hình học máy được sử dụng để phân tích dữ liệu và rút ra các mẫu, quy luật hoặc kiến thức từ dữ liệu đó.

Trong Machine Learning, Learning from Observations có các yếu tố chính bao gồm các tác nhân học tập (Learning agents), phương pháp học quy nạp (Inductive learning) và phương pháp cây quyết định (Decision tree learning).

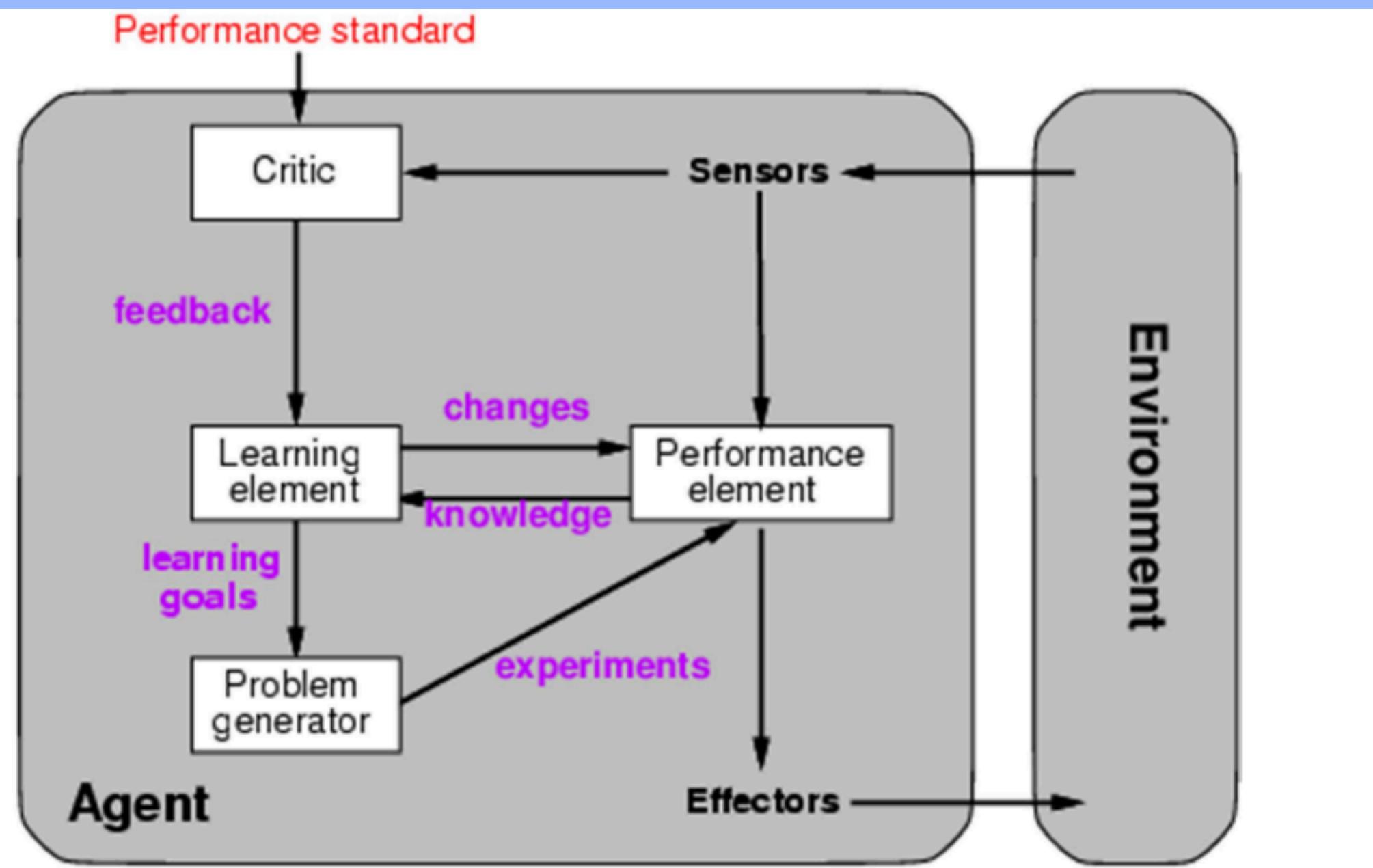


2

Các tác nhân học tập (Learning agents)

Một learning agent là một hệ thống tự động hoạt động trong một môi trường, tự học và cải thiện hiệu suất của mình thông qua kinh nghiệm.

Cụ thể, một learning agent bao gồm các thành phần sau:



Environment (Môi trường): Không gian mà Learning agent hoạt động và tương tác.

Sensors (Cảm biến): Là các thiết bị hoặc công cụ mà learning agent sử dụng để quan sát trạng thái của môi trường.

Effectors (Công cụ thực thi): Là các phương tiện mà learning agent sử dụng để tương tác với môi trường.

Learning Element (Bộ phận học tập): Đây là trí tuệ của learning agent, nơi mà kiến thức và kinh nghiệm được tích lũy và sử dụng để cải thiện hiệu suất trong tương lai.

Critic (Bộ phận đánh giá) không thực sự thực hiện hành động như Effectors, mà chỉ đánh giá các hành động dựa trên hiểu biết về môi trường và mục tiêu của hệ thống.

Một "problem generator" được thiết kế để tạo ra các bài toán hoặc tình huống thử thách để kiểm tra hoặc đào tạo hệ thống.

Performance Element (Yếu tố hiệu suất): Là phần đo lường và đánh giá hiệu suất của các hành động và quyết định.

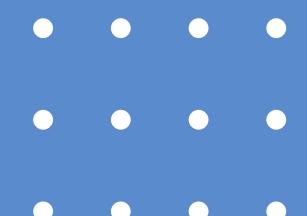
⋮ ⋮ ⋮ Thiết kế của một bộ phận học tập (Learning Element) bị ảnh hưởng bởi :



- Những thành phần nào của yếu tố hiệu suất cần được học.
- Những phản hồi nào có sẵn để tìm hiểu những điều này các thành phần.
- Kiểu biểu diễn (Representation) nào được sử dụng cho các thành phần.

Các loại phản hồi (Feedback)

- Học có giám sát (Supervised learning): đưa ra câu trả lời đúng cho mỗi câu hỏi.
- Học không giám sát (Unsupervised learning): không đưa ra câu trả lời đúng
- Học tăng cường (Reinforcement learning): đưa ra một tín hiệu phần thưởng (reward signal).



3

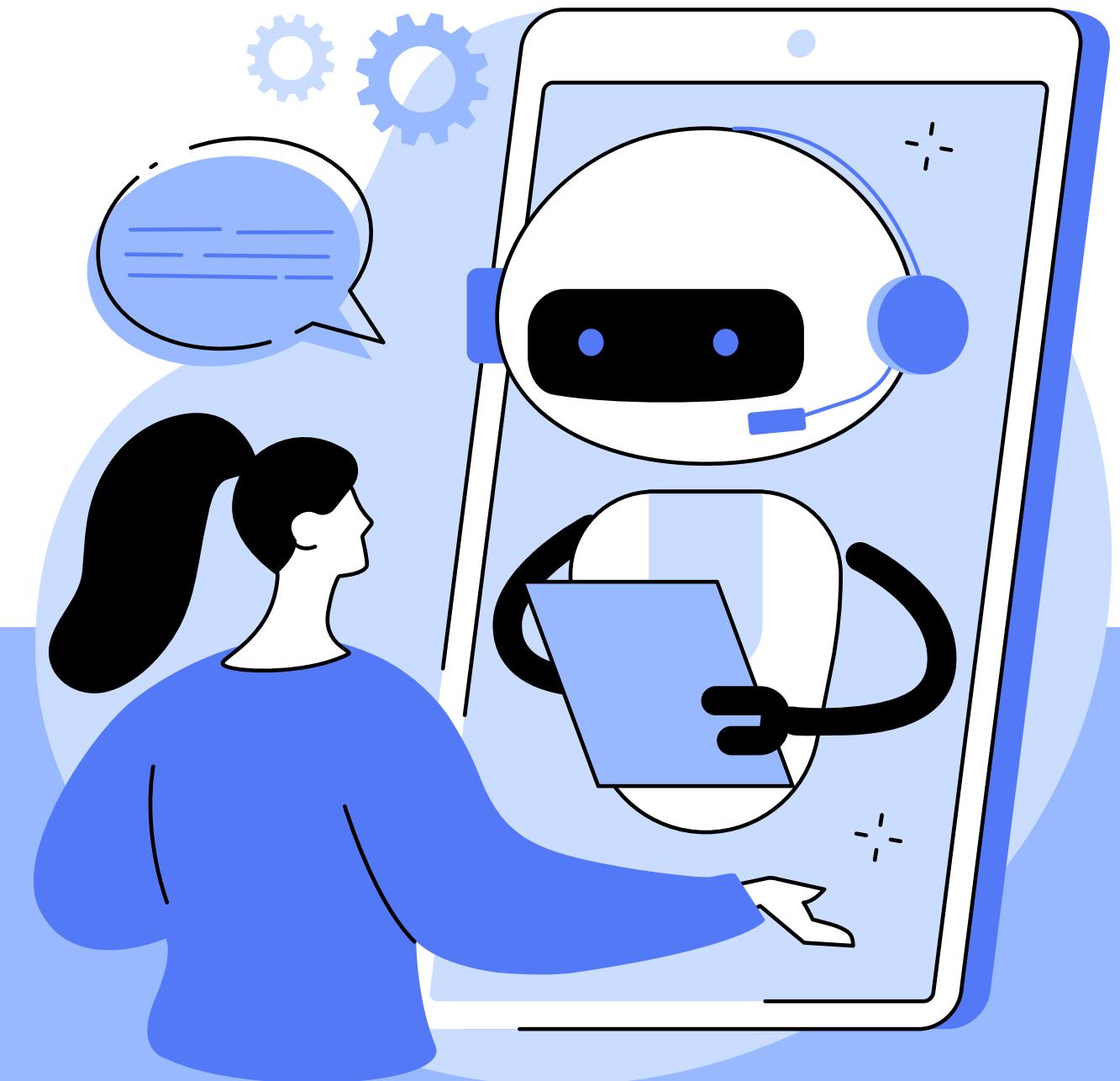
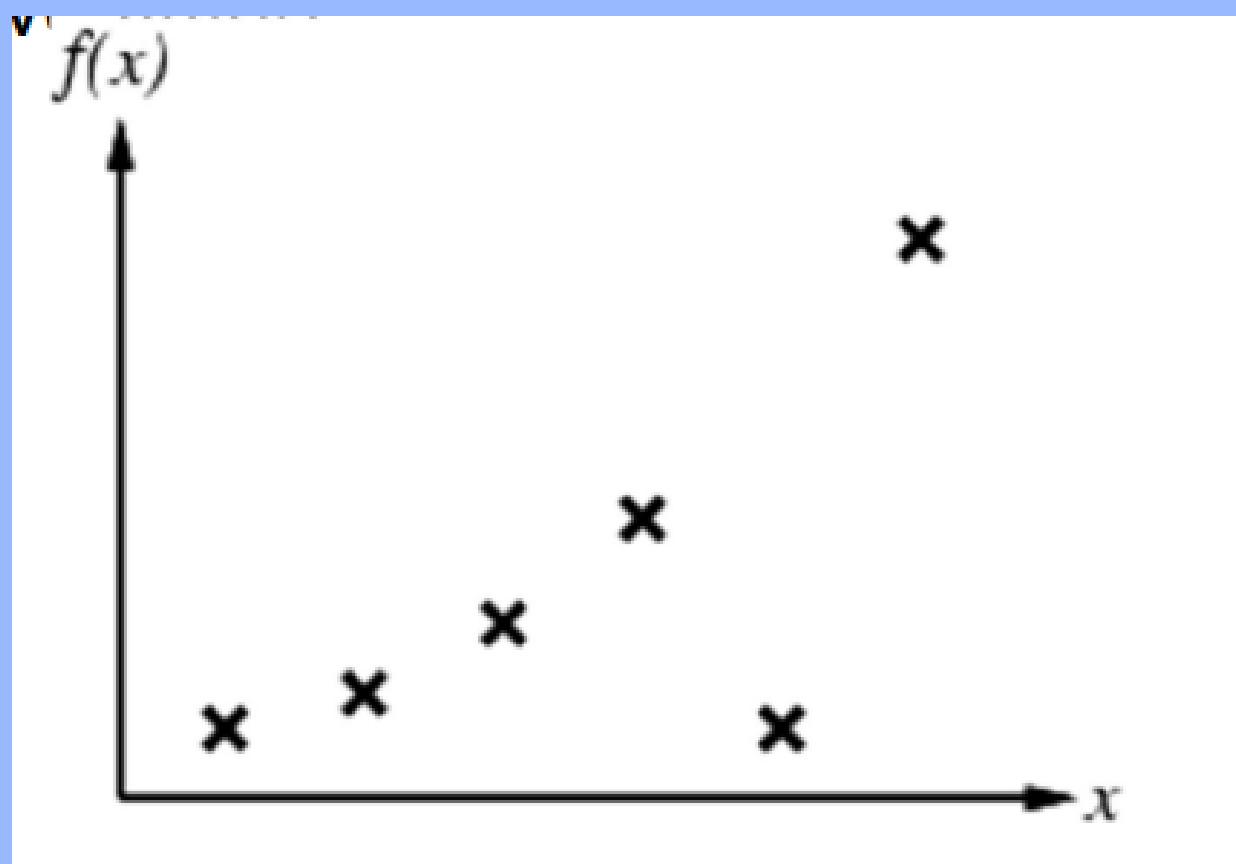
Inductive learning.

Inductive learning (Học quy nạp) là một phương pháp học máy mà học viên học từ dữ liệu huấn luyện để tạo ra một mô hình hoặc các quy tắc tổng quát mà có thể áp dụng cho dữ liệu mới mà nó chưa được huấn luyện. Điều quan trọng trong phương pháp này là tạo ra các dự đoán chính xác cho dữ liệu mới mà chưa được thấy trong dữ liệu huấn luyện.

Ta nghiên cứu bài toán : Xác định hàm số từ các giá trị x tương ứng cho trước cho trước.

Với $f(x)$ là hàm số cần tìm.

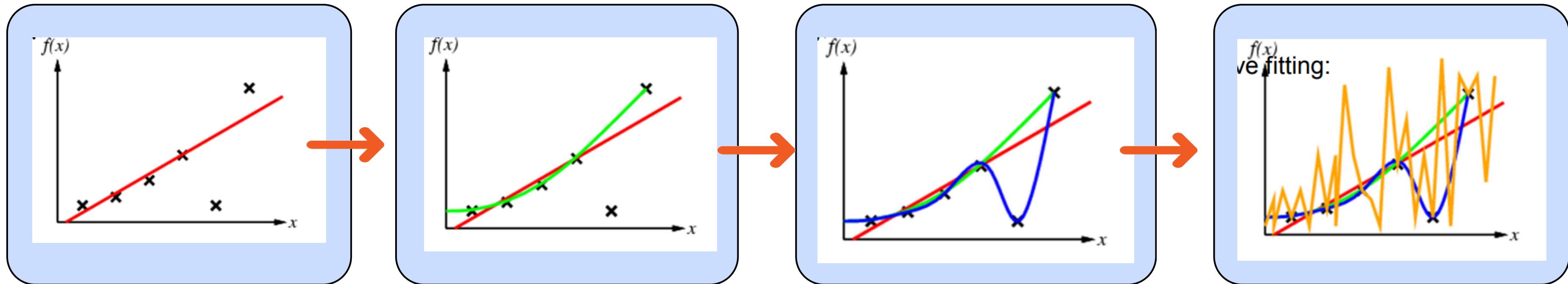
Dựa vào các giá trị $f(x)$ ứng với x co trước xuất hiện, ta sử dụng phương pháp nối liền các giá trị x và lập phương trình dựa vào đồ thị mà nó đi qua.



Đây là một mô hình học tập thực sự được đơn giản hóa cao độ:

- Bỏ qua những kiến thức đã có.
- Giả sử đưa ra ví dụ từ dữ liệu thu thập được trong bài toán.

Ta xây dựng được các trường hợp sau:



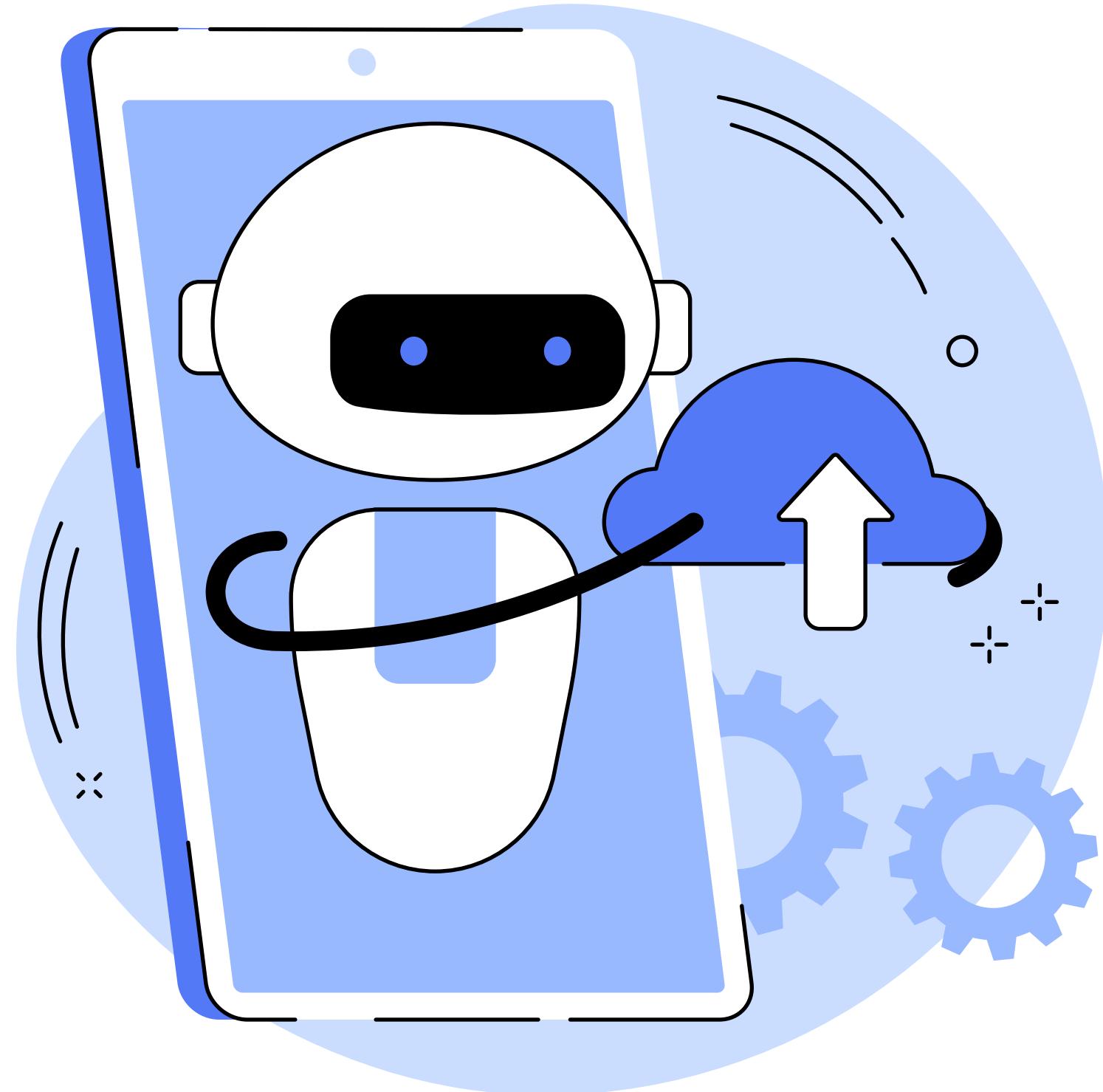
Theo nguyên lý đơn giản hóa của Ockham, chúng ta sẽ chọn giả thuyết đơn giản nhất phù hợp với dữ liệu thu thập được.

4

Decision tree learning

Learning decision trees là quá trình huấn luyện một loại mô hình dự đoán trong học máy được gọi là cây quyết định (decision tree). Cây quyết định là một cấu trúc cây có thể được sử dụng để đưa ra các quyết định dựa trên các quy tắc học từ dữ liệu

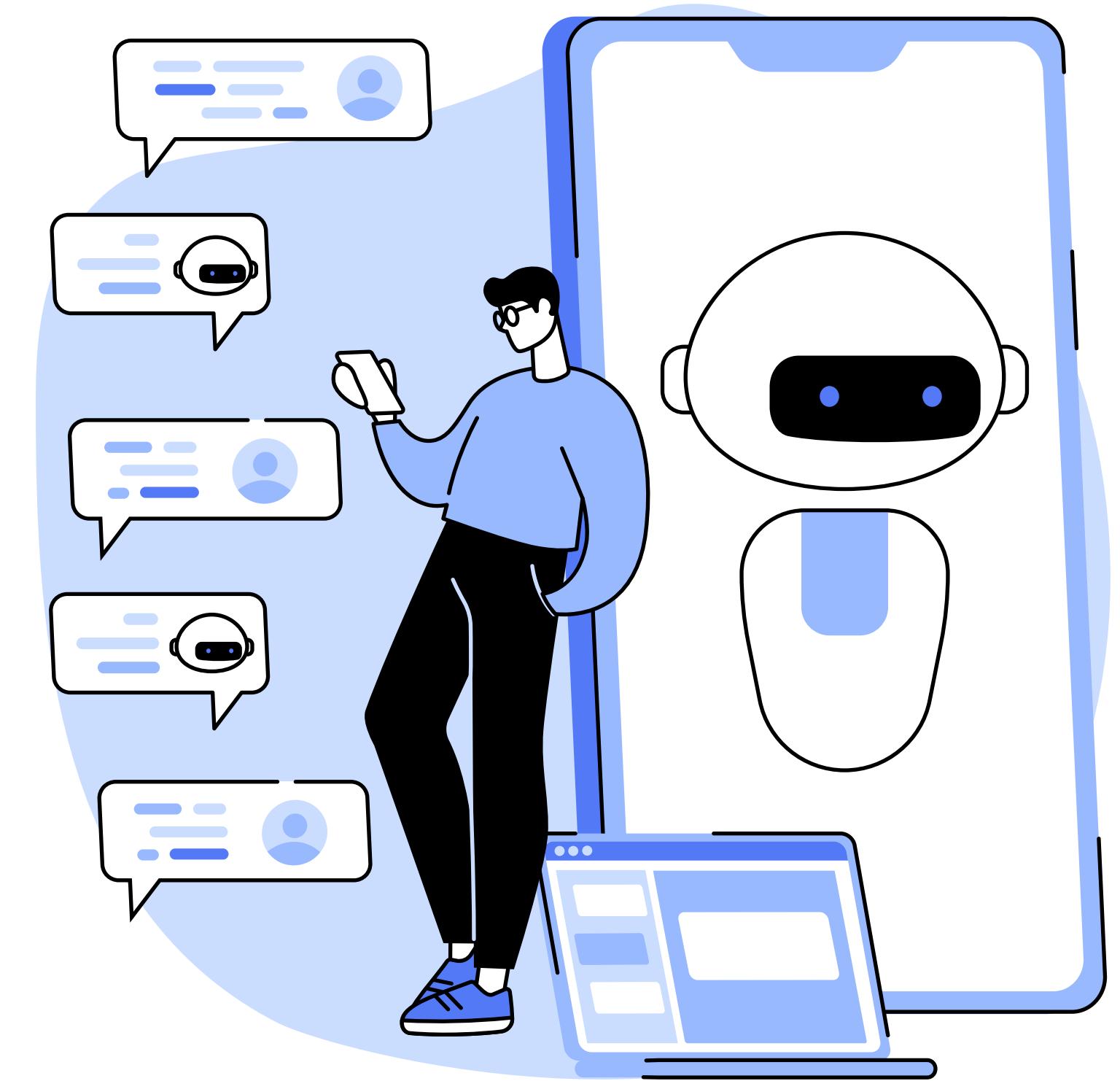
Quá trình huấn luyện cây quyết định bao gồm việc phân chia tập dữ liệu thành các phần nhỏ hơn dựa trên các thuộc tính hoặc đặc trưng của dữ liệu. Mục tiêu là tìm ra các quy tắc hoặc "câu hỏi" để phân loại dữ liệu một cách hiệu quả nhất. Quy trình này được tiếp tục đệ quy cho đến khi mỗi nhánh của cây đều chứa các điều kiện dừng (ví dụ: tất cả các điểm dữ liệu trong một nhóm đều thuộc về cùng một lớp).



Ta phân tích vấn đề sau: Có nên đợi bàn ở nhà hàng hay không?

Dựa trên các tiêu chí:

1. Thay thế: có nhà hàng thay thế nào gần đó không?
2. Quầy bar: Có khu vực quầy bar nào thoải mái để chờ không?
3. Thứ Sáu/Thứ Bảy: hôm nay là thứ Sáu hay thứ Bảy?
4. Đói: chúng ta có đói không?
5. Khách hàng (Patrons): số lượng người trong nhà hàng (None, Some, Full)?
6. Giá: khoảng giá (\$, \$\$, \$\$\$)?
7. Raining: ngoài trời có mưa không?
8. Đặt chỗ: chúng ta đã đặt chỗ chưa?
9. Kiểu nhà hàng: loại hình nhà hàng (Pháp, Ý, Thái, Burger)?
10. Thời gian đợi: thời gian chờ ước tính (0-10, 10-30, 30-60, >60)?



Biểu diễn dựa trên thuộc tính

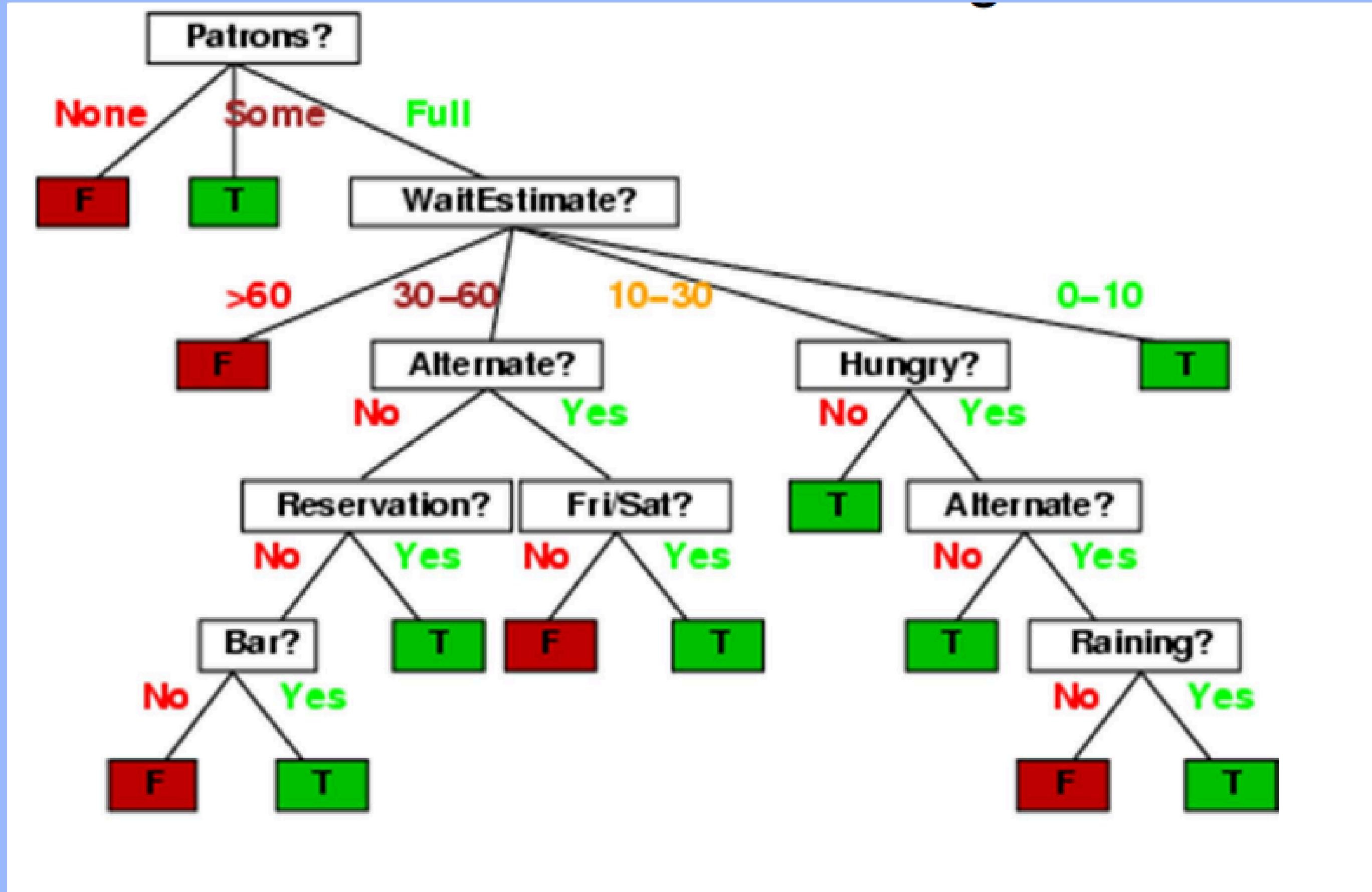
Ví dụ mô tả bằng giá trị thuộc tính (Boolean, rời rạc, liên tục).

Giả định những tình huống tôi sẽ/không đợi bàn:

Example	Attributes										Target Wait
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
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

Giá trị được đưa dưới dạng Positive (T) hoặc Negative (F).

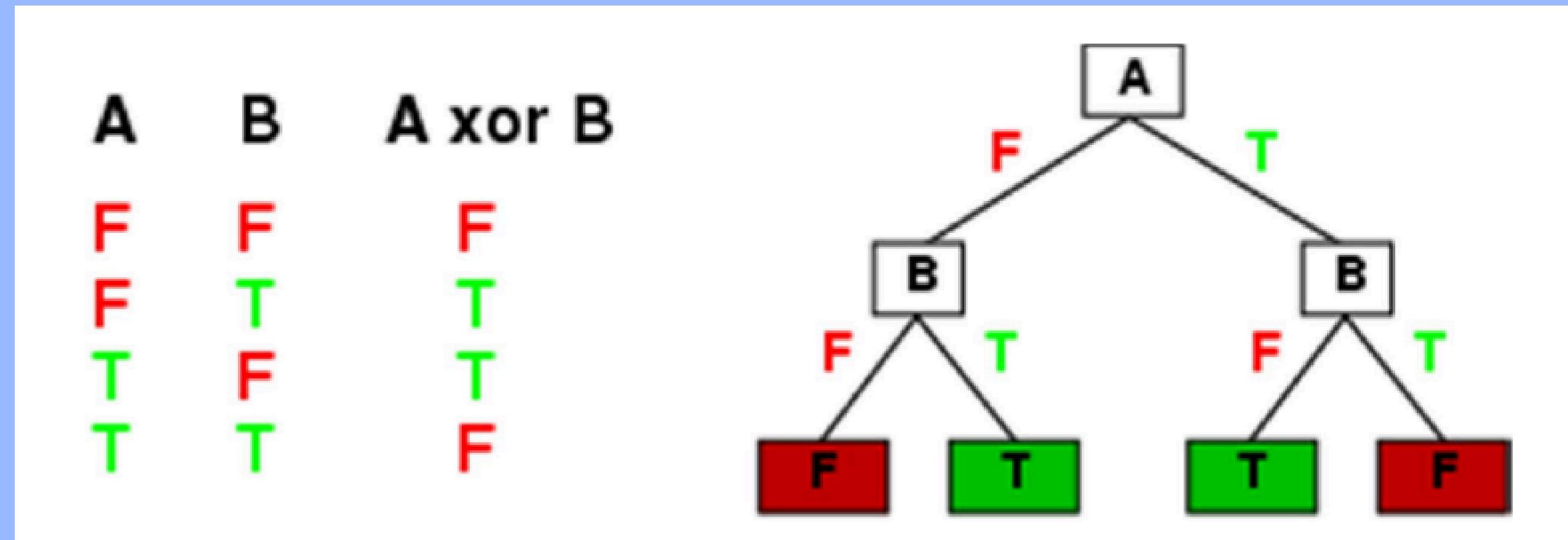
Ta lập được sơ đồ cây (Decision trees) như sau



Giải thích

Cây quyết định có thể thể hiện bất kỳ chức năng nào của thuộc tính đầu vào.

Ví dụ: đối với các hàm Boolean, hàng của bảng chân lý → đường dẫn đến các nhánh.



Điều đáng chú ý là có một cây quyết định nhất quán cho bất kỳ tập huấn luyện (training set) nào đó sẽ tạo một đường dẫn tới nhánh cho mỗi ví dụ (trừ khi hàm $f(x)$ không xác định) nhưng có lẽ nó sẽ không tạo ra các ví dụ mới gây phức tạp bài toán.

Chúng ta ưu tiên tìm cây quyết định nhỏ gọn hơn.

Vấn đề tiếp theo được nhắc tới là:

1. “Có bao nhiêu giả thuyết có thể xảy ra “

Ta lập được bao nhiêu cây quyết định riêng biệt với n thuộc tính Boolean?

= số lượng hàm Boolean

= số bảng chân lý phân biệt có 2^n hàng = 2^{2n}

Ví dụ: với 6 thuộc tính Boolean, có
18.446.744.073.709.551.616 cây

2. Có bao nhiêu giả thuyết liên kết thuận túy (ví dụ, Đói ^ Mưa)?

Mỗi thuộc tính có thể in (Positive), in (Negative) hoặc out

=> 3^n giả thuyết liên kết riêng biệt

Không gian giả thuyết mở rộng hơn

– Tăng khả năng hàm mục tiêu có thể được thể hiện

– Tăng số lượng giả thuyết phù hợp với tập huấn luyện

=> có thể nhận được những dự đoán tồi tệ hơn



Decision tree learning

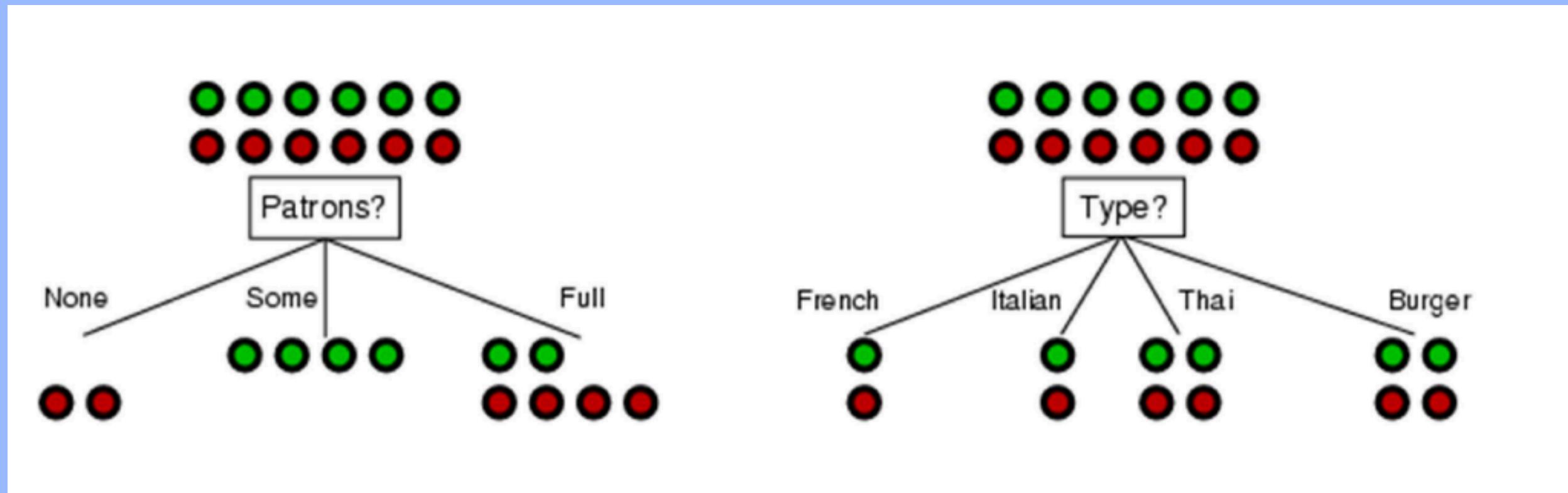
Mục tiêu: tìm một cây nhở phù hợp với các ví dụ đào tạo

Ý tưởng: (đệ quy) chọn thuộc tính “quan trọng nhất” làm gốc của cây phụ/nhánh.

```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
    best ← CHOOSE-ATTRIBUTE(attributes, examples)
    tree ← a new decision tree with root test best
    for each value  $v_i$  of best do
      examplesi ← {elements of examples with best =  $v_i$ }
      subtree ← DTL(examplesi, attributes – best, MODE(examples))
      add a branch to tree with label  $v_i$  and subtree subtree
  return tree
```

Chọn một thuộc tính

Ý tưởng: một thuộc tính tốt sẽ chia các ví dụ thành các tập con đó là (lý tưởng nhất) "All positive" hoặc "All negative"



Vậy, việc lựa chọn theo Patrons sẽ tối ưu hơn bởi:

- Ít phương án lựa chọn hơn.
- Việc chọn lựa rõ ràng hơn.
- So sánh được giữa các khả năng có thể xảy ra.

Sử dụng lý thuyết về thông tin (Information theory)

=> Để triển khai việc “ Chọn thuộc tính ” trong thuật toán DTL (Decision Tree Learning)

Information Content (Entropy)

Công thức tính Entropy của một tập dữ liệu với các lớp C_1, C_2, \dots, C_n có xác suất tương ứng là p_1, p_2, \dots, p_n là:

$$\text{Entropy}(S) = - \sum_{i=1}^n p_i \log_2(p_i)$$

Trong đó:

- S là tập dữ liệu hoặc phân phối lớp cần tính Entropy.
- p_i là xác suất của lớp C_i .

Đối với tập huấn luyện chứa p ví dụ positive và n ví dụ negative

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

Ta thu được các thông tin sau

Thuộc tính được chọn A chia tập huấn luyện E thành các tập con E1, ..., Ev theo giá trị của chúng đối với A, trong đó A có v giá trị khác nhau.

$$remainder(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

Thông tin nhận được Information Gain (IG) hoặc giảm entropy từ kiểm tra thuộc tính:

$$IG(A) = I\left(\frac{p}{p + n}, \frac{n}{p + n}\right) - remainder(A)$$

=> Chọn thuộc tính có IG lớn nhất

Ví dụ, đối với tập huấn luyện, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

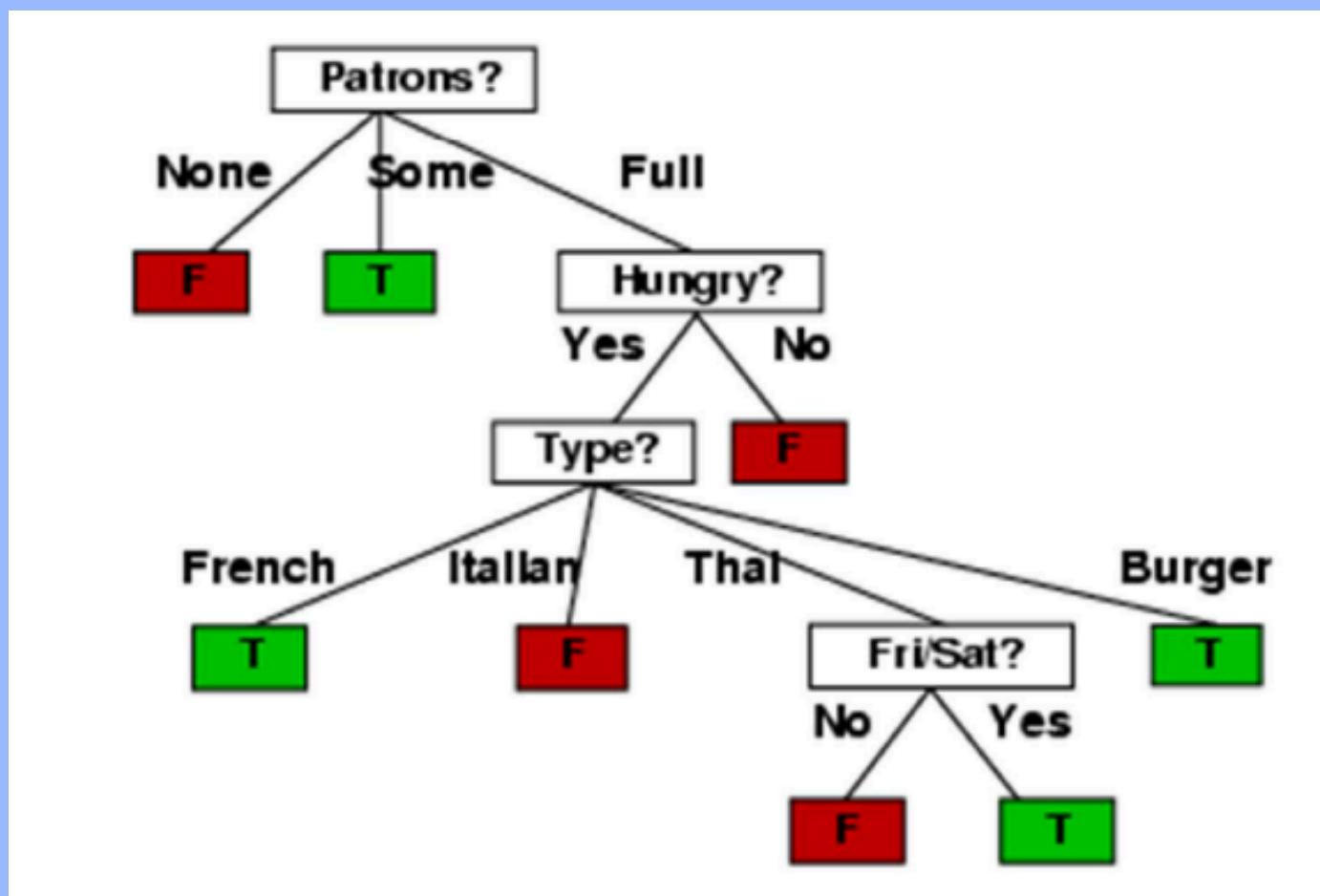
Hãy xem xét các thuộc tính của Patrons và Types (Slide 15) (và cả các thuộc tính khác nữa):

$$IG(Patrons) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .0541 \text{ bits}$$

$$IG(Type) = 1 - \left[\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0 \text{ bits}$$

=> Chọn Patrons vì có IG lớn nhất

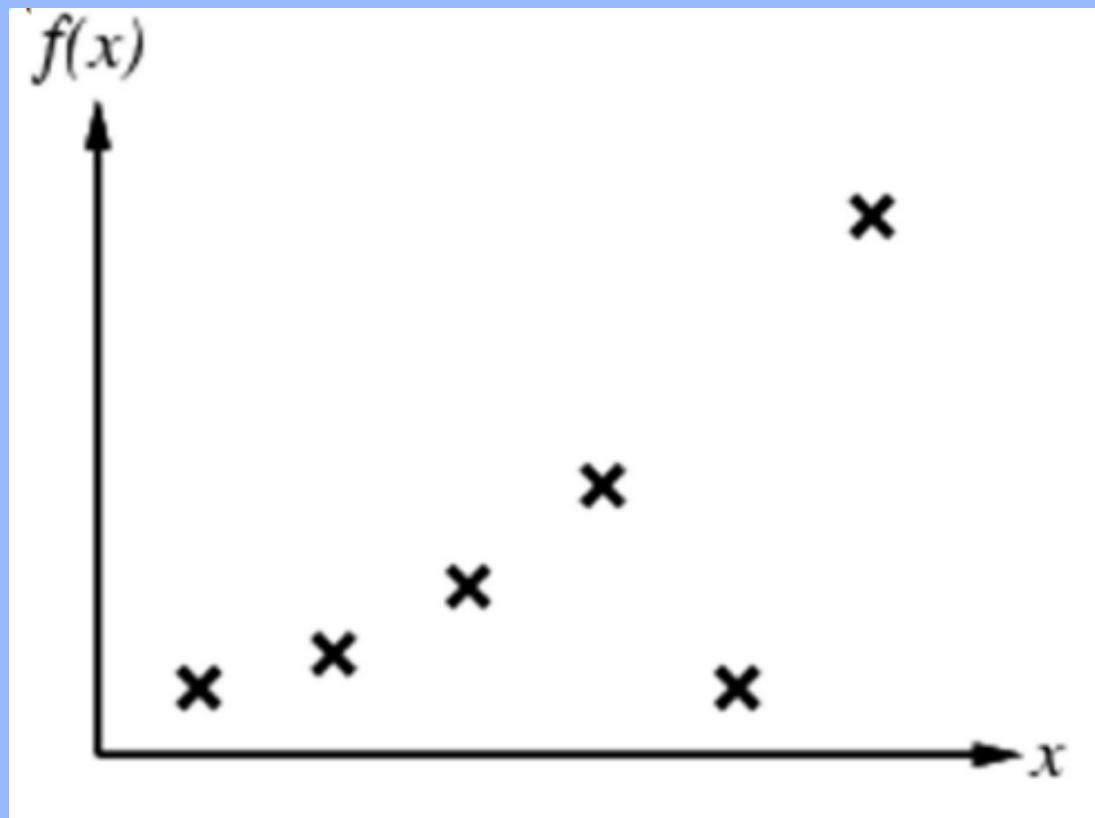
Ví dụ tiếp Cây quyết định rút ra từ 12 ví dụ:



Về cơ bản, cây quyết định đơn giản hơn cây tổng quát từ ban đầu phức tạp hơn, dẫn tới kết quả nhanh hơn. Song giả thuyết không được chứng minh bằng lượng nhỏ dữ liệu mà phải sử dụng tất cả dữ liệu như ban đầu

Đo lường hiệu suất (Performance measurement)

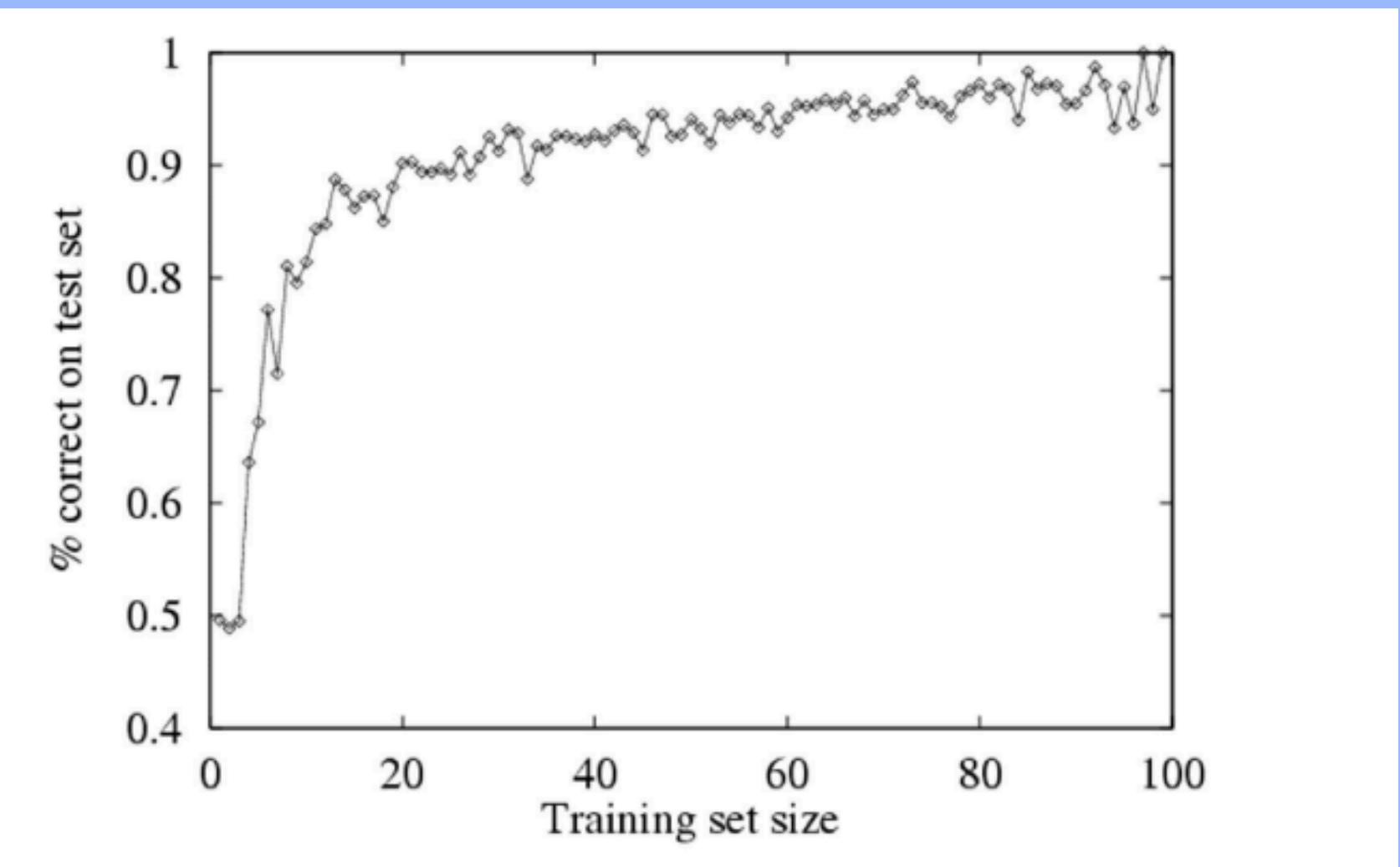
Quay lại bài toán Learning Agents, ta có vấn đề như sau “ Tìm giả thuyết cho h sao cho $h \approx f$ ”



Đường cong học được (Learning Curve) = % đúng trên tập kiểm tra như một hàm của kích thước tập huấn luyện

Vậy làm sao chúng ta biết được $h \approx f$?

1. Vận dụng các định lý của lý thuyết học tính toán/thống kê.
2. Thử h trên một tập hợp các ví dụ kiểm tra mới (sử dụng cùng một phân phối trên không gian mẫu như tập huấn luyện).

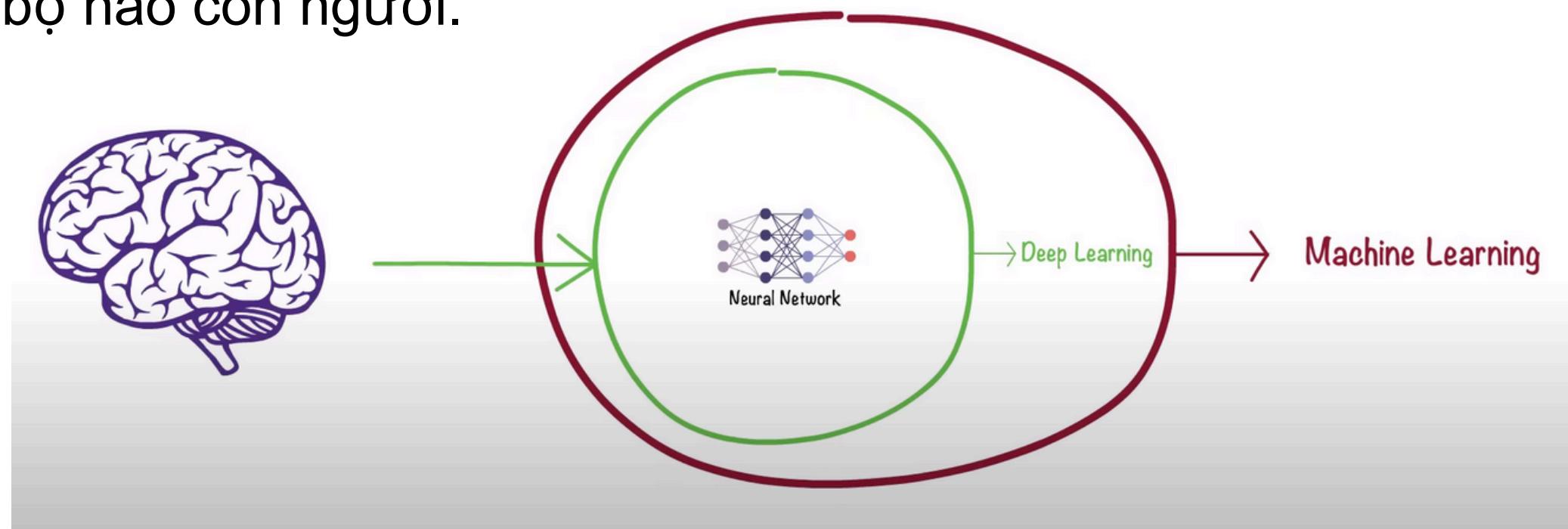


II. LEARNING IN NEURAL NETWORKS.

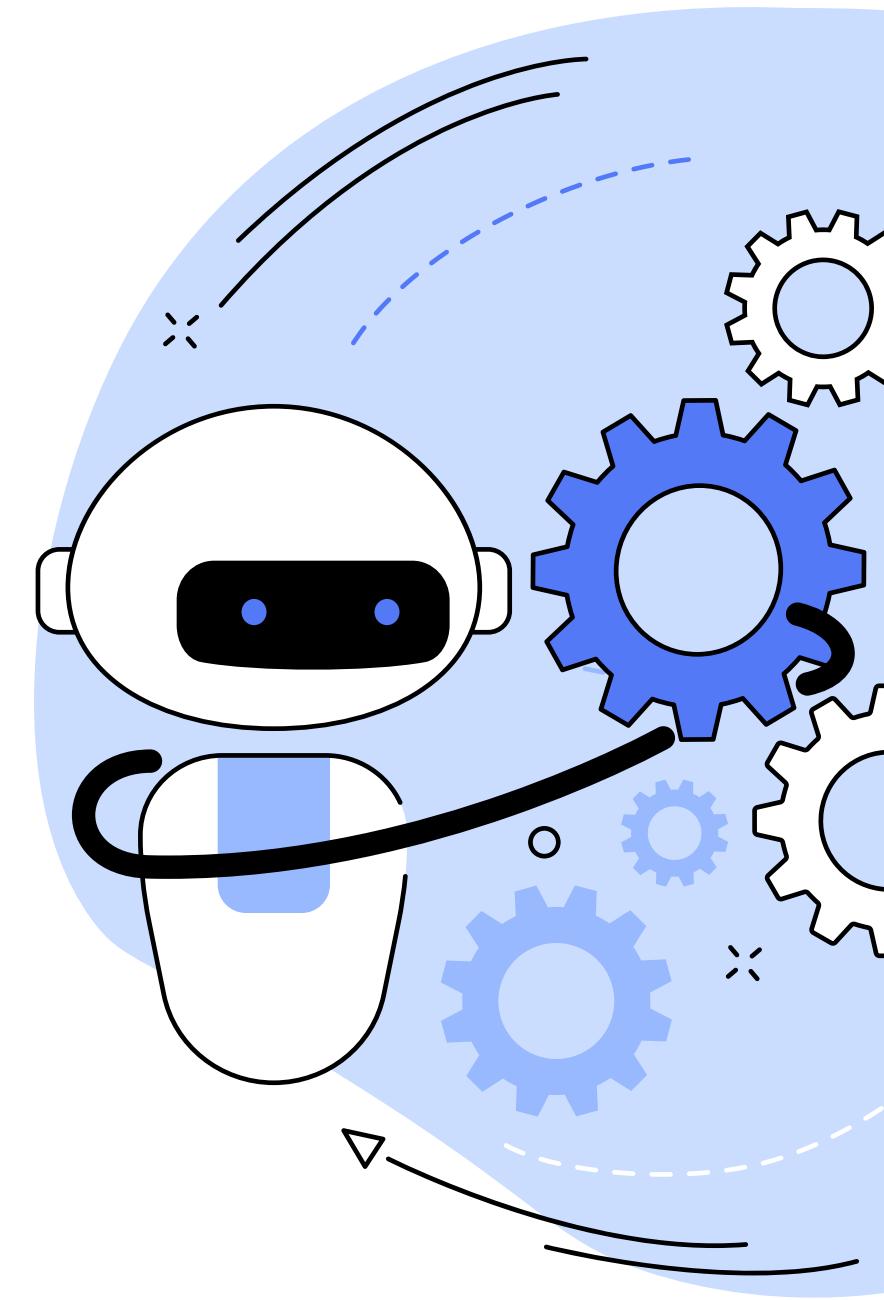
1. Neural Network là gì?

"Learning in neural networks" (học trong mạng thần kinh) là quá trình mà một mạng neural được huấn luyện để hiểu và tự động hóa một nhiệm vụ cụ thể thông qua dữ liệu mẫu.

Từ góc độ sinh học, chương này trình bày về mô hình toán học cho hoạt động của bộ não con người.



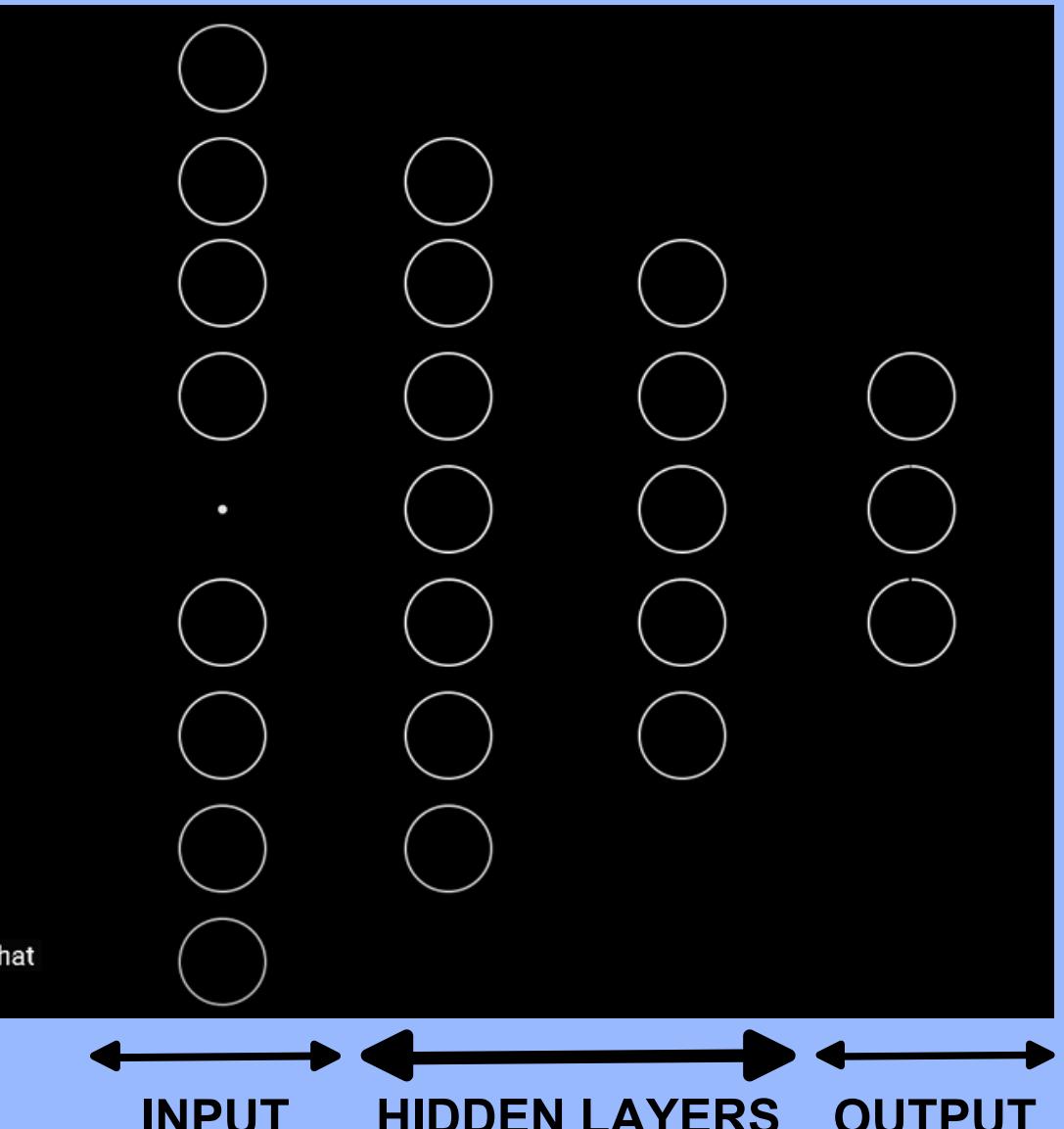
Mạng neural thường trông giống như hệ thần kinh của con người, các dữ liệu được liên kết chặt chẽ với nhau thông qua các con đường.



Neural Network vận hành như thế nào?

Mạng neural hoạt động bằng cách mô phỏng cách não bộ xử lý thông tin. Một mạng neural thường bao gồm một số lớp (layers) của các neuron được tổ chức thành các cấu trúc phức tạp, và mỗi neuron kết nối với các neuron trong các lớp khác thông qua các trọng số.

Dưới đây là cách một mạng neural hoạt động theo một cách tổng quan:



1. Đầu vào (Input):

Dữ liệu đầu vào được truyền vào mạng neural. Mỗi điểm dữ liệu thường được biểu diễn dưới dạng một vector đặc trưng, với mỗi thành phần của vector tương ứng với một thuộc tính hoặc đặc điểm của dữ liệu.

2. Lan truyền tín hiệu (Forwardpropagation):

- Dữ liệu đầu vào được truyền qua mạng neural từ lớp đầu vào (input layer) đến lớp đầu ra (output layer) thông qua các lớp ẩn (hidden layers).
- Trong quá trình này, các trọng số được gán cho mỗi liên kết giữa các neuron, và tích tức của đầu vào và trọng số được tính toán để tạo ra một giá trị đầu ra cho mỗi neuron.

3. Tính toán (Activation Function):

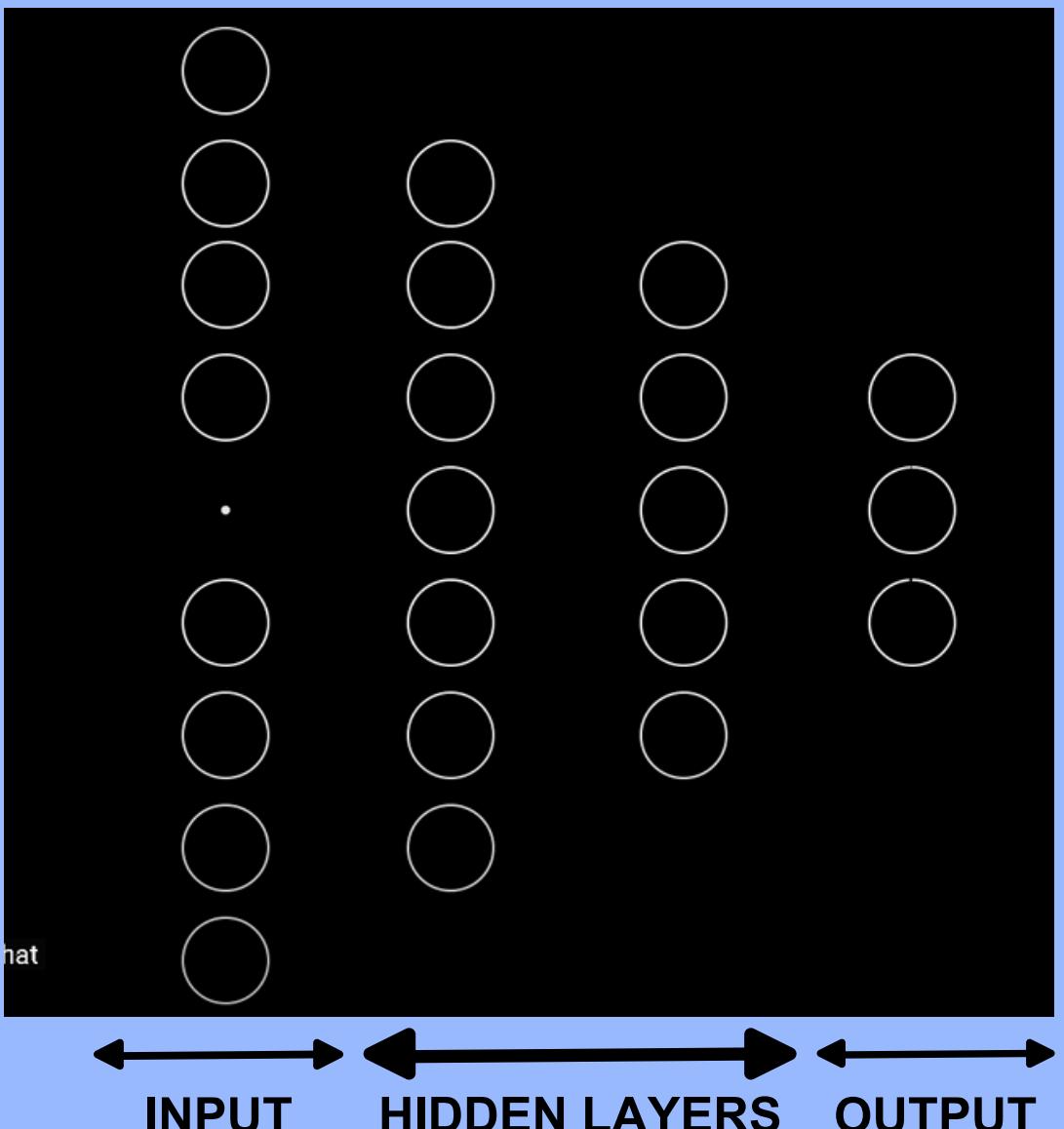
- Sau khi tích tức của đầu vào và trọng số được tính toán, một hàm kích hoạt được áp dụng cho mỗi giá trị đầu ra của neuron.
- Hàm kích hoạt thường là một hàm phi tuyến tính như hàm sigmoid, hàm tanh, hoặc hàm ReLU (Rectified Linear Unit), và nó giúp mạng neural có khả năng học các mô hình phức tạp.

2

Neural Network vận hành như thế nào?

Mạng neural hoạt động bằng cách mô phỏng cách não bộ xử lý thông tin. Một mạng neural thường bao gồm một số lớp (layers) của các neuron được tổ chức thành các cấu trúc phức tạp, và mỗi neuron kết nối với các neuron trong các lớp khác thông qua các trọng số.

Dưới đây là cách một mạng neural hoạt động theo một cách tổng quan:



=> Mạng neural hoạt động như vậy để học và áp dụng các mô hình phức tạp, từ việc nhận dạng hình ảnh đến dự đoán giá cổ phiếu, và nó đã chứng minh được khả năng của mình trong nhiều lĩnh vực khác nhau của khoa học máy tính và trí tuệ nhân tạo.

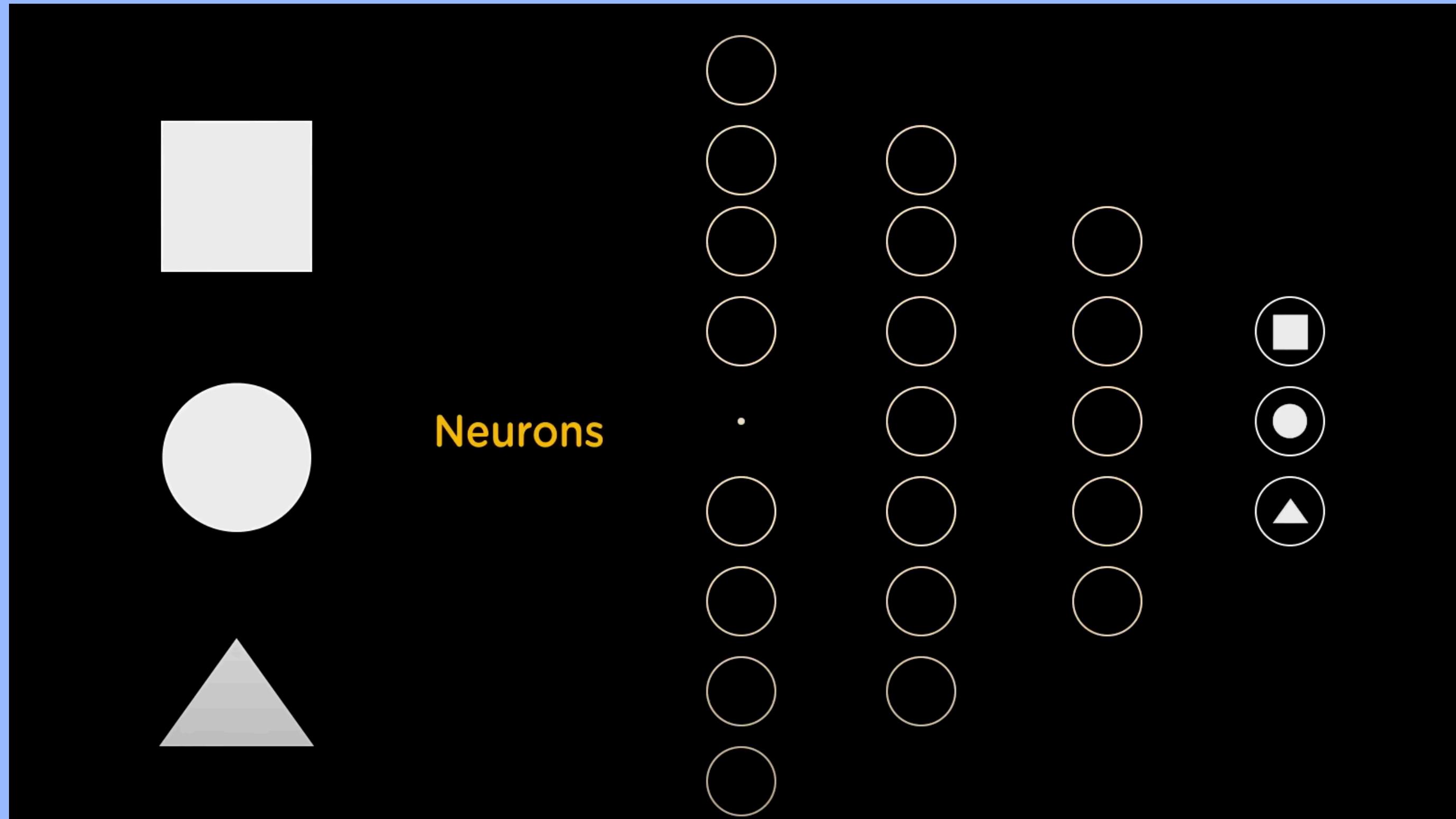
4. Lan truyền ngược (Backpropagation):

- Sau khi có được đầu ra từ mạng neural, sai số giữa đầu ra thực tế và đầu ra mong muốn (được gọi là lỗi) được tính toán.
- Sau đó, thuật toán lan truyền ngược được sử dụng để điều chỉnh các trọng số của mạng neural để giảm thiểu lỗi này.
- Quá trình này liên tục được lặp lại trên nhiều lần lặp lại (epochs) cho đến khi mạng neural đạt được một mức độ học tập chấp nhận được.

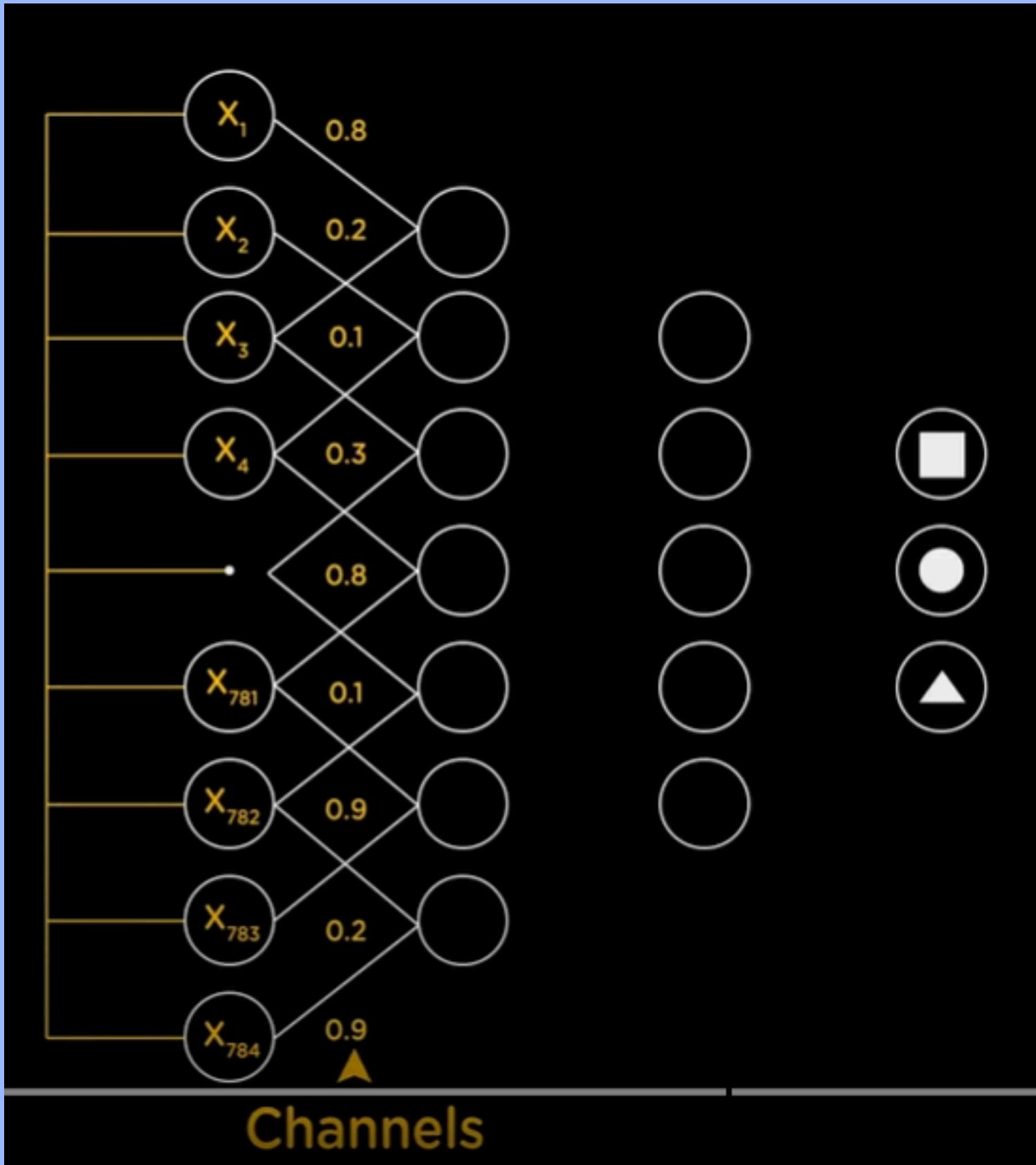
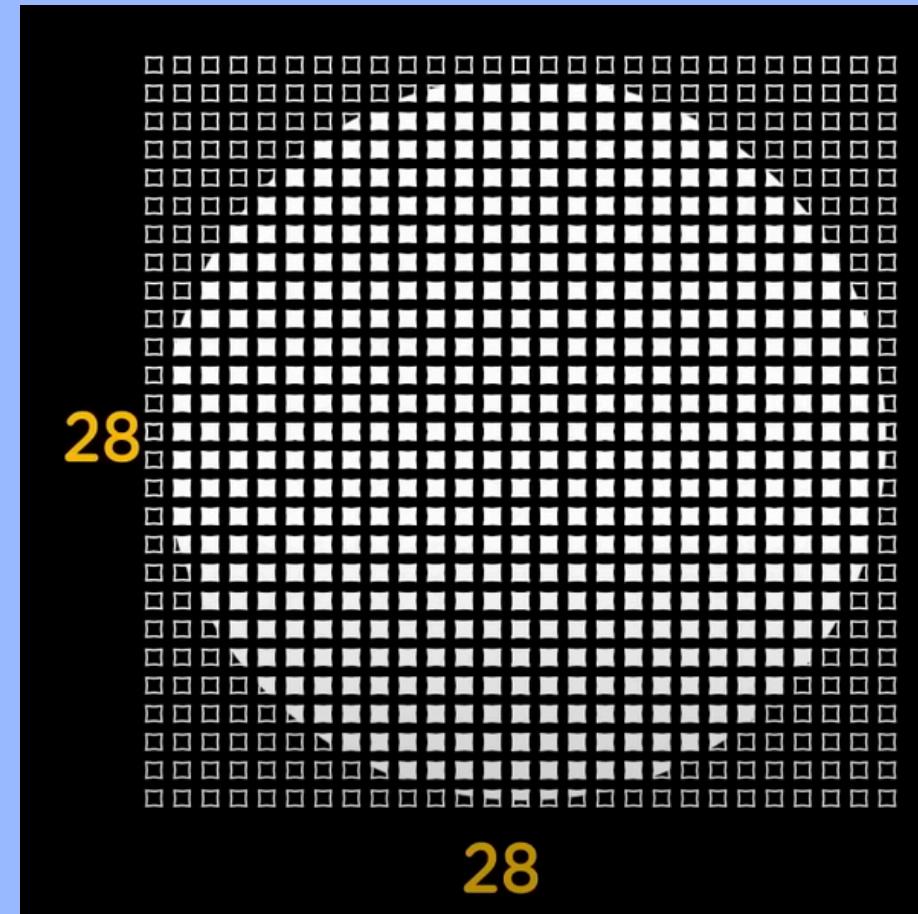
5. Đầu ra (Output):

- Sau khi mạng neural đã được huấn luyện, nó có thể được sử dụng để đưa ra dự đoán trên dữ liệu mới bằng cách truyền ngược và tính toán đầu ra từ các giá trị đầu vào mới.

Xét bài toán sau: Sử dụng Neural Network để phân biệt hình vuông, hình tròn và hình tam giác.



Neural Network
được tạo bởi các
Neurons, đây chính
là vị trí cốt lõi để xử
lí thông tin của
thuật toán này.

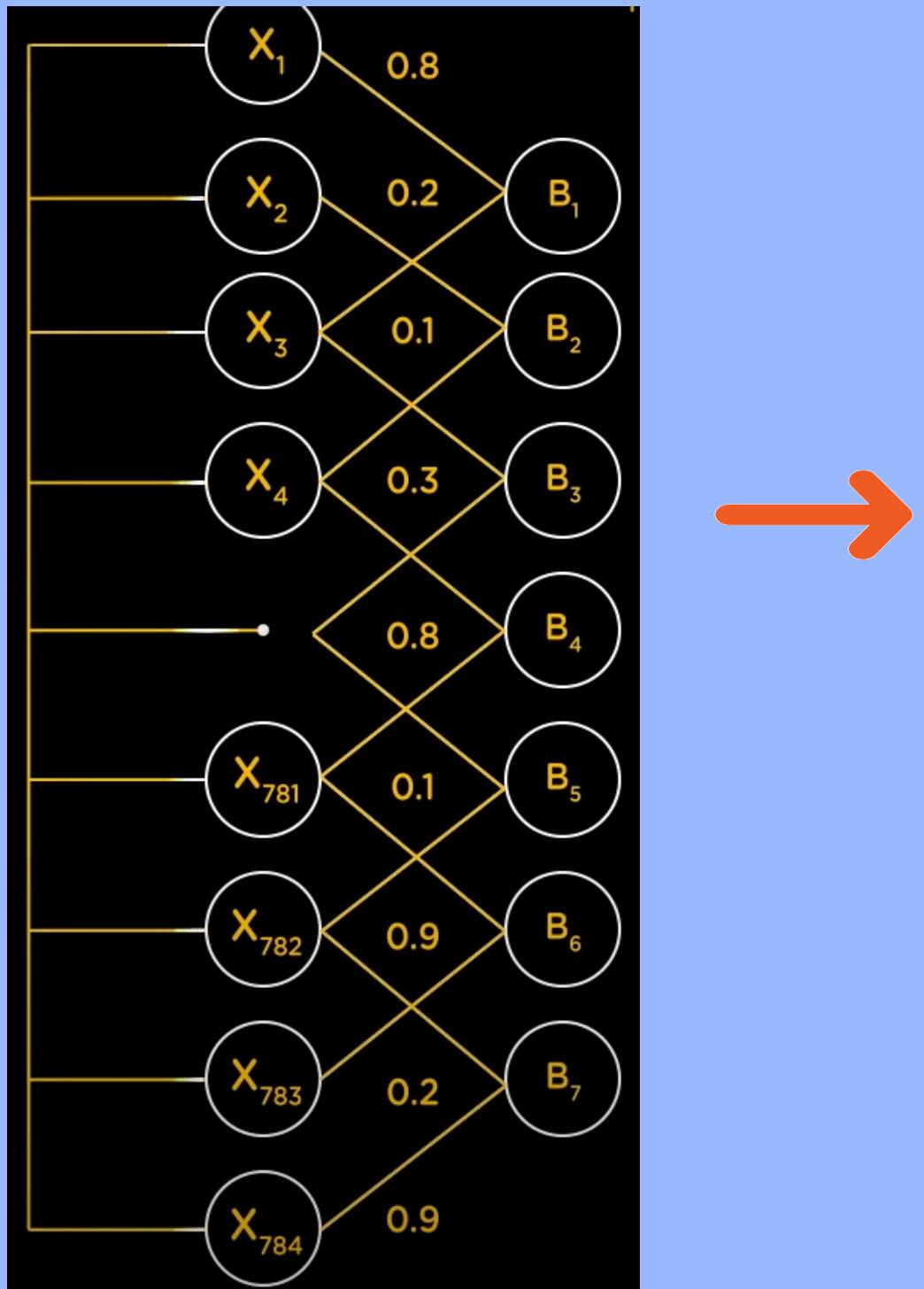


B1: Chia hình tròn ra thành các ô pixel.

Với $d = 28$ pixel

=> Cần sử dụng $28 \times 28 = 784$ pixel

B2: Ghép các pixel vào các input layer; các input layer nối với các layer tiếp theo bằng các Channels mà mỗi Channels này đều mang một giá trị đại số được hiểu là “Weight” (Trọng số).



$$(X_1 * 0.8 + X_3 * 0.2) + B_1 \quad \Rightarrow \text{Activation Function}$$

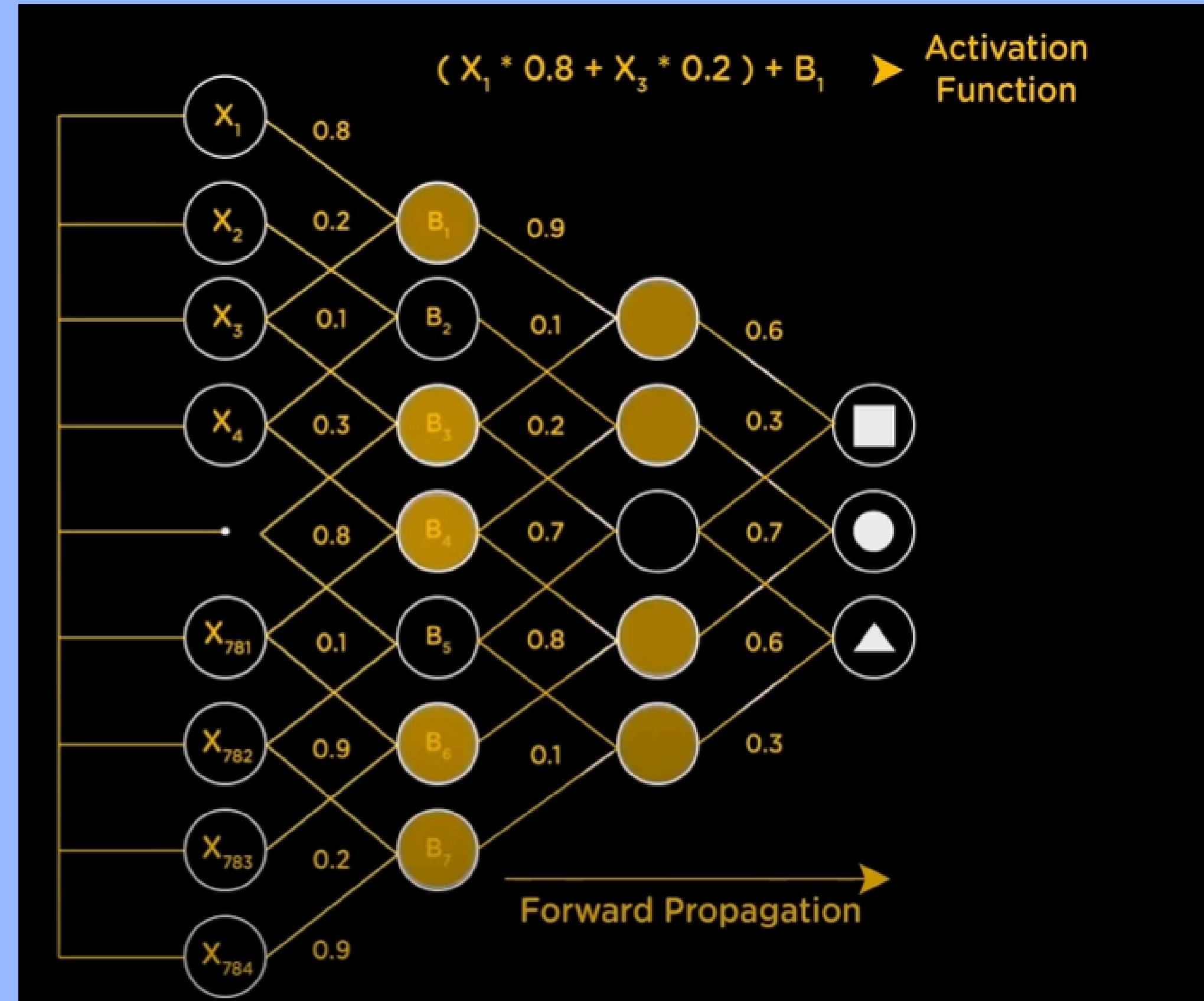
Tổng ở trên được đưa 1 luồng các hàm gọi là Activation Function

Các neurons trong Hidden layers được gán 1 giá trị đại số gọi là bias (độ dốc).

Activation Function sẽ quyết định xem neuron đó có được hoạt động tiếp hay không (Activated or Not).

Activated neuron sẽ tiếp tục chuyển data tới các layer tiếp theo thông qua các channels.

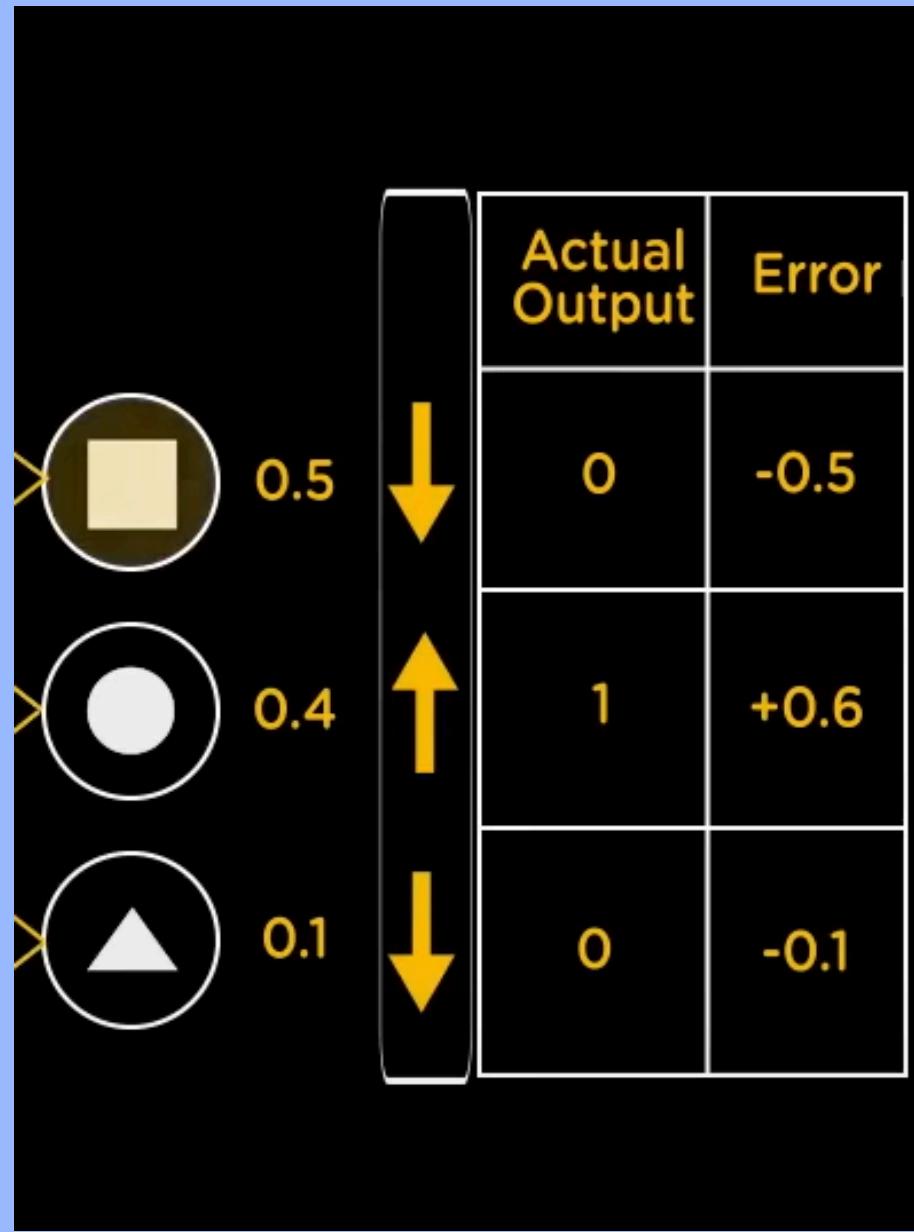
=> Việc dữ liệu xuyên suốt mạng lưới như vậy được gọi là Forward Propagation.





Tại layer OUTPUT, hệ thống đã tự tính toán ra xác suất của đáp án vấn đề được đặt ra. Hình có xác suất lớn nhất là hình mà hệ thống đã chọn để gắn cho hình input đầu vào.

Trong trường hợp này, hệ thống đã đưa ra được đáp án hình ban đầu là hình chữ nhật, song chúng ta đều biết đó là đáp án sai.
Vậy làm sao để hệ thống có thể hiệu chỉnh lại đáp án của nó?

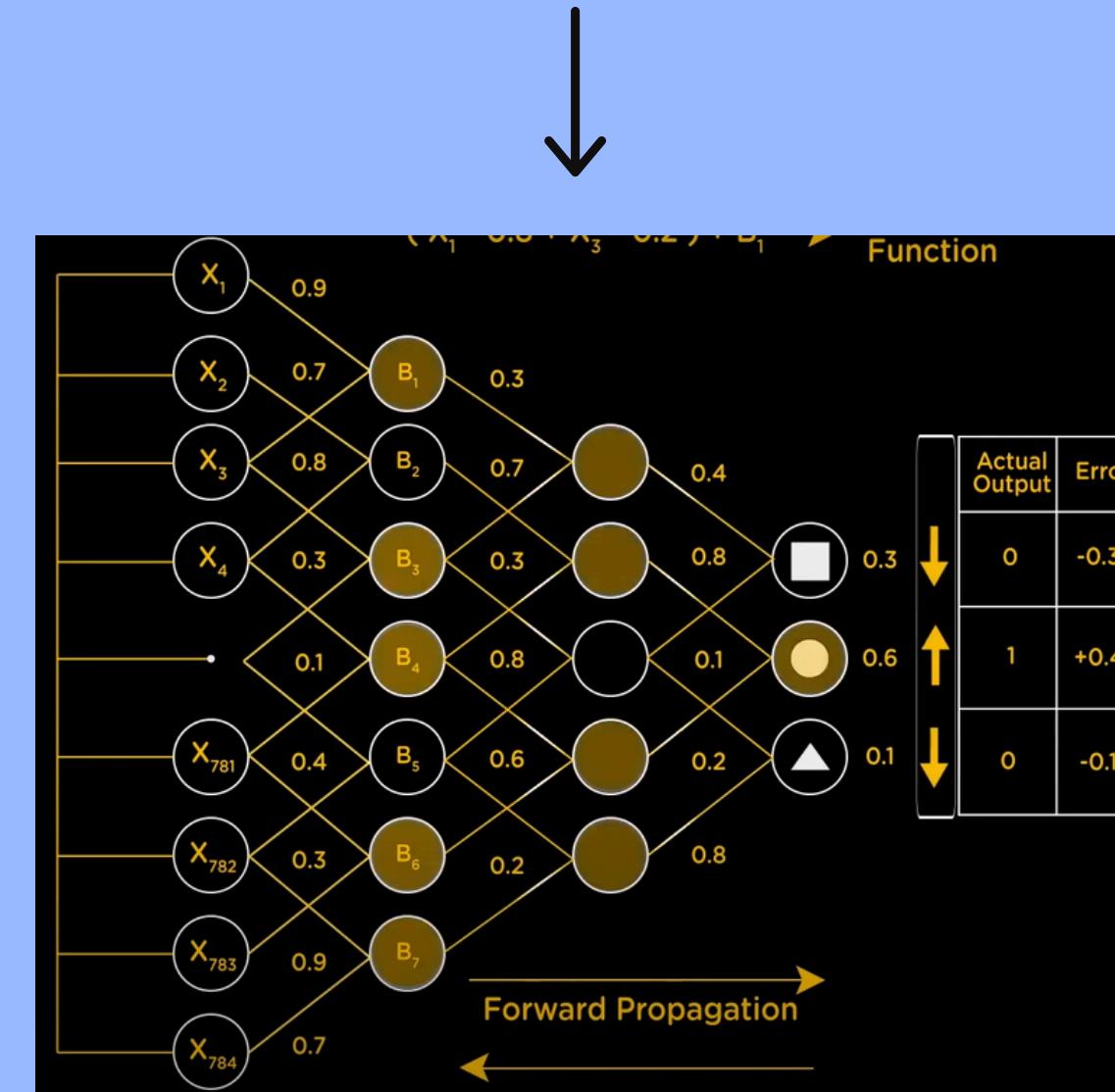


Ta lập được bảng so sánh giữa giá trị lý thuyết và giá trị thực tế như sau.

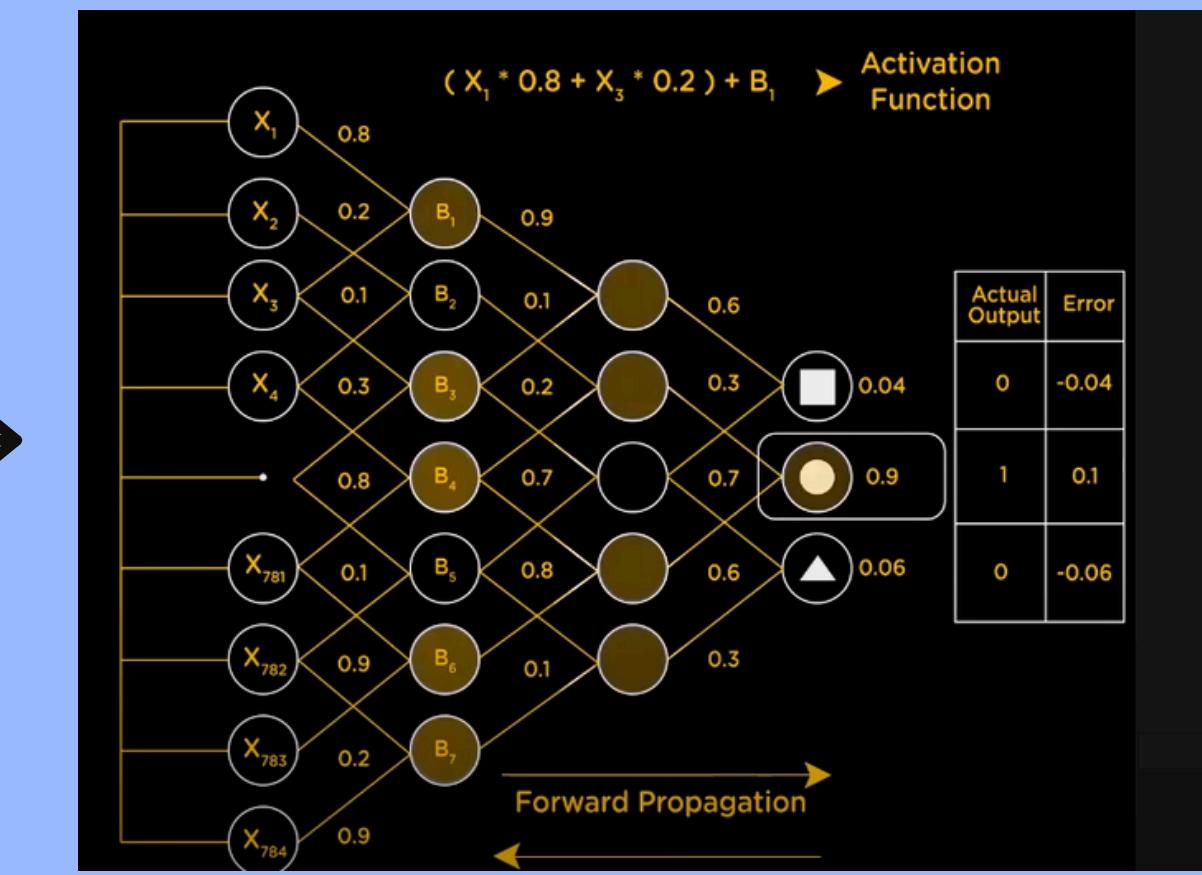
Hướng mũi tên chỉ ra giá trị lý thuyết cần tăng hay giảm để đạt được giá trị thực tế đúng.

Thông tin này sẽ được phản hồi lại về INPUT, quá trình này được gọi là Back Propagation.

Nhận được tín hiệu, các trọng số sẽ được thay đổi để phù hợp với kết quả đầu ra của bài toán.



Khi giá trị đầu ra đúng với giá trị thực tế, training process sẽ kết thúc và hệ thống sẽ đưa ra được đáp án của bài toán.



3

Ứng dụng của Neural Network

Mạng neural đã được áp dụng rộng rãi trong nhiều lĩnh vực khác nhau do khả năng của chúng trong việc học từ dữ liệu và tạo ra dự đoán chính xác trên dữ liệu mới. Dưới đây là một số ứng dụng phổ biến của mạng neural:

1 Nhận dạng hình ảnh và video:

Mạng neural có thể được sử dụng để nhận dạng và phân loại các đối tượng, khuôn mặt, định dạng chữ viết, biển số xe, và nhiều loại hình ảnh và video khác.

2. Xử lý ngôn ngữ tự nhiên (Natural Language Processing, NLP):

Trong lĩnh vực NLP, mạng neural được sử dụng để xây dựng các mô hình dịch máy, nhận diện ngôn ngữ tự nhiên, phân loại văn bản, và sinh văn bản tự động.

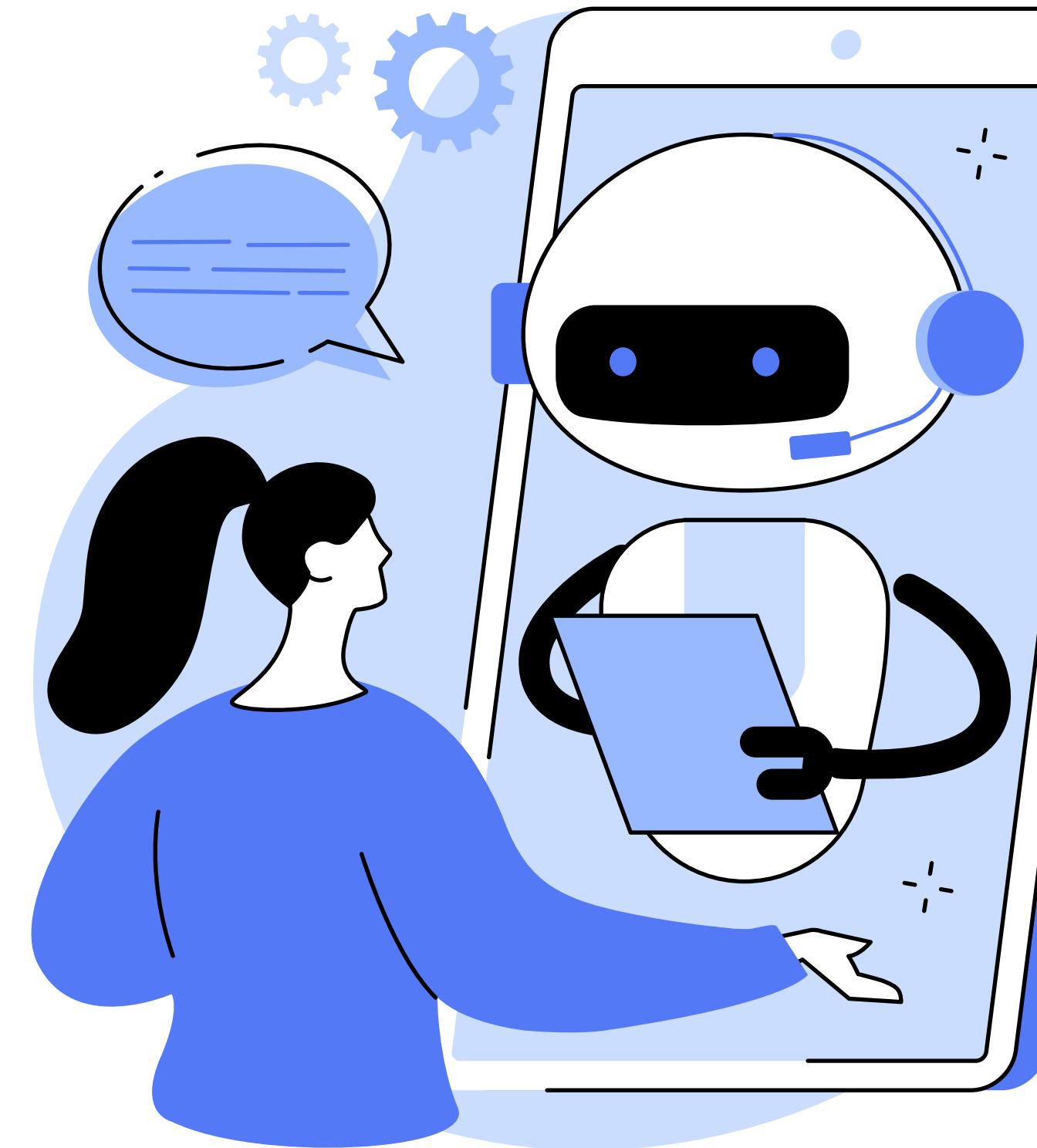
3. Dự đoán và phân tích dữ liệu:

Mạng neural có thể được sử dụng để dự đoán các xu hướng trong tài chính, phân tích dữ liệu y tế để dự đoán bệnh lý, và các ứng dụng khác trong dự báo và phân tích dữ liệu.

4. Tự lái xe và robot hợp tác tự động (Autonomous Driving and Robotics):

Trong ngành tự động hóa, mạng neural được sử dụng để phát hiện và phản ứng với các biến cố trên đường, dự đoán hành vi của người điều khiển và các đối tượng xung quanh, và điều khiển các thiết bị tự động di chuyển.

5. Mạng Neural cũng được sử dụng rộng rãi trong dược phẩm và y học, văn hóa và nghệ thuật cũng như trong các vấn đề môi trường ...

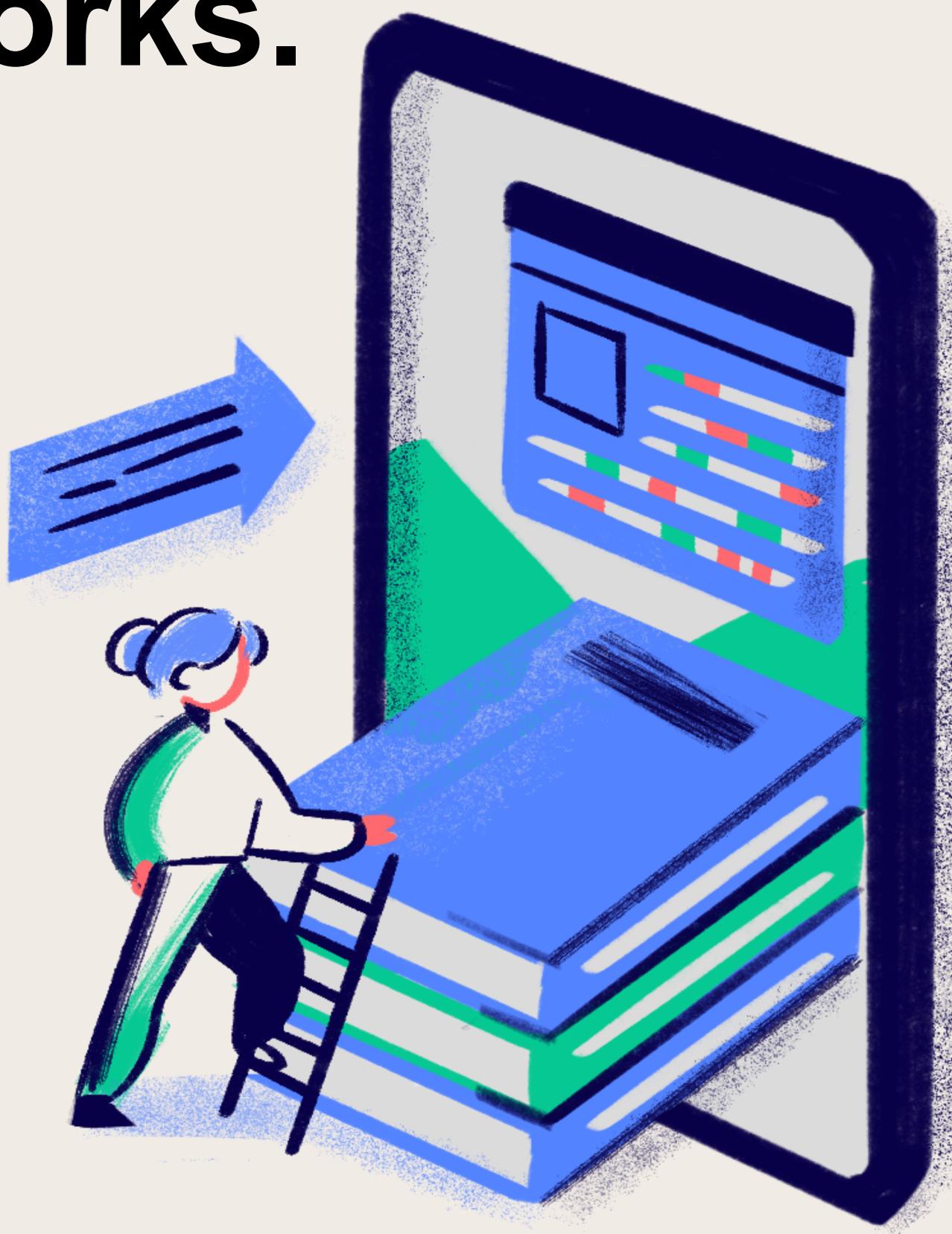


III. Learning in Belief networks.

1

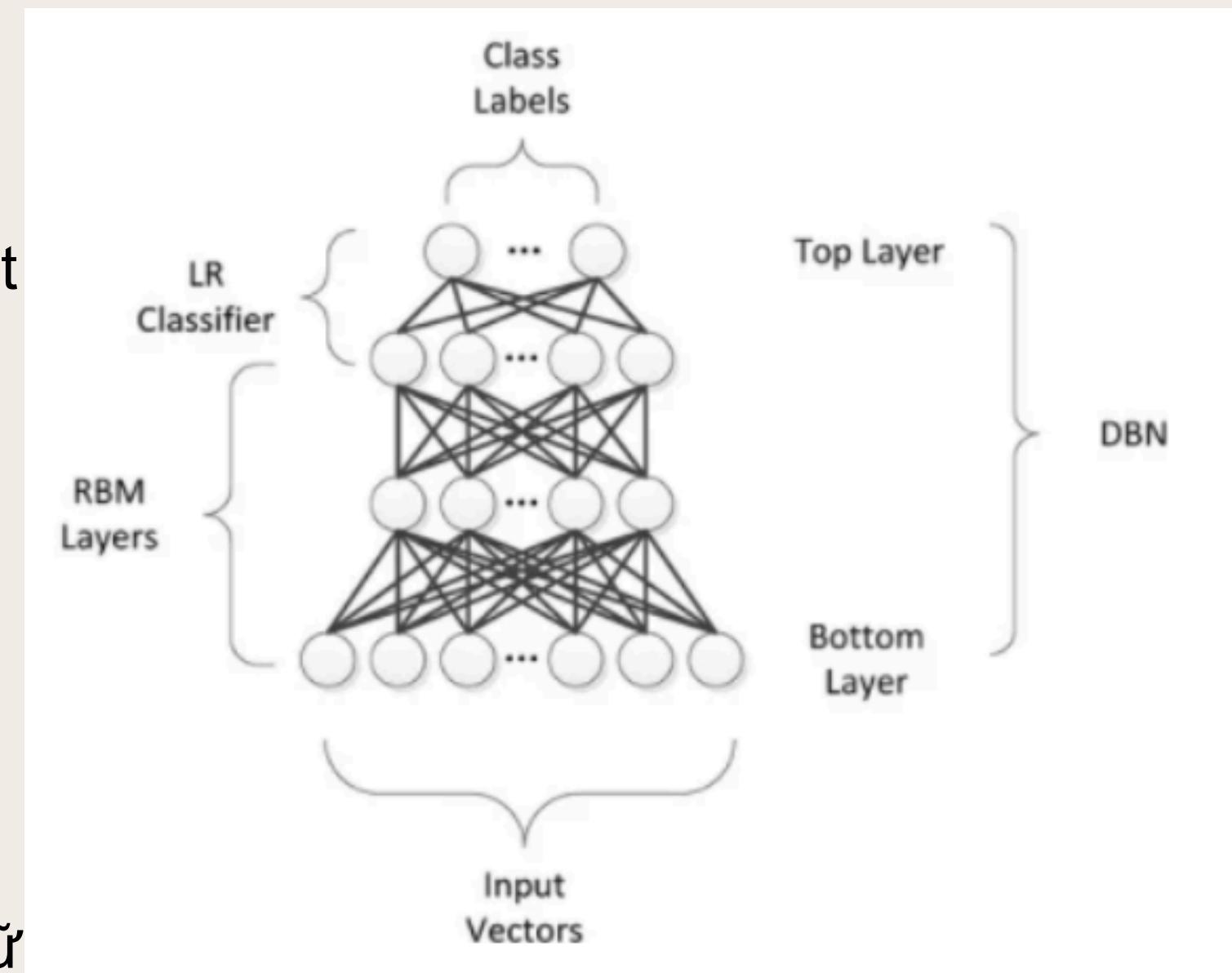
Deep Belief networks là gì?

Deep Belief Networks (DBNs) là mạng thần kinh nhân tạo phức tạp được sử dụng trong lĩnh vực Deep Learning , một tập hợp con của Machine Learning. Chúng được thiết kế để tự động khám phá và tìm hiểu các mẫu trong tập hợp dữ liệu lớn. Hãy tưởng tượng chúng như các mạng nhiều lớp, trong đó mỗi lớp có khả năng hiểu được thông tin nhận được từ lớp trước, dần dần xây dựng sự hiểu biết phức tạp về dữ liệu tổng thể.



DBN bao gồm nhiều lớp đơn vị ngẫu nhiên hoặc được xác định ngẫu nhiên. Các thiết bị này được gọi là Máy Boltzmann bị hạn chế (RBM) hoặc các cấu trúc tương tự khác. Mỗi lớp trong DBN nhằm mục đích trích xuất các tính năng khác nhau từ dữ liệu đầu vào, với các lớp thấp hơn xác định các mẫu cơ bản và các lớp cao hơn nhận biết các khái niệm trừu tượng hơn. Cấu trúc này cho phép DBN học cách biểu diễn dữ liệu phức tạp một cách hiệu quả, điều này khiến chúng đặc biệt hữu ích cho các tác vụ như nhận dạng hình ảnh và giọng nói, trong đó dữ liệu đầu vào có nhiều chiều và đòi hỏi mức độ hiểu biết sâu sắc.

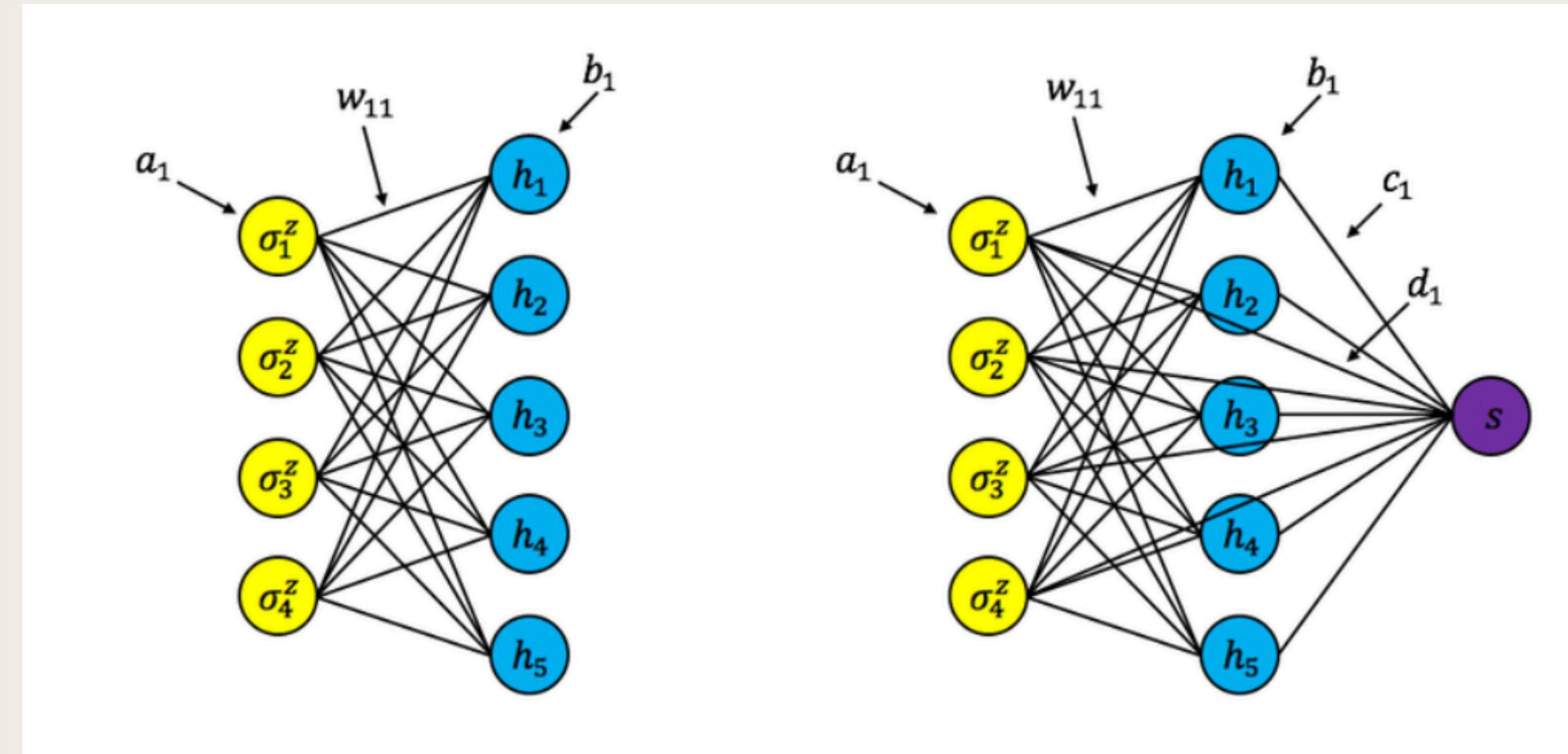
Kiến trúc của DBN cũng giúp chúng thực hiện tốt việc học không giám sát (Unsupervised learning), trong đó mục tiêu là hiểu và gắn nhãn dữ liệu đầu vào mà không cần hướng dẫn rõ ràng. Đặc điểm này đặc biệt hữu ích trong các tình huống mà dữ liệu được gắn nhãn khan hiếm hoặc khi mục tiêu là khám phá cấu trúc của dữ liệu mà không có bất kỳ nhãn định trước nào.



2

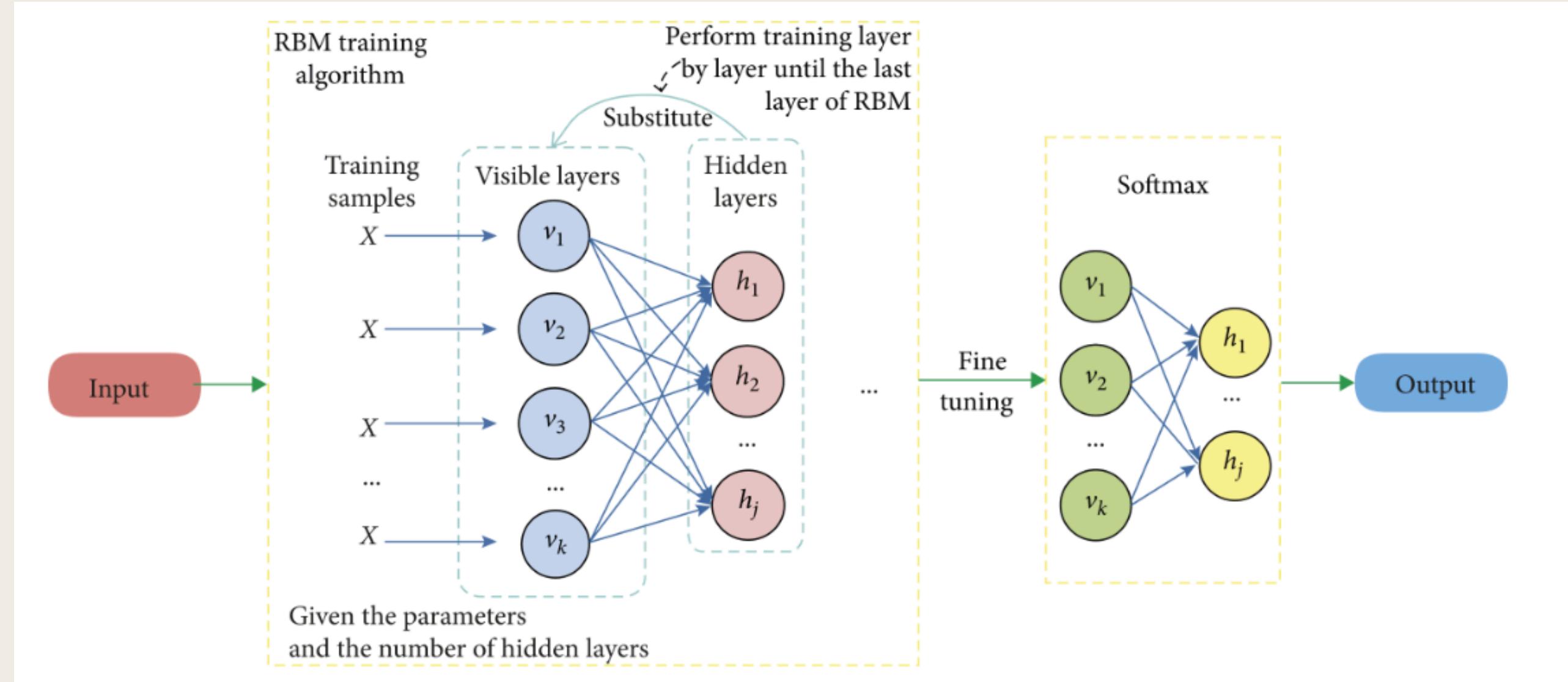
Sự phát triển của Deep Belief Network.

Sử dụng Perceptron trong thế hệ mạng thần kinh đầu tiên để xác định một đối tượng nhất định hoặc bất kỳ thứ gì khác bằng cách xem xét trọng số. Tuy nhiên, Perceptron có thể chỉ có lợi cho công nghệ cơ bản chứ không phải cho công nghệ phức tạp. Để giải quyết những vấn đề này, Mạng thần kinh thế hệ thứ hai đã đưa ra khái niệm Lan truyền ngược, so sánh đầu ra nhận được với đầu ra mong muốn và giảm giá trị lỗi xuống 0. Sau đó, xuất hiện các biểu đồ tuần hoàn có hướng được gọi là Belief Network, hỗ trợ giải quyết các vấn đề suy luận và học tập. Sau đó, sẽ sử dụng Deep Belief Network để giúp xây dựng các giá trị không thiên vị có thể lưu trữ trong các nút lá.



3

Deep Belief Network hoạt động như thế nào?



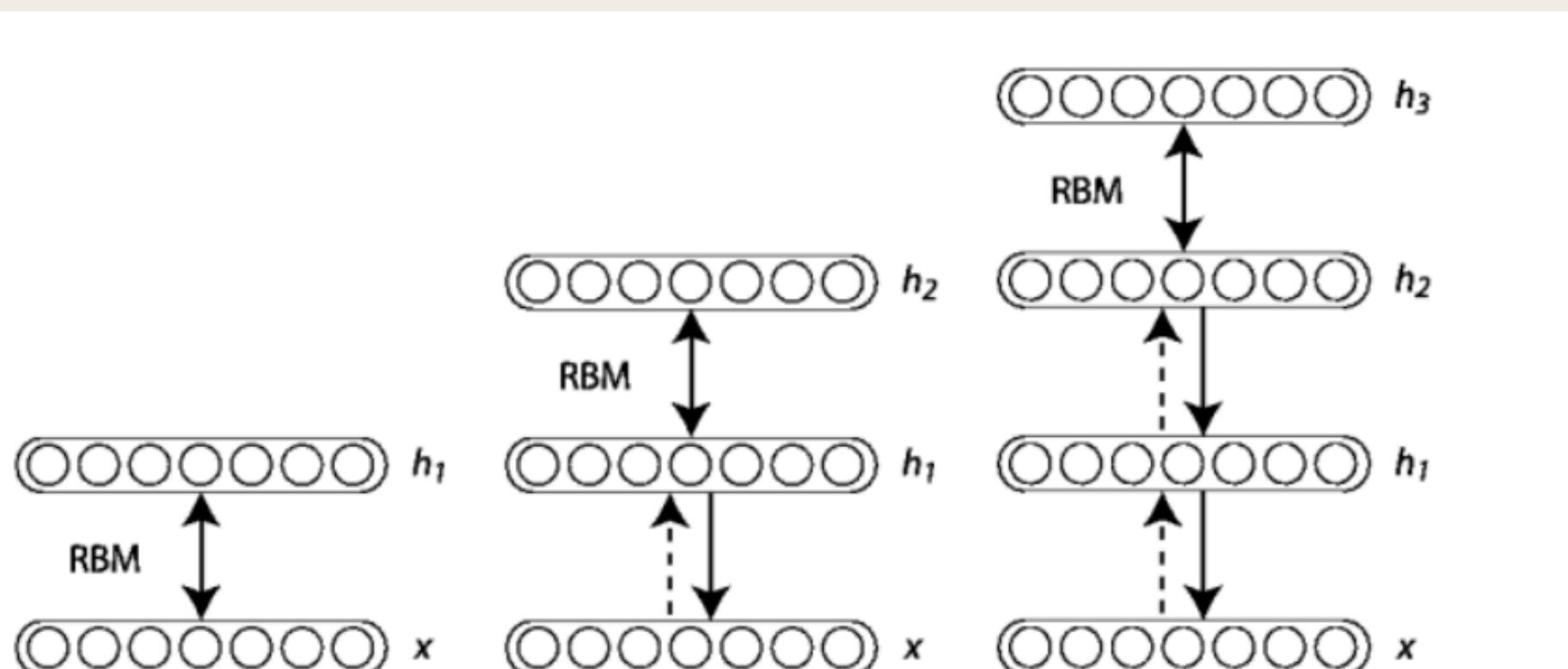
Giai đoạn đầu tiên là huấn luyện lớp thuộc tính có thể thu trực tiếp tín hiệu đầu vào từ pixel. Trong một phân lớp đã nghỉ hưu thay thế, hãy tìm hiểu các tính năng của các tính năng đạt được sơ bộ bằng cách coi các giá trị của phân lớp này là pixel. Giới hạn dưới về trách nhiệm pháp lý đối với nhật ký của tập dữ liệu huấn luyện sẽ cải thiện mỗi khi có một nhóm nhỏ các bưu kiện hoặc tính năng mới mà chúng tôi thêm vào mạng.

3

Deep Belief Network hoạt động như thế nào?

Quy trình hoạt động của Deep Belief Network như sau:

- Sử dụng thuật toán Greedy learning để đào tạo trước DBN. Để học các trọng số tổng quát từ trên xuống - phương pháp học tham lam sử dụng cách tiếp cận từng lớp. Các trọng số tổng quát này xác định mối quan hệ giữa các biến trong một lớp và các biến trong lớp trên.
- Ở hai lớp ẩn trên cùng, hệ thống chạy nhiều bước lấy mẫu Gibbs trong DBN. Hai lớp ẩn trên cùng xác định RBM, do đó, giai đoạn này sẽ trích xuất một mẫu từ nó một cách hiệu quả.
- Sau đó, tạo mẫu từ các đơn vị hiển thị bằng cách sử dụng một lần lấy mẫu tổ tiên qua phần còn lại của mô hình.
- Sử dụng một lần chuyển từ dưới lên để suy ra giá trị của các biến tiềm ẩn trong mỗi lớp. Ở lớp dưới cùng, quá trình tiền huấn luyện tham lam bắt đầu bằng một vectơ dữ liệu được quan sát. Sau đó, nó sẽ tinh chỉnh các trọng số tổng hợp theo cách ngược lại.



Cần phải nhớ rằng việc xây dựng Mạng lưới niềm tin
sâu sắc đòi hỏi phải đào tạo từng lớp RBM.

4

Xây dựng Deep Belief Network

Để tạo một deep belief network (DBN), bạn có thể sử dụng các thư viện và framework học sâu như TensorFlow, PyTorch hoặc scikit-learn. Dưới đây là một ví dụ về cách tạo một DBN bằng sử dụng thư viện TensorFlow và Keras:

Trong đó:

- **input_dim** là số chiều của đầu vào, ví dụ: số lượng pixel của mỗi hình ảnh.
- **hidden_layers** là một danh sách của số lượng unit trong mỗi hidden layer.
- **output_dim** là số lượng lớp đầu ra, ví dụ: số lượng lớp trong tập dữ liệu.

python

```

import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models

# Tạo một mô hình DBN
def create_DBN(input_dim, hidden_layers, output_dim):
    model = models.Sequential()

    # Thêm lớp visible layer
    model.add(layers.Dense(hidden_layers[0], activation='relu', input_shape=(input_dim,)))

    # Thêm các lớp hidden layer
    for units in hidden_layers[1:]:
        model.add(layers.Dense(units, activation='relu'))

    # Thêm lớp output layer
    model.add(layers.Dense(output_dim, activation='softmax'))

    return model

# Tham số của mạng
input_dim = 784 # Số lượng pixel của ảnh MNIST
hidden_layers = [256, 128, 64] # Số lượng unit trong các hidden layers
output_dim = 10 # Số lượng lớp đầu ra (số lượng lớp trong MNIST)

# Tạo DBN
dbn_model = create_DBN(input_dim, hidden_layers, output_dim)

# Compile mô hình
dbn_model.compile(optimizer='adam',
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])

# In thông tin về kiến trúc mô hình
dbn_model.summary()

```

5

Learning in Deep Belief Network

"Learning in belief networks" là quá trình huấn luyện các mô hình mạng lưới tin tưởng (belief networks) từ dữ liệu mẫu. Mạng lưới tin tưởng là một dạng mô hình đồ thị được sử dụng để biểu diễn và rút ra các mối quan hệ giữa các biến ngẫu nhiên. Trong mạng lưới tin tưởng, các nút trong đồ thị đại diện cho các biến ngẫu nhiên và các cạnh đại diện cho các mối quan hệ giữa chúng.

Quá trình "learning in belief networks" thường bao gồm việc:

1. Xây dựng mô hình: Xác định cấu trúc của mạng lưới tin tưởng, bao gồm việc xác định các nút và các cạnh của đồ thị, cũng như xác định loại mối quan hệ giữa chúng (ví dụ: phụ thuộc có điều kiện, độc lập có điều kiện).
- 2.. Thu thập dữ liệu: Thu thập dữ liệu mẫu để huấn luyện mô hình. Dữ liệu này bao gồm các giá trị quan sát cho các biến trong mạng lưới tin tưởng.
- 3.. Ước lượng tham số: Dựa trên dữ liệu mẫu, ước lượng các tham số của mô hình, bao gồm xác suất biên, xác suất có điều kiện và các tham số khác liên quan đến mối quan hệ giữa các biến.
- 4.. Kiểm tra và điều chỉnh: Kiểm tra hiệu suất của mô hình trên dữ liệu kiểm tra và điều chỉnh cấu trúc và tham số của mạng lưới tin tưởng để cải thiện khả năng dự đoán và khả năng tổng quát hóa.

Quá trình này thường là một quá trình lặp lại, trong đó mô hình được điều chỉnh dựa trên các đánh giá của hiệu suất trên dữ liệu kiểm tra và các kỹ thuật học máy để cải thiện mô hình.

Hãy xem xét một ví dụ đơn giản về quá trình "learning in belief network", cụ thể là việc huấn luyện một mô hình Gaussian Bayesian Network để dự đoán thời tiết dựa trên dữ liệu quan sát.

Giả sử chúng ta có một bộ dữ liệu với các biến như sau:

1. Nhiệt độ (T)
2. Độ ẩm (H)
3. Tình trạng mây (C)
4. Có mưa hay không (R)

Mục tiêu là xây dựng một mạng Bayesian Network để dự đoán xác suất của việc có mưa (R) dựa trên các biến khác.

```
python

import pymc3 as pm
import numpy as np

# Dữ liệu mẫu (nhiệt độ, độ ẩm, tình trạng mây, có mưa không)
data = np.random.randn(1000, 4)

# Xây dựng mô hình Bayesian Network
with pm.Model() as weather_model:
    # Đặc tả phân phối của các biến
    temp = pm.Normal('temp', mu=0, sd=1)
    humidity = pm.Normal('humidity', mu=0, sd=1)
    cloudiness = pm.Normal('cloudiness', mu=0, sd=1)

    # Đặc tả xác suất có mưa dựa trên nhiệt độ, độ ẩm và tình trạng mây
    rain_prob = pm.Deterministic('rain_prob', pm.math.sigmoid(temp + humidity + cloudiness))

    # Biến nhị phân dự đoán việc có mưa
    rain = pm.Bernoulli('rain', p=rain_prob, observed=data[:, 3])

# Huấn luyện mô hình
trace = pm.sample(1000, tune=1000)

# Trích xuất thông tin từ trace
pm.summary(trace)
```

Xác suất của việc có mưa (rain_prob) được xác định dựa trên các biến khác và được tính toán bằng hàm sigmoid.



**THANK YOU FOR
LISTENING!**