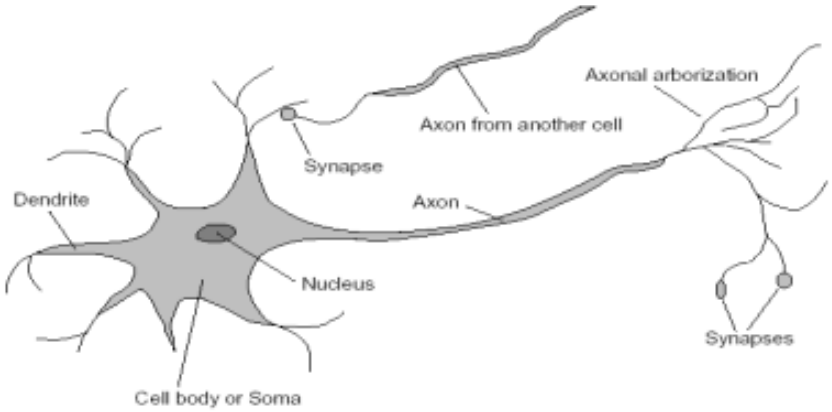
BÀI THỰC HÀNH CSTTNT – BUỔI 8
PHẦN 2- NEURAL NETWORK

I. LÝ THUYẾT

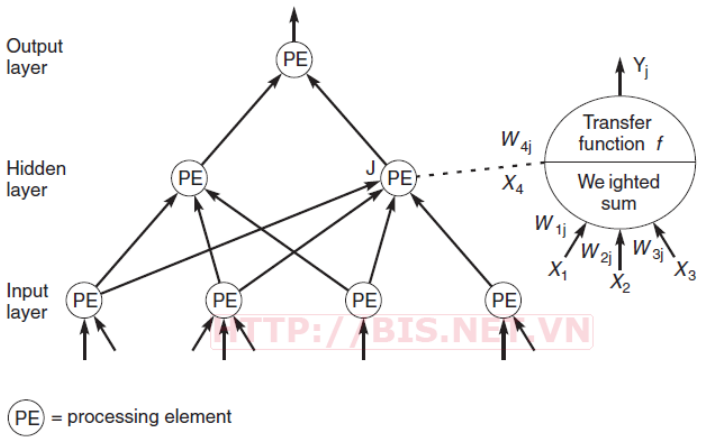
Mạng Neural sinh học



Mạng Neuron nhân tạo (Artificial Neural Network- ANN) là mô hình xử lý thông tin được mô phỏng dựa trên hoạt động của hệ thống thần kinh của sinh vật, bao gồm số lượng lớn các Neuron được gắn kết để xử lý thông tin. ANN giống như bộ não con người, được học bởi kinh nghiệm (thông qua huấn luyện), có khả năng lưu giữ những kinh nghiệm hiểu biết (tri thức) và sử dụng những tri thức đó trong việc dự đoán các dữ liệu chưa biết (unseen data).

Các ứng dụng của mạng Neuron được sử dụng trong rất nhiều lĩnh vực như điện, điện tử, kinh tế, quân sự,… để giải quyết các bài toán có độ phức tạp và đòi hỏi có độ chính xác cao như điều khiển tự động, khai phá dữ liệu, nhận dạng,…

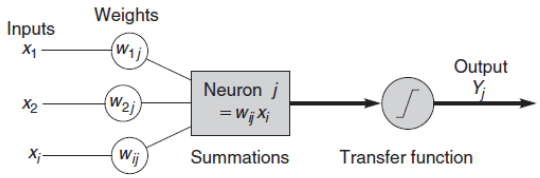
1. Kiến trúc tổng quát của một ANN như sau



Processing Elements (PE): Các PE của ANN gọi là Neuron, mỗi Neuron nhận các dữ liệu vào (Inputs) xử lý chúng và cho ra một kết quả (output) duy nhất. Kết quả xử lý của một Neuron có thể làm Input cho các Neuron khác

Kiến trúc chung của một ANN gồm 3 thành phần đó là ***Input Layer***, ***Hidden Layer*** *và* ***Output Layer*** Trong đó, lớp ẩn (Hidden Layer) gồm các Neuron, nhận dữ liệu input từ các Neuron ở lớp (Layer) trước đó và chuyển đổi các input này cho các lớp xử lý tiếp theo. Trong một ANN có thể có nhiều Hidden Layer.

Quá trình xử lý thông tin của một ANN



Inputs: Mỗi Input tương ứng với 1 thuộc tính (attribute) của dữ liệu (patterns). Ví dụ như trong ứng dụng của ngân hàng xem xét có chấp nhận cho khách hàng vay tiền hay không thì mỗi Input là một thuộc tính của khách hàng như thu nhập, nghề nghiệp, tuổi, số con,…

Output: Kết quả của một ANN là một giải pháp cho một vấn đề, ví dụ như với bài toán xem xét chấp nhận cho khách hàng vay tiền hay không thì output là yes (cho vay) hoặc no (không cho vay).

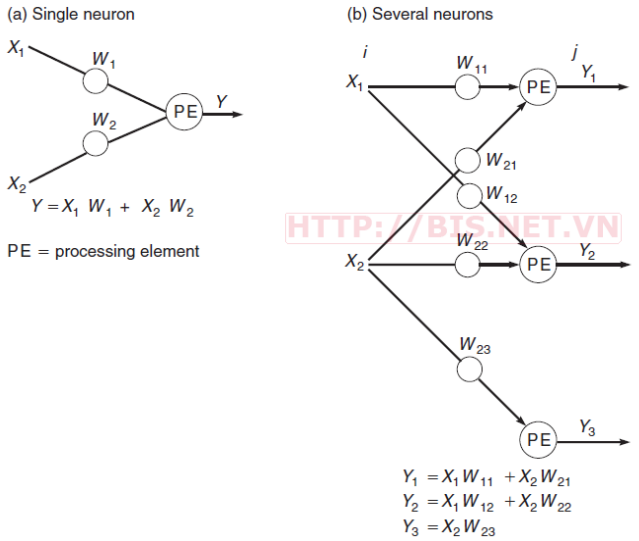
Connection Weights (Trọng số liên kết) : Đây là thành phần rất quan trọng của một ANN, nó thể hiện mức độ quan trọng (độ mạnh) của dữ liệu đầu vào đối với quá trình xử lý thông tin (quá trình chuyển đổi dữ liệu từ Layer này sang layer khác). Quá trình học (Learning Processing) của ANN thực ra là quá trình điều chỉnh các trọng số (Weight) của các input data để có được kết quả mong muốn.

Summation Function (Hàm tổng): Tính tổng trọng số của tất cả các input được đưa vào mỗi Neuron (phần tử xử lý PE). Hàm tổng của một Neuron đối với n input được tính theo công thức sau:

$$Y = \sum_{i=1}^{n} X_i W_i$$

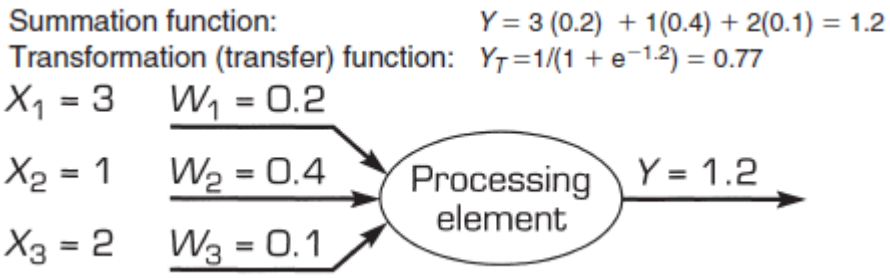
Hàm tổng đối với nhiều Neurons trong cùng một Layer (Xem hình b):

$$Yj = \sum_{i=1}^{n} X_i W_{ij}$$



Transformation (Transfer) Function (Hàm chuyển đổi)

Hàm tổng (Summation Function) của một Neuron cho biết khả năng kích hoạt (Activation) của neuron đó còn gọi là kích hoạt bên trong (internal activation). Các Neuron này có thể sinh ra một output hoặc không trong ANN (nói cách khác rằng có thể output của 1 Neuron có thể được chuyển đến layer tiếp trong mạng Neuron theo hoặc không). Mối quan hệ giữa Internal Activation và kết quả (output) được thể hiện bằng hàm chuyển đổi (Transfer Function).



Việc lựa chọn Transfer Function có tác động lớn đến kết quả của ANN. Hàm chuyển đổi phi tuyến được sử dụng phổ biến trong ANN là **sigmoid (logical activation) function.**

$$Y_T = 1/(1 + e^{-Y})$$

Trong đó :

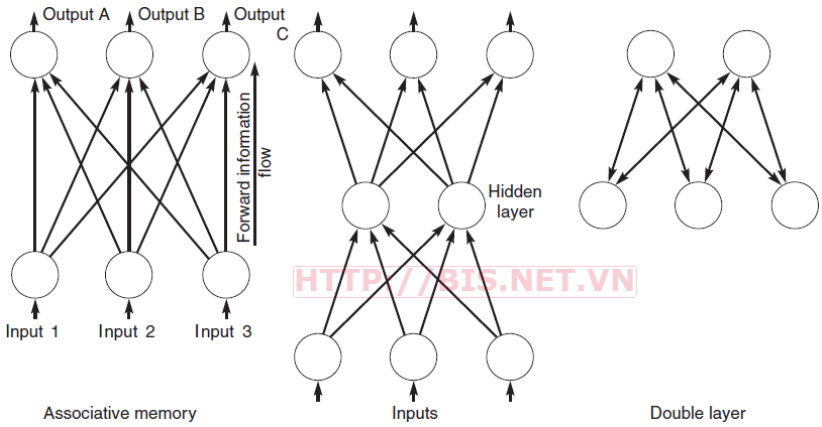
Y_T: Hàm chuyển đổi

Y: Hàm tổng

Kết quả của Sigmoid Function thuộc khoảng [0,1] nên còn gọi là hàm chuẩn hóa (Normalized Function).

Đôi khi thay vì sử dụng Transfer Function người ta sử dụng giá trị ngưỡng (Threshold value) để kiểm soát các output của các neuron tại một layer nào đó trước khi chuyển các output này đến các Layer tiếp theo. Nếu output của một neuron nào đó nhỏ hơn Threshold thì nó sẻ không được chuyển đến Layer tiếp theo.

Một số kiến trúc của ANN



2. Quá trình học (Learning Processing) của ANN

ANN được huấn luyện (Training) hay được học (Learning) theo 2 kỹ thuật cơ bản đó là học có giám sát (Supervised Learning) và học không giám sát (Unsupervised Learning).

Supervised learning: Quá trình Training được lặp lại cho đến kết quả (output) của ANN đạt được giá trị mong muốn (Desired value) đã biết. Điển hình cho kỹ thuật này là mạng Neuron lan truyền ngược (**Backpropagation**).

Unsupervised learning: Không sử dụng tri thức bên ngoài trong quá trình học (Learning), nên còn gọi là tự tổ chức (Self – Organizing). Mạng Neuron điển hình được huấn luyện theo kiểu Unsupervised là Sefl – Organizing Map (SOM).

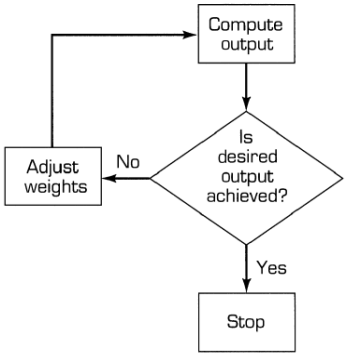
Thuật ngữ **"epoch"** được dùng để mô tả quá trình khi tất cả các input patterns của training set được đưa để huấn luyện mạng. Nói cách khách 1 epoch được hoàn thành khi tất cả các dữ liệu trong training set được đưa vào huấn luyện mạng. Vì vậy số lượng **"epoch"** xác định số lần mạng được huấn luyện (hay số lần đưa tất cả các dữ liệu trong training set vào mạng).

Quá trình học của Supervised ANN được mô tả như sau:

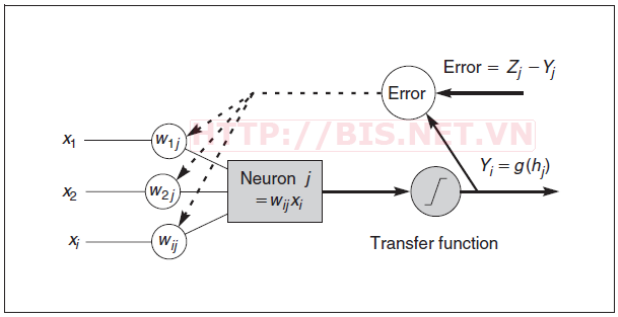
1.Tính giá trị output .

2. So sánh output với giá trị mong muốn (desired value).

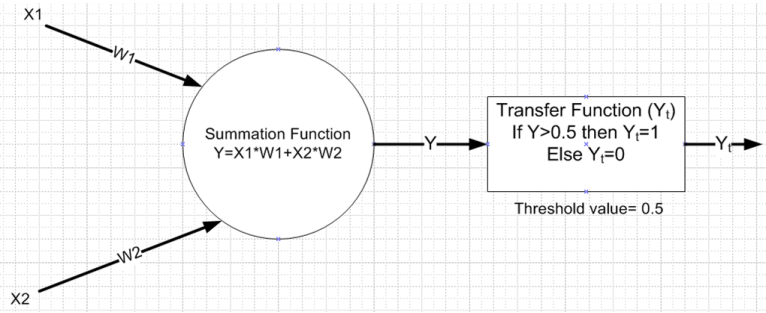
3. Nếu chưa đạt được giá trị mong muốn thì hiệu chỉnh trọng số (weights) và tính lại output



Giả sử sau khi tính toán từ các input value đã cho, ta có output là Y. Giá trị mong muốn (Desired) là Z đã biết trước. Sự chênh lệch giữa Y và kết quả mong muốn Z được biểu diễn bởi tham số delta (gọi là lỗi) = **Z-Y**. Mục đích của việc Learning là làm sao cho delta càng nhỏ càng tốt (Nếu delta =0 là hoàn hảo nhất) bằng cách điều chỉnh trọng số (weights) của các dữ liệu vào.



Sau đây minh họa việc huấn luyện một ANN đơn giản gồm 1 Neuron học để thực hiện phép toán giữa 2 input là 2 toán hạng X1 và X2. Nếu một trong 2 input là positive (1) thì kết quả là Positive. Kiến trúc ANN như sau :



Ví dụ đơn giản về ANN

Các Input patterns và Desired value (dự đoán) như sau:

	Inputs		
Case	X_1	X_2	*Desired Results*
1	0	0	0
2	0	1	1 (positive)
3	1	0	1 (positive)
4	1	1	1 (positive)

Neuron phải được huấn luyện để nhận biết các dữ liệu vào (input pattern) ở đây là một bộ gồm 2 giá trị X1 và X2 (trong ví dụ trên có 4 bộ dữ liệu) và phân loại chúng vào các kết quả đã biết (Desired Results)

Quá trình học được diễn ra như sau:

4 Input patterns được lần lượt đưa vào Neuron, các trọng số ban đầu được khởi tạo ngẫu nhiên và được điều chỉnh sau mỗi vòng lặp. Bước này lặp lại cho đến khi các trọng số hội tụ đến tập các giá trị cho phép ANN phân loại chính xác 4 input patterns.

	A	B	C	D	E	F	G	H	I	J
1				Dự đoán	initial weights				Final Weights	
2	**Bước**	**X1**	**X2**	**Z**	**W1**	**W2**	**Y**	**delta**	**W1**	**W2**
3		0	0	0	0.1	0.3	0	0.0	0.1	0.3
4	**1**	0	1	1	0.1	0.3	0	1.0	0.1	0.5
5		1	0	1	0.1	0.5	0	1.0	0.3	0.5
6		1	1	1	0.3	0.5	1	0.0	0.3	0.5
7		0	0	0	0.1	0.3	0	0.0	0.1	0.3
8	**2**	0	1	1	0.1	0.5	0	1.0	0.1	0.7
9		1	0	1	0.3	0.5	0	1.0	0.5	0.5
10		1	1	1	0.3	0.5	1	0.0	0.3	0.5
11		0	0	0	0.1	0.3	0	0.0	0.1	0.3
12	**3**	0	1	1	0.1	0.7	1	0.0	0.1	0.7
13		1	0	1	0.5	0.5	0	1.0	0.7	0.5
14		1	1	1	0.3	0.5	1	0.0	0.3	0.5
15		0	0	**0**	0.1	0.3	**0**	0.0	0.1	0.3
16	**4**	0	1	**1**	0.1	0.7	**1**	0.0	0.1	0.7
17		1	0	**1**	0.7	0.5	**1**	0.0	0.7	0.5
18		1	1	**1**	0.3	0.5	**1**	0.0	0.3	0.5

Trong ví dụ này giá trị ngưỡng (Threshold) để đánh giá hàm tổng của các input là 0.5. Sau khi tính toán các output (Y), và sử dụng Kết quả dự đoán (Desired Result - Z) để tính toán lỗi delta và delta được sử dụng để cập nhật trọng số của các input: **delta = Z_j-Y_j**

Hiệu chỉnh trọng số: **W_i(final) = W_i(initial) + alpha*delta*X_i**

Trong đó alpha =0.2 là tham số kiểm soát tốc độ học của ANN gọi là **Learning rate**. Việc chọn tham số Learning rate phù hợp (làm tăng độ chính xác) là vấn đề rất quan trọng khi triển khai ANN.

Learning rate – Tốc độ học là một siêu tham số sử dụng trong việc huấn luyện các mạng nơ ron. Giá trị của nó là một số dương, thường nằm trong khoảng giữa 0 và 1. Tốc độ học kiểm soát tốc độ mô hình thay đổi các trọng số để phù hợp với bài toán. Tốc độ học lớn giúp mạng nơ ron được huấn luyện nhanh hơn nhưng cũng có thể làm giảm độ chính xác.

II. BÀI TẬP

Bài 1: Sử dụng Perceptron Neural Network để huấn luyện

```
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import Perceptron
import sklearn.metrics as metric
import numpy as np

X_training=[[1, 1],
            [1, 0],
            [0, 1],
            [0, 0]
           ]
y_training=[1,
            1,
            1,
            0
           ]
X_testing=X_training
y_true=y_training

ptn = Perceptron(max_iter=500)                          # set the method
ptn.fit(X_training, y_training)                         # training
y_pred=ptn.predict(X_testing)                           # prediction
print(y_pred)                                           # show the output
accuracy=metric.accuracy_score(y_true, y_pred, normalize=True)
print('acuracy=',accuracy)                              # show accracy score
```

```
[1 1 1 0]
acuracy= 1.0
```

Bài 2: Sử dụng Multi-Layer Perceptron (MLP) để huấn luyện

```
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import Perceptron
import sklearn.metrics as metric
import numpy as np

X_training=[[1, 1],
            [1, 0],
            [0, 1],
            [0, 0]
           ]
y_training=[1,
            1,
            1,
            0
           ]
X_testing=X_training
y_true=y_training


mlp = MLPClassifier(solver='lbfgs', hidden_layer_sizes=[1,1], activation='logistic') # set the method
mlp.fit(X_training, y_training)                    # training
y_pred=mlp.predict(X_testing)                      # prediction
print(y_pred)                                      # show the output
accuracy=metric.accuracy_score(np.array(y_true).flatten(), np.array(y_pred).flatten(), normalize=True)
print('acuracy=',accuracy)                         # show accuracy score
```

```
[1 1 1 0]
acuracy= 1.0
```

Bài 3: Neural Network có 3 tham số đầu vào và 2 tham số đầu ra

```
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import Perceptron
import sklearn.metrics as metric
import numpy as np

X_training=[[ 1,  1,  0],
            [ 1, -1, -1],
            [-1,  1,  1],
            [-1, -1,  1],
            [ 0,  1, -1],
            [ 0, -1, -1],
            [ 1,  1,  1]
           ]
y_training=[[1, 0],
            [0, 1],
            [1, 1],
            [1, 0],
            [1, 0],
            [1, 1],
            [1, 1]
           ]
X_testing=X_training
y_true=y_training

mlp = MLPClassifier(solver='lbfgs', hidden_layer_sizes=[3,2], activation='logistic') # set the method
mlp.fit(X_training, y_training)                    # training
y_pred=mlp.predict(X_testing)                      # prediction
print(y_pred)                                      # show the output
accuracy=metric.accuracy_score(np.array(y_true).flatten(), np.array(y_pred).flatten(), normalize=True)
print('acuracy=',accuracy)                         # show accuracy score
```

```
[[1 0]
 [0 1]
 [1 1]
 [1 0]
 [1 0]
 [1 1]
 [1 1]]
acuracy= 1.0
```

Bài 4: Áp dụng Neural Network để huấn luyện bài toán sau

Tạo file .csv trong Excel như sau:

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Dùng Neural Network để giải quyết bài toán sau:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?