ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

-----🙰 🙲-----

BÁO CÁO BÀI TẬP LỚN

Môn: Trí tuệ nhân tạo

ĐỀ TÀI: Learning from Observations. Learning in Neural and Belief Network.

Ngành: **CÔNG NGHỆ KỸ THUẬT CƠ - ĐIỆN TỬ**

Giảng viên hướng dẫn: **TS. Trần Hồng Việt**

- Lớp: 2324II_INT3401_20

Nhóm Sinh viên thực hiện:

Phạm Thành Long	MSSV: 21021337
Đoàn Hữu Mạnh	MSSV: 21021339
Lê Vũ Đức Mạnh	MSSV: 21021340
Nguyễn Bình Minh	MSSV: 21020156
Phạm Quang Minh	MSSV: 21021343

MỤC LỤC

LỜI NÓI ĐẦU

Trong thời đại của sự tiến bộ công nghệ ngày nay, Trí Tuệ Nhân Tạo (AI) không chỉ là một chủ đề đang nổi lên mạnh mẽ mà còn là một lực lượng định hình tương lai của con người và xã hội. Từ các hệ thống tự động đơn giản đến các thuật toán phức tạp học máy, AI đã và đang tiếp tục thay đổi cách chúng ta làm việc, tư duy và tương tác với thế giới xung quanh.

Báo cáo này sẽ đi sâu vào các khái niệm về Deep Learning bao gồm Learning from Observations và Learning in Neural and Belief Network. Bên cạnh đó, chúng ta cũng sẽ đặt ra câu hỏi về ảnh hưởng của AI đối với đời sống cá nhân, doanh nghiệp và xã hội.

Trong quá trình thực hiện đề tài không tránh khỏi những sai sót, nhóm mong sẽ nhận được sự góp ý và đánh giá của thầy.

Xin chân thành cảm ơn!

I. Giới thiệu chung

1. Trí tuệ nhân tạo là gì?

Trí tuệ nhân tạo (AI) là lĩnh vực khoa học máy tính chuyên giải quyết các vấn đề nhận thức thường liên quan đến trí tuệ con người, chẳng hạn như học tập, sáng tạo và nhận diện hình ảnh. Các tổ chức hiện đại thu thập vô số dữ liệu từ nhiều nguồn khác nhau như cảm biến thông minh, nội dung do con người tạo, công cụ giám sát và nhật ký hệ thống. Mục tiêu của AI là tạo ra các hệ thống tự học có thể tìm ra ý nghĩa của dữ liệu. Sau đó, AI áp dụng kiến thức thu được để giải quyết các vấn đề mới theo cách giống như con người.

Ví dụ: công nghệ AI có thể trả lời cuộc trò chuyện với con người một cách hợp lý, tạo hình ảnh và văn bản gốc cũng như đưa ra quyết định dựa trên đầu vào dữ liệu theo thời gian thực. Tổ chức bạn có thể tích hợp tính năng AI vào ứng dụng để tối ưu hóa quy trình kinh doanh, nâng cao trải nghiệm khách hàng và đẩy mạnh quá trình đổi mới.

2. Học tập là gì?

Học tập là quá trình tiếp nhận, thu thập, và nắm bắt kiến thức, kỹ năng, và thông tin mới thông qua các phương tiện như sách, giáo viên, bài giảng, trải nghiệm thực tế, hoặc các tài nguyên trực tuyến. Đây là quá trình giúp cá nhân hiểu biết và phát triển, từ việc nhận thức cơ bản đến việc áp dụng kiến thức vào thực tế và phát triển kỹ năng để giải quyết các vấn đề. Học tập không chỉ diễn ra trong các tình huống học thuật mà còn xảy ra liên tục trong cuộc sống hàng ngày. 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.

3. Làm thế nào để AI có thể học được?

AI học thông qua các thuật toán và mô hình học máy. Dưới đây là một số phương pháp học của AI:

1. Học có giám sát (Supervised Learning): Trong loại học này, mô hình được cung cấp ví dụ đầu ra và ví dụ đầu vào mong muốn. Nó học từ các cặp dữ liệu này để dự đoán đầu ra cho các dữ liệu mới.
2. Học không giám sát (Unsupervised Learning): Ở đây, không có dữ liệu đầu ra được cung cấp cho mô hình. Thay vào đó, mô hình phải tự học cấu trúc và đặc điểm của dữ liệu.
3. Học tăng cường (Reinforcement Learning): Mô hình học từ kinh nghiệm và thử nghiệm sai lầm. Nó nhận được phản hồi từ môi trường và tối ưu hoá hành vi của mình để đạt được mục tiêu hoặc tối đa hoá phần thưởng.
4. Học bán giám sát (Semi-supervised Learning): Loại học này sử dụng một phần dữ liệu được gán nhãn và một phần không được gán nhãn. Mô hình sẽ học từ cả hai loại dữ liệu này để đưa ra dự đoán. Điều này hữu ích khi việc thu nhập dữ liệu gán nhãn có thể tốn kém hoặc khó khăn.
5. Học truyền cảm hứng (Transfer Learning): Trong trường hợp này, mô hình được huấn luyện trên một tác vụ và sau đó được chuyển giao kiến thức đã học sang một tác vụ khác. Điều này làm tăng tốc quá trình huấn luyện và cải thiện hiệu suất huấn luyện chương trình.

Những phương pháp này thường được kết hợp và điều chỉnh để tạo ra các mô hình AI hiệu quả trong nhiều ứng dụng khác nhau.

II. Learning from Observation.

1. Learning from Observation là gì?

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.

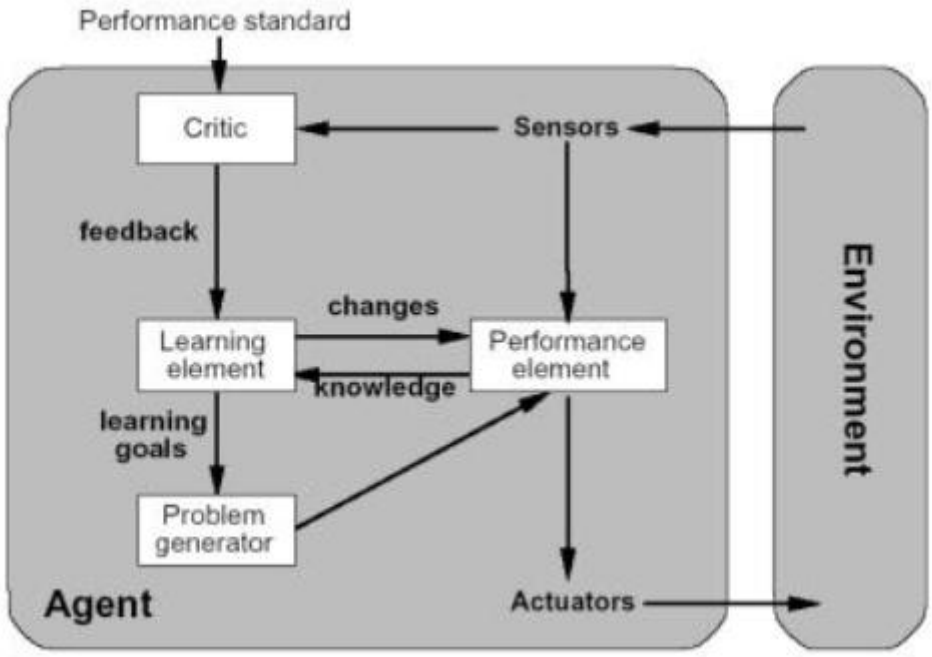
Quá trình "học từ quan sát" thường bao gồm việc sử dụng các kỹ thuật máy học như học có giám sát, học không giám sát hoặc học tăng cường để phân tích dữ liệu và tạo ra mô hình hoặc hệ thống có khả năng dự đoán, phân loại, hoặc thực hiện các nhiệm vụ khác dựa trên thông tin thu thập được.

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).

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:



Trong đó

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

2. Sensors (Cảm biến): Là các thiết bị hoặc công cụ mà learning agent sử dụng để quan sát trạng thái của môi trường.
3. Effectors (Công cụ thực thi): Là các phương tiện mà learning agent sử dụng để tương tác với môi trường.
4. 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.

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

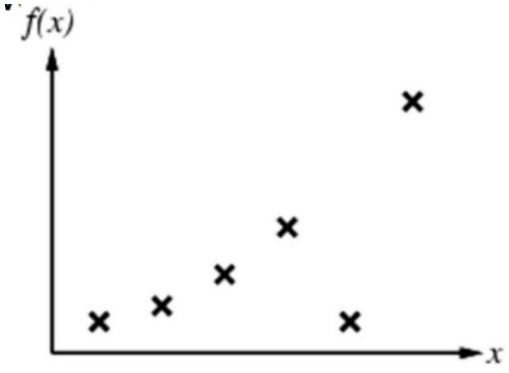
   - Những thành phần nào của yếu tố hiệu suất cần được học.
   - Những phản hồi nào có sẵn để tìm hiểu những điều này các thành phần.
   - Kiểu biểu diễn (Representation) nào được sử dụng cho các thành phần.

5. 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.
6. 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.
7. Performance Element (Yếu tố hiệu suất): Là phần đo lường và đánh giá hiệu suất của các hành động và quyết định.

2. Phương pháp học quy nạp (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 xét bài toán sau: **Xác đinh hàm số từ các giá trị x và f(x) tương ứng cho trước cho trước trên đồ thị.**



Với f(x) là hàm số cần tìm. Dựa vào các giá trị f(x) ứng với x cho 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.

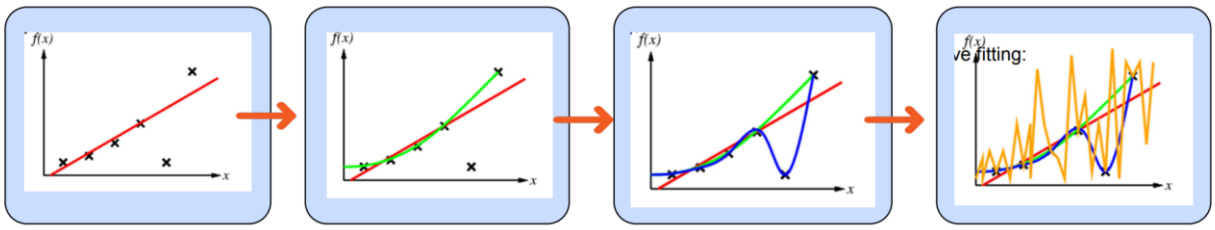
Đây là một mô hình học tập thực sự

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:



Ví dụ: Trong hình 1, đồ thị hàm số đi qua các điểm có toạ độ (1;0) và (5;5). Đồ thị là một đường thẳng có dạng ax + by + c = 0. Ta hoàn toàn xác định được dạng chính xác của đồ thị đi qua 2 điểm cho trước là đường thẳng (d) có dạng:

$$(d)\ \ 5x - 4y\ -\ 5\ = 0.$$

Tương tự với các đồ thị bậc 2, bậc 3 và cao hơn nữa ứng với các hình 2,3,4.

Trong trường hợp này, 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.

3. Phương pháp cây quyết định (Decision Tree).

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.

Cây quyết định bao gồm các nút và các cạnh. Mỗi nút trong cây đại diện cho một thuộc tính và mỗi cạnh đại diện cho một quyết định hoặc một kết quả của thuộc tính đó. Ở mỗi nút, một quyết định được đưa ra dựa trên giá trị của thuộc tính tương ứng trong dữ liệu. Quyết định này dẫn đến việc di chuyển đến các nút con khác trên cây, cho đến khi đạt được một nút lá, tại đó kết quả cuối cùng được dự đoán.

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).

Cây quyết định có thể dễ dàng hiểu và giải thích, nên được sử dụng rộng rãi trong nhiều ứng dụng trong các lĩnh vực như y học, tài chính, marketing, và hệ thống thông tin.

Ta xét vấn đề sau: **Có nên đợi bàn ở nhà hàng hay không?**

Dựa trên các tiêu chí:

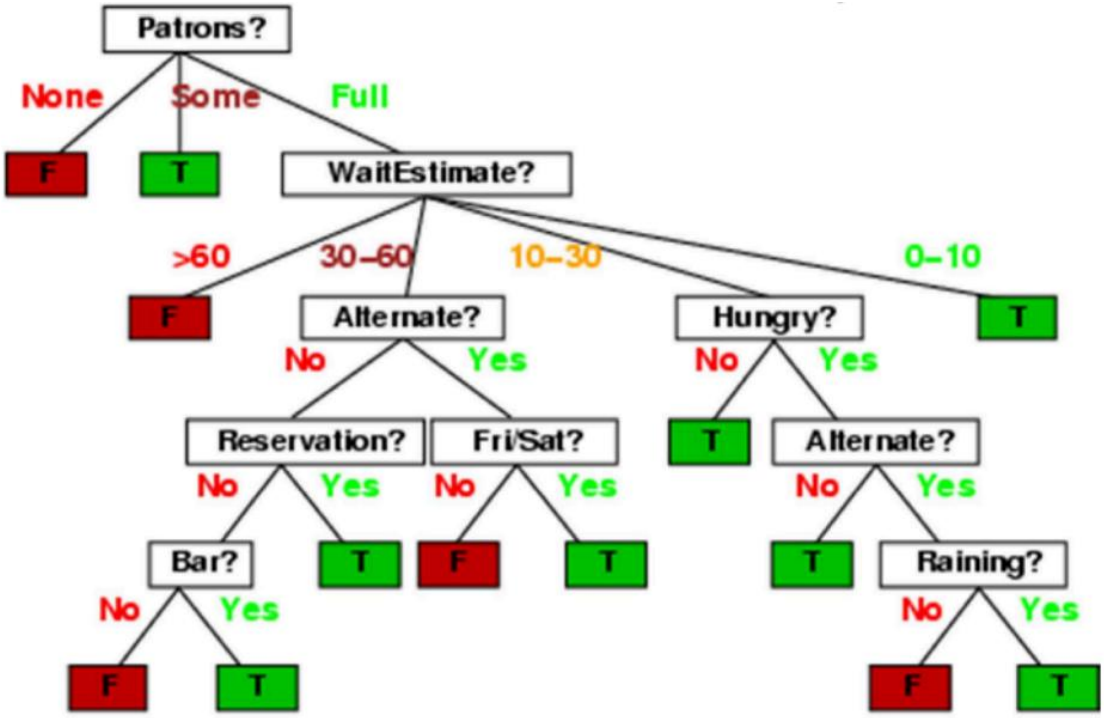
1. Thay thế: có nhà hàng thay thế nào gần đó không?
2. Quầy bar: Có khu vực quầy bar nào thoải mái để chờ không?
3. Thứ Sáu/Thứ Bảy: hôm nay là thứ Sáu hay thứ Bảy?
4. Đói: chúng ta có đói không?
5. Khách hàng (Patrons): số lượng người trong nhà hàng (None, Some, Full)?
6. Giá: khoảng giá ($, $$, $$$)?
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)?

Ta giả định các tình huống và lập được bảng thuộc tính như sau:

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

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

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



- **Giải thích.**

Xây dựng một cây quyết định là quá trình chia tập dữ liệu thành các phần con và xây dựng cấu trúc cây dựa trên các thuộc tính của dữ liệu. Dưới đây là các bước cơ bản để xây dựng một cây quyết định:

B1: Chọn thuộc tính: Bắt đầu với tất cả các thuộc tính trong tập dữ liệu. Thuộc tính nào nên được chọn làm nút gốc của cây là quan trọng nhất. Một số phương pháp đánh giá thuộc tính như Information Gain, Gini Impurity hoặc Entropy có thể được sử dụng để quyết định thuộc tính vào được chọn làm gốc.

B2: Chia tập dữ liệu: Sử dụng thuộc tính đã chọn để chia tập dữ liệu thành các phần còn dựa trên các giá trị thuộc tính đó. Mỗi phần con tương ứng với một giá trị của thuộc tính.

B3: Lặp hoặc dừng lại: Tiếp tục quá trình lặp việc chọn thuộc tính và chia tập dữ liệu cho mỗi phần con, tạo ra các nhánh mới trên cây cho đến khi một điều kiện dừng được đáp ứng. Các điều kiện có thể là:

- Tất cả các mẫu thuộc cùng một lớp.
- Không còn thuộc tính nào để chia.
- Đã đạt đến một số lượng giới hạn độ sâu đã được xác định trước.
- Số lượng mẫu ở dưới một ngưỡng nhất định.

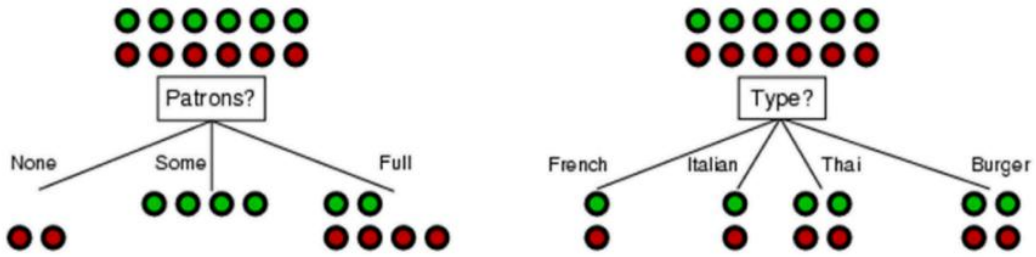
B4: Gán nhãn cho các nút lá: Khi một nút lá được tạo ra, một nhãn được gán cho nó dựa trên lớp phổ biến nhất trong tập dữ liệu con tương ứng.

Kết quả cuối cùng là một cây quyết định mà mỗi nút bao gồm một thuộc tính và mỗi cạnh dẫn đến một giá trị của thuộc tính đó. Khi một dữ liệu mới được đưa vào, nó được đưa qua cây từ nút gốc đến nút lá và dự đoán kết quả dựa trên nhãn của nút lá mà nó đến.

- **Cách chọn thuộc tính gốc.**

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". Ở đây ta xét hai thuộc tính “ Patrons “ và “ Type “.

a, Dựa trên khảo sát, ta thu được kết quả sau:



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.

b, Sử dụng lý thuyết về thông tin (Information theory)

Information Content (Entropy)

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

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

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

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

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.

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

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

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

→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):

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

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

→Chọn Patrons vì có IG lớn nhất.

III. Learning in Neural Network.

1. Giới thiệu nạng Neuron sinh học.

Não bộ con người gồm:

• 100 tỉ tế bào thần kinh (neuron), 6x1014 khớp thần kinh (synapse) .

• Mỗi tế bào có cấu trúc đơn giản.

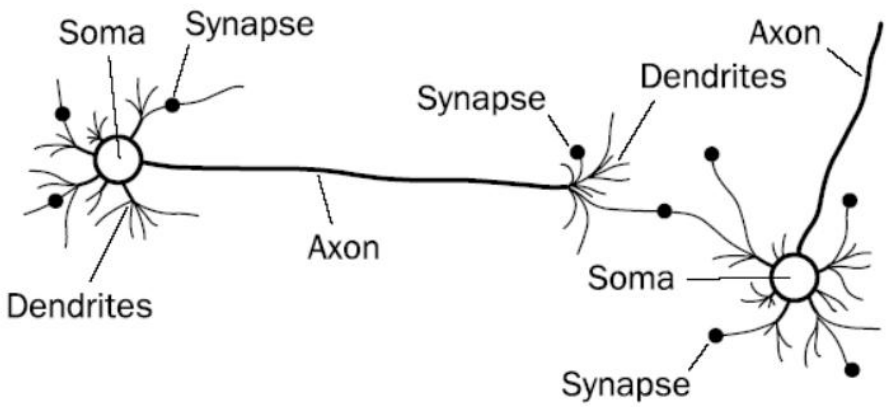
• Một tế bào thần kinh bao gồm: thân – soma, nhiều sợi thần kinh – dendrite và một sợi trục chính – axon.

Cơ chế học của neuron thần kinh:

• Tín hiệu được lan truyền giữa các neuron.

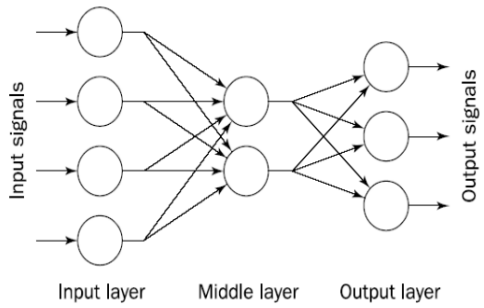
• Một neuron nhận tín hiệu kích thích từ các khớp nối (synapse) và phát tín hiệu qua thân (soma) đến các neuron khác.

• Mối liên hệ giữa các neuron quy định chức năng của mạng neuron và được hình thành từ từ qua quá trình học.



2. Mạng Neuron nhân tạo.

Mạng neuron nhân tạo (Artificial Neural Netwok – ANN): là một tập hợp các bộ xử lý rất đơn giản – neuron – và nối với nhau.



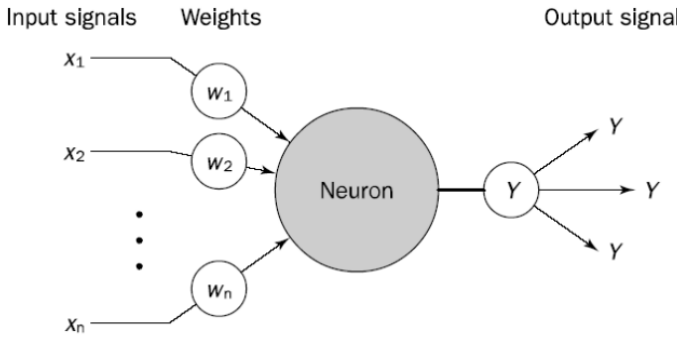
Mạng neuron sinh học	Mạng neuron nhân tạo
Soma (thân)	Neuron
Denrite (sợi thần kinh)	Input
Axon (sợi trục chính)	Output
Synapse (khớp thần kinh)	Weight (trọng số)

• Cấu trúc và phương thức hoạt động của ANN mô phỏng tương tự mạng neuron sinh học.

• Các tín hiệu liên kết với các trọng số tương ứng. Các trọng số ứng với bộ nhớ dài hạn của ANN.

• ANN "học" bằng cách điều chỉnh từ từ các trọng số này qua quá trình tương tác với môi trường (huấn luyện).

Mỗi một neuron là một thành phần tính toán đơn giản.



$$X = \sum_{i=1}^{n} x_i w_i$$

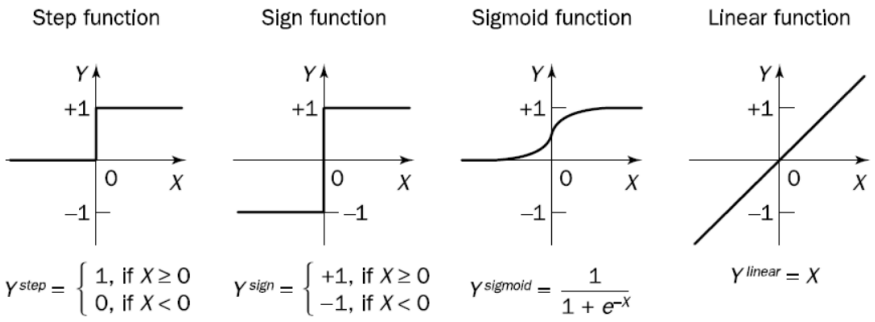
$$Y = \begin{cases} +1 & if\ X \geq t \\ -1 & if\ X < t \end{cases}$$

t: ngưỡng hoạt hoá

Y được gọi là hàm kích hoạt hay hàm truyền.

$$Y = sign\left[\sum_{i=1}^{n} x_i w_i - t\right]$$

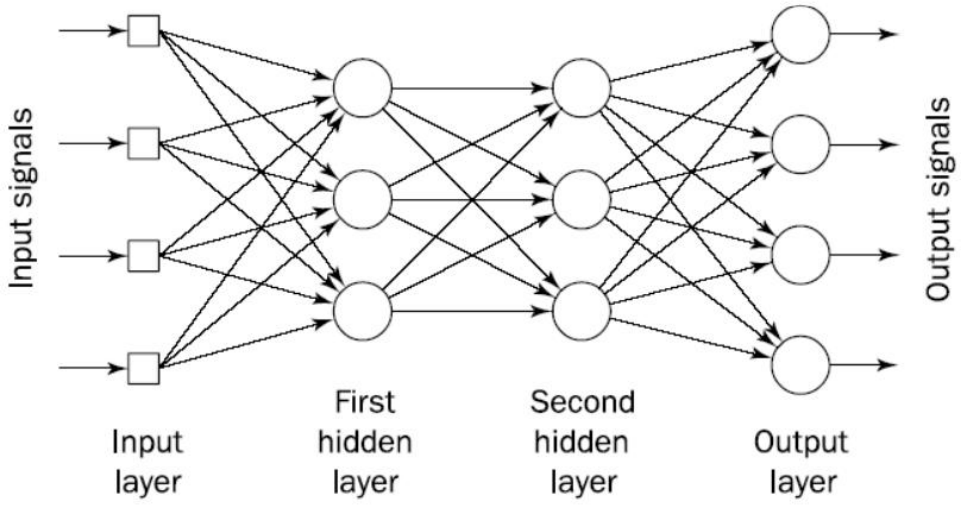
Bốn hàm truyền có ứng dụng thực tế:



Hàm sign và step được gọi là hàm giới hạn cứng.

3. Mạng neuron nhiều lớp.

Một mạng neuron lan truyền tiến gồm một lớp nhập (Input Layer), ít nhất một lớp ẩn (Hidden Layer), và một lớp xuất (Output Layer).



Trong đó:

- Lớp nhập: nhận các input và phân phối chúng cho tất cả neuron trong lớp ẩn
- Lớp xuất: biểu diễn kết quả của toàn mạng
- Lớp ẩn:

  • Dò tìm các đặc trưng.

  • Các neuron trong lớp này "ẩn" các kết xuất mong muốn của chúng.

  • Mạng một lớp ẩn có thể biểu diễn bất kỳ hàm liên tục nào.

  • Mạng hai lớp ẩn có thể biểu diễn các hàm không liên tục.

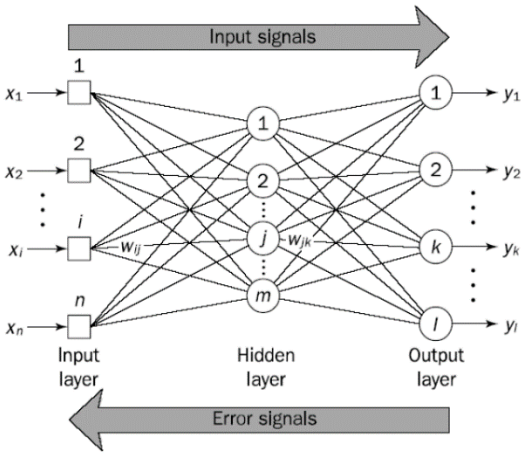
4. Học trong mạng neuron nhiều lớp.

a, Quá trình học gồm hai pha:

- Lan truyền các mẫu input từ lớp nhập đến lớp xuất, tại mỗi neuron tính:

$$X = \sum_{i=1}^{n} x_i w_i - \theta$$

$$Y^{sigmoid} = \frac{1}{1 + e^{-X}}$$



- Lan truyền ngược sai số từ lớp xuất và

cập nhật các trọng số.

b, Thuật toán lan truyền ngược.

B1: Khởi tạo. Đặt giá trị ngẫu nhiên các trọng số và ngưỡng của mạng.

B2: Kích hoạt.

a) Tính kết xuất thực sự của các neuron trong lớp ẩn:

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \theta_j\right]$$

b, Tính kết xuất thực sự của các neuron trong lớp xuất:

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p) \times w_{jk}(p) - \theta_k\right]$$

B3: Huấn luyện trọng số.

a) Tính gradient sai số cho các neuron lớp xuất:

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p)$$
$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p)$$

Cập nhật các trọng số của neuron lớp xuất:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$$

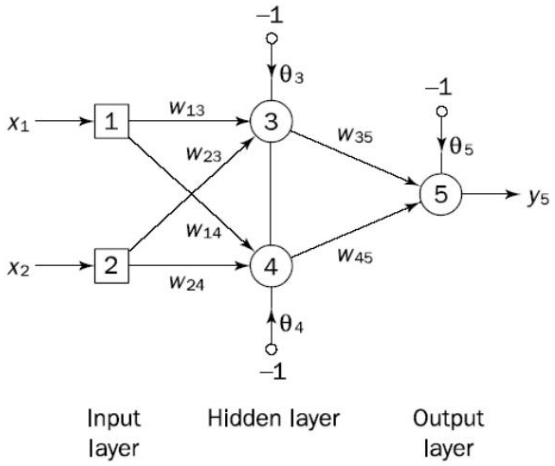
b) Tính gradien sai số và cập nhật trọng số lớp ẩn.

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{i} \delta_k(p) \times w_{jk}(p)$$
$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p)$$
$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

B4: Lặp.

5. Vi dụ.

- Mạng neuron ba lớp thực hiện phép logic XOR.
- Neuron 1, 2 của lớp nhập nhận input x1 và x2 và phân phối các input này đến lớp ẩn: x13 = x14 = x1 và x23 = x24 = x2 .
- Các giá trị ngưỡng được biểu diễn bởi các trọng số θ và được kết nối với input -1.



B1: Khởi tạo.

w13 = 0.5	θ3 = 0.8
w14 = 0.9	θ4 = -0.1
w23 = 0.4	θ5 = 0.3
w24 = 1.0	
w35 = -1.2	
w45 = 1.1	

Xét mẫu huấn luyện x1=x2=1 và kết xuất mong muốn y=0.

B2: Tính kết xuất thực tại nút ẩn.

$$y_3 = sigmoid\,(x_1 w_{13} + x_2 w_{23} - \theta_3) = 1/[1 + e^{-(1\times 0.5 + 1\times 0.4 - 1\times 0.8)}] = 0.5250$$

$$y_4 = sigmoid\,(x_1 w_{14} + x_2 w_{24} - \theta_4) = 1/[1 + e^{-(1\times 0.9 + 1\times 1.0 + 1\times 0.1)}] = 0.8808$$

Kết xuất tại nút xuất.

$$y_5 = sigmoid\,(y_3 w_{35} + y_4 w_{45} - \theta_5)$$
$$= 1/[1 + e^{-(-0.5250\times 1.2 + 0.8808\times 1.1 - 1\times 0.3)}] = 0.5097$$

Sai số.

$$e = y_{d,5} - y_5 = 0 - 0.5097 = -0.5097$$

B3. Tính gradient sai số tại các neuron 5 lớp xuất.

$$\delta_5 = y_5(1 - y_5)e = 0.5097 \times (1 - 0.5097) \times (-0.5097) = -0.1274$$

và tính các giá trị điều chỉnh trọng số

$$\Delta w_{35} = \alpha \times y_3 \times \delta_5 = 0.1 \times 0.5250 \times (-0.1274) = -0.0067$$

$$\Delta w_{45} = \alpha \times y_4 \times \delta_5 = 0.1 \times 0.8808 \times (-0.1274) = -0.0112$$

$$\Delta \theta_5 = \alpha \times (-1) \times \delta_5 = 0.1 \times (-1) \times (-0.1274) = 0.0127$$

cập nhật trọng số

$$w_{35} = w_{35} + \Delta w_{35} = -1.2 - 0.0067 = -1.2067$$

$$w_{45} = w_{45} + \Delta w_{45} = 1.1 - 0.0112 = 1.0888$$

$$\theta_5 = \theta_5 + \Delta \theta_5 = 0.3 + 0.0127 = 0.3127$$

Tại lớp ẩn, neuron 3, tính gradient sai số

$$\begin{aligned}\delta_3 &= y_3(1 - y_3) \times \delta_5 \times w_{35} \\ &= 0.5250 \times (1 - 0.5250) \times (-0.1274) \times (-1.2) = 0.0381\end{aligned}$$

tính các giá trị điều chỉnh

$$\Delta w_{13} = \alpha \times x_1 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$

$$\Delta w_{23} = \alpha \times x_2 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$

$$\Delta \theta_3 = \alpha \times (-1) \times \delta_3 = 0.1 \times (-1) \times 0.0381 = -0.0038$$

cập nhật trọng số

$$w_{13} = w_{13} + \Delta w_{13} = 0.5 + 0.0038 = 0.5038$$
$$w_{23} = w_{23} + \Delta w_{23} = 0.4 + 0.0038 = 0.4038$$
$$\theta_3 = \theta_3 + \Delta \theta_3 = 0.8 - 0.0038 = 0.7962$$

B4: Kết thúc.

- Quá trình huấn luyện trải qua 224 thế hệ hay 894 vòng lặp kết thúc khi tổng bình phương sai số bằng 0.001 với các giá trị trọng số cuối cùng w13= 4.7621, w14= 6.3917, w23= 4.7618, w24= 6.3917, w35= -10.3788, w45= 9.7691, θ3= 7.3061, θ4= 2.8441, θ5= 4.5589
- Mạng hoạt động bằng cách sử dụng quá trình kích hoạt (bước 2) trên tập dữ liệu thử nghiệm.

Input		Kết xuất mong muốn	Kết xuất thực	Sai số
x_1	x_2	Y_d	Y	e
1	1	0	0.0155	-0.0155
0	1	1	0.9849	0.0151
1	0	1	0.9849	0.0151
0	0	0	0.0175	-0.0175

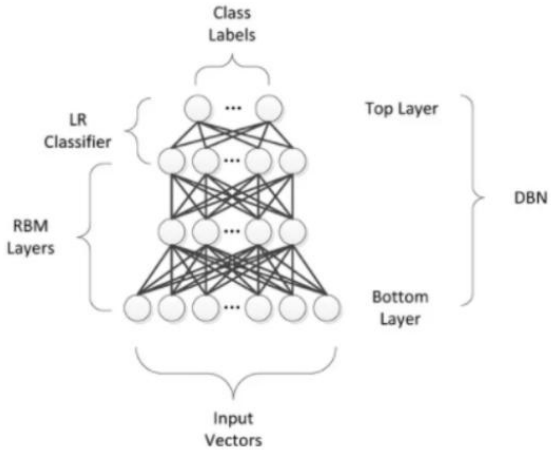
- Lưu ý: mạng sẽ thu được các giá trị trọng số và ngưỡng khác nhau trong những lần huấn luyện khác nhau.

IV. 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. Phương pháp Bayesian trong Learning Belief Network.

Công thức cơ bản của mô hình Naïve Bayes được dùng để tính xác suất có điều kiện của một biến mục tiêu (target variable) dựa trên các biến độc lập (independent variables) được gọi là "naïve" vì giả định rằng các biến độc lập này là hoàn toàn độc lập với nhau, điều này thường không đúng trong thực tế, nhưng giả định này giúp đơn giản hóa mô hình và tính toán.

Giả sử bạn có một biến mục tiêu (Y) và một tập hợp các biến độc lập (X1, X2, ..., Xn). Công thức cơ bản của Naïve Bayes cho việc tính xác suất có điều kiện của biến mục tiêu (Y) dưới điều kiện các biến độc lập (X1, X2, ..., Xn) là:

$$P(Y|X_1, X_2, ..., X_n) = \frac{P(Y)\cdot P(X_1|Y)\cdot P(X_2|Y)\cdot ...\cdot P(X_n|Y)}{P(X_1)\cdot P(X_2)\cdot ...\cdot P(X_n)}$$

Trong đó:

- (P(Y)) là xác suất tiên nghiệm của biến mục tiêu \(Y \).

- (P(Xi | Y)) là xác suất của biến độc lập \(X_i \) trong điều kiện biến mục tiêu \(Y \).

- (P(Xi) là xác suất tiên nghiệm của biến độc lập \(X_i \).

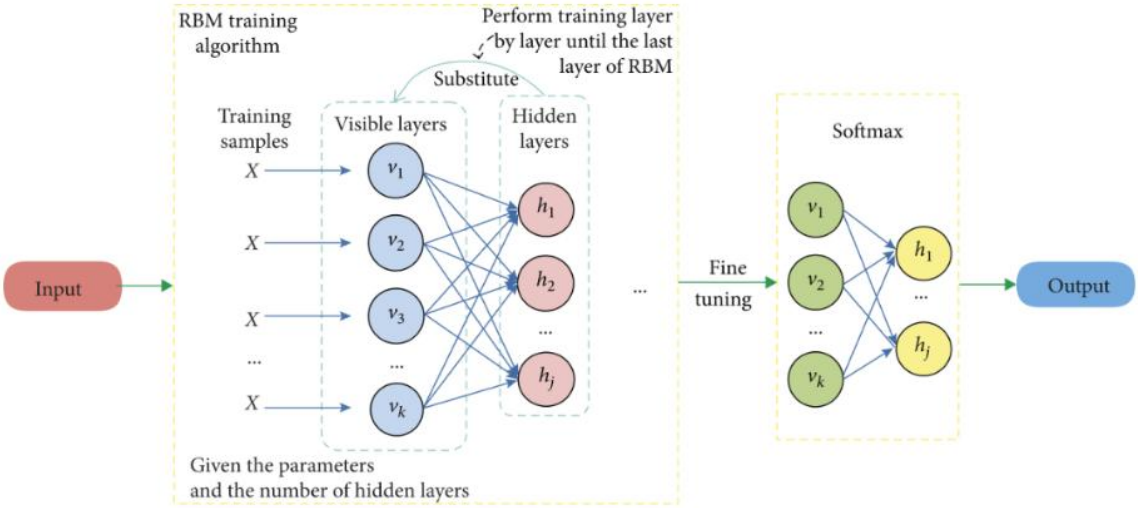
- (P(Y | X1, X2, ..., Xn)) là xác suất có điều kiện của biến mục tiêu (Y) dưới điều kiện các biến độc lập (X1, X2, ..., Xn).

Trong Naïve Bayes, giả định rằng các biến độc lập (X1, X2, ..., Xn) là độc lập có điều kiện với biến mục tiêu (Y), vì vậy ta có thể rút gọn công thức trên thành:

$$P(Y|X_1, X_2, ..., X_n) \propto P(Y) \cdot \prod_{i=1}^{n} P(X_i|Y)$$

Ở đây, $\propto$ biểu thị cho việc "proportional to", nghĩa là các xác suất này được tỉ lệ thuận với xác suất có điều kiện của biến mục tiêu (Y).

3. Học trong Deep Belief Network.



Deep Belief Networks (DBNs) thường được học bằng cách sử dụng một kỹ thuật gọi là "greedy layer-wise pretraining" (huấn luyện theo từng tầng một theo cách tham lam). Phương pháp này thường áp dụng cho mạng Deep Belief Network được cấu tạo từ các Restricted Boltzmann Machines (RBMs).

Dưới đây là quy trình tổng quát của cách học DBNs:

1. Huấn luyện từng tầng RBM một cách tuần tự:

- Bắt đầu với tầng đầu tiên của DBN, một RBM được huấn luyện với dữ liệu đầu vào. Trong quá trình này, dữ liệu được đưa vào RBM và RBM được huấn luyện để học một biểu diễn hiệu quả của dữ liệu đó thông qua việc điều chỉnh các trọng số và các tham số khác của mạng.

- Khi RBM đầu tiên được huấn luyện, đầu ra của nó (biểu diễn) được sử dụng làm dữ liệu đầu vào cho tầng RBM tiếp theo.

2. Lặp lại quá trình cho các tầng RBM tiếp theo:

- Khi RBM đầu tiên đã được huấn luyện, tiếp tục với việc huấn luyện các RBM tiếp theo trong DBN. Mỗi RBM sau được huấn luyện bằng cách sử dụng biểu diễn được tạo ra từ RBM trước đó làm dữ liệu đầu vào.

- Quá trình này lặp lại cho đến khi tất cả các tầng RBM đã được huấn luyện.

3. Fine-tuning (tinh chỉnh cuối cùng):

- Sau khi tất cả các tầng RBM đã được huấn luyện, một quá trình fine-tuning được thực hiện để cải thiện hiệu suất toàn bộ của DBN. Trong quá trình này, một thuật toán học có giám sát như lan truyền ngược được sử dụng để điều chỉnh các tham số của toàn bộ mạng, bao gồm cả các trọng số và các tham số khác, dựa trên một hàm mất mát (loss function) và dữ liệu huấn luyện.

Quá trình này giúp DBN học được biểu diễn phức tạp của dữ liệu bằng cách lặp lại việc học từng tầng một cách tuần tự và sau đó tinh chỉnh toàn bộ mạng thông qua quá trình fine-tuning.

4. Xây dựng Deep Belief Network.

Xây dựng một Deep Belief Network (DBN) đòi hỏi quá trình lặp đi lặp lại giữa hai giai đoạn chính: huấn luyện từng tầng Restricted Boltzmann Machines (RBMs) và sau đó là tinh chỉnh toàn bộ mạng. Dưới đây là các bước chi tiết để xây dựng một DBN:

1. Xây dựng từng tầng RBM:

- Chọn số lượng tầng và kích thước của mỗi tầng: Quyết định về số lượng và kích thước của mỗi RBM trong DBN, tức là số lượng nơ-ron ẩn và số lượng nơ-ron đầu vào.

- Huấn luyện RBM đầu tiên: Sử dụng dữ liệu đầu vào để huấn luyện RBM đầu tiên bằng cách cập nhật các trọng số của mạng dựa trên kỹ thuật Markov Chain Monte Carlo (MCMC) hoặc Gradient Descent.

- Sử dụng biểu diễn của RBM đầu tiên làm đầu vào cho RBM tiếp theo: Khi RBM đầu tiên đã được huấn luyện, sử dụng biểu diễn của nó làm đầu vào cho RBM tiếp theo. Lặp lại quá trình này cho đến khi tất cả các RBM đã được huấn luyện.

2. Fine-tuning toàn bộ mạng:

- Kết hợp các RBM thành một DBN: Khi tất cả các RBM đã được huấn luyện, kết hợp chúng lại với nhau để tạo thành một DBN.

- Fine-tuning toàn bộ mạng: Sử dụng một thuật toán học có giám sát như lan truyền ngược để điều chỉnh toàn bộ mạng dựa trên một hàm mất mát và dữ liệu huấn luyện. Quá trình này sẽ điều chỉnh các trọng số và các tham số khác của toàn bộ mạng để cải thiện hiệu suất của DBN.

3. Kiểm tra và đánh giá:

- Kiểm tra hiệu suất: Đánh giá hiệu suất của DBN trên dữ liệu kiểm tra để đảm bảo rằng nó hoạt động tốt trên dữ liệu mới mà nó chưa từng thấy trước đó.

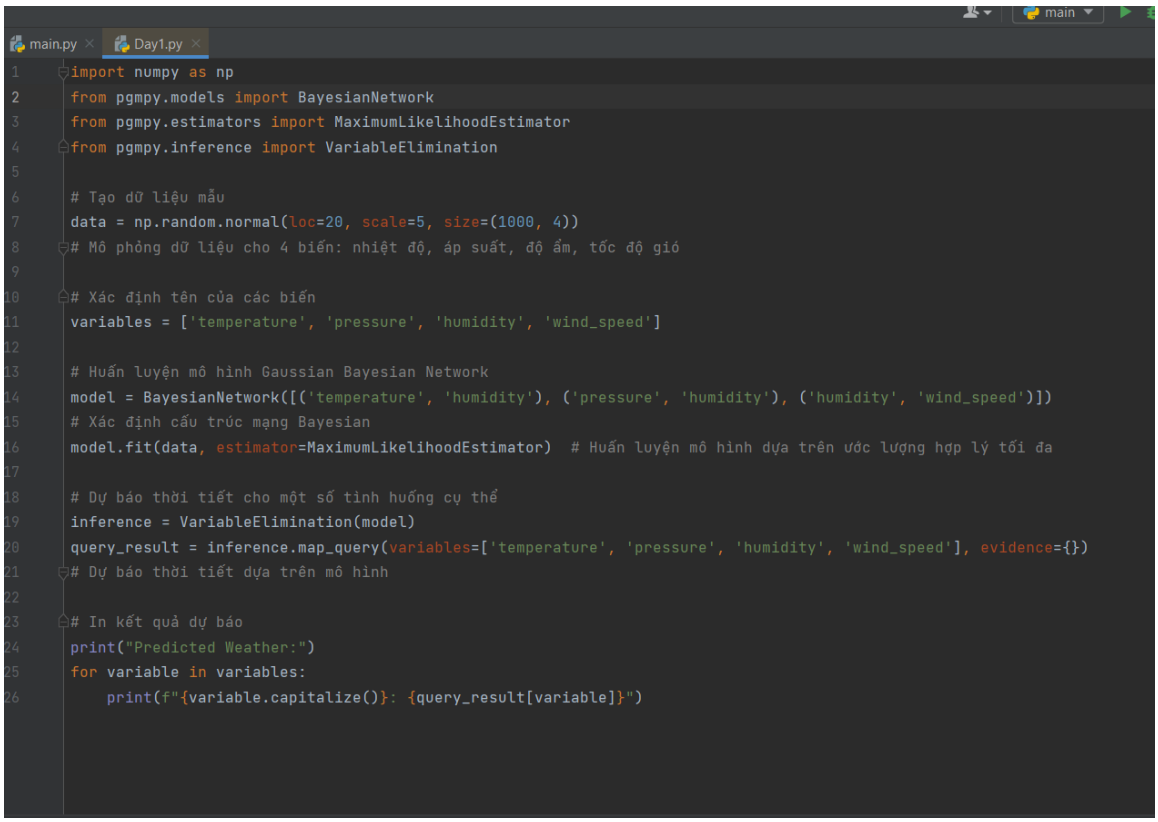
- Điều chỉnh siêu tham số (nếu cần): Nếu cần, điều chỉnh các siêu tham số như số lượng tầng và kích thước của mỗi tầng để cải thiện hiệu suất của mô hình.

Quá trình xây dựng một DBN là một công việc phức tạp đòi hỏi sự hiểu biết sâu về lý thuyết và thuật toán của mạng nơ-ron, cùng với kỹ năng thực hành trong việc triển khai và tinh chỉnh mô hình trên dữ liệu thực tế.

5. Ví dụ.

Xây dựng một chương trình dự báo thời tiết sử dụng Deep 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.

Trong trường hợp dự báo thời tiết, chúng ta có thể xem xét các biến như nhiệt độ, áp suất không khí, độ ẩm, và tốc độ gió.



Trong ví dụ này, chúng ta sử dụng thư viện **pgmpy** để tạo và huấn luyện một mô hình Gaussian Bayesian Network dựa trên dữ liệu mẫu. Đầu tiên, chúng ta tạo dữ liệu mẫu với 4 biến: nhiệt độ, áp suất, độ ẩm và tốc độ gió. Sau đó, chúng ta xác định cấu trúc mạng Bayesian và huấn luyện mô hình bằng cách sử dụng ước lượng hợp lý tối đa (Maximum Likelihood Estimation). Cuối cùng, chúng ta sử dụng mô hình đã huấn luyện để dự báo thời tiết cho một số tình huống cụ thể.