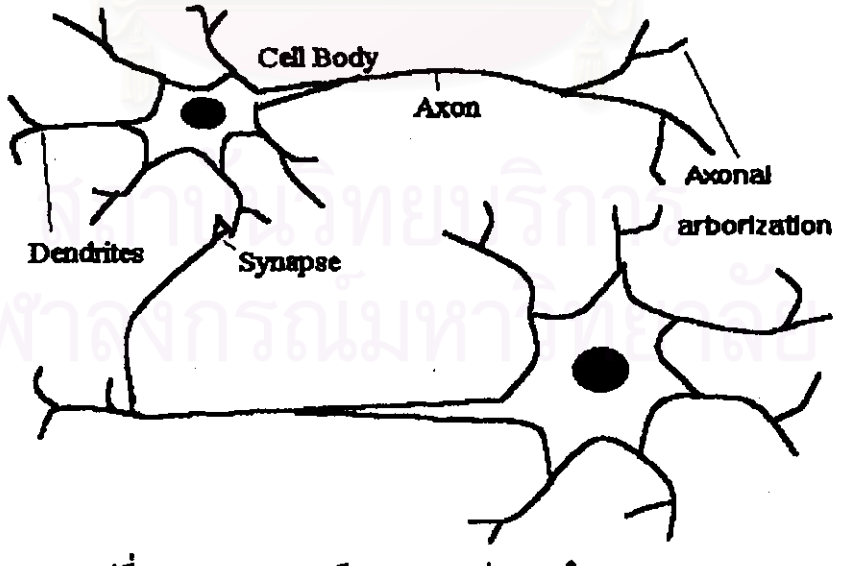
บทที่ 2
ความรู้เบื้องต้นของทฤษฎีเครือข่ายนิวรอล

2.1 ประวัติความเป็นมาของเครือข่ายนิวรอล

เครือข่ายนิวรอลเกิดขึ้นเนื่องจากนักวิทยาศาสตร์ได้ศึกษาการทำงานของสมองมนุษย์จนกระทั่งรู้ว่าสมองประกอบด้วยนิวรอน(Neuron)หรือเซลสมอง[3] ซึ่งเป็นหน่วยพื้นฐานที่ประกอบขึ้นเป็นเนื้อเยื่อสมอง นิวรอนมีส่วนประกอบสำคัญคือตัวเซล (Cell Body) ที่มีนิวเคลียส (Nucleus) อยู่ภายในถัดจากตัวเซลคือแอกซอน (Axon) และเดนไดรต์ (Dendrites) เป็นเส้นใยประสาท (Fiber)ทำหน้าที่รับส่งสัญญาณประสาท ซึ่งเดนไดรทส์ประกอบด้วยเส้นใยประสาท (Fiber) จำนวนมากอยู่รอบๆตัวเซลทำหน้าที่รับสัญญาณประสาทจากเซลอื่นๆ ส่วนแอกซอนเป็นเส้นใยประสาทเส้นเดี่ยวยื่นยาวออกจากตัวเซลทำหน้าที่ส่งสัญญาณประสาทออกจากตัวเซลไปยังเซลอื่นๆ ที่ส่วนปลายของแอกซอนจะแยกเป็นแขนงเรียกว่า Axonal Arborization ซึ่งส่วนปลายของ Axonal Arborization มีจุดเชื่อมต่อเล็กๆเรียกว่าจุดประสานประสาท (Synapses) ในแต่ละ Axonal Arborization มีจำนวนจุดประสานประสาทไม่เท่ากันซึ่งทำให้เกิดระดับการเชื่อมต่อ (Connection Strength) ที่ไม่เท่ากันอีกทั้งจำนวน จุดประสานประสาท ดังกล่าวสามารถเปลี่ยนแปลงได้ตลอดช่วงชีวิต ซึ่งระดับการเชื่อมต่อนี้เองคือความสามารถในการจดจำ (Memory) ความคิด(Thinking) ฯลฯ ของสมอง[4] และเกิดขึ้นโดยการเรียนรู้ (Learning) ซึ่งจะทำให้จำนวนจุดประสานประสาทเปลี่ยนแปลงได้ รูปที่ 2.1 แสดงถึงโครงสร้างสมองทั้งหมดที่กล่าวมาข้างต้น



รูปที่ 2.1 แสดงรายละเอียดของเซลประสาทในสมอง

สมองประกอบด้วยนิวรอนเป็นจำนวนถึง 10^{11} นิวรอน และด้วยจำนวนนิวรอนที่มากมายดังกล่าวพร้อมกับการประมวลผลแบบขนานหมายถึงแต่ละนิวรอนช่วยกันประมวลผลพร้อมๆกัน ทำให้

หอสมุดกลาง สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

สมองมีความสามารถมากกว่าเครื่องคอมพิวเตอร์ ถึงแม้ว่าในส่วนย่อยแล้วการประมวลผลของนิวรอนจะช้ากว่าวงจรตรรกะ(Logic Circuit) มากก็ตาม

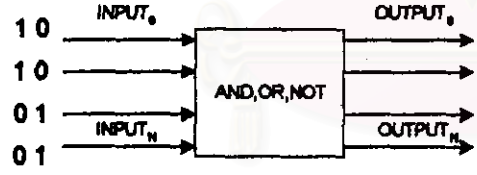
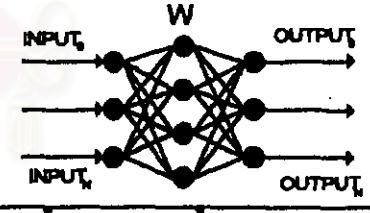
หลังจากที่นักวิทยาศาสตร์ทำการศึกษาจนเข้าใจหลักการประมวลผลของสมองแล้ว จึงสร้างแบบจำลองเชิงคณิตศาสตร์ (Mathematical Model) ขึ้นมาเพื่อจำลองการทำงานของเซลสมองเรียกว่าเครือข่ายนิวรอล อย่างไรก็ตามถึงปัจจุบันนี้เครือข่ายนิวรอลก็ยังไม่สามารถเทียบเคียงได้กับระบบเครือข่ายนิวรอนในสมองจริง แต่เครือข่ายนิวรอลขนาดเล็กสามารถนำไปประยุกต์ใช้ในงานต่างๆ เช่น

- ใช้ในงานการรู้จำแบบ (Pattern Recognition) เช่น วิเคราะห์ตัวอักษร รูปภาพ คำพูด[7] เป็นต้น
- ใช้ในงานระบบควบคุมเช่น ระบบขับขี่อัตโนมัติ (Automatic Driving)[7] เป็นต้น

2.2 หลักการทำงานของเครือข่ายนิวรอล

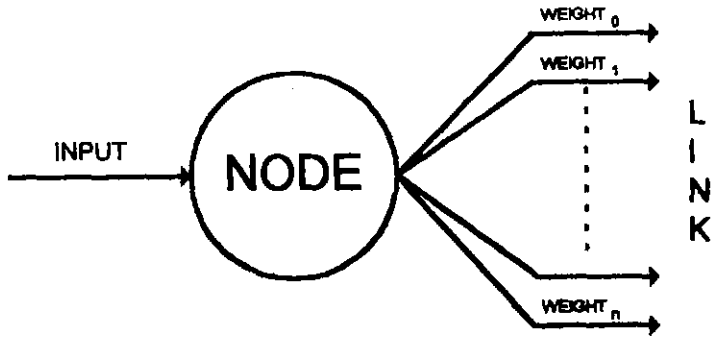
หลักการทำงานพื้นฐานของเครือข่ายนิวรอลมีลักษณะคล้ายกับวงจรตรรกะความแตกต่างที่สำคัญของเครือข่ายนิวรอลกับวงจรตรรกะอยู่ที่สัญญาณนำเข้า(Input)ของเครือข่ายนิวรอลเป็นสัญญาณเชิงอุปมาน (Analog) ในขณะที่สัญญาณนำเข้าของวงจรตรรกะเป็นสัญญาณเชิงเลข (Digital) และมีข้อปลีกย่อยตามตารางที่ 2.1

ตารางที่ 2.1 เปรียบเทียบการทำงานของเครือข่ายนิวรอลกับวงจรตรรกะ

วงจรตรรกะ	เครือข่ายนิวรอล
	
คำนวณโดยใช้พีชคณิตแบบบูล(Boolean Algebra)	คำนวณโดยใช้พีชคณิตเชิงเส้น(Linear Algebra)
สัญญาณคำนำเข้า/คำนำออก(Output)เป็นสัญญาณเชิงเลข	สัญญาณคำนำเข้าเป็นแบบเชิงอุปมานสัญญาณคำนำออกเป็นแบบเชิงอุปมานหรือแบบเชิงเลข

ส่วนประกอบของเครือข่ายนิวรอล มี 3 ส่วนคือ โนด(Node) ส่วนเชื่อมโยง (Links) และ ค่าน้ำหนัก (Weight)

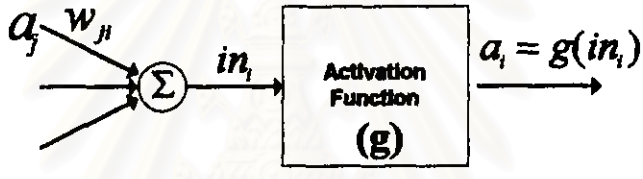
โนดเทียบได้กับตัวเซลซึ่งแต่ละโนดติดต่อกันด้วย ส่วนเชื่อมโยง และแต่ละส่วนเชื่อมโยงมีค่าประจำตัวคือ ค่าน้ำหนักซึ่งเทียบได้กับ Synapse ของแอกซอนดังรูป 2.2



รูปที่ 2.2 แสดงโนดพื้นฐานของเครือข่ายนิวรอล

2.3 โครงสร้างของนิวรอล

โครงสร้างของนิวรอลแสดงในรูปที่ 2.3



รูปที่ 2.3 รูปแบบการคำนวณในโนดพื้นฐาน

จากรูปแสดงถึงนิวรอลเพียง 1 หน่วยซึ่งเป็นรูปแบบการคำนวณพื้นฐานที่สุดในระบบเครือข่ายนิวรอลตามสมการต่อไปนี้

$$in_i = \Sigma(w_{ji} a_j) \quad \text{.....(2.1)}$$

$$a_i = g(\Sigma w_{ji} a_j) = g(in_i) \quad \text{.....(2.2)}$$

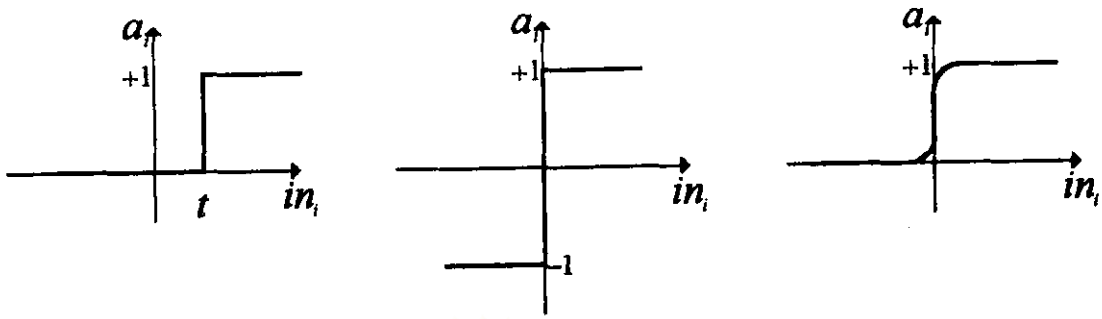
จากสมการดังกล่าวสามารถแยกการคำนวณออกเป็น 2 ส่วน

- ส่วน Linear Component เรียกว่า Input Function $in_i = \Sigma(w_{ji} a_j)$
- ส่วน Non Linear Component เรียกว่า Activation Function หรือฟังก์ชันการถ่ายโอน (Transfer Function) $a_i = g(in_i)$ ซึ่งทำหน้าที่ควบคุมสัญญาณค่านำออกให้อยู่ในขอบเขตจำกัดไม่สูงเกินไปเนื่องจากสัญญาณค่านำออก จะถูกส่งต่อไปยังโนดถัดไปอีกหลายโนด

ฟังก์ชันการถ่ายโอนมีหลายชนิดแต่ที่สำคัญคือ

- Step Function
- Sign Function
- Sigmoid Function

รูปที่ 2.4 แสดงถึงฟังก์ชันการถ่ายโอน แบบต่างๆ



a) Step Function b) Sign Function c) Sigmoid Function

รูปที่ 2.4 แสดงฟังก์ชันการถ่ายโอนพื้นฐานของเครือข่ายนิวรอล

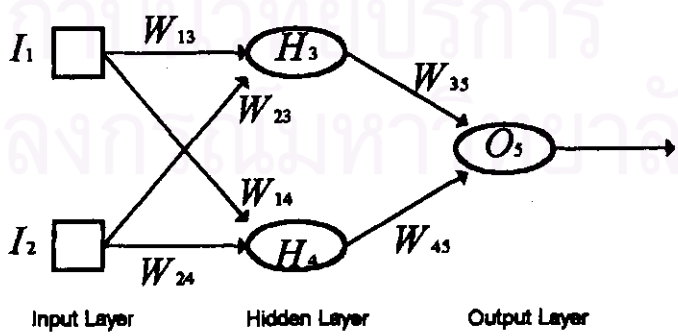
2.4 ชนิดของเครือข่ายนิวรอล

เครือข่ายนิวรอลมีหลายชนิดแต่เครือข่ายนิวรอลที่นำมาประยุกต์ใช้งานอย่างกว้างขวางคือแบบ Feed - Forward Network ซึ่งประกอบด้วย

- Input Layer
- Hidden Layer
- Output Layer

ลักษณะการคำนวณเป็นแบบทิศทางเดียว (Unidirectional) โดยที่โนดในแต่ละชั้น(Layer) คำนวณและส่งผลลัพท์ไปที่ชั้นถัดไปไม่มีการย้อนกลับ

รูปแบบการคำนวนของ Feed - Forward Network มีอยู่จริงในส่วนของสมองที่ทำหน้าที่เก็บความทรงจำ[7] เนื่องจากมีคุณสมบัติที่สำคัญคือ ความสามารถในการเก็บแบบรูปในลักษณะ Long-Term Storage รูปที่ 2.5 แสดงถึงโครงสร้างของ Feed - Forward Network



รูปที่ 2.5 ลักษณะโครงสร้างทั่วไปของ Feed-Forward Network

จากรูปที่ 2.5 ชั้นซ้ายมือเป็นส่วนของ Input Units ในขณะที่ ชั้น ขวามือสุดเป็นส่วนของ Output Units และกลุ่มของ ชั้น ตรงกลางระหว่าง Input Layer กับ Output Layer เรียกว่า Hidden Layer

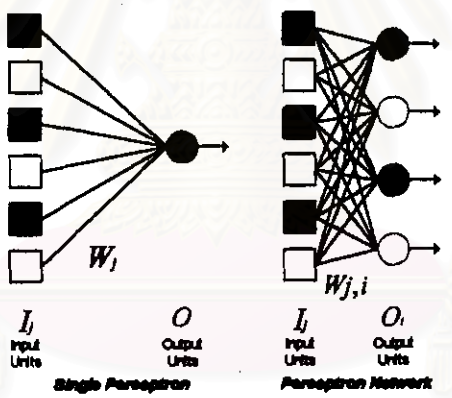
Feed-Forward Network ที่ไม่มีชั้น Hidden Layer เป็น Feed-Forward Network แบบที่สามารถทำความเข้าใจได้ง่ายการทำงานไม่ซับซ้อนซึ่งจากรูปที่ 2.5 สามารถเขียนสมการ การคำนวณของ Feed-Forward Network ดังนี้

$$a_5 = g(w_{35}a_3 + w_{45}a_4)$$
$$= g(w_{35}g(w_{13}a_1 + w_{23}a_2)) + w_{45}g(w_{14}a_1 + w_{24}a_2)) \quad \text{.....(2.3)}$$

เมื่อ g คือ ฟังก์ชันการถ่ายโอน และ a_i เป็นค่าน่าออกของโนดที่ i

2.5 Perceptron

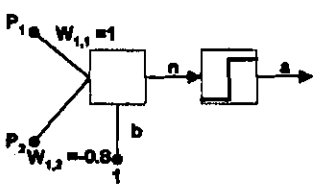
เหตุผลที่ต้องศึกษา Perceptron เพราะว่า Perceptron เป็นหน่วยพื้นฐานของเครือข่ายนิวรอลทุกชนิดการศึกษาจนเข้าใจหลักการทำงานของ Perceptron เป็นอย่างดีจะช่วยให้เราสามารถวิเคราะห์การทำงานของเครือข่ายนิวรอลชนิดต่างๆได้ง่ายรูปที่ 2.6 แสดงแผนภาพบล็อก(Block Diagram)ของ Perceptron และ Perceptron Network



รูปที่ 2.6 เปรียบเทียบการคำนวณแบบ Single Perceptron กับ Perceptron Network

2.5.1 การทำงานของ Perceptron

พิจารณา Single Perceptron แบบ 2 ค่านำเข้าแสดงดังรูปที่ 2.7



รูปที่ 2.7 ลักษณะทั่วไปของ Single Perceptron

สมมุติว่า Perceptron ดังกล่าวมีค่าค่าน้ำหนักเริ่มต้นเป็น $W_{11} = 1.0$; $W_{12} = -0.8$ และเมื่อผ่านกระบวนการเรียนรู้เพื่อให้ Perceptron สามารถคำนวณความสัมพันธ์ระหว่างค่านำเข้ากับค่านำออกตามเซต (Set) ต่อไปนี้ได้ถูกต้อง

$$\left(P_1=\begin{bmatrix}1\\2\end{bmatrix},t_1=1\right),\left(P_2=\begin{bmatrix}-1\\2\end{bmatrix},t_2=0\right),\left(P_3=\begin{bmatrix}0\\-1\end{bmatrix},t_3=0\right)$$

ในกรณีที่ค่า เมทริกซ์ของน้ำหนักเริ่มต้นเป็น $W_1=\begin{bmatrix}1\\-0.8\end{bmatrix}$ และ Input Matrix เป็น $P_1=\begin{bmatrix}1\\2\end{bmatrix}$

$$a = hardlim((W_1)^t P_1) = hardlim\left([1-0.8]\begin{bmatrix}1\\2\end{bmatrix}\right)$$

$$a = hardlim(-0.6) = 0$$

ซึ่งค่า a ที่ได้ไม่ตรงกับที่ต้องการดังนั้นจึงจะต้องมีการเปลี่ยนแปลงค่าน้ำหนักเพื่อทำให้ Perceptron สามารถคำนวณค่าได้ถูกต้อง โดยสมการในการปรับเปลี่ยนค่าน้ำหนัก เป็นดังนี้

$$W^{new} = W^{old} + (t-a)P \quad \text{.....(2.4)}$$

จาก $\left(P_1=\begin{bmatrix}1\\2\end{bmatrix},t_1=1\right)$

$$W^{new} = W^{old} + (1-0)P_1 = \begin{bmatrix}1\\-0.8\end{bmatrix}+\begin{bmatrix}1\\2\end{bmatrix}=\begin{bmatrix}2\\1.2\end{bmatrix}$$

$$a_1 = hardlim((W^{new})^t P_1) = hardlim\left([2 \quad 1.2]\begin{bmatrix}1\\2\end{bmatrix}\right) = hardlim(4.4)$$

ดังนั้นได้ค่า $a_1 = 1$

เมื่อ $\left(P_2=\begin{bmatrix}-1\\2\end{bmatrix},t_2=0\right)$

$$a = hardlim((W_2)^t P_2) = hardlim\left([2 \quad 1.2]\begin{bmatrix}-1\\2\end{bmatrix}\right) = hardlim(0.4) = 1$$

ดังนั้นได้ค่า $a = 1$

ดังนั้น $W^{new} = W^{old} + (t_2 - a)P_2 = \begin{bmatrix}2\\1.2\end{bmatrix}+(0-1)\begin{bmatrix}-1\\2\end{bmatrix}=\begin{bmatrix}3\\-0.8\end{bmatrix}$

เมื่อ $\left(P_3=\begin{bmatrix}0\\-1\end{bmatrix},t_3=0\right)$

$$a = hardlim(0.8) = 1$$

ดังนั้น $W^{new} = W^{old} + (t_3 - a)P_3 = \begin{bmatrix}3\\-0.8\end{bmatrix}+(0-1)\begin{bmatrix}0\\-1\end{bmatrix}=\begin{bmatrix}3\\0.2\end{bmatrix}$

เมทริกซ์ของน้ำหนัก ที่ได้ในขั้นตอนสุดท้ายเท่ากับ

$$W=\begin{bmatrix}3\\0.2\end{bmatrix}$$

ขั้นตอนสุดท้ายเราจะได้เมทริกซ์ของน้ำหนักที่ทำให้ Perceptron สามารถคำนวณผลลัพท์ตรงกับที่กำหนด ซึ่งทดสอบได้ดังนี้

จากเซ็ท $\left(P_1=\begin{bmatrix}1\\2\end{bmatrix},t_1=1\right)$

$$a=hardlim((W)^tP)=hardlim\left(\begin{bmatrix}3 & 0.2\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}\right)=1$$

จากเซ็ท $\left(P_2=\begin{bmatrix}-1\\2\end{bmatrix},t_2=0\right)$

$$a=hardlim((W)^tP)=hardlim\left(\begin{bmatrix}3 & 0.2\end{bmatrix}\begin{bmatrix}-1\\2\end{bmatrix}\right)=0$$

และจากเซ็ท $\left(P_3=\begin{bmatrix}0\\-1\end{bmatrix},t_3=0\right)$

$$a=hardlim((W)^tP)=hardlim\left(\begin{bmatrix}3 & 0.2\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\right)=0$$

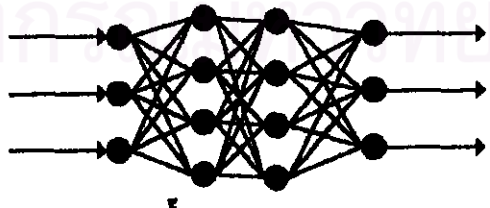
จากรายละเอียดดังกล่าวสามารถสรุปได้ว่าสมการที่ใช้ในการเปลี่ยนค่าน้ำหนักของ Perceptron เขียนได้เป็น

$$W^{new}=W^{old}+(t-a)P=W^{old}+(error)P \quad .....(2.5)$$

2.6 Multilayer Feed-Forward Network

Multilayer Feed-Forward Network ถูกพัฒนาต่อมาหลังจากปี 1950 หลังจากที่พบว่า Perceptron ไม่สามารถเรียนรู้แบบรูปที่มีจำนวนมาก และซับซ้อนได้

โครงสร้างของ Multilayer Feed-Forward Network ประกอบด้วยชั้นจำนวน 3 ชั้นคือ Input Layer , Hidden Layer , Output Layer โดยเฉพาะในส่วนของ Hidden Layer มีได้หลาย ชั้น ดังรูปที่ 2.9



รูปที่ 2.8 แสดงลักษณะพื้นฐานของ Multilayer Feed-Forward Network

กฎเกณฑ์การเรียนรู้ของ Multilayer Feed-Forward Network มีอยู่หลายแบบซึ่งกฎเกณฑ์ที่นิยมมากที่สุดคือ Back-Propagation ที่พัฒนาขึ้นในปี 1969 โดย Bryson&Ho ซึ่งมีการนำมาใช้งานได้จริง

หลังจากปี 1980 แต่เนื่องจากวิธี Back-Propagation ต้องการการคำนวณแบบวนซ้ำ (Loop) ซึ่งต้องใช้เครื่อง คอมพิวเตอร์ช่วยในการคำนวณ

2.6.1 Back-Propagation Learning

การใช้เครือข่ายนิวรอลในการเรียนรู้แบบรูปที่มีขนาดใหญ่อีกทั้งมีหลายแบบรูปตัวอย่างเช่น แบบรูปที่เกิดจากสมการเชิงเส้น

$$z = 3x + 2y \; ; \; x \& y \text{ เป็นจำนวนเต็มบวก}$$

แบบรูปที่เกิดจากสมการตัวอย่างมีจำนวนไม่จำกัดดังในตาราง

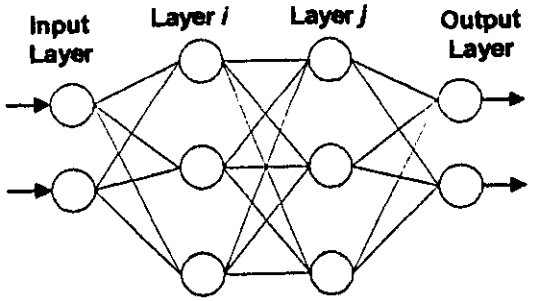
ตารางที่ 2.2 ค่าของตัวเลขที่ต้องการให้เครือข่ายนิวรอลเรียนรู้

Z	X	Y
0	0	0
2	0	1
4	0	2
↓	↓	↓
∞	∞	∞

เครือข่ายนิวรอลแบบ Perceptron ไม่สามารถเรียนรู้แบบรูปที่มีจำนวนมากดังกล่าวข้างต้นได้ แต่เมื่อถูกพัฒนาขึ้นเป็นเครือข่ายนิวรอลแบบ Multilayer Feed-forward Network ซึ่งมีการพัฒนากฎเกณฑ์ของการเรียนรู้ (Learning Method) ที่ใช้กับ Multilayer Feed-Forward Network ด้วยกฎเกณฑ์ของการเรียนรู้ที่นิยมใช้กันมากคือ Back-Propagation Learning

หลักการของ Back-Propagation Learning คือใช้การคำนวณค่าแตกต่างระหว่างค่านำออกที่คำนวณได้จากเครือข่ายนิวรอล กับค่าที่ต้องการ(Target) แล้วนำค่าแตกต่างดังกล่าวใช้คำนวณเปลี่ยนน้ำหนักของ Perceptron ในแต่ละชั้นลักษณะการเปลี่ยนน้ำหนักเป็นไปในทางที่ทำให้ค่าแตกต่างลดลงแบบเฉลี่ยในทุกค่าน้ำหนัก

ใน Perceptron การปรับเปลี่ยนค่าน้ำหนักทำได้ง่ายเนื่องจากมีค่าน้ำหนัก เพียงชุดเดียวระหว่างค่านำเข้ากับค่านำออก แต่ใน Multilayer Network มีค่าน้ำหนักประจำชั้นที่อยู่ระหว่างค่ารับเข้ากับค่านำออก รูปที่ 2.9 แสดง Two Layer Feed-Forward Network



รูปที่ 2.9 ลักษณะการเรียนรู้แบบ Back- Propagation Learning

กฎการเปลี่ยนค่าน้ำหนักในช่วงการเชื่อมโยงจาก Unit j ไปยัง Unit i เขียนเป็นสมการดังนี้

$$W_{ji} \longleftarrow W_{ji} + \alpha \times a_j \times Err_i \times g'(in_i) \quad \text{.....(2.6)}$$

โดยที่ $Err_i = (T_i - O_i)$ และ a_j คือค่านำเข้าของ Perceptron ในชั้นนั้นๆ

$g'(in_i)$ เป็นค่าอนุพันธ์ (Derivative) ของฟังก์ชันการถ่ายโอน $g(in_i)$

กำหนดให้ $\Delta_i = Err_i \times g'(in_i)$

จากสมการ (2.6) เขียนใหม่เป็น

$$W_{ji} \longleftarrow W_{ji} + \alpha \times a_j \times \Delta_i \quad \text{.....(2.7)}$$

ในการปรับเปลี่ยนค่าน้ำหนักของ Hidden Layer และ Input Layer โดยที่ชั้นดังกล่าวไม่ใช่ชั้นสุดท้ายที่คำนวณค่านำออกจะทำให้ค่าผิดพลาดประจำชั้นต้องคำนวณจากค่าผิดพลาดจากชั้นของ Output Layer ซึ่งมีนิยามดังสมการต่อไปนี้

$$\Delta_j = g'(in_j) \sum_i W_{ji} \Delta_i$$

ดังนั้นกฎการเปลี่ยนน้ำหนักสำหรับ Hidden Layers และ Input Layer เขียนได้เป็น

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times I_k \times \Delta_j$$

จากขั้นตอนวิธี (Algorithm) ที่กล่าวมาข้างต้นสรุปได้ดังนี้

- คำนวณค่า Δ_i สำหรับ Output Layers ตามสมการ $Err_i = (T_i - O_i)$
- คำนวณค่า Δ_j สำหรับแต่ละ ชั้นที่คัดเข้ามาจาก Output Layers โดยใช้ค่า Δ_i เป็นหลัก
- เปลี่ยนค่าน้ำหนักระหว่างชั้นเป็นค่าใหม่

รูปที่ 2.10 แสดงถึงโครงสร้างของขั้นตอนวิธีสำหรับ วิธี Back-Propagation Learning ใน Multilayer Network

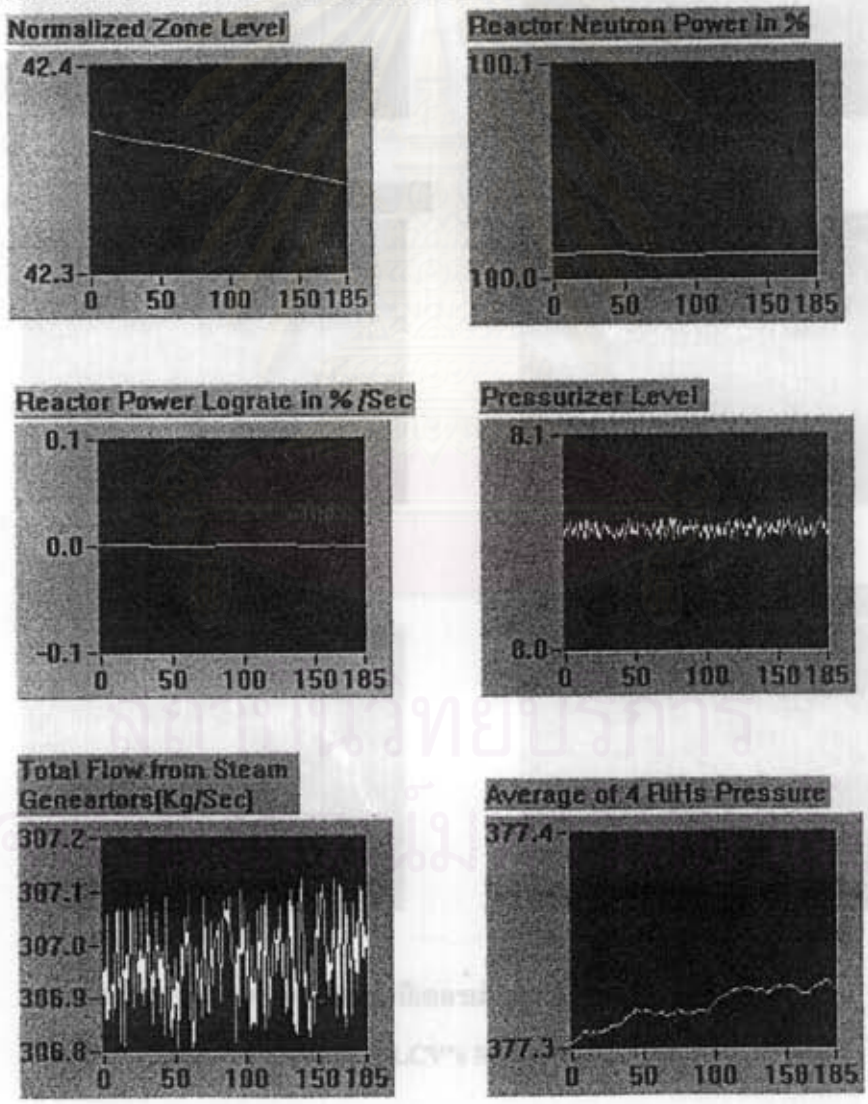
```
FUNCTION Back-Propagation-Update(Network,Examples,α) RETURN a Network with Modified
Weights
INPUT: Network , a Multilayer Network
       Examples , a Set of Input/Output Pairs
       α , the Learning Rate
REPEAT
       FOR EACH e IN Examples DO
       /* Compute the Output for this Example */
         O ← Run-Network (Network , I^e)
       /* Compute the Error and Δ for Units in the Output Layer */
         ERR^e ← T^e - O
       /* Update the Weights Leading to the Output Layer */
         W_j,i ← W_j,i + α x a_j x Err_i^e x g'(in_i)
       FOR EACH Subsequent Layer IN Network DO
       /* Compute the Error at Each Node */
         Δ_j ← g'(in_j) Σ_i W_j,i Δ_i
       /* Update the Weights Leading into the Layer */
         W_k,j ← W_k,j + α x I_k x Δ_j
       END
       END
       UNTIL Network has Converged
       RETURN Network
```

รูปที่ 2.10 แสดงขั้นตอนการคำนวณของ Back_propagation Learning (Russell, S. J., and Norvig, P. Artificial Intelligence .Prentice-Hall International,1995. pp-581.)

2.7 รูปแบบของเครือข่ายนิวรอลที่เหมาะสมกับแบบรูปพารามิเตอร์ (Parameter Pattern) จากโรงไฟฟ้านิวเคลียร์

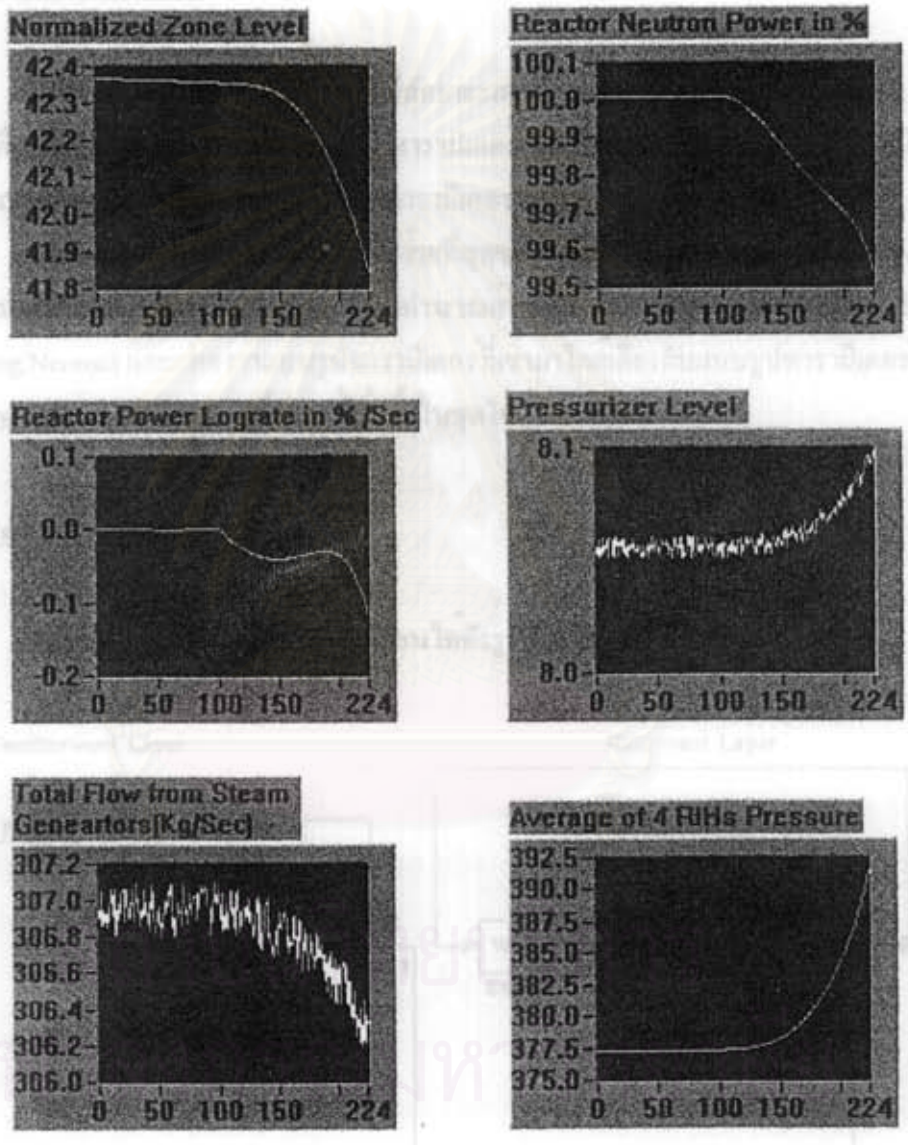
ในปัจจุบันเครือข่ายนิวรอลมีการพัฒนาออกไปในหลายรูปแบบเช่น ปรับปรุงกฎเกณฑ์การเรียนรู้ พัฒนาโครงสร้างของเครือข่าย (Network) พัฒนารูปแบบของนิวรอล ฯลฯ การพิจารณาว่าเครือข่ายนิวรอลแบบใดเหมาะสมกับแบบรูปพารามิเตอร์ของโรงไฟฟ้านิวเคลียร์นั้น จะต้องทำความเข้าใจในคุณสมบัติของข้อมูลดังกล่าวก่อน

แบบรูปพารามิเตอร์จากการเดินเครื่องโรงไฟฟ้านิวเคลียร์ เมื่อนำมาเขียนเป็นกราฟเทียบกับเวลาจนเข้าสู่ภาวะเสถียร (Steady State) แล้วส่วนใหญ่จะได้กราฟที่ต่อเนื่องและมีลักษณะราบเรียบหรืออาจจะเปลี่ยนแปลงอยู่ในช่วงๆหนึ่งเท่านั้น ตัวอย่างดังกราฟในรูปที่ 2.11 ซึ่งเป็นตัวอย่างกราฟของพารามิเตอร์ บางส่วน ที่ได้จาก Plants Overview Screen ของโปรแกรม CASSIM[10,11]



รูปที่ 2.11 แสดงรูปกราฟของพารามิเตอร์ของโรงไฟฟ้านิวเคลียร์แคนดู-9ในสภาวะเดินเครื่องปกติที่ได้จาก Plants Overview Screen ของโปรแกรม CASSIM

แต่เมื่อเกิดภาวะทรานเซียนต์ขึ้นจะมีพารามิเตอร์บางตัวที่เกี่ยวข้องกับภาวะทรานเซียนต์นั้นๆ จะออกจากภาวะปกติ ซึ่งพารามิเตอร์ดังกล่าวจะเปลี่ยนแปลงออกจากขอบเขตที่เคยอยู่รูปกราฟที่ได้จะไม่เหมือนเดิมและมีรูปแบบเฉพาะตัวของแต่ละชนิดของภาวะทรานเซียนต์ที่เกิดขึ้นตัวอย่างดังกราฟในรูปที่ 2.12 ซึ่งเป็นพารามิเตอร์ที่ได้จาก Plants Overview Screen เมื่อเกิดภาวะทรานเซียนต์ชนิด Fail Closed All Feedwater LCV's Mvs ขึ้น



รูปที่ 2.12 แสดงลักษณะของพารามิเตอร์เมื่อเกิดภาวะทรานเซียนต์ชนิด Fail Closed All Feedwater LCV's Mvs ขึ้น

ตัวอย่างจากรูปที่ 2.12 เมื่อเกิดภาวะทรานเซียนต์ขึ้นจะเกิดรูปกราฟที่แตกต่างจากภาวะปกติ แต่จะคล้ายกันในภาวะทรานเซียนต์เดียวกันที่อาจจะเกิดขึ้นในช่วงเวลาที่แตกต่างกัน

จากคุณสมบัติของชุดพารามิเตอร์ที่ยกตัวอย่างมาข้างต้นนี้ เครือข่ายนิวรอลที่เลือกใช้จะต้องมีคุณสมบัติที่เรียกว่าการรู้จำแบบ โดยที่แบบรูปพารามิเตอร์จะต้องอยู่ในรูปแบบของรูปกราฟซึ่งจากการศึกษาพบว่าเครือข่ายนิวรอลแบบ SOFM มีคุณสมบัติดังกล่าว SOFM ถูกพัฒนาขึ้นจาก Competitive Network โดยการรวม 1 โนดของ Competitive Network เป็น Map เรียกว่า Feature Map (FM) ดังนั้นจะกล่าวถึง Competitive Network ในหัวข้อต่อไปก่อน

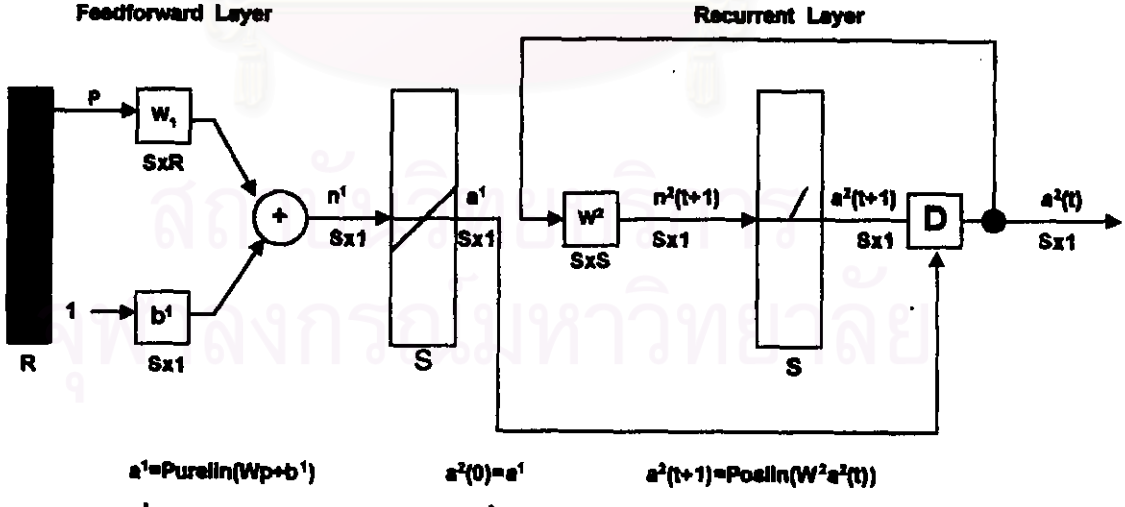
2.8 Competitive Network

เป็นรูปแบบของเครือข่ายนิวรอลที่มีลักษณะการเรียนรู้แบบ Adaptively Learning ซึ่งเป็นการเรียนรู้เพื่อปรับเครือข่ายให้สอดคล้องกับชุดพารามิเตอร์ที่ต่อเนื่องลักษณะ Real-Time โดยมีจุดประสงค์เพื่อจัดแบ่งรูปแบบ (Classify Patterns) ของพารามิเตอร์

Hamming Network เป็นตัวอย่างที่ง่ายที่สุดของเครือข่ายนิวรอลแบบ Competitive Network โดยนิวรอลที่อยู่บน Output Layer จะคำนวณค่านำออกและเปรียบเทียบกันเพื่อหานิวรอลที่ชนะ (Winning Neuron) และแสดงว่าแบบรูปพารามิเตอร์ที่เข้ามาใกล้เคียงกับแบบรูปพารามิเตอร์ต้นแบบ (Prototype Parameter Pattern) ที่เรียนรู้ไว้แล้วในชุดใด

2.8.1 Hamming Network

ลักษณะของ Hamming Network เขียนได้ดังรูปที่ 2.13



รูปที่ 2.13 แสดงแผนภาพบล็อกทั่วไปของ Hamming Nework(Hagan, M.T., Demuth, H. B., and Beale, M. Neural Network Design . PWS Publishing ,1996. pp 14-3)

Hamming Network ประกอบด้วย 2 ชั้นในชั้นแรกทำหน้าที่จับคู่ความสัมพันธ์ระหว่างเวกเตอร์ค่านำเข้า (Input Vector) กับ เวกเตอร์ต้นแบบ(Prototype Vectors) ซึ่งถูกเก็บไว้ในรูปเมทริกซ์ของน้ำหนัก

สมมุติว่าเราต้องการทำให้ Hamming Network สามารถเก็บรูปแบบของเวกเตอร์ต้นแบบที่เขียนอยู่ในรูปของเซตดังต่อไปนี้ $\{p_1, p_2, \ldots\ldots, p_q\}$

และกำหนดให้ เมทริกซ์ของน้ำหนัก เริ่มต้น W' และ Bias Matrix b' ของ ชั้น 1 เป็นดังนี้

$$W' = \begin{bmatrix} W_1^T \\ W_2^T \\ \vdots \\ W_S^T \end{bmatrix} = \begin{bmatrix} P_1^T \\ P_2^T \\ \vdots \\ P_q^T \end{bmatrix} b' = \begin{bmatrix} R \\ R \\ \vdots \\ R \end{bmatrix}$$

เมื่อแต่ละแถวของ W' แทนเวกเตอร์ต้นแบบซึ่งเราต้องการให้เครือข่ายนิวรอลเกิดการรู้จำ ดังนั้นจำนวนนิวรอลจึงเท่ากับจำนวน เวกเตอร์ต้นแบบที่จะรู้จำและค่านำออกของชั้นที่ 1 เป็นดังนี้

$$a' = Purelin(W'P + b') = \begin{bmatrix} P_1^T P + R \\ P_2^T P + R \\ \vdots \\ P_q^T P + R \end{bmatrix}$$

ในชั้น 2 ของ Hamming Network เป็นส่วนที่คำนวณว่าเวกเตอร์ค่านำเข้าเมื่อผ่านการวิเคราะห์แล้วจะมีค่าใกล้เคียงกับเวกเตอร์ต้นแบบชุดใดมากที่สุด

ใน ชั้น 2 เป็น Competitive Network ทำหน้าที่คำนวณหานิวรอลที่อยู่ใน ชั้น 1 ว่านิวรอลไหนเป็นนิวรอลที่ชนะหรือเป็นนิวรอลที่ให้ค่าใกล้เคียงกับเวกเตอร์ต้นแบบมากที่สุด

กำหนดให้ค่านำออกของ ชั้น 1 คือ a' ซึ่งเป็นค่าเริ่มต้นสำหรับชั้น 2

$$a^2(0) = a'$$

$$a^2(t+1) = Poslin(W^2 a^2(t))$$

ใน ชั้น 2 มีเมทริกซ์ของน้ำหนัก W^2 เขียนเป็นเมทริกซ์ทแยงมุม (Diagonal Matrix)โดยมีสมาชิกในแนวทแยงมุม (Diagonal Elements) เป็น 1 และสมาชิกนอกแนวทแยงมุม (Off-Diagonal Elements) มีค่าเป็นลบ

$$W_{ij}^2 = \begin{cases} 1; i = j \\ -\varepsilon; otherwise \end{cases} \text{ โดย } 0 < \varepsilon < \frac{1}{(s-1)}$$

- Competitive Layer

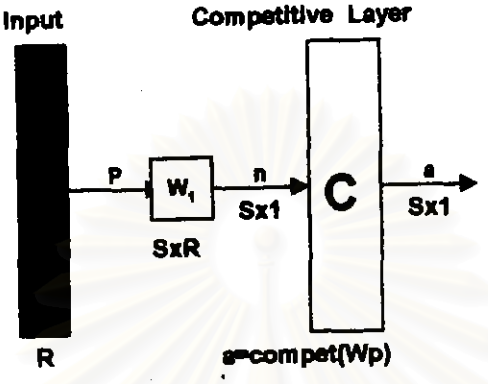
ใน ชั้นที่ 2 ของ Hamming Network เรียกว่า Competitive Layer เนื่องจากแต่ละนิวรอล ในชั้น นั้นจะถูกเปรียบเทียบผลการคำนวณระหว่างกัน เพื่อหาว่าเวกเตอร์ค่านำเข้าเหมือนกับเวกเตอร์ต้นแบบชุดไหนมากที่สุด ซึ่งสามารถเขียนอยู่ในรูปของฟังก์ชันการถ่านโอน ดังนี้

$$a = Compet(n)$$

ลักษณะการทำงานของฟังก์ชันดังกล่าวเพื่อหาว่านิวรอลตัวใดคำนวณค่า i^* ของค่านำเข้าสูงที่สุดซึ่งจะถูกให้มีค่านำออก a เป็น 1 ในขณะที่นิวรอลตัวที่เหลือจะถูกกำหนดให้มีค่าเป็น 0 หมดดังนี้

$$a_i = \begin{cases} 1; i = i^* \\ 0; i \neq i^* \end{cases}, \quad \begin{array}{l} where\ n_{i^*} \geq n_i, \forall i \\ \&\ i^* \leq i, \forall n_i = n_{i^*} \end{array}$$

จากนั้นใช้ Competitive Layer แทนที่ใน ชั้น ที่ 2 ของ Hamming Network ดังแสดงในรูปที่ 2.14



รูปที่ 2.14 แสดง แผนภาพบล็อก ทั่วไปของ Competitive Network(Hagan, M.T., Demuth, H. B., and Beale, M. Neural Network Design . PWS Publishing ,1996. pp 14-6)

การทำงานของ Network ใหม่มีลักษณะคล้ายกับ Hamming Network ซึ่งเวกเตอร์ต้นแบบถูกเก็บอยู่เป็นแถวในรูปแบบของเมทริกซ์ของน้ำหนัก W โดยที่ n สุทธิเป็นผลการคำนวณของระยะห่างระหว่างเวกเตอร์ค่านำเข้า $\bar{P}$ กับแต่ละเวกเตอร์ต้นแบบ W_i โดยที่เวกเตอร์ที่นำมาคำนวณจะต้องผ่านการทำให้เป็นบรรทัดฐาน (Normalize) ในกรณีของขนาดเวกเตอร์เสียก่อน ค่า n_i สุทธิของแต่ละนิวรอล i ขึ้นกับมุม θ_i ระหว่าง $\bar{P}$ และ เวกเตอร์ต้นแบบ W_i

$$n = WP = \begin{bmatrix} W_1^T \\ W_2^T \\ \vdots \\ W_S^T \end{bmatrix} P = \begin{bmatrix} W_1^T P \\ W_2^T P \\ \vdots \\ W_S^T P \end{bmatrix} = \begin{bmatrix} \cos\theta_1 \\ \cos\theta_2 \\ \vdots \\ \cos\theta_S \end{bmatrix}$$

Competitive Transfer Function ให้ค่านำออกเป็น 1 เมื่อนิวรอลที่มีเวกเตอร์ค่านำเข้าอยู่ในทิศเดียวกับเวกเตอร์ของน้ำหนัก (Parameter Vector) โดยที่

$$a = Compet(WP)$$

- Competitive Learning

ในกรณีที่เราต้องการบันทึกเวกเตอร์ค่านำเข้าเพื่อเก็บไว้เป็นชุดของเวกเตอร์ต้นแบบที่ต้องผ่านขั้นตอนการเรียนรู้เพื่อปรับเปลี่ยนเมทริกซ์ของน้ำหนักให้สอดคล้องกับเวกเตอร์ต้นแบบที่ถูกใส่เข้ามาใหม่ซึ่งมีสมการดังต่อไปนี้

$$W_i(q) = W_i(q-1) + \alpha a_i(q)\left[P(q) - W_i(q-1)\right];\ 0 < \alpha \leq 1$$

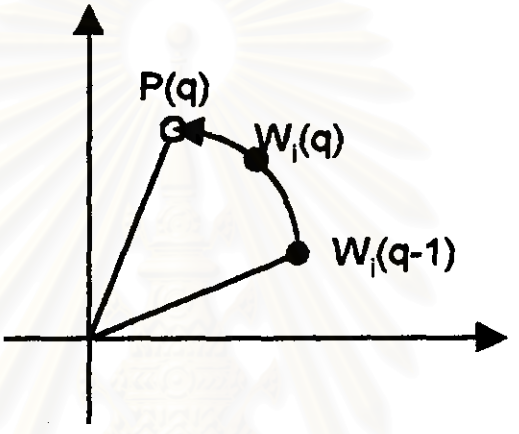
เนื่องจากใน Competitive Network $a = 1$ สำหรับนิวรอลที่ชนะ $(i = i^*)$ ดังนั้น

$$W_i(q) = W_i(q-1) + \alpha[P(q) - W_i(q-1)]$$

$$W_i(q) = (1-\alpha)W_i(q-1) + \alpha P(q)$$

$$W_i(q) = W_i(q-1) \qquad \text{เมื่อ } i \neq i^*$$

จากสมการข้างต้นจะเห็นว่าแถวของเมทริกซ์ของน้ำหนักที่ใกล้กับเวกเตอร์ค่านำเข้า จะเปลี่ยนค่าตามสมการโดยเป็นไปในรูปแบบที่เลื่อนเวกเตอร์ของน้ำหนักเข้าใกล้เวกเตอร์ค่านำเข้า โดยจะเลื่อนอยู่ในแนวระหว่าง เวกเตอร์ของน้ำหนักเก่ากับเวกเตอร์ค่านำเข้า ซึ่งอธิบายได้ดังรูปต่อไปนี้



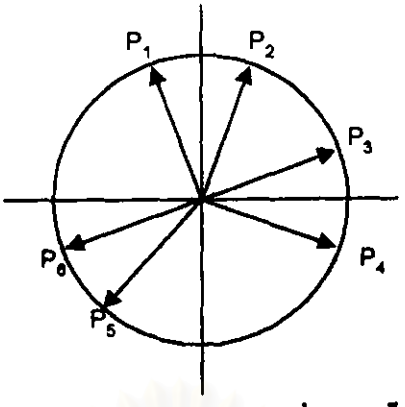
รูปที่ 2.15 แสดงถึงลักษณะการปรับตัวเข้าหา เวกเตอร์ของพารามิเตอร์ ของ Weight Vector (Hagan, M.T., Demuth, H. B., and Beale, M. Neural Network Design . PWS Publishing ,1996.pp 14-11)

ตัวอย่างแสดงการทำงานของ Competitive Learning ที่ใช้จัดจำพวกของเวกเตอร์ซึ่งมีเวกเตอร์ 6 ชุดดังนี้

$$P_1 = \begin{bmatrix} -0.1961 \\ 0.9806 \end{bmatrix} \quad P_2 = \begin{bmatrix} 0.1961 \\ 0.9806 \end{bmatrix} \quad P_3 = \begin{bmatrix} 0.9806 \\ 0.1961 \end{bmatrix}$$

$$P_4 = \begin{bmatrix} 0.9806 \\ -0.1961 \end{bmatrix} \quad P_5 = \begin{bmatrix} -0.5811 \\ -0.8137 \end{bmatrix} \quad P_6 = \begin{bmatrix} -0.8137 \\ -0.5812 \end{bmatrix}$$

เวกเตอร์ทั้ง 6 ชุด เป็นเวกเตอร์ขนาด 2 มิติซึ่งสามารถเขียนบนวงกลม 1 หน่วย ดังรูป

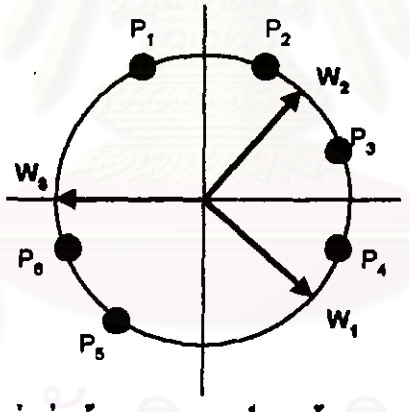


รูปที่ 2.16 แสดงถึง Competitive Network เมื่อเรียนรู้พารามิเตอร์หลายชุด

Competitive Network ในตัวอย่างนี้มี 3 นิวรอลโดยเมื่อผ่านการเรียนรู้แล้วเวกเตอร์ค่านำเข้า จะถูกจัดแบ่งเป็น 3 จำพวก โดยที่แต่ละนิวรอลมีเมทริกซ์ของน้ำหนักแบบสุ่ม (Random) ดังนี้

$$W_1=\begin{bmatrix}0.7071\\-0.7071\end{bmatrix} W_2=\begin{bmatrix}0.7071\\0.7071\end{bmatrix} W_3=\begin{bmatrix}-1\\0\end{bmatrix}$$

เมื่อให้ $W=\begin{bmatrix}w_1^t\\w_2^t\\w_3^t\end{bmatrix}$ ซึ่งสามารถแสดงได้ดังรูปที่ 2.17



รูปที่ 2.17 แสดงค่าเริ่มต้นของเวกเตอร์ของน้ำหนัก (Hagan, M.T., Demuth, H. B., and Beale, M. Neural Network Design . PWS Publishing ,1996.pp14-12)

ดังนั้น Competitive Transfer Function ของอินพุทเวคเตอร์ P_2 จะเท่ากับ

$$a=compet(WP_2)=compet\left(\begin{bmatrix}0.7071 & -0.7071\\0.7071 & 0.7071\\-1.000 & 0.000\end{bmatrix}\begin{bmatrix}0.1961\\0.9806\end{bmatrix}\right)$$

$$a=compet\left(\begin{bmatrix}-0.5547\\0.8321\\-0.1961\end{bmatrix}\right)=\begin{pmatrix}0\\1\\0\end{pmatrix}$$

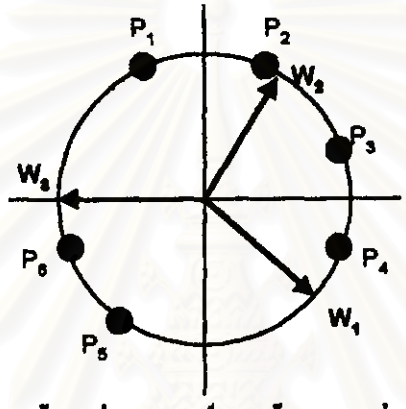
หอสมุดกลาง สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

จะเห็นว่านิวรอลตัวที่ 2 มีค่าเวกเตอร์ของน้ำหนักใกล้กับ P_2 มากที่สุดดังนั้นนิวรอลตัวดังกล่าวเป็น นิวรอลที่เป็นนิวรอลที่ชนะ ($i^* = 2$) สำหรับรอบการเรียนรู้รอบแรกดังนั้น

$$W_2^{new} = W_2^{old} + \alpha(P_2 - W_2^{old}) \quad \text{กำหนดให้ } \alpha = 0.5$$

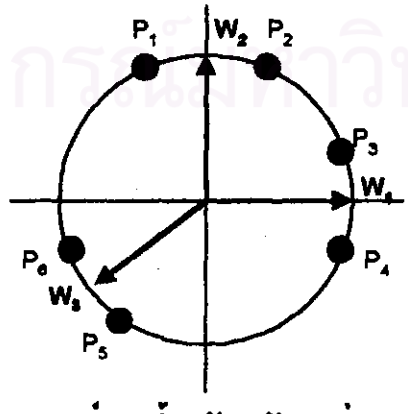
$$W_2^{new} = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix} + 0.5\left(\begin{bmatrix} 0.1961 \\ 0.9806 \end{bmatrix} - \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}\right) = \begin{bmatrix} 0.4516 \\ 0.8438 \end{bmatrix}$$

จะเห็นว่า W_2 จะเลื่อนเข้าใกล้ P_2 ดังรูปที่ 2.18



รูปที่ 2.18 แสดงให้เห็นว่าเวกเตอร์ของน้ำหนักเปลี่ยนไปโดยเลื่อนเข้าหาเวกเตอร์ของพารามิเตอร์

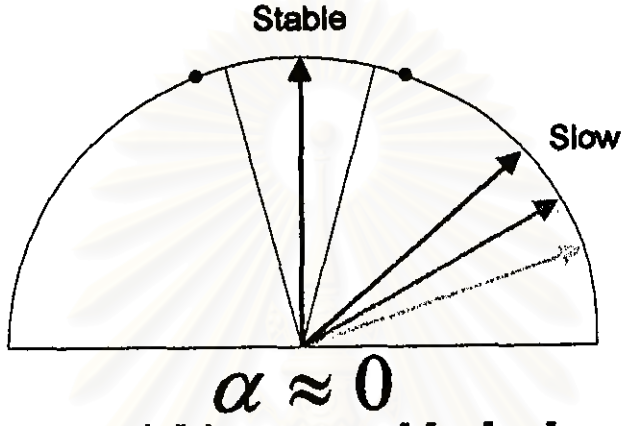
ซึ่งเมทริกซ์ของน้ำหนักที่ได้คือ เวกเตอร์ต้นแบบ และถ้าเรายังทำการเรียนรู้ต่อไปเรื่อยๆ จะพบว่าในแต่ละรอบการเรียนรู้ (Iteration) ค่าเวกเตอร์ของน้ำหนักจะเลื่อนเข้าใกล้เวกเตอร์ค่านำเข้า ถึงแม้ว่าแต่ละเวกเตอร์ของน้ำหนัก จะชี้ไปทางกลุ่มของเวกเตอร์ค่านำเข้าที่ต่างกันเวกเตอร์ของน้ำหนัก นั้นจะเป็นเวกเตอร์ต้นแบบของกลุ่มซึ่งเราสามารถเดาได้ว่าเวกเตอร์ของน้ำหนักอันไหนจะชี้ไปยังกลุ่มของเวกเตอร์ค่านำเข้าซึ่งในที่สุดจะได้ทิศทางของ Final Weight ดังรูปที่ 2.19



รูปที่ 2.19 แสดง เวกเตอร์ของน้ำหนัก หลังจากผ่านการเรียนรู้ในครั้งสุดท้าย

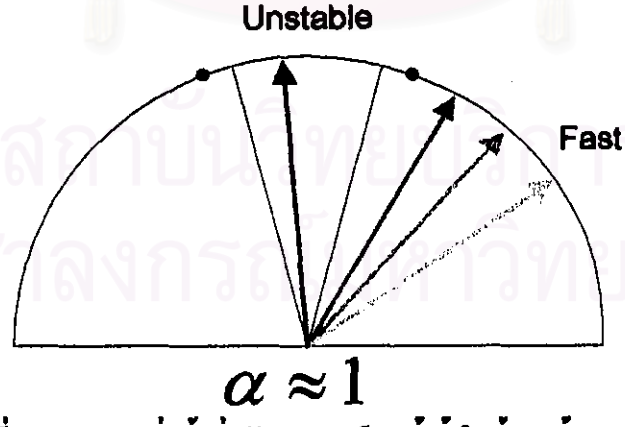
2.8.2 <u>ปัญหาของการใช้ Competitive Layer</u>

Competitive Layer ช่วยในการจัดจำพวกของ เวกเตอร์ค่าน่าเข้า ได้แต่ก็ยังมีปัญหาบางอย่างในการใช้งานจริงคือการเลือกค่าอัตราการเรียนรู้ซึ่งจะมีการแข่งขันกันระหว่างความเร็วของการเรียนรู้กับความเสถียรของ Final Weight ซึ่งอัตราการเรียนรู้ที่ใกล้เคียงกับ 0 จะช้าอย่างไรก็ตามเมื่อเวกเตอร์ของน้ำหนักเข้าใกล้ศูนย์กลางของกลุ่มมันจะมีแนวโน้มที่จะหยุดอยู่ตำแหน่งตรงกลางดังรูป



รูปที่ 2.20 แสดงว่าถ้าค่าอัตราการเรียนรู้เข้าใกล้ 0 แล้วเวกเตอร์ของน้ำหนักจะเข้าใกล้เวกเตอร์ของพารามิเตอร์ช้า แต่จะเข้าสู่ภาวะเสถียรได้เร็ว

ในทางกลับกันอัตราการเรียนรู้ที่เข้าใกล้ 1 จะเร็วมากแต่จะทำให้เมื่อเวกเตอร์ของน้ำหนักเลื่อนเข้าใกล้กลุ่มแล้วเกิดการเลื่อนไปเลื่อนมาคล้ายกับการแกว่ง (Oscillate) ขึ้น



รูปที่ 2.21 แสดงว่าถ้าค่าอัตราการเรียนรู้เข้าใกล้ 1 แล้วเวกเตอร์ของน้ำหนักจะเข้าใกล้เวกเตอร์ของพารามิเตอร์เร็วแต่อาจเกิดการแกว่งขึ้นได้

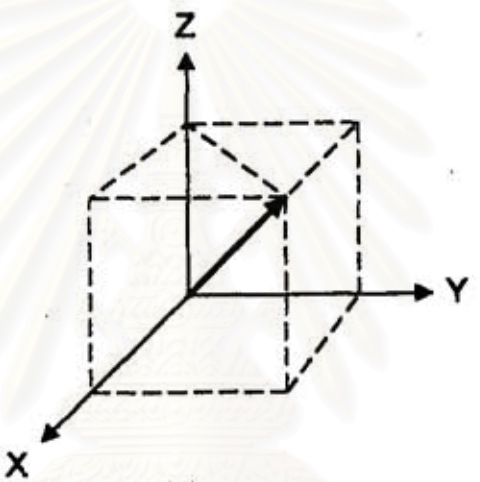
2.9 Self-Organizing Feature Maps (SOFM)

SOFM เป็นทฤษฎีที่พัฒนาจาก Hamming Network ซึ่งมีการกำหนดค่า เมทริกซ์ของน้ำหนักเริ่มต้นเป็นค่าคงที่อยู่ในรูปของ Map เรียกว่า FM

ตัวอย่างเช่น SOFM ขนาด 3 มิติ ซึ่งมีค่านำเข้า 3 ตัวแปร และใช้เมทริกซ์ของค่าน้ำหนักขนาด 3 มิติซึ่งสามารถแทนค่าเมทริกซ์ของน้ำหนัก ได้ด้วยเวกเตอร์บนทรงกลม 1 หน่วยดังรูป

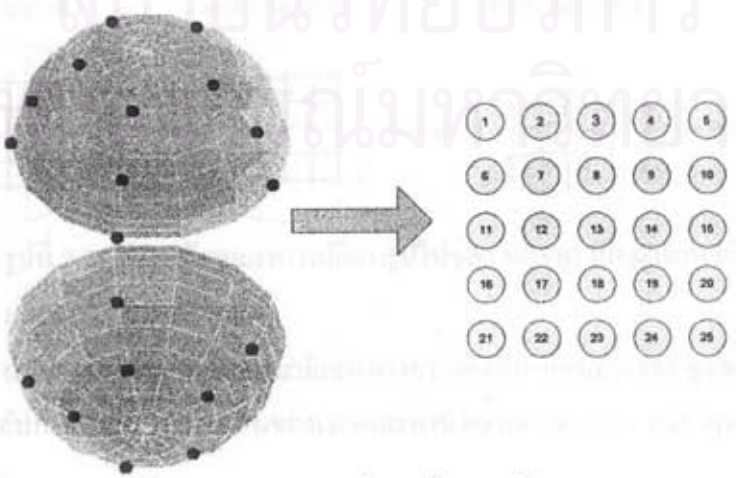
$$W_i(q)=\begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \bar{W}(q)=x\hat{a}_x+y\hat{a}_y+z\hat{a}_z$$

โดยที่ $\sqrt{x^2+y^2+z^2}=1$



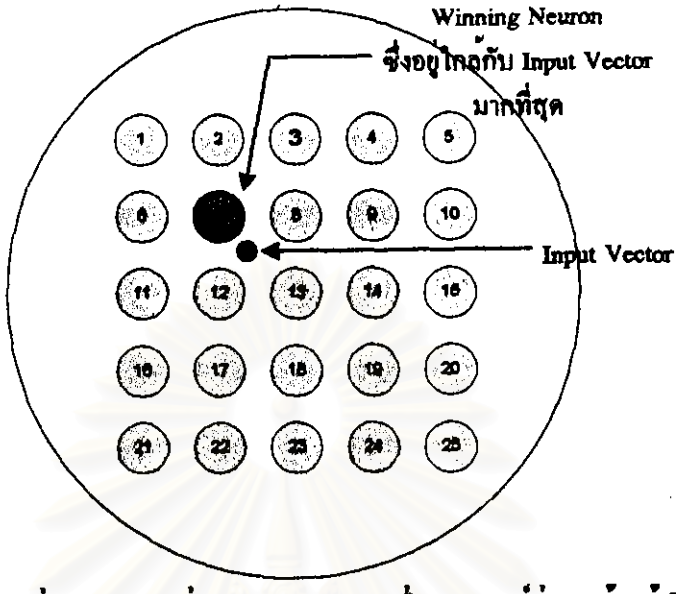
รูปที่ 2.22 แสดง เมทริกซ์ของน้ำหนัก อยู่ในรูปของเวกเตอร์ 3 มิติ

ตัวอย่างเช่น กรณีที่ใช้นิวรอลในเครือข่ายทั้งหมด 25 นิวรอลชุดของเวกเตอร์ของน้ำหนักที่อยู่บนทรงกลม 1 หน่วย มีรูปแบบเริ่มต้นดังนี้



รูปที่ 2.23 แทนเวกเตอร์ของน้ำหนัก ด้วยจุดบนทรงกลม 3 มิติ

หลักการเรียนรู้ของ SOFM คือการหานิวรอลที่ชนะ i^* ที่มีเวกเตอร์ของน้ำหนักประจำอยู่ซึ่งใกล้กับ เวกเตอร์ค่านำเข้ามากที่สุดดังรูปที่ 2.24



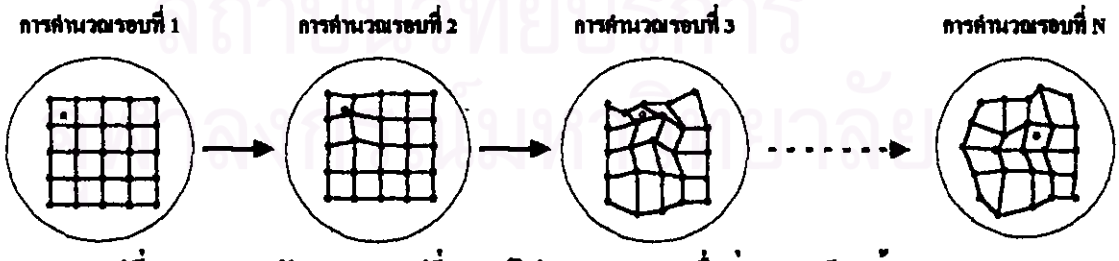
รูปที่ 2.24 แสดงว่า โนด จะ Active เมื่อ เวกเตอร์ค่านำเข้า เข้าใกล้ โนด นั้น

เมื่อได้นิวรอลที่ชนะแล้วค่าเมทริกซ์ของน้ำหนักจะถูกปรับเปลี่ยนเพื่อทำให้เวกเตอร์ของน้ำหนักของนิวรอลที่ชนะเลื่อนเข้าใกล้เวกเตอร์ค่านำเข้าตามสมการ

$$W_i(q) = W_i(q-1) + \alpha[P(q) - W_i(q-1)]$$

$$W_i(q) = (1-\alpha)W_i(q-1) + \alpha P(q)$$

การคำนวณตามสมการข้างต้นกระทำทุกครั้งในทุกรอบของการเรียนรู้เมื่อเวกเตอร์ค่านำเข้าที่เข้ามาเปลี่ยนแปลงไปเรื่อยๆ และถ้าเวกเตอร์ค่านำเข้าเลื่อนเข้าใกล้นิวรอลตัวใดก็ถือว่านิวรอลตัวนั้นเป็นนิวรอลที่ชนะสำหรับรอบการคำนวณนั้น เมื่อทำการคำนวณหลายๆรอบ FM ที่ได้ก็จะเปลี่ยนไปเรื่อยๆดังรูปที่ 2.25



รูปที่ 2.25 แสดงลักษณะการเปลี่ยนรูปไปของ SOFM เมื่อผ่านการเรียนรู้แต่ละครั้ง

ถ้าเวกเตอร์ค่านำเข้าที่เข้ามามีเส้นทางการเดินที่เหมือนกัน FM สุดท้ายที่ได้ก็จะเหมือนกัน และในทางกลับกันถ้าเส้นทางการเดินของเวกเตอร์ค่านำเข้าแตกต่างกัน FM สุดท้ายก็จะแตกต่างกันอย่างมีนัยสำคัญ

จากคุณสมบัติของ SOFM ดังกล่าวสามารถนำมาประยุกต์ใช้ได้กับพารามิเตอร์จากโรงไฟฟ้านิวเคลียร์ซึ่งส่วนใหญ่เป็นพารามิเตอร์ที่ต่อเนื่อง ซึ่งเมื่อจัดกลุ่มของข้อมูลและเปลี่ยนให้อยู่ในรูปของเวกเตอร์หลายมิติก็สามารถใช้ SOFM วิเคราะห์ได้

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย