การพัฒนาโปรแกรมการจดจำตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษโดยโครงข่ายประสาทเทียม

Printed Thai and Printed English Recognition using Neural Network Software Development

โดย

นางสาวจรรยา สุภาพ

รหัส 41067170

อาจารย์ที่ปรึกษา

ดร. อาริต ธรรมโน

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 1 ปีการศึกษา 2543

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การพัฒนาโปรแกรมจดจำตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษ โดยใช้โครงข่ายประสาทเทียม
นักศึกษา	นางสาวจรรยา สุภาพ
อาจารย์ที่ปรึกษา	ดร.อาริต ธรรมโน
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2543

บทคัดย่อ

การจดจำตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษ โดยใช้โครงข่ายประสาทเทียม เป็นความพยายามในการฝึกสอนให้คอมพิวเตอร์มีความสามารถ หรือความรู้ในการแยกแยะตัวอักษร แม้จะมีขนาดหรือแบบอักษรต่างกันได้อย่างถูกต้อง ซึ่งในการพัฒนาโปรแกรมนี้จะใช้แบบจำลอง Adaptive Resonance Theory 1 เพื่อนำมาประยุกต์ใช้ในการจดจำตัวพิมพ์

Title	Printed Thai and Printed English Recognition Using Neural Network Software Development
Student	Miss Chanya Suphab
Advisor	Dr.Arit Thammano
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2000

ABSTRACT

Printed Thai Character and Printed English Character recognition using neural network is one of the process in technology or practical on computer in order to enhance the ability of computer in the seperation the characters, both Thai and English correctly. Although the characters vaied in size or type. This paper cites as the printed characters, Thai and English, and uses the neural networks, also using the Adaptive Resonance theory 1 Network Model in the recognition the printed characters.

กิตติกรรมประกาศ

การพัฒนาโปรแกรมรู้จำตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษนี้ประสบความสำเร็จได้ด้วยดีจากการสนับสนุนและอนุเคราะห์จากบุคคลหลายฝ่าย ผู้จัดทำจึงใคร่ขอขอบพระคุณบุคคลดังต่อไปนี้

1. บิดา มารดา และพี่สาว ที่คอยเป็นกำลังใจให้เสมอมา
2. ดร.อาริต ธรรมโน อาจารย์ที่ปรึกษา ที่กรุณาให้คำแนะนำ ในด้านต่าง ๆ ที่เกี่ยวกับการพัฒนาโปรแกรมนี้ จนกระทั่งการพัฒนาโปรแกรมลุล่วงไปได้ด้วยดี
3. ผู้อำนวยการ หัวหน้าฝ่าย หัวหน้างาน และเพื่อนร่วมงานทุกคนในส่วนเทคโนโลยีสารสนเทศ สำนักงานสรรพากรภาค 5 ที่เป็นกำลังใจให้ตลอดมา
4. กรมสรรพากร ซึ่งผู้จัดทำได้ทำงานอยู่ในสังกัด ที่กรุณาให้ทุนการศึกษาตลอดโครงการ
5. พี่ ๆ เพื่อน ๆ และน้อง ๆ แขนงวิชาวิทยาการสารสนเทศ รุ่น 6 (สรรพากร) ที่ให้คำปรึกษาแนะนำ และเป็นกำลังใจให้ตลอดระยะเวลาที่ทำการพัฒนาโปรแกรม รวมถึงเจ้าหน้าที่คณะเทคโนโลยีสารสนเทศ ที่ให้ความช่วยเหลือและอำนวยความสะดวกในด้านอื่น ๆ ผู้จัดทำขอขอบพระคุณไว้ ณ โอกาสนี้ด้วย

นางสาวจรรยา สุภาพ

สารบัญ

หน้า

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

ภาพที่

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาของการพัฒนาโปรแกรมจดจำตัวพิมพ์อักษรโดยใช้โครงข่ายประสาทเทียม

ในปัจจุบันรูปแบบการป้อนข้อมูลให้กับคอมพิวเตอร์ ยังคงเป็นวิธีการที่ง่ายและสะดวกสำหรับผู้ที่มีความชำนาญ แต่ถึงอย่างไรความผิดพลาดก็ยังคงมีตามมา อีกทั้งยังต้องเสียเวลาค่อนข้างมาก ถ้าข้อมูลมีปริมาณมากทำให้สิ้นเปลืองงบประมาณ จึงทำให้เกิดการพัฒนาเครื่องอ่านตัวอักษรด้วยแสงขึ้น (Optical Character Reader) มาใช้งานเพื่อแก้ปัญหาที่กล่าวมาข้างต้น

กระบวนการทำงานของเครื่องอ่านตัวอักษรด้วยแสงประกอบด้วย 4 ขั้นตอนคือ

1. การสแกน (Scanning) เป็นกระบวนการแปลงสัญญาณไฟฟ้าให้อยู่ในรูปของดิจิตอล
2. การประมวลผลส่วนหน้า (Preprocessing) เป็นกระบวนการในการตัดสิ่งรบกวนออกจากการสแกน ของภาพและทำการแปลงข้อมูลให้อยู่ในรูปขาว ดำ
3. การทำเซกเมนต์และการลงทะเบียนตัวอักษร (Segmentation and Registration) ทำหน้าที่ในการแยกภาพตัวอักษรและบันทึกตำแหน่งของภาพเพื่อนำไปใช้ในกระบวนการถัดไป
4. การจดจำตัวอักษร (Recognition) ซึ่งประกอบด้วย 2 ส่วนย่อยคือ ในส่วนของการวิเคราะห์ หรือแยกแยะตัวอักษร (Classification) ซึ่งส่วนนี้ทำหน้าที่วิเคราะห์ภาพตัวอักษรที่ได้จากกระบวนการที่สามว่าคือตัวอักษรอะไร แล้วก็แปลงให้เป็นรหัสแอสกี ส่วนย่อยที่สองคือการตรวจสอบว่าตัวอักษรที่ได้จากการวิเคราะห์มีความถูกต้องหรือไม่ และจัดเรียงลำดับตัวอักษรซึ่งการจัดเรียงนี้จะนำตำแหน่งที่บันทึกในขั้นตอนของกระบวนการเซกเมนต์มาจัดลำดับ

1.2 แนวทางการพัฒนาโปรแกรม

เป็นการพัฒนาโปรแกรมเพื่อวิเคราะห์ตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษโดยใช้โครงข่ายประสาทเทียม (Neural Network) ใช้วิธีการเรียนรู้ตัวอย่างข้อมูลเช่นเดียวกับสมองมนุษย์ โดยตัวโครงข่ายจะปรับตัวเองเพื่อสร้างฐานความรู้ให้กับตัวเอง จากนั้นจึงนำความรู้ที่สอนนี้

ไปใช้ในการวิเคราะห์ข้อมูล ตัวแบบของโครงข่ายประสาทเทียมที่นำมาประยุกต์ใช้ในการพัฒนาโปรแกรมนี้คือ Adaptive Resonance Theory 1 ได้นำมาประยุกต์ใช้เพื่อการจดจำตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษ

1.3 วัตถุประสงค์ของการพัฒนาโปรแกรม

1. เพื่อศึกษาการนำโครงข่ายประสาทเทียมมาใช้ในการจดจำตัวอักษร
2. เพื่อศึกษาคุณสมบัติและขั้นตอนการนำตัวแบบ ART 1 มาประยุกต์ใช้งาน
3. เพื่อศึกษาการพัฒนาโปรแกรมโดยใช้ ART 1

1.4 โครงสร้างของการพัฒนาโปรแกรม

การพัฒนาโปรแกรมนี้ประกอบด้วยบทที่ 2 จะกล่าวถึงทฤษฎีเบื้องต้นของโครงข่ายประสาทเทียม บทที่ 3 กล่าวถึงทฤษฎีและขั้นตอนการนำเอา ART 1 มาใช้ในการพัฒนาโปรแกรมการรู้จำตัวพิมพ์อักษรทั้งภาษาไทยและภาษาอังกฤษ บทที่ 4 การประยุกต์ใช้ ART 1 เพื่อรู้จำตัวพิมพ์อักษรภาษาไทยและตัวพิมพ์อักษรภาษาอังกฤษ และบทที่ 5 บทสรุปและข้อเสนอแนะ

บทที่ 2
ทฤษฎีเบื้องต้นของโครงข่ายประสาทเทียม

2.1 ประวัติของโครงข่ายประสาทเทียม

การศึกษาทางด้านโครงข่ายประสาทเทียมได้เริ่มมีการคิดค้นและดำเนินการวิจัยมาตั้งแต่ปี ค.ศ. 1940 ซึ่งมีลำดับของการเกิดสถาปัตยกรรมของโครงข่ายประสาทเทียมชนิดต่าง ๆ ดังนี้

ในปี ค.ศ. 1943 วอร์เรน แมคคูลล๊อค (Warren MacCulloch) และ วอลเตอร์ พิทท์ (Walter Pitts) ได้ออกแบบนิวรอนที่ถือเป็นต้นกำเนิดและพื้นฐานของโครงข่ายประสาทเทียมในปัจจุบัน โครงข่ายของแมคคูลล๊อคและพิทท์ ประกอบด้วยชั้นของโครงข่ายจำนวน 2 ชั้นคือชั้น อินพุทและชั้นเอาต์พุท โครงข่ายจะมีการเชื่อมโยงจากชั้นอินพุทไปยังชั้นเอาต์พุท โดยใช้ค่าถ่วงน้ำหนักซึ่งจะถูกกำหนดไว้ตายตัว การทำงานเบื้องต้นของโครงข่ายคือการนำไปใช้จำลองฟังก์ชั่นทางลอจิกพื้นฐาน คือฟังก์ชั่น AND, ฟังก์ชั่น OR, ฟังก์ชั่น NOT และฟังก์ชั่น XOR โดยนำเสนอการกำหนดค่าถ่วงน้ำหนักเพื่อให้โครงข่ายทำงานได้กับการจำลองตามฟังก์ชั่นดังกล่าว

ในปี ค.ศ. 1949 โดแนลด์ เฮบบ์ (Donald Hebb) ได้ออกแบบกฎการเรียนรู้สำหรับโครงข่ายประสาทเทียมคนแรก กฎการเรียนรู้นี้คือขั้นตอนของกระบวนการปรับแต่งค่าถ่วงน้ำหนัก เพื่อทำให้เกิดค่าเอาต์พุทที่ต้องการ ซึ่งเป็นแนวทางในการพัฒนากฎการเรียนรู้ภาพที่แบบอื่น ๆ ตามมา

ในปี ค.ศ. 1957 แฟรงค์ โรเซนบลัทท์ (Frank Rosenblatt) ได้นำเสนอภาพที่แบบที่สำคัญภาพที่แบบหนึ่งของโครงข่ายประสาทเทียมนั่นคือภาพที่แบบที่เรียกว่า เพอร์เซพตรอน เป็นสถาปัตยกรรมอันประกอบไปด้วยชั้นอินพุทต่อผ่านน้ำหนักไปยังชั้นนิวรอนที่ชั้นเอาต์พุท และใช้กฎการเรียนรู้ของเพอร์เซพตรอนในการปรับแต่งค่าถ่วงน้ำหนักที่มีประสิทธิภาพดีกว่ากฎการเรียนรู้ของเฮบบ์

ในปี ค.ศ. 1960 เบอร์นาร์ด วิดโรว์ (Bernard Widrow) และมาเชียน ฮอฟฟ์ (Marcian Hoff) ได้พัฒนากฎการเรียนรู้ซึ่งใกล้เคียงกับการเรียนรู้ของเพอร์เซพตรอนและนำกฎนี้ไปใช้กับระบบหรือภาพที่แบบที่ชื่อว่าอะดาไลน์ ภาพที่แบบมาลาไดน์คือภาพที่แบบที่พัฒนามาจากอะดาไลน์โดยวิดโรว์ เช่นกัน

ในปี ค.ศ. 1972 ทูโว คาโคเนน (Teuvo Kohonen) ได้เริ่มพัฒนาการเรียนรู้แบบจัดตัวเอง (Self-organized) สถาปัตยกรรมของโครงข่ายคาโคเนนนี้ประกอบด้วยจำนวนชั้น 2 ชั้น คือชั้นอินพุทและชั้นเอาต์พุท การเชื่อมโยงที่ชั้นอินพุทไปยังนิวรอนที่ชั้นเอาต์พุทนั้นจะต่อผ่านน้ำหนัก

ข้อแตกต่างจากโครงข่ายอื่น ๆ คือกระบวนการเรียนรู้ที่สามารถจัดกลุ่มข้อมูลอินพุทได้ด้วยตัวเองและเรียกการเรียนรู้นี้ว่าการเรียนรู้แบบแข่งขัน (Competitive Learning) หลังจากปี ค.ศ .1980 เป็นต้นมา ได้มีการพัฒนาภาพที่แบบของโครงข่ายประสาทเทียมชนิดต่าง ๆ เช่นภาพที่แบบการจัดตัวเองที่เรียกว่า ART คิดค้นโดยสตีเฟน กรอสเบิร์ก (Stephen Grossberg) และ เกียล คาร์เพนเตอร์ (Gail Carpenter) ภาพที่แบบของโครงข่ายฮอปฟิลด์ คิดค้นโดยจอน ฮอปฟิลด์ (John Hopfield) นีโอคอก นิตรอน คิดค้นโดย คูนิฮิโกะ ฟูกูชิมา (Kunihiko Fukushima) ภาพที่แบบโบลสแมนน์แมชชีน (Boltzman Machine) คิดค้นโดยนักวิจัยหลายคน และภาพที่แบบที่นิยมใช้กันมากคือภาพที่แบบแพร่ย้อนกลับ (Back-Propagation) คิดค้นโดยเดวิด ปาร์คเคอร์ (David Parker) และเดวิด รูเมลฮาร์ท (David Rumelhard)

ชื่อแบบจำลอง	ผู้คิดค้น	ปีที่พัฒนา	การใช้งาน
1. Perceptron	F.Resenblatt	1957	อ่านตัวพิมพ์ดีด
2. Adaline และ Madaline	B.Widrow	1960-1962	Adaptive Equalizer
3. Avalanche	S.Grossberg	1967	เข้าใจคำพูด
4. Cerebellatron	D.Mar, J.Albus และ A.Pellionez	1969-1982	ควบคุมการเคลื่อนไหวของแขนหุ่นยนต์
5. Back Propagtion	P.Werbos, D.Parker และ D.Rumelhart	1974-1985	เลียนแบบฟังก์ชั่นทางคณิตศาสตร์, อ่านตัวเขียน, คำพูด, ทำนายหุ้น
6. Brain-State-in-a-box	J.Anderson	1977	สกัดข้อมูลบางอย่างออกจากฐานข้อมูล
7. Neocognition	K.Fukushima	1978-1984	อ่านตัวเลขและตัวเขียน
8. Adaptive Resonance	G.Carpenter และ S.Grossberg	1978-1986	Pattern Recognition

ตารางที่ 1 แสดงแบบจำลองโครงข่ายประสาทเทียมในช่วงปี ค.ศ. 1950-1986

ชื่อแบบจำลอง	ผู้คิดค้น	ปีที่พัฒนา	การใช้งาน
9. Self-Organizing Map	T.Kohonen	1980	จำลองลักษณะพื้นผิวของวัตถุเช่น ปีกเครื่องบิน
10. Hopfield	J.Hopfield	1982	ทำนายส่วนของข้อมูลที่หายไป
11. Bidirectional Associative Memory	B.Kosko	1985	Content-addressable
12. Boltzman และ Cauchy Machine	J.Hinton, T. Sejnowsky และ H.Szu	1985-1986	Pattern recognition
13. Counterpropagation	R.Hencht-Nielsen	1986	อัดข้อมูลของภาพให้น้อยลง, Table-lookup

ตารางที่ 1 แสดงแบบจำลองโครงข่ายประสาทเทียมในช่วงปี ค.ศ. 1950-1986 (ต่อ)

2.2 การทำงานของเซลประสาทเทียม

ปัจจุบันถึงแม้ว่าคอมพิวเตอร์จะมีความสามารถในการทำงานมากขึ้นแต่ความสามารถอย่างหนึ่งที่คอมพิวเตอร์ไม่สามารถทำได้ดีเท่ามนุษย์ ก็คือความสามารถในการจดจำสิ่งต่าง ๆ ที่เกิดขึ้นมาแล้ว เมื่อเปรียบเทียบกับมนุษย์ มนุษย์สามารถที่จะจดจำหน้าตาของบุคคลอื่นได้อย่างรวดเร็ว ถึงแม้ว่าคน คนนั้นจะมีการเปลี่ยนแปลงบางสิ่งบางอย่างไปจากเดิม ซึ่งคอมพิวเตอร์อาจจะทำไม่ได้ถึงแม้ทำได้ก็ต้องใช้เวลาในการประมวลผลนาน

จากที่กล่าวมาแล้วข้างต้นสิ่งนี้เองเกิดจากสถาปัตยกรรมของสมองมนุษย์ซึ่งต่างจากคอมพิวเตอร์ สมองมนุษย์ประกอบด้วยเซลประสาทจำนวนมหาศาลเชื่อมต่อกัน ซึ่งเซลเหล่านี้ทำหน้าที่เป็นหน่วยประมวลผลให้กับสมองมนุษย์ การส่งสัญญาณของเซลประสาทระหว่างเซลหนึ่งสู่เซลหนึ่งใช้เวลาเร็วมากคือประมาณสิบส่วนล้านวินาที ซึ่งภาพที่แบบการส่งสัญญาณทำในลักษณะแบบขนาน และวิธีการแก้ปัญหาของสมองมนุษย์ใช้ประสบการณ์การเรียนรู้จากอดีตที่ผ่านมา

เพื่อนำมาวิเคราะห์ในการแก้ปัญหานั้น ๆ แต่สถาปัตยกรรมคอมพิวเตอร์ประกอบด้วยหน่วยประมวลผลเพียงหน่วยเดียว และการทำงานใช้ชุดคำสั่งสั่งงานเป็นลำดับขั้น อีกทั้งวิธีการแก้ปัญหาต่าง ๆ นั้นจะต้องทราบลำดับขั้นตอนการทำงานที่แน่นอนของปัญหานั้น

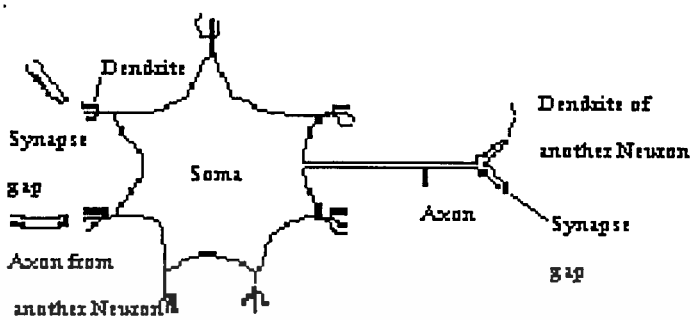
จากความต้องการที่ต้องการให้คอมพิวเตอร์สามารถจัดการปัญหาในลักษณะที่กล่าวข้างต้น จึงได้มีการนำเอาสถาปัตยกรรมสมองมนุษย์มาเป็นตัวแบบในการประมวลผล ซึ่งเรียกว่า Artificial Neural Systems หรือโครงข่ายประสาทเทียม (Neural Network)

โครงข่ายประสาทเทียมประกอบด้วยหน่วยประมวลผลแบบง่าย ๆ จำนวนมหาศาลที่เชื่อมต่อกันซึ่งเรียกว่า Neural, Cell หรือ Nodes และเรียกส่วนที่เชื่อมต่อกันระหว่างโหนดว่า Interconnection โดยที่ Interconnection จะเก็บความรู้ที่ใช้ ในการแก้ปัญหานั้น ๆ ของโครงข่าย ซึ่งความรู้ที่ใช้ในการแก้ปัญหานี้ได้มาจากการเรียนรู้ตัวอย่างของปัญหานั้น ข้อเปรียบเทียบระหว่างโครงข่ายประสาทเทียมกับสมองมนุษย์นั้น คือการเก็บข้อมูลต่าง ๆ ในภาพที่แบบของแพทเทิร์นของ Interconnection และลักษณะการแก้ปัญหาด้วยวิธีการเรียนตัวอย่างซึ่งเทียบได้กับการสะสมประสบการณ์ของมนุษย์

โครงข่ายประสาทเทียมได้ถูกนำมาประยุกต์ใช้งานด้านต่าง ๆ หลายด้าน เช่น การจัดเก็บ, การค้นหาแพทเทิร์น, การแยกแยะแพทเทิร์น, การแบ่งกลุ่มแพทเทิร์นที่มีลักษณะเหมือนกันเป็นต้น คุณสมบัติที่สำคัญอย่างหนึ่งของโครงข่ายประสาทเทียมคือ Generalization โครงข่ายสามารถวิเคราะห์หรือสร้างคำตอบสำหรับปัญหาซึ่งมีลักษณะที่เหมือนกันหรือคล้ายกับตัวอย่างที่นำมาสอนได้อย่างมีประสิทธิภาพ โดยที่ปัญหานั้นไม่เคยถูกนำมาสอนให้กับโครงข่ายเลย

2.3 ตัวแบบของเซลประสาทเทียม (Artificial Neural Model)

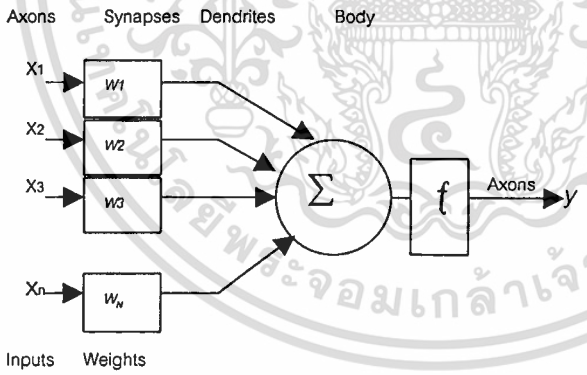
ลักษณะเซลประสาททางชีวภาพประกอบด้วย 3 ส่วนหลัก คือ 1. เดนไดรซ์ (Dendrites) 2. โซมา หรือตัวเซล (Somas or Cell Body) ซึ่งจะมีนิวเคลียสอยู่ตรงกลาง 3. แอคซอน (Axon) เป็นส่วนที่ยื่นออกไปเพื่อรับและส่งสัญญาณจากเซลประสาทอื่น ๆ 4. ไซแนปส์ (Synapse) จะอยู่บริเวณรอยต่อระหว่างก้านของเซลล์ประสาทที่ต่างกัน ซึ่งสามารถเปลี่ยนค่าความต้านทานได้ตามสัญญาณที่ส่งระหว่างกันของเซลประสาท การส่งสัญญาณระหว่างเซลประสาททำได้โดยการถ่ายเทสารประกอบโซเดียมและโพแทสเซียม เดนไดรซ์จะรับสัญญาณจากเซลประสาทที่อยู่รอบข้าง ผ่านทางไซแนปส์ด้วยปฏิกิริยาทางเคมีที่เกิดขึ้นจะทำการปรับเปลี่ยนความถี่ของสัญยาณที่เข้ามา ตัวเซลทำหน้าที่รวมสัญญาณที่เข้ามาแล้วทำการส่งสัญญาณออกให้กับแอคซอนเพื่อผ่านต่อให้กับเซลอื่น ๆ ซึ่งสัญญาณที่ออกจากตัวเซลมีลักษณะเป็นแบบสัญญาณกระตุ้นให้กับเซลอื่น กล่าวคือสัญญาณที่ส่งออกมาจากตัวเซลมีจำนวน 100 ครั้งต่อวินาทีถือว่าสัญญาณที่ออกมาเป็นสถานะ



ภาพที่ 2.1 แสดงตัวแบบของเซลประสาทเทียม

การกระตุ้น (Fire) โดยทั่วไปเราสนใจสัญญาณที่ออกจากเซลประสาทในภาพที่ของสถานะการกระตุ้นและไม่กระตุ้น (Not Fire) ให้กับเซลที่อยู่รอบข้างมากกว่าสนใจเงื่อนไขที่เกี่ยวข้องทำให้เซลประสาทเกิดสถานะนี้ขึ้นมา ภาพที่ 2.1 แสดงลักษณะของเซลประสาททางชีววิทยา

ตัวแบบของเซลประสาทเทียมที่ใช้ใน Artificial Neural Network มีคุณลักษณะพื้นฐานเหมือนกับเซลประสาททางชีววิทยา



ภาพที่ 2.2 แสดงลักษณะของตัวแบบเซลประสาทเทียมที่ใช้กันโดยทั่วไปซึ่งใช้พื้นฐานจากตัวแบบที่เสนอโดย MacCulloch และ Pitt ในปี 1943

จากภาพที่ 2.2 เซลประสาทเทียมรับสัญญาณ X1-Xn จากเซลรอบข้าง โดยไซแนปส์หรืออินเตอร์คอนเน็กชันจะทำการปรับเปลี่ยน หรือถ่วงน้ำหนักสัญญาณที่เข้ามาด้วย W1-Wn ซึ่งจะแสดงการหาค่าผลรวมสัญญาณที่เข้าสู่เซลประสาทเทียมโหนดที่ I ซึ่งเรียกว่า Net Input และ Y คือค่าสัญญาณที่ออกจากเซลประสาทเทียมซึ่งจะปรากฏที่แอคซอน โดยค่าสัญญาณนี้จะส่งผ่านทางฟังก์ชั่นกระตุ้น (Activation Function ($f(\ )$)) ซึ่งฟังก์ชั่นนี้ทำหน้าที่เลือกผ่านค่าสัญญาณให้กับ

แอคซอน โดยจะทำการย่อขนาดผลรวมสัญญาณที่ออกจากตัวเซลให้อยู่ในช่วง 0 ถึง 1 ในบางครั้งอาจไม่มีการใช้ฟังก์ชั่นกระตุ้นก็ได้

2.4 สถาปัตยกรรมของโครงข่ายประสาทเทียม (Neural Network Architecture)

โครงข่ายประสาทเทียมประกอบด้วยเซลประสาทเทียมหรือโหนดจำนวนมากที่เชื่อมต่อกัน ซึ่งการเชื่อมต่อจะแบ่งออกเป็นกลุ่มย่อยเรียกว่าชั้น (Layer) โดยทั่วไปแล้วแบ่งออกเป็น 2 แบบ คือโครงข่ายแบบชั้นเดียว (Single Layer) และโครงข่ายแบบหลายชั้น (Multilayer) การกำหนดจำนวนชั้นของโครงข่ายนี้ส่วนที่เป็นหน่วยรับข้อมูลเข้า (Input Unit) จะไม่ถูกนับด้วยเนื่องจากเป็นส่วนที่ไม่มีการคำนวณ ดังนั้นเราอาจกล่าวได้ว่าจำนวนชั้นของโครงข่ายคือ จำนวนชั้นของอินเตอร์คอนเน็กชันที่ถูกถ่วงน้ำหนัก โดยปกติแล้วโครงข่ายแบบหลายชั้นสามารถแก้ปัญหาที่มีความซับซ้อนได้ดีกว่าโครงข่ายแบบชั้นเดียว และในบางปัญหานั้นไม่สามารถแก้ปัญหาด้วยโครงข่ายแบบชั้นเดียวได้

2.5 การฝึกสอนโครงข่ายประสาทเทียม

การนำโครงข่ายประสาทเทียมไปใช้งานจะต้องมีการสร้างความรู้ (Knowledge) ให้กับโครงข่ายก่อนซึ่งขั้นตอนนี้เรียกว่า การฝึกสอนโครงข่าย ข้อมูลที่นำมาฝึกสอนให้แก่โครงข่าย คือตัวอย่างข้อมูลของปัญหาที่ต้องการนำโครงข่ายไปใช้ในการแก้ปัญหา โดยการเรียนรู้ของโครงข่ายจะทำการปรับค่าถ่วงน้ำหนักของอินเตอร์คอนเน็กชันที่เชื่อมต่อกันในแต่ละชั้นของโครงข่าย เพื่อสร้างฐานความรู้ของปัญหานั้น ๆ ให้กับโครงข่าย ลักษณะการสอนของโครงข่ายประสาทเทียมแบ่งได้ 2 แบบ คือ การฝึกแบบมีผู้สอน (Supervised Training) และการฝึกแบบไม่มีผู้สอน (Unsupervised Training)

การฝึกแบบมีผู้สอน (Supervised Training)

ข้อมูลที่เข้าสู่โครงข่ายด้วยการสอนวิธีนี้ประกอบด้วย 2 ส่วนคือ อินพุทเวกเตอร์ (Input Vector) ที่เป็นตัวอย่างข้อมูลของปัญหานั้น ๆ และเวกเตอร์เป้าหมาย (Target Vector) ซึ่งเป็นผลลัพธ์ที่เราต้องการให้โครงข่ายสร้างเมื่อมีการนำอินพุทเวกเตอร์ของตัวอย่างข้อมูลมาสอนให้กับโครงข่าย ขณะสอนโครงข่ายด้วยวิธีนี้จะมีการกำหนดค่าผลลัพธ์เป้าหมายให้กับแต่ละอินพุทเวกเตอร์ที่นำเข้ามาสอน โดยโครงข่ายจะนำค่าผิดพลาดที่ได้จากการคำนวณของโครงข่าย (Actual Output) กับค่าผลลัพธ์เป้าหมายที่กำหนดขึ้นมาใช้ในการเรียน หรือปรับค่าถ่วงน้ำหนักของอินเตอร์คอนเน็กชัน ตัวแบบโครงข่ายที่มีการเรียนรู้ลักษณะนี้ได้แก่ Adaline, Perceptron, Backpropagation เป็นต้น

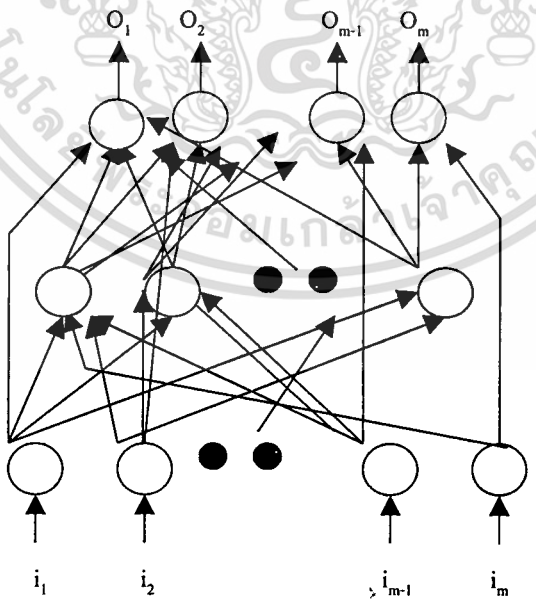
การฝึกแบบไม่มีผู้สอน (Unsupervised Training)

ข้อมูลที่นำมาใช้ในการสอนโครงข่ายด้วยวิธีนี้มีเพียงอินพุทเวกเตอร์ที่เป็นตัวอย่างข้อมูลของปัญหานั้น ๆ เพียงอย่างเดียว โดยขณะสอนจะไม่มีการกำหนดค่าผลลัพธ์เป้าหมายที่ต้องการเพื่อใช้ในการเรียนหรือปรับค่าถ่วงน้ำหนักให้กับโครงข่าย ซึ่งลักษณะการปรับค่าถ่วงน้ำหนักของโครงข่ายจะใช้ข้อมูลที่นำมาสอนในการปรับเปลี่ยนค่า โดยกลุ่มของโหนดที่ถูกปรับค่าถ่วงน้ำหนักนี้จะเป็นกลุ่มเดียวกับกลุ่มข้อมูลที่ให้ผลลัพธ์คล้าย ๆ กัน การฝึกสอนด้วยวิธีนี้ใช้เวลาในการเรียนตัวอย่างได้รวดเร็วกว่าการฝึกสอนแบบมีผู้สอน ตัวอย่างตัวแบบที่ใช้ในการเรียนลักษณะนี้ เช่น Kohonen Self-Organiation Maps (SOM), Counterpropagation Network (CPN), Adaptive Resonance Theory (ART)

2.6 พื้นฐานของ Neural Network

โดยทั่วๆไป Neural Network เป็นการรวบรวมของ Single Analog Processors ที่ทำการเชื่อมต่อผ่านการ Link ที่เรียกว่า Interconnect หรือเรียกง่าย ๆ ว่า Connection

Neural Network จะแทนในภาพที่แบบของ Direct Graph โดยที่ Node ใช้แทน Processing Elements เส้นโค้งใช้แทน Modulating Connection หัวลูกศรที่อยู่บนเส้นโค้งใช้แทน Normal Direction ของการไหลของสัญญาณ แสดงดังภาพที่ 2.3



ภาพที่ 2.3 แสดงถึงลักษณะโครงสร้างโดยทั่วไปของ Layer ลักษณะโครงข่ายแบบ แพร่ไปข้างหน้า

จากภาพที่ 2.3 Processing Elements โดยปกติจะใช้การรวมกลุ่มเข้าสู่โครงสร้างของ Layer ที่เรียกว่า Slab หรือ Layer โดยที่แต่ละ Processing Elements อยู่บนแต่ละ Layer จะทำงานแบบ Analog Integration กับอินพุทของมัน เพื่ออธิบาย (กำหนดค่า activation)

การประมวลผลเริ่มต้นกับ Entire Network ในสถานะที่ไม่มีอะไรเปลี่ยนแปลง External Pattern ประกอบด้วยกลุ่มของสัญญาณที่จะทำการประมวลผลโดยโครงข่าย เมื่อทำการประยุกต์เข้าสู่ อินพุท Layer โดยที่สัญญาณแต่ละตัวจะกระตุ้น Processing Element 1 ตัว บน อินพุทเลเยอร์ ในแต่ละ Processing Element บนอินพุทเลเยอร์ทำการสร้างเพียง 1 สัญญาณเอาต์พุท Magnitude ที่เป็นหน้าที่ของการกระตุ้นทั้งหมดจะได้รับโดย Unit เอาต์พุทจะถูกสร้างโดยทั้งหมดของ Processing Element บน Layer ที่ผ่านไปบน อินพุตแพทเทิร์น ไปยังlayer ถัดไป ของ Processing Element การประมวลผลนี้จะถูกทำซ้ำอีกครั้งหนึ่งจนกระทั่ง layer สุดท้ายของ Processing Element จะถูกสร้างเป็นเอาต์พุทสำหรับ เวกเตอร์แพทเทิร์นปัจจุบัน

จากภาพที่ 2.3 อธิบาย Typical Model ของ Neural Network Processing Element (เรียกว่า โหนดหรือ Unit) อินพุทของ Unit จะมีลักษณะเหมือน Analog Integrator โดยที่ Modulated สัญญาณอินพุททำการเชื่อมทั้งภาพที่แบบของสัญญาณอินพุท เพื่อเข้าสู่ Unit สิ่งที่สำคัญของ Typical Unit เป็นสิ่งที่บ่งบอกว่าทั้ง Magnitude และ Polarity ของสัญญาณอินพุทเป็นส่วนช่วยให้ Net Input ได้รับโดย Unit ของโครงข่าย จะมีกลไกที่สัญญาณอินพุทจะมีขนาดใหญ่หรือเล็ก Excitorator หรือ Inhibitory ที่มีผลกับ Unit

ในบางภาพที่แบบของ Neural Network เช่นเดียวกับ model เริ่มแรกของ McCulloch Pitts Unit จะถูกกำหนดว่าเป็นสัญญาณอินพุทแบบ Single Inhibitory ที่พอเพียงจะระงับ Output จาก Unit อื่นๆ ความเป็นอิสระของ อินพุทแบบ Excitorator ทั้งหมด เพราะว่า model ของ Neural Network ในปัจจุบันก็ใช้วิธีการเดียวกัน เราจะพิจารณาความต้องการของเราบน Unit ที่ทำการเชื่อมโยง Input ในสัดส่วนโดยตรงเพื่อให้สัญญาณแรงไปจนถึง Connection

ในระบบ Analog แต่ละ Unit จะทำการสร้างแบบต่อเนื่องจากอินพุท เพื่อที่จะจำลองโครงข่ายเหล่านี้บนระบบของดิจิตอล คอมพิวเตอร์ จะทำการ model พฤติกรรมของ Unit ถึงแม้ว่าจะถูกสร้างใน discrete time มีการจำแนกเรียบร้อยแล้ว แต่ละ Unit ได้รับอินพุท ซิมูเลเตอร์ ของUnit อื่นๆ กับสิ่งที่ยกเว้นของ Input Layer ที่ได้รับ Input จากเพียง 1 ที่ การคำนวณจะกระทำโดย Unit ที่ได้กำหนด อินพุท ซิมูเลเตอร์ของมัน ที่ทั้งหมดจะเป็นแบบเดียวกัน สัญญาณที่มายังแต่ละ Connection จะถูกคูณ (Modulator) โดยค่าถ่วงน้ำหนักของ Connection โดยแต่ละ Modulator Input จะถูกนำมารวมกัน (Sum) โดย Unit จะสร้าง สัญญาณซิมูเลเตอร์ไปยัง Unit

ในการคำนวณต้องทำการคำนวณ อินพุท จะอธิบายแต่ละ Unit I ในโครงข่ายที่เวลา t ดังสมการที่ (2.1)

$$net_I(t) = \sum_{j=1}^{n} w_{ij}(t) o_j(t) \quad \text{.....(2.1)}$$

โดยที่ $net_I(t)$ ใช้แทน net input signal ไปยัง unit ที่ I ในโครงข่าย

$o_j(t)$ ใช้แทนเอาต์พุทจาก Unit ที่ j ในโครงข่าย

$w_{ij}(t)$ ใช้แทนค่าถ่วงน้ำหนักที่มีความสัมพันธ์กับ connection ที่มาจาก Unit ที่ j ไปยัง Unit ที่ I

n ใช้แทนจำนวนของ Unit อื่น ๆ ที่ทำการ Connection ไปยัง Input ของ Unit ที่ I

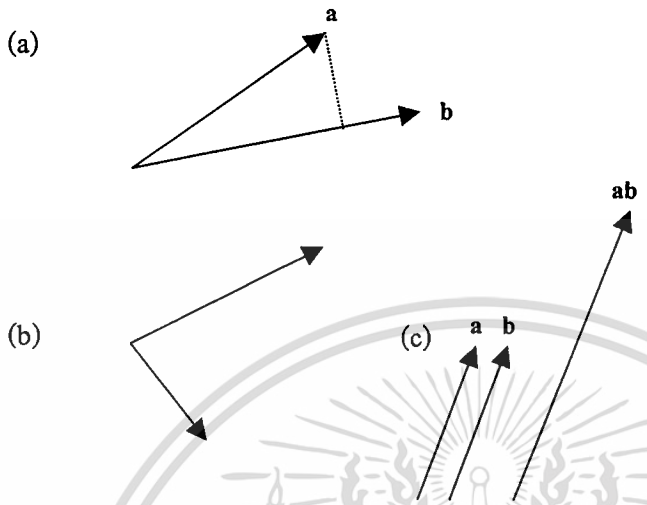
การใช้ Model นี้ เราจะเห็นความคล้ายคลึงระหว่างการคำนวณอินพุทที่กระทำโดย Unit ของโครงข่าย และการคำนวณ Inner-Product (Vector) O เป็นกลุ่มของสัญญาณอินพุทไปยังเป็นเวกเตอร์ที่มี Dimension n ถ้าเราพิจารณาค่าถ่วงน้ำหนักของ อินพุท Connection ไปยัง Unit I W_I จะมี Dimension n อินพุทจะคำนวณได้ดังสมการที่ 2.2 และสมการที่ 2.3

$$Net_I = O.\ W_I \quad \text{.....(2.2)}$$

$$= \| O \| \| W_i \| \cos(\theta) \quad \text{.....(2.3)}$$

โดยที่ θ ใช้แทนมุมระหว่าง 2 เวกเตอร์ใน n Dimension Euclidian Space

พิจารณา Inner-Product แสดงในภาพที่ 2.4 Inner-Product ระหว่าง 2 เวกเตอร์ใน n Dimension Hyperspace ในการทำการ Projection ของ 1 เวกเตอร์ หรือมากกว่านั้น



ภาพที่ 2.4 แสดงการ Projection ของ Vector (a) การ Projection ของ Vector a บน Vector b (b) Vector ที่มีการทำมุมกันแบบ Orthogonal เมื่อทำการ Inner Product กัน จะมีค่าเท่ากับ 0 (c) แสดงถึง Vector ในแบบขนานกัน เมื่อมีการ Inner Product กันจะมีค่าเท่ากับการรวมเอาทิศทางของทั้ง 2 Vector เข้าด้วยกัน

2.6.1 Neural Activation

จากภาพที่ 2.3 ที่ได้กล่าวมาแล้วนั้น ขั้นตอนถัดไปหลังจากที่ได้กำหนด อินพุทซิมูเลเตอร์ที่ได้รับจาก Unit เพื่อทำการเปรียบเทียบให้ค่าอินพุทเป็น Activation value (หรือเรียกว่า Activation) การ Activation ของ Unit เป็นลักษณะแบบต่อเนื่อง โดยเป็นระดับของ Excitator ของ Unit ความแรงของ Activation ขึ้นอยู่กับ Unit นั้น จะ Excitator (หรือ Inhibitory) กว่า Unit อื่นๆ ในโครงข่าย

Unit จะทำการกำหนด การ Activation โดยตรงจาก อินพุทซิมูเลชั่นตามสมการที่ 2.4

$$a_i(t) = F_i(a_i(t-1), net_i(t)) \quad \text{.....(2.4)}$$

จากสมการที่ 2.4 ค่าของ $net_i(t)$ ทำให้เรารู้ว่า Activation ของ Unit จะมีค่าเป็น Explicit Function ของ net input ไปยัง Unit ที่ ณ เวลาปัจจุบันเสมอ ค่าของ $a_i(t-1)$ เป็นค่า Activation ของ Unit ในเวลาก่อนหน้า สรุปได้ว่า Activation ของ Unit จะมีค่าเท่ากับ Current Input จะได้ดังสมการที่ 2.5

$$a_i(t) = net_i(t) \quad \text{.....(2.5)}$$

2.6.2 Activation Function

Unit ที่สร้างสัญญาณเอาต์พุทที่เกี่ยวข้องกับการ Activation โดยใช้ Transfer Function ที่รู้จักกันคือ Activation Function ซึ่งสามารถคำนวณได้ดังสมการที่ 2.6 และสมการที่ 2.7 ดังนี้

$$O_i(t) = f_i(a_i(t)) \quad \text{.....(2.6)}$$

$$= f_i(net_i(t)) \quad \text{.....(2.7)}$$

ภาพที่แบบที่เหมาะสมของ Activation Function จะถูกใช้โดย Unit ที่เป็นอิสระกับ Function ต่าง ๆ ซึ่งรวมไปถึงประเภทของโครงข่ายที่ใช้ประกอบด้วย Unit, Function ที่จะถูกกระทำโดย Unit, การตีความภายนอกของเอาต์พุทในโครงข่าย

2.6.2.1 The Linear Unit

Linear Unit เป็น Processing Elements ที่มีลักษณะแบบตรงไปตรงมา โดยพื้นฐานแล้ว ถ้ามีการรวม (Sum) ของทุก ๆสิ่ง ที่มาจาก Unit ผลลัพธ์ก็จะเป็นสิ่งที่ได้จากการ Sum นั่นเอง ดังสมการที่ 2.8

$$f_i^1(net_i(t)) = net_i(t) \quad \text{.....(2.8)}$$

โดยที่ ตัวที่ยกกำลัง 1 จะใช้แทน Linear Activation Function

Unit ที่ได้ทำงานแบบนี้ ในการตอบสนองจะแสดงบทบาทที่สำคัญที่หลากหลายใน Neural Network Application รวมถึงการใช้ในลักษณะของ

- Fan-Out Units เป็นเพียง Single Input ที่จะกระจายโดยไม่เปลี่ยนแปลงไปยัง Processing Elements หลายๆ ตัว Application ของ Device สามารถพบใน อินพุทเลเยอร์ของโครงสร้างโครงข่ายทั่ว ๆไป
- Linear Combiner มีหลายสัญญาณอินพุทจะทำการรวมกันในลักษณะของ Coherent โดยที่ Application ของ Device นี้จะรวมถึง Linear Pattern Interpolation และ Pattern Multiplexing
- Analog Output Device สัญญาณจะสร้างโดย Unit ที่ถูกตีความโดย Network Unit อื่น ๆ หรือจากภายนอก เป็นค่าตัวแปรแบบต่อเนื่อง

จากความสามารถรอบตัวของ Linear Function จึงเป็นวิธีการหนึ่งใน Activation Function ที่นำมาใช้ใน Neural Network

2.6.2.2 Binary-Threshold Units

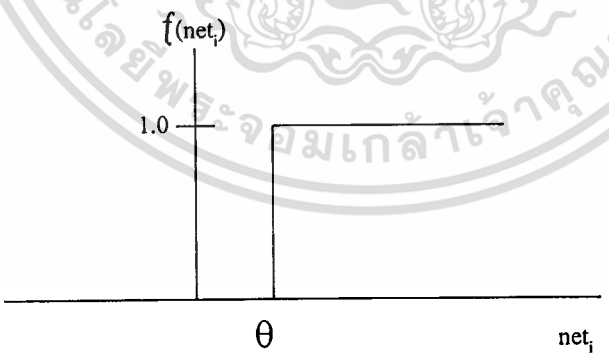
Binary-Threshold Units เป็น Unit ที่มี 2 Stable States (Active หรือ Inactive) Activation Function จะใช้สมการดังสมการที่ 2.9

$$f_i^{bt}(net_i(t)) = \begin{cases} 1 & \text{ถ้า } net_i(t) > \theta \\ 0 & \text{กรณีอื่นๆ} \end{cases} \quad \text{.....(2.9)}$$

โดยที่ θ ใช้แทนจุดของการ Transition ระหว่าง 2 State ในหลาย ๆภาพที่แบบของ Neural Network และจะถูก ตั้งค่าให้เป็น 0 ในกรณีที่ Unit กลายเป็นการตรวจจับ Polarity

ถ้าอินพุท Active จะมีค่าเป็น Positive (มีค่ามากกว่า 0) Unit จะถูกกระตุ้นและสร้าง Active Output ในหลายๆ กรณี

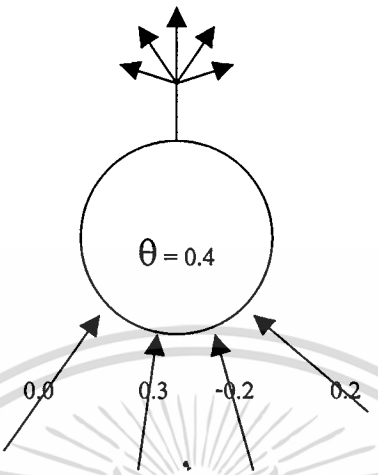
ถ้าเอาต์พุทจาก Unit จะ Inhibited (ถูกตั้งค่าให้เป็น 0) แทนที่ การซิมูเลเตอร์ที่ไม่มีอยู่ในอินพุท กลายเป็น Threshold ในภาพที่ 2.5 จะแสดงถึงกราฟที่แสดงถึงการตอบสนอง Activation Function นี้กับ Input Stimulate ที่ได้รับโดย Unit ที่ I



ภาพที่ 2.5 แสดงการตอบสนองของ Binary Threshold Unit

จากภาพที่2.3 จะพบว่า Output จาก Unit สามารถเป็นได้ 2 States

Neural Network Unit ที่แสดงถึงพฤติกรรมนี้ใช้เพื่อตรวจจับ Presence หรือ Absence ของสิ่งที่อยู่ใน Input Pattern ที่ถูกเสนอไปยัง Unit ตัวอย่างจากการพิจารณา Binary-Threshold Units อธิบายในภาพที่ 2.6



ภาพที่ 2.6 แสดงว่า Binary Threshold Unit ทำการ Detecture Feature ได้อย่างไร

จากภาพที่ 2.6 พิจารณาที่ค่าถ่วงน้ำหนักมีความสัมพันธ์กับแต่ละตัวของ Input Connection จะพบว่า Unit จะตอบสนองกับค่าเอาต์พุทที่เป็นบวกเท่านั้น ดังนั้นเมื่อสัญญาณอินพุท 2 และ 4 เข้ามาจะ Active โดยทันที ในขณะที่สัญญาณอินพุท 3 จะไม่ Active เราจะตีความว่าค่าบวกที่ตอบสนองจาก Unit นี้เป็นสิ่งที่แสดงว่า 2 และ 4 จะแสดงอยู่ใน Input pattern ในขณะที่ 3 จะไม่อยู่

2.6.2.3 Sigmoidal Units

Sigmoidal Units ทำการสร้างในลักษณะที่คล้ายคลึงกับ Binary Threshold Units ที่ได้อธิบายไปแล้ว โดยที่ Sigmoidal Units จะทำการสร้างสัญญาณเอาต์พุทที่ 2 Stable States และ Transition Regions

ความแตกต่างระหว่าง Sigmoidal Units กับ Binary Threshold Unis : Sigmoidal Units เป็นลักษณะทางคณิตศาสตร์แบบต่อเนื่อง สิ่งนี้เองเป็นความแตกต่าง มีหลายๆ function ที่สามารถใช้ model แบบ Sigmodal Units ที่เป็นแบบ Curve ได้ แสดงดังสมการที่ 2.10 และสมการที่ 2.11

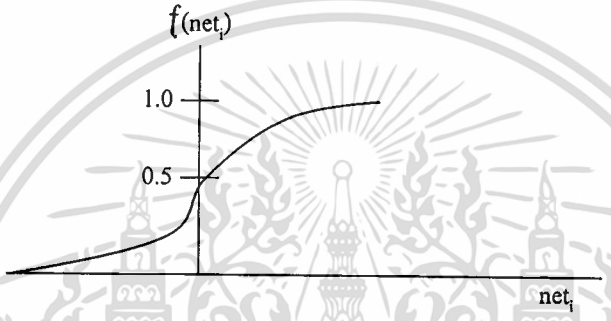
$$f_i^s(net_i(t)) = 1/\,1+e^{-(neti(t)-\theta/\Upsilon)} \quad \text{.....(2.10)}$$

สมการที่ 2.10 เป็นสมการในภาพที่แบบทั่ว ๆไป สำหรับ Sigmodal Function เพราะว่าสามารถควบคุมทั้งจุดของการ Transition และ ภาพที่ร่างของ Curve โดยอาศัยความแตกต่างของค่า

$f_i^s(net_i(t)) = ½(1+\tanh(\lambda\, net_i(t))$.....(2.11)

สมการที่ 2.11 จะควบคุมภาพที่ร่างของ Curve โดยใช้ความแตกต่างที่เป็นอิสระของ gain parameter

Function นี้จะถูกนำมาใช้สำหรับโครงข่ายที่ระบุโดยเฉพาะโดยที่ Gain จะต้องกำหนดตาม Application ในกรณีอื่นๆ ของการตอบสนองของ Unit ที่ได้ประโยชน์จาก Sigmoidal Activation Fuction อธิบายในภาพที่ 2.5



ภาพที่ 2.7 แสดงการตอบสนองของ Sigmoidal Unit จะสังเกตุได้ว่าภาพที่ร่างของ Curve ที่เป็นภาพที่ตัว S ค่อยเพิ่มขึ้น ซึ่งวิธีการนี้จะให้ค่า ความผิดพลาดน้อยมาก

พิจารณาความคล้ายคลึงกันระหว่าง Sigmoidal Function และ Binary Threshold Function

- Activation Function ทั้งคู่มี 2 Regions ที่มีความแตกต่างที่สำคัญในอินพุทคือไม่มีการเปลี่ยนแปลงอย่างรวดเร็ว สำหรับ Output ที่สร้างโดย Unit
- ความกว้างของ Transition Regions ในทั้ง 2 Function จะมีเพียงเล็กน้อย เมื่อเปรียบเทียบกับความกว้างของ Stable Regions
- ทั้ง 2 Function ใน Transition Regions จะให้ค่าอยู่ประมาณ 0 ถึงแม้ว่าจะมีความสามารถในการเลื่อนไปทางซ้ายหรือขวาซึ่งทำได้โดยการแก้ไขค่า ตามสมการที่ 2.10

ในกรณีของ Binary Threshold Units โหนดที่เป็น Sigmoidal จะถูกนำมาใช้เป็นตัวที่ใช้ในการตรวจจับและวัตถุประสงค์ในการแยกแยะ Pattern ใน Application เหล่านี้ เอาต์พุทจาก Sigmoidal Units จะถูกตีความในแบบเดียวกันกับเอาต์พุทจาก Binary Threshold Units :

โดยที่ Active Output จะถูกตีความว่าแสดงถึง Presence ของการเชื่อมโยงในลักษณะพิเศษของ Features ใน Input Pattern ในขณะที่ Inactive Output โดยปกติจะแสดงถึง Absence ของ Features เหล่านั้น

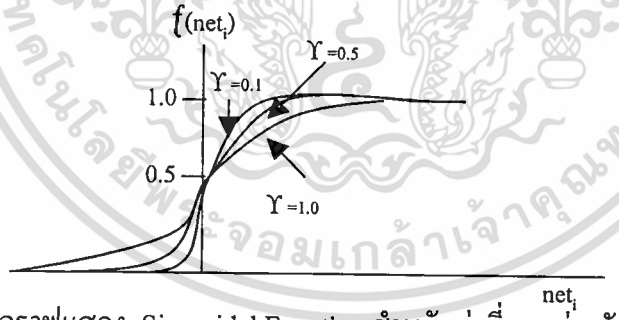
โดยธรรมชาติของ Sigmoidal Function จะทำการสร้างสภาวะเกี่ยวกับ Unit ที่ได้รับการแยกแยะถูกต้อง ใน Sigmoidal Activation Function จะมี 3 สถานะ สำหรับเอาต์พุทของ Unit สถานะที่ 3 ก็คืออยู่ระหว่าง Active และ Inactive

กลับไปพิจารณาภาพที่ 2.7 จะพบว่าเนื่องจาก Sigmoidal Function เป็นแบบต่อเนื่อง Unit ที่ใช้ Sigmoidal Activation Function สามารถสร้างค่า Output จริง ๆ ระหว่าง ค่าที่น้อยที่สุด (0.0) และค่าสูงสุด (1.0) เพราะว่าวัตถุประสงค์ของเราเพื่อทำการแยกแยะ Input ที่เข้าสู่ 1 ใน 2 State

สามารถตีความได้ว่า Output จาก Sigmoidal Units ที่มีค่ามากกว่า 0.8 จะเป็น Active Output ในขณะที่เราพิจารณาค่า Output ว่ามีค่าน้อยกว่า 0.2 เป็นค่า Inactive Output ค่า output จะตกลงระหว่าง 0.2 และ 0.8 จะได้รับการพิจารณาว่าเป็น Transition

ในการพิจารณา Output ที่สร้างโดย Sigmoidal Unit เมื่อ Input Pattern ที่เข้ามาเป็น Inactive (มีค่าเป็น 0) ค่า 0 จะถูก Modulated โดยค่าถ่วงน้ำหนัก สร้าง Zero Active

เมื่อเข้าสู่ Sigmoidal Unit จะสร้าง Output ที่ค่า 0.5 ซึ่งจะเป็นค่าที่อยู่กึ่งกลางของ Output State เราสามารถเปลี่ยนผลของ Transition Region บนสิ่งที่เรียกว่า พฤติกรรมของ Sigmoidal Unit โดยการเปลี่ยนค่าของ Υ ซึ่งจากภาพที่ 2.8 ใช้แสดงผลของ Υ บน Sigmoidal Function



ภาพที่ 2.8 กราฟแสดง Sigmoidal Function สำหรับค่าที่แตกต่างกันของ Υ

2.6.2.4 Competitve Units

Outstar เป็น Dynamic Unit ที่อยู่ใน Competitve Units จะเหมือนกับภาพที่แบบของ Binary Threshold Unit จะมี 2 state คือ Active (1) และ Inactive (0) เมื่อ Active เกิดขึ้น Outstar จะ Transmit ไปยัง pattern ที่อยู่ในหน่วยความจำ สามารถอธิบายได้โดยใช้ค่าถ่วงน้ำหนักที่อยู่ใน Output Connection จาก Unit ไปยัง Layer ต่อมาของ Unit นั้น ในทำนองเดียวกันถ้าเป็น Inactive จะพบว่า Outstar จะไม่สร้างสิ่งใดเพื่อไปยัง Layer ถัดไป

การทำงานของ Outstar เป็นสิ่งที่น่าสนใจ แต่ไม่นิยมมากนัก สำหรับ Unit ที่สามารถสร้าง Memory Pattern ได้จะพบค่าที่ได้มีค่าความแตกต่างน้อยมากจนเราสามารถเลือก Outstar ได้หลายตัว เพื่อให้การเลือกเป็นไปอย่างถูกต้องจะมีการเชื่อม Outstar Function กับ Neural Device ที่เรียกว่า Instar ใน Instar จะแสดงเพียง Input pattern ที่ระบุถึง Unit ที่อยู่บน Layer ของ Instar ซึ่งจะครบถ้วนเมื่อมี Input Pattern ตัวใหม่ที่แสดงอยู่ การแข่งขันอาจจะคิดได้ว่าเป็นกระบวนการเลือกจากความสัมพันธ์ของ Outstar ที่ระบุกับ Instar แต่ละตัว และเมื่อยอมให้ Instar แต่ละตัวแสดงความสอดคล้องกับ Outstar เพียงแต่ตัวที่ชนะการแข่งขันเราก็จะทำการสร้างโครงข่ายที่มีความสามารถในการ Storing และการเลือกเพื่อนำมาใช้ใหม่ได้ทันที

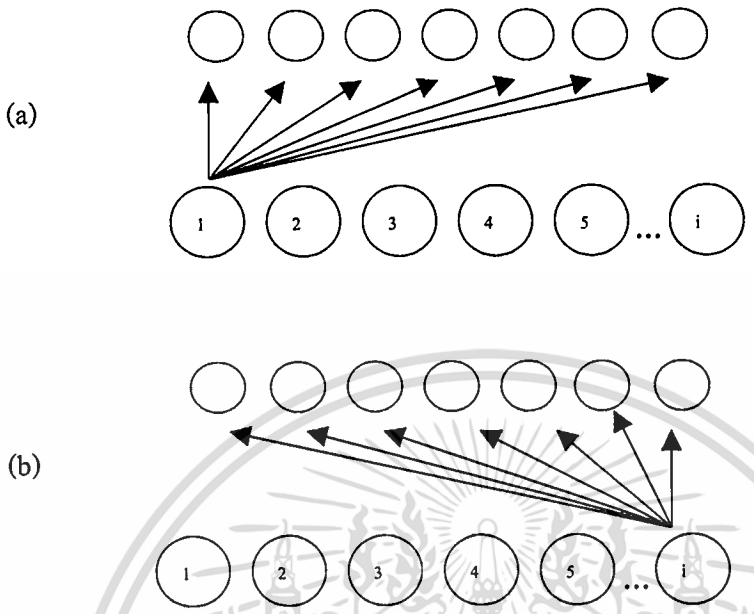
ถ้าพิจารณาจาก Output Memory Pattern ที่มีความสัมพันธ์กับ Input Pattern ที่เหมาะสม กระบวนการที่พิจารณาผู้ชนะในการแข่งขันจะเกิดขึ้นในระหว่างการคำนวณ Input-Activation ถ้าเราพิจารณาเกี่ยวกับการออกแบบใน Neural Network การออกแบบโครงข่ายจะมีการบันทึกการแข่งขัน ระหว่าง Unit ซึ่งจะเป็นส่วนหนึ่งของ Activation ของแต่ละ Competitive Units

ภาพที่แบบทั่วไปของ Activation Function ที่ใช้ใน Competitive Units ดังสมการที่ 2.12

$$f_i^c(net_i(t)) = \begin{cases} 1 & \text{ถ้า } net_i(t) = \max\{net_j(t)\},\ 1 \le j \le n \\ 0 & \text{กรณีอื่นๆ} \end{cases} \qquad \text{.....(2.12)}$$

พิจารณาสมการที่ 2.12 จะบอกได้ว่าทั้งหมด n Units บน Competitive layer มีเพียง Competitive Unit I (เป็นตัวหนึ่งที่ Lagest Activation ดังนั้นจึงชนะการแข่งขัน) จะถูกกระตุ้นในขณะที่ Unit อื่น ๆ ทั้งหมดบน Layer จะยังคง Inactive

การทำงานเช่นนี้มีความหมายว่าสำหรับทุก ๆ Input Pattern จะถูกนำไปสู่ Layer ของ Competitive Units มีเพียง 1 Unit ที่เด่นกว่าตัวอื่นๆ และยังเป็นเพียงตัวเดียวที่จะสร้าง Active Output ตัวอย่างดังภาพที่ 2.9



ภาพที่ 2.9 แสดงการประมวลผลของ Pattern (a) Unit 1 ชนะการแข่งขัน และส่งค่า Pattern ที่อยู่ในหน่วยความจำไปยัง Output Layer
(b) เมื่อ Unit 1 ชนะการแข่งขันแล้วOutput จากโครงข่ายจะกลายเป็น Pattern ที่ถูกจัดเก็บไว้ภายในการเชื่อมต่อระหว่าง Unit ที่ I และ Output Layer

จากภาพที่ 2.9 อธิบายใน Diagram เราจะยอมให้ Layer ของ Competitive Units ทำการแพร่ไปยัง Layer ของ Linear Unit โดยที่ Output จะถูกสร้างโดย Unit เป็นการ Sum ของ Modulated ซึ่งเป็น Input ทั้งหมดที่จะได้รับ ถ้ามีเพียง 1 Competitive Units จะ Active สำหรับทุก ๆ Input Pattern ที่เข้าสู่ Layer สำหรับ Linear Units จะมีเพียงตัวเดียวที่ได้รับ Active Input Signal

- มีเพียง Input ตัวเดียวที่จะ Connect ไปยัง Unit ที่ชนะบน Layer ของ Competitive Units ส่วนตัวที่ไม่ได้รับ Activation จาก Competitive Units อื่นๆ จะทำการแปลงสถานะของ Input Activation กลับดังเดิม

2.6.2.5 Guassian Units

Guassian Units เป็นสัญญาณ Output โดยการกลับ Activation Function ไปยัง Input ซึ่งจะไม่เหมือนกับ Unit ประเภทอื่น ๆ ที่เราได้ศึกษามาก่อนหน้านี้ Guassian Units ไม่ได้จำกัด Output ดังสมการที่ 2.13

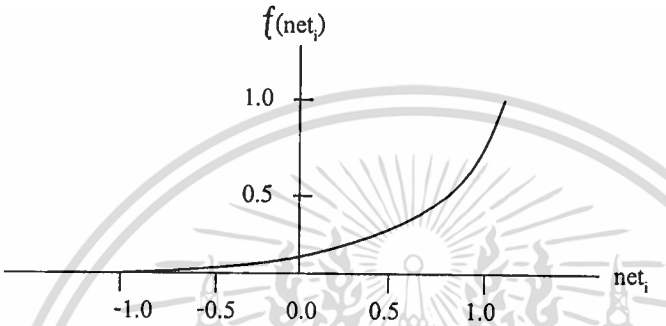
$$f_i^g(net_i) = e^{net\ I\text{-}1/\sigma * \sigma} \quad \text{.....(2.13)}$$

จากสมการที่ 2.13

net_i ใช้แทน net input ไปยัง Unit

σ เป็นค่าของ parameter ที่มีการเปลี่ยนแปลงค่าแบบช้า ๆ

ภาพที่แบบของ Function นี้แสดงในภาพที่ 2.10



ภาพที่ 2.10 จากกราฟแสดงในภาพที่แบบของ Guassian Activation Function เนื่องจากค่า Exponential Function ของ Guassian ค่า Output จะถูกจำกัดตามค่าของ Input

จากภาพที่ 2.10 พิจารณาจากค่าของ Net I อยู่ในช่วงระหว่าง –1 และ 1 สังเกตุได้จากกราฟของ Guassian Activation Function ที่มีช่วงแคบจากค่า Input ที่ยอมให้ Unit ทำการสร้าง Active Output ทุก ๆค่า Input จะมีค่าประมาณ 05 ถึง -∞ จะเปลี่ยนแปลงอย่างรวดเร็วในขณะที่ Input จะมีค่ามากกว่า 0.5 เป็นสาเหตุให้ Unit มีค่าเป็นไปตามที่ต้องการ (Active) อย่างรวดเร็ว

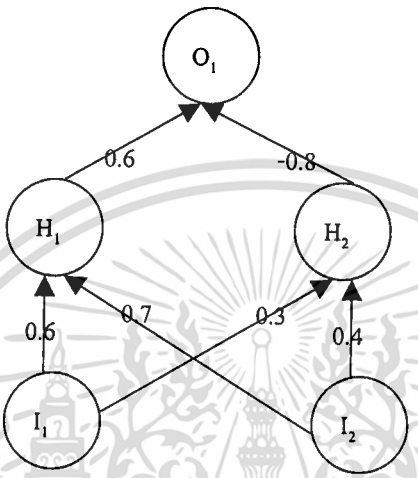
Guassian Units จะมีลักษณะเหมือนการกรอง โดยจะยอมให้ Input Pattern ที่สามารถสร้าง Active ในขอบเขตที่จำกัดจะสามารถผ่านไปได้ ในขณะเดียวกันจะปฏิเสธ Input ตัวอื่นทั้งหมด

พิจารณาพฤติกรรมของ Unit นี้ ใน Neural Network จะพบว่าในแต่ละ Unit บน Layer จะถูกกระตุ้นโดยจำนวนของ Connection ที่มาจาก Layer อื่น

Pattern จะพบอยู่ใน Input Connection ที่ไปยัง Unit โดยแสดงเป็น Tuner เมื่อ Layer ดำเนินการได้แล้วนั้น Guassian Units จะสร้าง Output Pattern ที่เข้ากันได้กับ Pattern ที่มีอยู่ใน Input Connection โดยเข้าสู่วิธีของ Guassian ก็คือในลักษณะของ Inner-Product Guassian จะตรวจหาตัวที่เข้ากันได้ จะสร้างสัญญาณที่แสดงว่า การเข้ากันได้เกิดขึ้นแล้วเพื่อที่จะได้ Output Pattern จาก Layer เมื่อใช้วิธีของ Guassian ถ้าพบว่ามีความแตกต่างจาก Pattern ที่มีอยู่ เมื่อพิจารณาจากค่าถ่วงน้ำหนักของ Input Guassian จะสร้างในสิ่งที่แสดงว่าไม่มี Output เพื่อที่จะบอกว่า Input ไม่สามารถเข้ากันได้

2.7 การคำนวณที่เกิดขึ้นในโครงข่าย (Network Computation)

โครงข่ายจะทำการคำนวณได้ขึ้นอยู่กับความสามารถของทุกๆ Single Processing Elements ซึ่งจะอธิบาย Network Computation ได้ดังภาพที่ 2.11



ภาพที่ 2.11 จากแผนภาพเป็นการแสดงว่า Neural Network สามารถทำการแปลงจากภาพที่ซับซ้อนได้

สมมติว่าก่อนที่ทุก ๆ Pattern จะได้รับการแสดงออกมา โครงข่ายจะยังไม่มีค่าใดอยู่ โดยที่แต่ละ Processing Elements ในโครงข่ายตามภาพที่ 2.9 จะทำงานตาม Algorithm 2 ขั้นตอนดังนี้

1. เมื่อปรากฏ Input Signal จะพบว่า Unit ที่ I ทำการคำนวณ Activation ของตัวเอง $a_i(t)$ โดยใช้สมการที่ 2.1 และสมการที่ 2.5 ตามที่ได้กล่าวมาแล้วนั้น จะได้สมการใหม่คือสมการที่ 2.15

$$a_i(t) = net_i(t) = \sum_{j=1}^{n} w_{ij}(t)\, o_i(t) \qquad \text{.....(2.15)}$$

2. เมื่อรู้ค่าของ $a_i(t)$ แล้ว Unit จะสร้าง Output ตาม Activation Function ดังสมการที่ 2.16

$$f_i(a_i(t)) = \begin{cases} 1 & \text{ถ้า } a_i(t) \geq 0.5 \\ 0 & \text{กรณีอื่น ๆ} \end{cases} \qquad \text{.....(2.16)}$$

ถ้าเราพิจารณาภาพที่แบบของโครงข่าย โดยขึ้นอยู่กับค่าถ่วงน้ำหนักของ Connection เราจะพบสิ่งที่น่าสนใจหลาย ๆ อย่าง เกี่ยวกับพฤติกรรมของโครงข่าย ตัวอย่างจากภาพที่ 2.9

- ถ้าสัญญาณอินพุท สมมติให้เป็น Binary (0 หรือ 1 เท่านั้น) Unit ที่อยู่บน Input Layer แสดงลักษณะเหมือนการกรองเสียง สำหรับสัญญาณอินพุท จะมีค่า Threshold เท่ากับ 0.5 เป็นค่าที่ยอมรับได้ สัญญาณที่มีค่าต่ำกว่า Threshold จะไม่ได้รับการยอมรับ
- Middle หรือ Hidden Layer ในส่วน Layer ของ Unit จะแสดงเป็นตัวตรวจจับการ Activate ซึ่ง Unit H1 จะทำการตรวจจับเงื่อนไขที่ Input Unit ตัวใดตัวหนึ่ง Active ในขณะที่ H2 จะตอบสนองเมื่อ Input Unit ทั้งคู่ Active พร้อมกัน
- Output Layer ในกรณีนี้ประกอบด้วยเพียง 1 Unit เป็นการกระทำการเชื่อมต่อแบบ Logical ระหว่าง ตัวที่ทำหน้าที่ตรวจจับ โดย Hidden –layer Units

จากที่กล่าวมาแล้ว สามารถแสดง Output ของโครงข่ายในตัวอย่างเป็น Boolean Function ของ Input Signal ไปยัง โครงข่ายดังสมการที่ 2.17

$$o(t) = (a(t-2)+b(t-2))(a(t-2)b(t-2))$$
$$= a(t-2)b(t-2) \quad \text{.....(2.17)}$$

โดยที่ สัญลักษณ์ o(t) จะถูกใช้เพื่อแสดงถึง Output Signal จาก Unit o ที่เวลา t

a(t-2) และ b(t-2) ใช้เพื่อแสดง Input Signals ไปยังโครงข่าย

สมมติว่าให้ Input และ Output signals จากโครงข่ายนี้เป็น Binary Patterns จะเห็นได้ว่าโครงข่ายจะทำงานในภาพที่แบบของ Exclusive-or (XOR) บน Input ที่แสดงถึงความสามารถในการใช้ Neural Network ให้กระทำ Function ที่ซับซ้อนในลักษณะ แบบขนาน

ในโครงข่ายแบบ XOR ดังภาพที่ 2.9 ที่ผ่านมา โดยที่ Hidden Layer Unit จะแสดงเป็นตัวที่ใช้ในการตรวจจับ ถ้า Processing Elements แต่ละตัวในโครงข่าย เป็นแบบ Discrete Process จะพบว่า 2 Unit ที่อยู่บน Hidden Layer จะกระทำบนข้อมูลเดียวกันในช่วงเวลาเดียวกัน ถัดมาจำนวนเวลาในความต้องการที่จะตรวจจับ 2 features ใน Input Pattern มีค่าเท่ากับ เวลาทั้งหมดที่ทำให้ 1 Unit ทำการคำนวณ Activation ของตัวเอง และทำการสร้าง Output ออกมา

ถ้ามีการพิจารณาจากตัวอย่างโครงข่ายอื่น ๆ ที่ได้ถูกออกแบบมาให้ทำหน้าที่ที่คล้ายคลึงกัน แต่สิ่งสำคัญที่มีความซับซ้อนมากกว่า XOR Function จะพบว่า Input Pattern Vector ที่อยู่ใน Application ใหม่ที่ประกอบด้วย 10,000 Input Elements ในการประมวลผลผ่านโครงข่ายแบบ XOR ใช้วิธีการพื้นฐานเช่นเดียวกัน เพียงแต่ต่างกันที่ความซับซ้อนของงานเท่านั้น

ถ้าแต่ละ Processing Elements ที่อยู่บนแต่ละ Layer ทำงานในลักษณะ Paralelle กับ Unit อื่นๆ บน Layer เดียวกันเวลาที่ใช้ในการ Detect Features ก็จะลดลง

2.8 Network Simulation

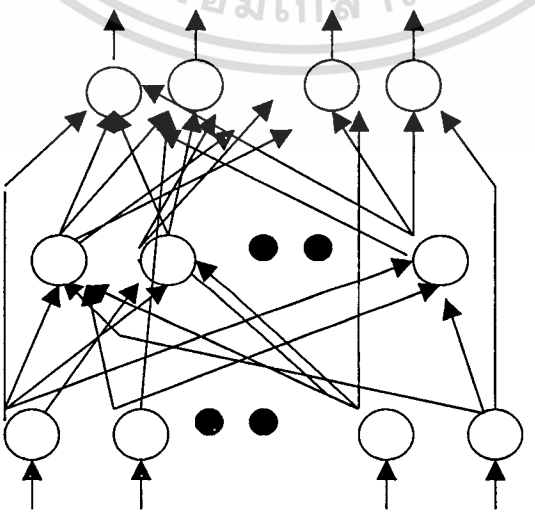
ในการสร้างโครงข่ายของ Analog Process ที่มีค่าถ่วงน้ำหนักในการติดต่อกับ Information จะพบเวลาทั้งหมดที่จำเป็นเพื่อให้ Pattern ที่เตรียมไว้ครบถ้วนผ่านโครงข่ายควรจะใช้เวลาน้อยคือใช้เวลาเท่าที่จำเป็น

ขั้นตอนแรกของการออกแบบ Application เมื่อโครงข่ายได้เรียนรู้พฤติกรรมในการตัดสินใจเกือบทั้งหมดในสิ่งที่จำเป็นจะต้องได้รับการแก้ไข (หรือออกแบบใหม่ให้ครบถ้วน) โครงข่ายจะทำอย่างไรให้ได้รับผลลัพธ์ที่ดีกว่าเดิม ดังนั้นในขั้นตอนแรกของการพัฒนา Application จะต้องมี Software Network Architecture ซึ่งเป็นสิ่งหนึ่งที่สามารถเปลี่ยนแปลงได้อย่างรวดเร็วทั้งขนาดของโครงข่ายและวิธีในการเรียนรู้

การตัดสินใจนี้เป็นความยืดหยุ่นที่สามารถจัดการได้โดย Network Simulators

- Program ที่ run (ทำงานบน Conventinal Computer) ทำให้ Computer เลียนแบบพฤติกรรมของสิ่งง่าย ๆ ก็คือ Analogs Processors ของโครงข่าย จะพบว่า Software โครงสร้างของโครงข่ายจะเป็น Model ของหน่วยความจำที่ยอมให้เปลี่ยนแปลงได้โดยง่าย ภาพที่แบบของโครงข่าย รวมทั้ง Algorithm ที่ใช้ในการทำงานของ Processing Elements ในโครงข่าย และพฤติกรรมของโครงข่ายเอง (เป็น Computer Program ง่าย ๆ ที่ทำงานซ้ำไปเรื่อย ๆ)

ต่อไปจะเป็นการอธิบายว่า Software Simulator ทำงานได้อย่างไร โดยพิจารณาจาก Neural Network Structure ที่แสดงในภาพที่ 2.12



ภาพที่ 2.12 แสดงโครงข่ายแบบ Multilayer เป็นโครงข่ายแบบแพร่ไปข้างหน้า

Algorithm ที่จำเป็น ในการ Simulate โครงสร้างโครงข่ายบน Conventinal Computer มีดังต่อไปนี้

Locate the input layer structure in memory

Copy the input pattern into output of the input layer

For each layer from the first non-input layer to the output

For each unit on the current layer

Set the accmulated input value for this unit to zero

For each input connection to this unit to zero

Compute the modulated input across the connection

Add the modulated input to the accumulated input

End loop

Convert the accumulated input to its corresponding output

Store the output value for the unit in the layer structure

End loop

End loop

Return the output values from the top-most layer structure

Software Simulator สำหรับ Neural Network ทำให้ผู้พัฒนา Application ทำการสร้าง Neural Network ที่แตกต่างได้อย่างรวดเร็ว และทดสอบพฤติกรรมของแต่ละโครงข่ายบนปัญหาที่ระบุ มีเพียงสิ่งสำคัญ 2 อย่าง ที่จำเป็นต้องอ้างถึง เมื่อพิจารณาบน Software สำหรับการ Simulate โครงข่ายเหล่านี้ มีดังนี้

- จำนวน Memory ที่ใช้ Connection แต่ละตัวในโครงข่าย โดยปกติจะ Model เป็นจำนวนจริง ใน Simulate Program ใช้ 10 Bytes ใน Memory คิดเป็น 80 Bits ของจำนวนจริง และมีค่าที่ต้องใช้เป็นจำนวนจริง 3 ตัว ในการ Connection ในโครงข่าย จำนวน Memory ที่จำเป็นในการสร้าง Neural Network Simulator จะมากขึ้นตามจำนวนของ Processing Elements ในโครงข่าย
- เวลาในการ Simulate Input ในแต่ละ Procressing Elements ในโครงข่าย โดยปกติแล้วจะคำนวณ โดยการใช้จำนวนจริงคูณและ จำนวนจริงที่บวกเพิ่มเข้าไปในแต่ละ Input Connection ที่ส่งไปยัง Unit ที่ต้องการในการ Simulate ถึงแม้จะเป็นโครงข่ายเล็กๆ หรือโครงข่ายที่มีมากกว่า 10,000 ก็สามารถทำงานได้อย่างรวดเร็ว

ในขณะที่มีข้อดีเหล่านี้แต่ก็มีปัญหาสำหรับการ Simulate Neural Network ขนาดใหญ่ ถ้าโครงข่ายของเรามีขนาดใหญ่เกินไป เราจะต้องใช้คอมพิวเตอร์ที่มีประสิทธิภาพสูงมาก ยิ่งกว่านั้นทางด้านการพัฒนา Application ปัญหาที่พบในการใช้ Software ก็คือปัญหาเกี่ยวกับ Host Computer ที่จะต้องมีความสามารถเพียงพอ และ Online Memory ในการ Simulate โครงข่าย

เมื่อ Application เสร็จสมบูรณ์แล้ว มาถึงจุดที่จะต้องทิ้ง จำเป็นต้องทำโครงข่ายใหม่ โครงข่ายสามารถนำมาใช้ในภาพที่ของ Software หรือทำการ Migrate โครงข่ายไปยัง Neural Network Device ก่อนหน้าได้

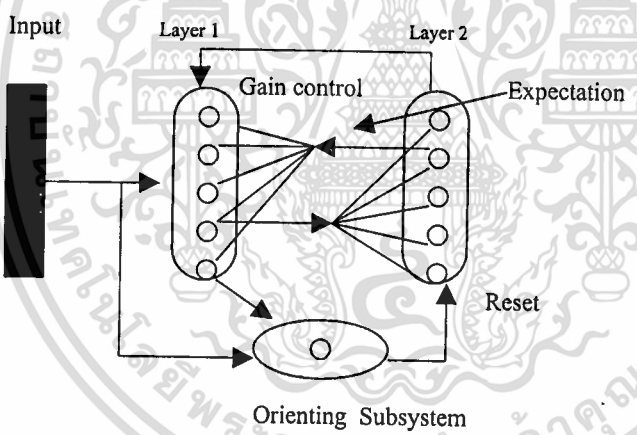
สำหรับในการศึกษาโครงข่ายประสาทเทียมนี้ จะทำการศึกษาโครงข่ายประสาทเทียมแบบ Adaptive Resonance Theory 1 ซึ่งจะกล่าวถึงรายละเอียดของโครงข่ายดังกล่าวในบทที่ 3

บทที่ 3

โครงข่ายประสาทเทียมแบบ Adaptive Resonance Theory (ART)

3.1 ทฤษฎีของ ART

โครงข่ายประสาทเทียมแบบ Adaptive Resonance Theory (ART) พัฒนาขึ้นโดย Carpenter และ Grossberg โดยที่ ART เป็นแบบจำลองที่มีโครงสร้างแบบมัลติเลเยอร์ (MultiLayer) ใช้รูปแบบในการเรียนรู้แบบไม่มีผู้ฝึกสอน สำหรับข้อดีของโครงข่าย ART คือใช้เวลาในการฝึกโครงข่ายน้อย และไม่มีปัญหาทางด้าน Memory Washout ก็คือสามารถที่จะเรียนรู้ Pattern ใหม่โดยที่ไม่ลืม Pattern เก่าที่เคยเรียนรู้ไปก่อนแล้ว สถาปัตยกรรมของ ART แสดงได้ดังภาพที่ 3.1



ภาพที่ 3.1 สถาปัตยกรรม ART

จากภาพที่ 3.1 สำหรับโครงข่ายแบบ ART การเรียนรู้จะเกิดขึ้นในกลุ่มของการแพร่ไปข้างหน้า และจาก Layer 2 ไปยัง Layer 1 การ ConNection เหล่านี้คือ Outstar เพื่อที่จะนำ Pattern กลับมาอีกครั้ง เมื่อโหนดที่อยู่ใน Layer 2 มีการ ActiVated จะมีการสร้าง Pattern ใหม่ (Expectation) ณ Layer ที่ 1 หลังจากนั้น Layer 1 จะทำการเปรียบเทียบระหว่าง Expectation และ Input Pattern

เมื่อ Expectation และ Input Pattern ไม่สามารถเข้ากันได้ Orienting Subsystem จะทำการตั้งค่าใหม่ใน Layer ที่ 2 การตั้งค่าใหม่นี้คือการ Disables สำหรับ Neuron ที่ชนะในปัจจุบัน และตัวที่คาดหวังไว้จะถูกเอาออกไป จะมีการแข่งขันใหม่เกิดขึ้นที่ Layer ที่ 2 ในขณะที่ Neuron ที่ชนะก่อนหน้านั้นจะถูก Disables ส่วน Neuron ที่ชนะใหม่ใน Layer ที่ 2 จะเลือกค่าที่คาดหวังตัวใหม่

ส่งไปยัง Layer 1 ผ่าน L2-L1 ConNection กระบวนการนี้จะดำเนินการไปจนกระทั่ง L2-L1 Expectition ใกล้เคียงกับ Input Pattern

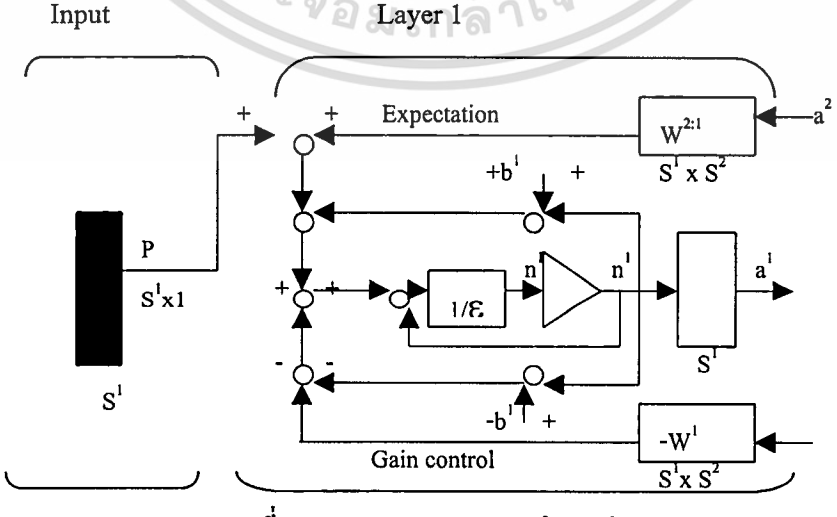
3.2 โครงสร้างของโครงข่าย ART

แบบจำลอง ART ได้พัฒนาเป็น 2 ประเภทคือ แบบจำลอง ART ซึ่งรับค่าอินพุทเป็นค่าไบนารี (ART1) และแบบจำลองที่รับค่าอินพุทในลักษณะ Continuous Value (ARTII) ในการศึกษานี้จะทำการศึกษาถึงโครงข่ายประสาทเทียมที่ใช้แบบจำลอง ART1 ซึ่ง ART1 มีโครงสร้างที่สำคัญดังนี้

1. Layer 1
2. Layer 2
3. Orienting Subsystem
4. Gain Control

3.2.1 Layer 1

วัตถุประสงค์ของ Layer 1 คือ การเปรียบเทียบอินพุทแพทเทิร์นกับ Expectation Pattern จากLayer 2 โดยที่ทั้ง 2 แพทเทิร์นเป็นไบนารี ถ้าแพทเทิร์นไม่ Match กัน Orienting Subsystem จะทำการตั้งค่าใหม่ในLayerที่ 2 ถ้าแพทเทิร์นมีความใกล้เคียงกันแล้วLayer 1 จะทำการรวมกับ ตัวที่คาดหวัง และ Input ให้อยู่ในรูปของ แพทเทิร์นต้นแบบตัวใหม่ Layer1 ของโครงข่าย ART1 ที่แสดงในภาพที่ 3.2 จะมีความคล้ายคลึงกันมากกับ Layer 1 ของ Grossberg Network



ภาพที่ 3.2 แสดง Layer 1 ของโครงข่าย ART1

สำหรับโครงข่าย ART 1 ไม่มีการนอร์มอลไลซ์ ที่จะกระทำที่ Layer 1 ดังนั้นเราจึงไม่ต้องทำการเชื่อมต่อแบบ On Center/Off Surround จาก Input Vector

Excitatory Input ที่จะไปยัง Layer 1 ของ ART1 ประกอบด้วยการรวมกันของ Input Pattern และ L1-L2 Expectation

Inhibitory Input ประกอบด้วยสัญญาณ Gain control จาก Layer 2 ตามที่กล่าวมาอธิบาย Input เหล่านี้ จะทำงานร่วมกันได้อย่างไร

สมการในการ Operation ของ Layer 1 คือสมการที่ 3.1

$$\varepsilon dn^1(t)/dt = -n^1(t)+(^+b^1-n^1(t))\{p+w^{2:1}a^2(t)\}-(n^1(t)+^-b^1[^-w^1]a^2(t) \quad \text{.....(3.1)}$$

และOutput ของ Layer 1 จะคำนวณจากสมการที่ 3.2 และสมการที่ 3.3

$$a^1 = \text{hardlim}^+(n^1) \quad \text{.....(3.2)}$$

โดยที่

$$\text{hardlim}^+(n) = \begin{cases} 1, & n>0 \\ 0, & n\leq 0 \end{cases} \quad \text{.....(3.3)}$$

สมการที่ 3.1 เป็น Shunting Model กับ Excitatory Input คือ $p+W^{2:1}a^2(t)$ เป็นการ Sum ของ Input Vector และจาก L2-L1 Expectation ตัวอย่างสมมติว่า Neural ที่ j ใน Layer 2 ชนะการแข่งขัน ดังนั้น Output ที่จะได้คือ 1 และ Neural อื่นๆ จะมี Output เป็น 0 กรณีนี้เราใช้สมการที่ 3.4

$$\begin{aligned} W^{2:1}a^2 &= [W_1^{2:1}\ W_2^{2:1} \dots\ W_j^{2:1} \dots\ W_{s^*s}^{2:1}]\ [0\ 0\ \dots 0 \dots 1..0]^T \\ &= W_j^{2:1} \quad \text{.....(3.4)} \end{aligned}$$

โดยที่ $W_j^{2:1}$ เป็นคอลัมน์ ที่ j ของเมตริกซ์ $W^{2:1}$ ($W^{2:1}$ เป็นเมตริกซ์ที่ได้รับการฝึกหัดโดยการใช้ กฎของ Outstar) เราจะพบว่า เป็นไปดังสมการที่ 3.5

$$p+W^{2:1}a^2 = p+W_j^{2:1} \quad \text{.....(3.5)}$$

ดังนั้น Excitatory Input ไปยัง Layer1 เป็นการ Sum ของ Input Pattern และ L2-L1 Expectation ในแต่ละคอลัมน์ของ เมตริกซ์ L2-L1 ใช้แสดง Expectation ที่แตกต่างกัน (แพทเทิร์นต้นแบบ)

Inhibitory Input ไปยัง Layer1 เป็น Gain Control Term ดังสมการที่ 3.6

$$\bar{W}^{1} = \begin{bmatrix} 1\ 1 \ldots 1 \\ 1\ 1 \ldots 1 \\ \ldots\ldots \end{bmatrix} \qquad \ldots\ldots(3.6)$$

ดังนั้น Inhibitory Input ที่ไปยังแต่ละ Neural ใน Layer 1 เป็นการ Sum ค่าทั้งหมดของ Output ของ Layer 2 ในขณะที่เรา Winner-Take-All การแข่งขันใน Layer 2 ถ้าค่าที่ Layer 2 เท่ากับ Active จะให้ค่าเท่ากับ 1 และมีสมาชิกเพียงตัวเดียวของ a^2 ที่ไม่ใช่ค่า 0 หลังจากการแข่งขัน

ดังนั้น Gain Control Input ไปยัง Layer1 จะเป็น 1 เมื่อ Layer 2 Active และจะเป็น 0 เมื่อ Layer 2 Inactive (Neural ทั้งหมดจะมี Output ที่มีค่าเป็น 0)

วัตถุประสงค์ของ Gain Control นี้จะกลายเป็นสิ่งที่ปรากฏที่เราจะนำมาวิเคราะห์พฤติกรรมของ Steady State ของ Layer 1

การวิเคราะห์ Steady State

การตอบสนองของ Neuron I ในLayer 1 อธิบายโดยใช้สมการที่ 3.7

$$\varepsilon dn_i^1(t)/dt = -n^1(t)+(\bar{b}^1-n_i^1(t)) \{p_i+ \sum_{j=1}^{s^2} w_{1,j}^{2:1} a_j^2\}- (n_i^1(t)+\bar{b}^1) \sum_{j=1}^{s^2} a_j^2 \qquad \ldots\ldots(3.7)$$

โดยที่ $\varepsilon << 1$ ดังนั้น ค่า Outputใน Short Term Memory จะมีการเปลี่ยนแปลงเร็วกว่า Long Term Memory (เมตริกซ์ของค่าถ่วงน้ำหนัก)

ต้องการให้ Steady State ตอบสนองกับระบบนี้สำหรับ 2 กรณี ที่แตกต่างกัน

ในกรณีแรก Layer 2 Inactive ดังนั้น $a_j^2=0$ สำหรับ j ทั้งหมด

ในกรณีที่ 2 Layer 2 Active ดังนั้นจะมี Neural 1 ตัวที่มี Output เป็น 1 และ Neural อื่นๆทั้งหมด Output เป็น 0

พิจารณากรณีแรกที่ Layer 2 Inactive เมื่อ a_j^2 แต่ละตัวมีค่าเท่ากับ 0 จากสมการที่ 3.7 ทำให้ง่ายขึ้น เป็นสมการที่(3.8)

$$\varepsilon dn_i^1(t)/dt = -n_i^1(t)+({}^+b^1-n_i^1(t))\{p_I\} \quad \text{.....(3.8)}$$

ใน Steady State เมื่อ $dn_i^1(t)/dt = 0$ จะได้สมการที่ 3.9

$$0 = -n_i^1(t)+({}^+b^1-n_i^1(t))p_I = -(1+p_I)n_i^1+{}^+b^1 p_I \quad \text{.....(3.9)}$$

ถ้าเราทำการพิสูจน์ใน Steady State Neural Output n_i^1 จะได้ดัง สมการที่ 3.10

$$n_i^1 = {}^+b^1 p_I/1+p_I \quad \text{.....(3.10)}$$

ดังนั้นถ้า $p_I = 0$ เมื่อ $n_i^1 = 0$ และถ้า $p_I = 1$ เมื่อ $n_i^1 = {}^+b^1/2>0$ จนกระทั่งเราเลือก Transfer Function สำหรับ Layer 1 ให้เป็น hardlim$^+$ Function เราจะได้สมการที่ 3.11

$$a^1 = p \quad \text{.....(3.11)}$$

ดังนั้นเมื่อLayer 2 Inactive Output ของ Layer 1 ก็เป็นเช่นเดียวกับ Input Pattern พิจารณากรณีที่ 2 ที่ซึ่ง Layer 2 Active สมมติว่า Neural j เป็น Neural ที่ชนะใน Layer 2 เมื่อ $a_j^2 = 1$ และ $a_k^2 = 0$ สำหรับ k ไม่เท่ากับ j สำหรับกรณีนี้สมการที่ 3.7 ทำในภาพที่ง่ายขึ้นได้เป็นสมการที่ 3.12

$$\varepsilon dn_i^1(t)/dt = -n_i^1+({}^+b^1-n_i^1)\{p_i+w_{I,j}^{2:1}\}-(n_i^1+{}^-b^1) \quad \text{.....(3.12)}$$

ใน Steady State เมื่อ $dn_i^1/dt = 0$ จะได้สมการที่ 3.13

$$\begin{aligned} 0 &= -n_i^1+({}^+b^1-n_i^1)\{p_i+w_{I,j}^{2:1}\}-(n_i^1+{}^-b^1) \\ &= -(1+p_i+w_{I,j}^{2:1}+1)n_i^1+({}^+b^1(p_i+w_{I,j}^{2:1})-{}^-b^1) \end{aligned} \quad \text{.....(3.13)}$$

ถ้าเราทำการแก้ปัญหาสำหรับ Steady State Neural Output n_i^1 จะพบว่าเป็นไปดังสมการที่3.14

$$n_i^1 = ({}^+b^1(p_i+w_{I,j}^{2:1})-{}^-b^1)/2+p_i+w_{I,j}^{2:1} \quad \text{.....(3.14)}$$

ย้อนกลับไปยัง Layer 1 ควรจะทำการ Combine Input Vector กับ Expectation จาก Layer 2 (แสดงโดยใช้ $w_j^{2:1}$) เมื่อเราอธิบายโดยใช้ Binary Pattern (ทั้ง Input และExpectation) เราจะใช้ Logical AND Operation ในการ Combine 2 Vector เราต้องการ n_i^1 ให้มีค่าน้อยกว่า 0 เมื่อ p_I หรือ $w_{i,j}^{2:1}$ มีค่ามากกว่า 0 (ตัวใดตัวหนึ่ง) ถ้าเราต้องการให้ n_i^1 มีค่ามากกว่า 0 เมื่อทั้ง p_I และ $w_{i,j}^{2:1}$ ต่างมีค่าเท่ากับ 1

ถ้าเราทำการประยุกต์เงื่อนไขเหล่านี้ตามสมการที่ 3.14 จะได้สมการใหม่ดังนี้คือสมการที่ 3.15 และสมการที่ 3.16

$$^{+}b^1 (2)- \bar{b}^1 > 0 \quad \text{.....(3.15)}$$

$$^{+}b^1 - \bar{b}^1 < 0 \quad \text{.....(3.16)}$$

ซึ่งเราสามารถ Combine เพื่อที่จะสร้างสมการที่ 3.17

$$^{+}b^1 (2)- \bar{b}^1 > {}^{+}b^1 \quad \text{.....(3.17)}$$

ตัวอย่างเราใช้ $^{+}b^1$ เท่ากับ 1 และ $\bar{b}^1$เท่ากับ 1.5 เพื่อใช้ในการปฏิบัติตามเงื่อนไขเหล่านี้ ดังนั้น ถ้าสมการ 3.17 จะถูกนำมาใช้ปฏิบัติ และ Neural j ของ Layer2 Active เมื่อ Output ของLayer1 เป็นไปดังสมการที่ 3.18

$$a^1 = p \cap w_j^{2:1} \quad \text{.....(3.18)}$$

โดยที่ $\cap$ ใช้แทน AND Operation

ให้สังเกตว่า เราจำเป็นที่จะใช้ Gain Control ทำการ Implement โดยใช้ AND

พิจารณาจำนวนเชิงตัวเลขตามสมการที่ 3.14 จะได้สมการใหม่คือสมการที่ 3.19

$$^{+}b^1 (p_i + w_{I,j}^{2:1}) - \bar{b}^1 \quad \text{.....(3.19)}$$

เทอม $\bar{b}^1$ เป็นการคูณโดยเทอม Gain Control ในกรณีนี้คือ ถ้าเทอมนี้ไม่คงอยู่ สมการ 3.19 ควรจะมีค่ามากกว่า 0 (และดังนั้น n_j^1 ควรจะมีค่ามากกว่า 0) เมื่อใดก็ตามที่ p_I หรือ $w_{ij}^{2:1}$ ตัวใดตัวหนึ่งมีค่ามากกว่า 0 สิ่งนี้แสดงโดย OR Operation มากกว่า AND Operation เป็นสิ่งที่เราพบเมื่อเราอธิบายใน Orienting subsystem มันจะเป็นเกณฑ์ให้ Layer 1 กระทำการโดยใช้ AND Operation

เมื่อ Layer 2 Inactive จะพบว่า Gain Control Term จะมีค่าเป็น 0 สิ่งนี้เป็นสิ่งที่จำเป็นเพราะว่าเป็นกรณีที่เราต้องการให้Layer 1 ตอบสนองกับ Input Pattern เพียงตัวเดียว จนกระทั่งไม่มีการ Expectation จะ Activated โดยLayer 2

สรุป Steady State Operation ของ Layer1

ถ้า Layer 2 ไม่ Active (เช่น a_j^2 แต่ละตัวมีค่า = 0)

$$a^1 = p \quad \text{.....(3.20)}$$

ถ้า Layer 2 Active (เช่นมี a_j^2 ตัวหนึ่งมีค่า = 1)

$$a^1 = p \cap w_j^{2:1} \quad \text{.....(3.21)}$$

การ Operation ของLayer 1 สมมติว่า parameter สำหรับโครงข่ายนี้คือ

$$\varepsilon = 0.1, {}^{+}b^1 = 1 \text{ และ } {}^{-}b^1 = 1.5 \quad \text{.....(3.22)}$$

สมมติว่าเรามี 2 Neural ในLayer 2 มี2 สมาชิกใน Input Vector และมีเมตริกซ์ค่า ถ่วงน้ำหนัก และInput ดังสมการที่ 3.23

$$w^{2:1} = [1\ 1, 0\ 1] \quad \text{และ} \quad p = [0\ 1]^T \quad \text{.....(3.23)}$$

ถ้าเราทำในกรณีที่ Layerที่ 2 Active และ Neural ทั้ง 2 ของ Layer 2ชนะการแข่งขัน สมการในการ Operation ของ Layer 1 คือ

$$\begin{aligned}(0.1)dn_i^1/dt &= -n_i^1 + (1-n_i^1)\{p_i + w_{i,j}^{2:1}\} - (n_i^1 + 1.5) \\ &= -n_i^1 + (1-n_i^1)\{0+1\} - (n_i^1 + 1.5) \\ &= -3\,n_i^1 - 0.5 \quad \text{.....(3.24)}\end{aligned}$$

$$\begin{aligned}(0.1)dn_2^1/dt &= -n_2^1 + (1-n_2^1)\{p_2 + w_{2,2}^{2:1}\} - (n_2^1 + 1.5) \\ &= -n_2^1 + (1-n_2^1)\{1+1\} - (n_2^1 + 1.5) = -4\,n_2^1 - 0.5 \quad \text{.....(3.25)}\end{aligned}$$

$$dn_1^1/dt = -30\,n_1^1 - 5 \quad \text{.....(3.26)}$$

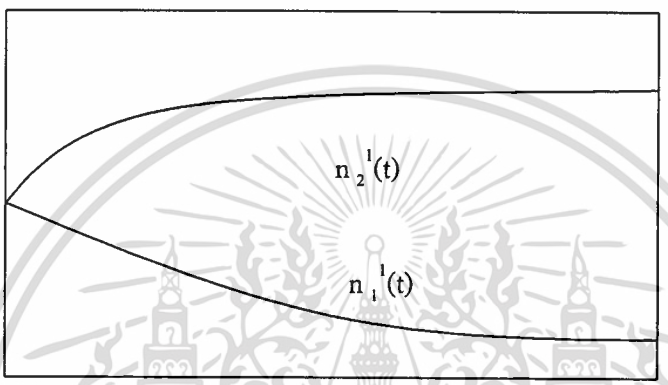
$$dn_2^1/dt = -40\,n_2^1 + 5 \quad \text{.....(3.27)}$$

วิธีที่ง่ายกว่าโดยการสมมติว่า Neural ทั้งคู่เริ่มต้นด้วยค่าเริ่มต้น = 0 แก้ปัญหาได้ดังสมการที่ 3.28 และสมการที่ 3.29

$$n_1^1(t) = -1/6[1-e^{-30t}] \quad \text{(3.28)}$$

$$n_2^1(t) = 1/8[1-e^{-40t}] \quad \text{(3.29)}$$

ซึ่งสิ่งเหล่านี้จะแสดงในภาพที่ 3.3



ภาพที่ 3.3 แสดงการตอบสนองของ Layer 1

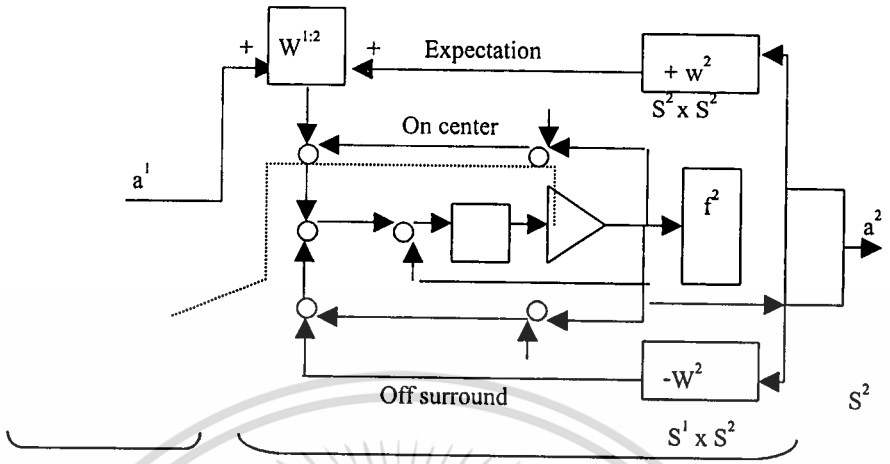
สังเกตว่า $n_1^1(t)$ จะมีค่าเป็นลบ และ $n_2^1(t)$ จะมีค่าเป็นบวก ดังนั้น $a_1^1(t)$ จะมีค่าเป็น 0 และ $a_2^1(t)$ จะมีค่าเป็น 1 ดังสมการที่ (3.30)

$$p \cap w_2^{2:1} = (0,1) \cap (1,1) = (0,1) = a^1 \quad \text{(3.30)}$$

3.2.2 Layer 2

วัตถุประสงค์หลักของ Layer 2 คือ ความสามารถที่ชัดแจ้งในรูปแบบ Output สำหรับการ Implement ในโครงข่ายของ ART1 ความสามารถที่เห็นได้ชัดจาก Winner-Take-All Competition ดังนั้นจึงมีเพียง Neural เดียวที่จะเป็นผู้ชนะการแข่งขัน และค่า Output ที่ได้มีค่าไม่เท่ากับ 0

หลักสำคัญที่แตกต่างระหว่าง Layer ที่ 2ของ Grossberg และ ART1 คือ Layerที่ 2 ของ ART1 สามารถที่จะทำการ Reset ค่าได้ เมื่อใดก็ตามที่ สัญญาณ a^0 มีค่าเป็นบวก Output ที่ Reset ก็ยังคงเป็น Inhibited ต่อไป



ภาพที่ 3.4 แสดง Layer 2 ของโครงข่าย ART1

จากภาพที่ 3.4 แสดงถึง Layerที่ 2 ของ ART1 สัญญาณการ Reset a^0 เป็น Output ของ Oreinting Subsystem จะเกิดการ Reset ขึ้นเมื่อพบว่ามีการ Mismatch เกิดขึ้นที่ Layer 1 ระหว่าง Input Signal และ L1-L2 Expectation

Output จากLayerที่ 2 เป็นการคำนวณจาก $a^2 = \text{hardlim}^+(n^2)$ โดยที่ Output ที่ได้ จาก Layer 2 ควรจะเป็น Binary

สมการการ Operation ของ Layer 2 คือ

$$\varepsilon dn^2(t)/dt = -n^2 + (^+b^2 - n^2)\{[^+w^2] f^2(n^2(t)) + w^{1:2}a^1\} - (n^2(t) + ^-b^2)\{[^-w^2] f^2(n^2(t))\} \quad \text{.....(3.31)}$$

จากสมการที่ 3.31 จะประกอบด้วยส่วนของ Shunting Model และ Excitatory Input $\{[^+w^2] f^2(n^2(t)) + w^{1:2}a^1\}$ โดย $^+w^2$ เป็นการกำหนด On-Center Feedback Connection และ $w^{1:2}$ ประกอบด้วยค่าถ่วงน้ำหนักที่ปรับค่าไปเรื่อยๆ จะพบว่าหลังจากที่ฝึกหัดแล้วแถวของ $w^{1:2}$ จะแสดง Pattern ต้นแบบ

Inhibitory Input ที่เข้าสู่ Shunting Model คือ $[^-w^2] f^2(n^2(t))$ โดยค่า $^-w^2$ ได้กำหนด Off-Surround Feedback Connection

เพื่อที่จะอธิบายประสิทธิภาพของ Layer 2 พิจารณาจาก 2 Neuron ดังสมการที่ 3.32

$$\varepsilon = 0.1,\ ^+b^2 = (1, 1),\ ^-b^2 = (1, 1),\ w^{1:2} = [(_1w^{1:2})^t (_2w^{1:2})^t] = [0.5\ 0.5\ 1\ 0] \quad \text{.....(3.32)}$$

และ $f^2(n) = \begin{cases} 10(n)^2, & n \geq 0 \\ 0, & n < 0 \end{cases}$ .....(3.33)

สมการการ Operation ของ Layer

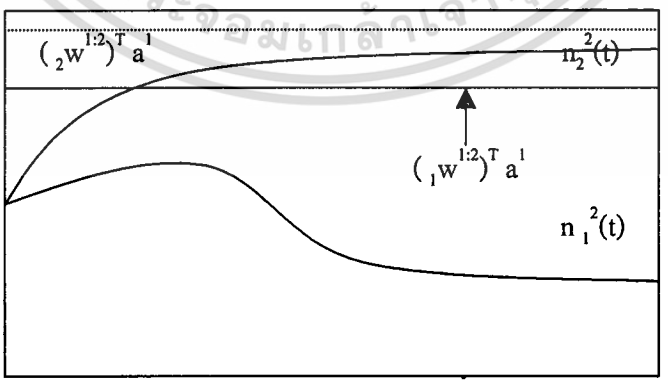
$$(0.1)dn_1^2/dt = -n_1^2 + (1-n_1^2(t)) \{ f^2(n_1^2(t)) + ({}_1w^{1:2})^t a^1 \}$$

$$= -(n_1^2(t)+1) \; f^2(n_1^2(t)) \quad \text{.....(3.34)}$$

$$(0.1)dn_2^2/dt = -n_2^2 + (1-n_2^2(t)) \{ f^2(n_2^2(t)) + ({}_2w^{1:2})^t a^1 \}$$

$$= -(n_2^2(t)+1) \; f^2(n_2^2(t)) \quad \text{.....(3.35)}$$

จากสมการข้างต้นค่าของ $b^2=1$ ซึ่งจะทำให้ค่าของ $n_1^2(t)$ และค่าของ $n_2^2(t)$ จะอยู่ระหว่าง –1 และ +1 Input ของ Layer2 เป็น Inner Product ของ Pattern ต้นแบบที่มีค่าใกล้เคียงกับ Output ของ Layer 1 โดยที่ Layer2 จะทำการแข่งขันระหว่าง Neurons และ Transfer Function $f^2(n)$ จะถูกเลือกให้เป็นตัวที่เร็วกว่า สิ่งนี้เองจะทำให้ Neurons มีค่า Input ที่สูงที่สุดคือ ค่า n ที่มีค่าเป็นบวก และ Neural อื่นๆ ที่มีค่า n เป็นลบ (โดยใช้ค่า parameter ที่เหมาะสม) หลังจากการแข่งขันมี 1 Neuron output ที่จะมีค่าเป็น 1 ส่วน Neural Output ตัวอื่น ๆ จะมีค่าเป็น 0



ภาพที่ 3.5 เป็นการอธิบายการตอบสนองของ Layer 2

จากภาพที่ 3.5 เป็นการอธิบายการตอบสนองของ Layer 2 เมื่อ Input Vector คือ $a^1 = [1\ 0]^t$ แถวที่ 2 ของ $w^{1:2}$ มีการคูณภายในจำนวนมากกับ a^1 ดังนั้น Neuron 2 ชนะการแข่งขันที่ Steady state, $n_2^2(t)$ มีค่าเป็นบวก และ $n_1^2(t)$ มีค่าเป็นลบ

Steady State ของ Layer 2 ดังสมการที่ 3.36

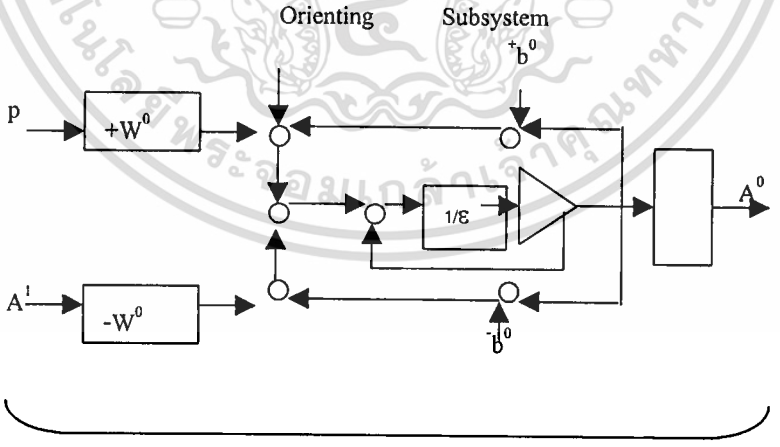
$$a^2 = \text{Vector}(0,1) \quad \text{.....(3.36)}$$

สรุป Steady State Operation ของ Layer 2 ได้ดังสมการที่ 3.37

$$a_i^2 = \begin{cases} 1 & \text{ถ้า } ({}_iw^{1:2})^T a^1 = \text{Max}[({}_jw^{1:2})^T a^1] \\ 0 & \text{กรณีอื่นๆ} \end{cases} \quad \text{.....(3.37)}$$

3.2.3 Orienting Subsystem

วัตถุประสงค์เป็นการพิจารณา ถ้ามีการ Match กันได้ค่าตามสมควรระหว่าง L2-L1 Connection และ Input Pattern เมื่อมีการ Match กันน้อยเกินไปในส่วนนี้ก็จะทำการส่งสัญญาณ Reset ไปยัง Layer 2 ลักษณะของ Orienting Subsystem แสดงดังภาพที่ 3.6



ภาพที่ 3.6 แสดง Orienting Subsystem ของโครงข่าย ART 1

สมการการ Operation ของ Orienting Subsystemคือ

$$\varepsilon dn^0(t)/dt = -n^0 + (^+b^0 - n^0)\{^+w^0 p\} - (n^0(t) + {}^-b^0)\{^-w^0 a^1\} \quad \text{.....(3.38)}$$

นี่คือ Shunting Model การ Excitatory Input ซึ่ง

$$^{+}w^{0} = [\alpha\ \alpha\ \alpha \ldots \alpha] \quad (3.39)$$

ดังนั้น Excitatory Input เขียนได้

$$^{+}w^{0}p = [\alpha\ \alpha\ \alpha \ldots \alpha]p = \alpha \sum_{j=1}^{s^{1}} p_j = \alpha \|p\|^{2} \quad (3.40)$$

จากสมการ 3.40 p เป็น Binary Vector

Inhibitory Input ที่ให้ยัง Orienting Subsystem คือ $^{-}w^{0}a^{1}$ ที่ซึ่ง

$$^{-}w^{0} = [\beta\beta \ldots \beta] \quad (3.41)$$

ดังนั้น Inhibitory Input เขียนได้

$$^{-}w^{0}a^{1} = [\beta\beta \ldots \beta]a^{1} = \beta \sum_{j=1}^{s^{1}} a_j^{1}(t) = \beta \| a^{1} \|^{2} \quad (3.42)$$

เมื่อใดก็ตามที่ Excitatory Input มีขนาดใหญ่กว่า Inhibitory Input Orienting Subsystem จะทำงานต่อเมื่อพิจารณาตาม Steady State Operation

$$0 = -n^{0} + (^{+}b^{0} - n^{0})\{\alpha \|p\|^{2}\} - (n^{0} + {}^{-}b^{0})\{\beta \| a^{1} \|^{2}\} \quad (3.43)$$

ถ้าเราพิสูจน์สำหรับ n^{0} เราจะพบดังสมการ 3.44

$$n^{0} = {}^{+}b^{0}(\alpha \|p\|^{2}) - {}^{-}b^{0})(\beta \| a^{1} \|^{2}) / (1 + \alpha \|p\|^{2} + \beta \| a^{1} \|^{2}) \quad (3.44)$$

ให้ $^{+}b^{0} = {}^{-}b^{0} = 1$ ดังนั้น $n^{0} > 0$ ถ้า $\alpha \|p\|^{2} - \beta \| a^{1} \|^{2} > 0$ หรือ

$$n^{0} > 0 \text{ ถ้า } \| a^{1} \|^{2} / \|p\|^{2} < \alpha/\beta = \rho \quad (3.45)$$

สิ่งนี้เองที่เป็นเงื่อนไขของสาเหตุในการ Reset ใน Layer 2 จนกระทั่ง $a^{0} = \text{hardlim}^{+}(n^{0})$ ρ จะถูกเรียกว่า Vigilance Parameter และค่าจะตกลงในช่วงของ $0 < \rho < 1$

ถ้า Vigilance Parameter มีค่าเข้าใกล้ 1 การ Reset จะเกิดขึ้นจนกระทั่ง a^{1} มีค่าเข้าใกล้ ρ

ถ้า Vigilance Parameter มีค่าเข้าใกล้ 0 a^{1} ไม่จำเป็นต้องมีค่าเข้าใกล้ ρ เพื่อป้องกันการ Reset โดยที่ค่า Vigilance Parameter จะถูกสร้างขึ้นโดย Vectors ต้นแบบ

กลับไปสมการที่ 3.21 ที่ $a^1 = p \cap w_j^{2:1}$

เมื่อใดก็ตามที่ Layer 2 Active ค่าของ $\| p \|^2$ จะมีค่ามากกว่าหรือเท่ากับ $\| a^1 \|^2$

อธิบายการ Operation ของ Orienting Subsystem โดยให้

p = Vector(1,1) และ a^1 =Vector(1,0)..... (3.46)

สมการในการ Operation จะได้ดังสมการที่ 3.47

$(0.1)dn^0(t)/dt = -n^0+(1-n^0(t)) \{ 3(p_1 + p_2)\}$..... .....(3.47)

หรือ $dn^0(t)/dt = -110n^0(t)+20$.....(3.48)

การตอบสนองจะดูได้จากการ Plott ลงไปในภาพที่ 3.7 ในกรณีนี้การ Reset จะส่งกลับไปยัง Layer 2 จนกระทั่ง $n^0(t)$ มีค่าเป็นบวก

ภาพที่ 3.7 แสดงการตอบสนองของ Orienting Subsystem

Steady State ของ Orienting Subsystem สามารถสรุปได้ดังนี้

$$a^0 = \begin{cases} 1 & \text{ถ้า } [\| a^1 \|^2 / \|p\|^2 < \rho] \\ \mathbf{0} & \text{กรณีอื่น ๆ} \end{cases} \quad \text{.....(3.49)}$$

3.3 กฎการเรียนรู้ (Learning law : L1-L2)

กฎการเรียนรู้สำหรับ $w^{1:2}$ คือ

$$d[_{i}w^{1:2}(t)]/dt = a_i^2(t)[\{^{+}b-{}_{i}w^{1:2}(t)\}\zeta[^{+}w]/\,a^1(t) - \{{}_{i}w^{1:2}(t)+{}^{+}b\}[^{-}w]/\,a^1(t)] \quad \text{.....(3.50)}$$

เมื่อ Neural I ของ Layer 2 เกิดการ Active แถวที่ I ของ $w^{1:2}$ และ $w^{2:1}$ จะถูกย้ายให้มาอยู่ที่ a^1 จากสมการที่ 3.50 ค่า Excitatory bias คือ $b^+ = 1$ และ Inhibitory bias คือ $b^- = 0$ การศึกษาจาก Steady State Operation วิเคราะห์โดยการสมมติว่า Output ของ Layer 1 และ Layer 2 ยังคงเป็นค่าคงที่จนกระทั่งค่าถ่วงน้ำหนักเข้าสู่ Steady State เรียกว่า Fast Learning สมการสำหรับสมาชิก $w_{I,j}^{1:2}$ คือสมการที่ 3.51

$$dw_{I,j}^{1:2}(t)]/dt = a_i^2(t)[(1- w_{I,j}^{1:2}(t))\zeta\, a_j^1(t)- w_{I,j}^{1:2}(t)\sum_{k\neq j} a_k^1(t)] \quad \text{.....(3.51)}$$

ถ้าสมมติว่า Neuron I มีค่า Active ใน Layer 2 ($a_i^2(t) = 1$) และกำหนดค่าให้เท่ากับ 0 ในสมการที่ 3.51 จะได้สมการใหม่คือสมการที่ 3.52

$$0 = [(1- w_{I,j}^{1:2}(t))\zeta\, a_j^1(t)- w_{I,j}^{1:2}(t)\sum_{k\neq j} a_k^1(t)] \quad \text{.....(3.52)}$$

ในการหาค่าของ Steady State $w_{I,j}^{1:2}$ เราจะพิจารณาใน 2 กรณี อันดับแรกสมมติว่า $a_j^1 = 1$ จะได้ว่า

$$0 = (1- w_{I,j}^{1:2}(t)\zeta - w_{I,j}^{1:2}(\|a^1\|^2-1) = -(\zeta+\|a^1\|^2-1)w_{I,j}^{1:2}+\zeta \quad \text{.....(3.53)}$$

หรือ $$w_{I,j}^{1:2} = \zeta/\zeta+\|a^1\|^2-1 \quad \text{.....(3.54)}$$

สังเกตว่าเมื่อค่าของ a^1 เป็น Binary Vector ถ้า $a_j^1 = 0$ สมการที่ 3.52 จะลดลง จนได้สมการที่ 3.55 และสมการที่ 3.56

$$0 = - w_{I,j}^{1:2} = \|a^1\|^2 \quad \text{.....(3.55)}$$

หรือ $$w_{I,j}^{1:2} = 0 \quad \text{.....(3.56)}$$

เมื่อทำการรวมสมการที่ 3.54 และสมการที่ 3.3 จะได้สมการที่ 3.57

$$_{I}w^{1:2} = \zeta\, a^{1}/\zeta + \|a^{1}\|^{2} - 1 \quad \text{.....(3.57)}$$

โดยที่ค่า $\zeta > 1$ เพื่อให้แน่ใจว่าตัวหารจะไม่ใช่ค่า 0
ดังนั้น Pattern ต้นแบบจะถูก Normalized เพื่อช่วยแก้ปัญหาบางอย่างที่เกิดขึ้น

3.4 กฎการเรียนรู้ L2-L1

L2-L1 Connection คือ $w^{2:1}$ ได้รับการฝึกหัดโดยใช้กฎของ Outstar Learning
วัตถุประสงค์ของ L2-L1 Connection เป็นการเอา Pattern ต้นแบบที่เหมาะสมกลับมา ดังนั้นจึงเป็นการรวมกันและทำการเปรียบเทียบ ใน Layer 1 สำหรับการใช้ Input Pattern เมื่อพบว่า Expectation และ Input Pattern ไม่ Match กัน การ Reset ก็จะถูกส่งไปยัง Layer 2 ดังนั้น Pattern ต้นแบบตัวใหม่จะถูกเลือกขึ้นมา
กฎการเรียนรู้สำหรับ $w^{2:1}$ เป็นดังสมการที่ 3.58

$$dw_{j}^{2:1}(t)]/dt = a_{j}^{2}(t)[(- w_{j}^{2:1}(t) + a^{1}(t)] \quad \text{.....(3.58)}$$

ดังนั้นถ้า Neural j ใน Layer 2 Active (ชนะการแข่งขัน) คอลัมน์ที่ j ของ $w^{2:1}$ จะเข้าสู่ a^{1} Pattern $w^{1:2}$ และ $w^{2:1}$ จะถูกปรับค่าในเวลาเดียวกัน เมื่อ Neural ของ Layer 2 มีค่าเป็น Active และ Match กันระหว่าง Expectative และ Input Pattern แถวที่ j ของ $w^{1:2}$ และคอลัมน์ที่ j ของ $w^{2:1}$ จะมีการปรับเปลี่ยนค่า โดยค่าของคอลัมน์ที่ j ของ $w^{2:1}$ จะเป็นค่าที่มาจาก a^{1} ส่วนค่าแถวที่ j ของ $w^{1:2}$ เป็นตัวที่ถูก Normalized ของ a^{1}

3.5 อัลกอริทึมของ ART1

อัลกอริทึมของ ART1 เริ่มต้นโดยการกำหนดเมตริกซ์ค่าถ่วงน้ำหนัก $w^{1:2}$ และ $w^{2:1}$ โดยเมตริกซ์ $w^{2:1}$ จะถูกกำหนดค่าเริ่มต้นให้เป็น 1 ทั้งหมด Resonance จะเกิดขึ้นเมื่อ $a^{1} = p \cap w_{j}^{2:1} = p$ ดังนั้น $\|a^{1}\|^{2} / \|p^{2}\| = 1 > \rho$ ความหมายก็คือว่า ทุกๆ คอลัมน์ที่ยังไม่ได้ฝึกหัดใน $w^{2:1}$ จะถูกแปลงให้เป็นช่องว่าง และจะเป็นตัวที่ใช้ในการจับคู่กับทุก ๆ Input Pattern แถวของเมตริกซ์ $w^{1:2}$ ควรจะถูก Normalized ก่อนโดยทุก ๆ ค่าของ เมตริกซ์ $w^{1:2}$ จะถูก set ให้มีค่า $= \zeta/(\zeta + S^{1} - 1)$

อัลกอริทึมมีขั้นตอนดังนี้

1. ใส่ Input Pattern เข้าไปในโครงข่ายเป็นลักษณะของ Binary โดยที่ Output ของ Layer 1 คือ $\mathbf{a}^1 = \mathbf{p}$

2. ทำการคำนวณ Input ที่จะเข้าสู่ Layer 2

$$\mathbf{w}^{1:2}\,\mathbf{a}^1$$

Neuron ที่อยู่ใน Layer 2 จะเลือกตัวที่ Active จาก

$$a_j^{\,2} = \begin{cases} 1 & \text{ถ้า } ({}_i\mathbf{w}^{1:2})^t\,\mathbf{a}^1 = \mathrm{Max}[({}_k\mathbf{w}^{1:2})^t\,\mathbf{a}^1] \\ 0 & \text{กรณีอื่น ๆ} \end{cases}$$

ถ้าพบว่าค่าที่ได้ออกมาเท่ากันหมดตัวที่มี Index น้อยที่สุดจะเป็นผู้ชนะ

3. ทำการคำนวณ L2-L1 Expectation

$$\mathbf{w}^{2:1}\,\mathbf{a}^2 = \mathbf{w}_j^{\,2:1}$$

4. เมื่อ Layer 2 Active ทำการปรับ Layer 1 output

$$\mathbf{a}^1 = \mathbf{p} \cap \mathbf{w}_j^{\,2:1}$$

5. ต่อมาในส่วนของ Orienting Subsystem หาจำนวนที่ Match กันระหว่าง Expectation และ Input Pattern

$$a^0 = \begin{cases} 1 & \text{ถ้า } [\|\mathbf{a}^1\|^2/\|\mathbf{p}\|^2 < \rho] \\ 0 & \text{กรณีอื่นๆ} \end{cases}$$

6. ถ้า $a^0 = 1$ เราจะ set ค่า $a_j^{\,2} = 0$ ย้อนกลับไปขั้นตอนที่ 1 ถ้า $a^0 = 0$ ทำต่อในขั้นตอนที่ 7

7. Resonance เกิดขึ้นแล้ว ดังนั้นเราจึงทำการ Update แถวที่ j ของ $\mathbf{w}^{1:2}$

$${}_j\mathbf{w}^{1:2} = \zeta\,\mathbf{a}^1/\zeta + \|\mathbf{a}^1\|^2 - 1$$

8. ทำการ Update Column ที่ j ของ $\mathbf{w}^{2:1}$

$${}_j\mathbf{w}^{2:1} = \mathbf{a}^1$$

9. Remove Input Pattern กลับไปทำ Step 1 ใหม่พร้อมกับ Input Pattern ตัวใหม่ตามที่เราต้องการ

ต่อไปในบทที่ 4 จะกล่าวถึงการนำเอาอัลกอริทึมข้างต้นไปประยุกต์ใช้งานในการรู้จำตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษ

บทที่ 4
การประยุกต์ใช้ ART 1 เพื่อจดจำตัวพิมพ์อักษรภาษาไทย และตัวพิมพ์อักษรภาษาอังกฤษ

ในการนำ Art 1 มาประยุกต์ใช้งานเพื่อจดจำตัวพิมพ์อักษรภาษาไทยและตัวพิมพ์อักษรภาษาอังกฤษนั้น ข้อมูลที่เข้าสู่โครงข่ายเป็นข้อมูลภาพของตัวอักษรที่อยู่ในรูปแบบของอินพุทเมตริก (Input Matrix) สำหรับภาษาไทยนั้นจะใช้ตัวพิมพ์อักษรภาษาไทยประกอบด้วยพยัญชนะ ก ถึง ฮ จำนวน 46 ตัวรวมทั้งตัว ฤ ฦ โดยใช้ Font AngsanaUPC ขนาด 20 พอยท์ เพื่อนำเข้าใช้ในการจดจำตัวอักษร และทดสอบการแยกแยะตัวอักษร โดยใช้ Font CordiaUPC ขนาด 20 พอยท์ โดยที่อักษรจะอยู่ในรูปของเมตริก 2 มิติขนาด 15 X 10 โดย 0 แทนพื้นตัวอักษร และ 1 แทนเนื้อตัวอักษร และตัวพิมพ์อักษรภาษาอังกฤษประกอบด้วยพยัญชนะ A ถึง Z จำนวน 26 ตัว โดยใช้ Font AngsanaUPC ขนาด 20 พอยท์ เพื่อนำเข้าใช้ในการจดจำตัวอักษร และทดสอบการแยกแยะตัวอักษร โดยใช้ Font FreesiaUPC ขนาด 20 พอยท์ อักษรจะอยู่ในรูปของเมตริก 2 มิติขนาด 15 X 10 โดย 0 แทนพื้นตัวอักษร และ 1 แทนเนื้อตัวอักษรเช่นเดียวกับตัวพิมพ์อักษรภาษาไทย

กำหนดค่า Parameter ของโครงข่ายประสาทเทียมก่อนเข้าสู่ขั้นตอนในการฝึกสอนโครงข่าย

ตัวพิมพ์อักษรภาษาไทย

T1 =150 จำนวน Input ของ 0 และ 1 ของตัวอักษร

T2 = 46 จำนวน Node ที่ Input ใช้เพื่อการจดจำต่อไป

K = 4

$\rho = 0.75$

ตัวพิมพ์อักษรภาษาอังกฤษ

S1 =150 จำนวน Input ของ 0 และ 1 ของตัวอักษร

S2 = 26 จำนวน Node ที่ Input ใช้เพื่อการจดจำต่อไป

L = 3

$\rho = 0.6$

กำหนด Weight1_2 ($W^{1:2}$) เป็นเมตริกซ์ขนาด 26 x 150 และ 46 x 150 จะเป็นค่า weight เพื่อใช้ในการคำนวณหา winner node ต่อไป กำหนดค่าจากสมการ

สำหรับตัวอักษรภาษาอังกฤษ = L/(L-1+150)

สำหรับตัวอักษรภาษาไทย = K/(K-1+150)

ซึ่งจะได้ค่าของเมตริกซ์ $W^{1:2}$ ดังนี้

สำหรับภาษาอังกฤษ = [0.02 0.02 0.02 ... 0.02
0.02 0.02 0.02 ... 0.02
0.02 0.02 0.02 ... 0.02] $_{(26x150)}$

สำหรับภาษาไทย = [0.026 0.026 0.026 ... 0.026
0.026 0.026 0.026 ... 0.026
0.026 0.026 0.026 ... 0.026] $_{(26x150)}$

กำหนดเมตริกซ์ Weight2_1 ($W^{2:1}$) ขนาด 150 x 26 และ 150 x 46 แทนค่าที่ได้จากการฝึกสอนซึ่งในตอนแรกจะกำหนดให้มีค่า = 1

4.1 การเรียนรู้เพื่อจดจำตัวอักษร

ขั้นตอนการทำงานตามอัลกอริทึมของ ART 1 มีดังต่อไปนี้

1. ทำการคำนวณเพื่อหา Output จาก Layer ที่ 1 อินพุทที่เข้าสู่โครงข่ายคือ ตัวพิมพ์อักษรภาษาไทย และตัวพิมพ์อักษรภาษาอังกฤษ โดยเข้าสู่โครงข่ายครั้งละ 1 ตัวอักษร จำนวน 150 บิต

a^1 = P1 =

[000001000000001110000000111000000011100000010010000001001000000100100000010
010000010000100001111110000100001000100000010010000001001000000100100000010]^T

ในที่นี้คือตัวอักษร A ซึ่งตัวพิมพ์อักษรภาษาไทยก็มีลักษณะเช่นเดียวกัน

2. คำนวณ Output ของ Layer 1 เพื่อเป็นอินพุตของ Layer 2

$W^{1:2} =$ [0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02
…………………
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02]

นำมา dot กับ a^1 จะได้ผลลัพธ์เป็นเวกเตอร์ขนาด 26 Dimension

$W^{1:2} a^1$ = [0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.02 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 0.72 $0.72]^T$

จะพบว่าค่าที่ได้มีค่าเท่ากันทั้งหมด ซึ่งในรอบนี้คือรอบแรกส่วนในรอบต่อไปจะพบว่าค่าที่ได้ออกมาจะมีค่าที่ค่อนข้างแตกต่างกัน ซึ่งจากผลในที่นี้พบว่าค่าที่ได้ออกมามีค่าเท่ากันหมดเราจะเลือกตัวที่มีค่า Index น้อยที่สุด เป็นโหนดที่ชนะการแข่งขัน ถ้าเป็นกรณีที่มีค่าที่แตกต่างกันจะเลือกเอาตัวที่มีค่าสูงสุดเป็นผู้ชนะเพราะฉะนั้นจะได้ผลลัพธ์ของ 26 ตัวอักษรภาษาอังกฤษดังนี้

ในรอบแรก a^2 = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $0]^T$
ในรอบที่สอง a^2 = [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $0]^T$
ในรอบที่สาม a^2 = [0 01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $0]^T$
ในรอบที่สี่ a^2 = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $0]^T$
ในรอบที่ห้า a^2 = [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $0]^T$
ในรอบที่หก a^2 = [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $0]^T$
ในรอบที่เจ็ด a^2 = [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $0]^T$

ในรอบที่แปด $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่เก้า $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบ $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบเอ็ด $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบสอง $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบสาม $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบสี่ $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบห้า $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบหก $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบเจ็ด $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบแปด $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่สิบเก้า $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่ยี่สิบ $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่ยี่สิบเอ็ด $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]^T$

ในรอบที่ยี่สิบสอง $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$

ในรอบที่ยี่สิบสาม $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$

ในรอบที่ยี่สิบสี่ $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^T$

ในรอบที่ยี่สิบห้า $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]^T$

ในรอบที่ยี่สิบหก $a^2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^T$

3. คำนวณ L2-L1 Expectation จากการ dot ของเมตริกซ์ $W^{2:1}$ กับค่าของเวกเตอร์ a^2 ที่ได้มาจากขั้นตอนที่ 2 ผลลัพธ์ที่ได้เป็นค่าของ $W_{winner}{}^{2:1}$ เป็นเวกเตอร์ขนาด 150 Dimension

$W^{2:1}$ เป็นเมตริกขนาด 150 X 26 และ a^2 เป็นเวกเตอร์ขนาด 26 Dimension

$W_{winner}{}^{2:1} = [\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \ldots 1]^T$

4. ทำการปรับค่า Output จาก Layer 1 กับค่าของ L2-L1 Expectation โดยที่ a^1 เป็นเวกเตอร์ขนาด 150 Dimension

$a^1 = P1 \cap W_{winner}{}^{2:1} = P1$

ผลลัพธ์ได้ดังนี้

[0000010000000001110000000011100000000111000000010010000000100100000001001000000010

0100000100001000011111100001000010001000000010010000000100100000001001000000010]^T

5. จากที่เราให้ส่วนของ Orienting Subsystem เป็นส่วนในการเปรียบเทียบค่าของการ Match กันระหว่างค่าที่คาดหวังและ Input Pattern โดยนำมาเทียบกับค่าของ Viligance Parameter จะได้ผลลัพธ์ดังนี้

$||a^1||^2 / ||p||^2 = 40/40 = 1$ ซึ่งมีค่ามากกว่า $\rho = 0.6$ ดังนั้นค่าของ $a^0 = 0$

6. เมื่อ $a^0 = 0$ ไปทำขั้นตอนที่ 7 ถ้า $a^0 = 1$ ไปทำขั้นตอนที่ 1 ใหม่

7. เมื่อเกิด Resonance ($a^0 = 0$) จะทำการ Update $W^{1:2}$ แถวที่ตรงกับค่าของโหนดที่ชนะ โดยทำการนอร์มอลไลซ์ a^1 จากสมการข้างล่าง

$${}_{winner}W^{1:2} = L\,a^1/L+||a^1||^2-1$$

$W^{1:2}$ = [0 0 0 0.079 0 0 0 0 0 0 0 0 0.079 0.079 0.079 0.079 0 …0

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 … 0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02

…

…

…

0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 …0.02] $_{(26x150)}$

8. ทำการ Update คอลัมน์ของ $W^{2:1}$ ขนาด 150 X 26 ที่ตรงกับค่าของโหนดที่ชนะ

$$W_{winner}^{2:1} = a^1;$$

$W^{2:1}$ = [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

...

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]$_{(150x26)}$

9. เคลียร์ค่าของ P1 (Input ตัวเก่า) หลังจากนั้นย้อนกลับไปยังขั้นตอนที่ 1 เพื่อรับ Input ตัวใหม่เข้ามา ในที่นี้ต่อไปคืออักษรตัว B

ซึ่งในตัวพิมพ์ภาษาอังกฤษรับ Input เข้ามาทั้งหมด 26 ตัวอักษร ส่วนในตัวพิมพ์ภาษาไทยจะทำการรับ Input ทั้งหมด 46 ตัวอักษร ส่วนกระบวนการทำงานจะทำเช่นเดียวกับตัวพิมพ์ภาษาอังกฤษมีเพียงค่า Parameter บางตัวเท่านั้นที่ต้องทำการเปลี่ยนแปลงเนื่องจากป้องกันการจำผิดโหนดของตัวอักษร ผลที่ได้จากกระบวนการขั้นต้นจะได้แพทเทิร์นที่จดจำตัวอักษรตัวพิมพ์ภาษาอังกฤษและตัวพิมพ์อักษรภาษาไทย เพื่อนำไปใช้ในกระบวนการต่อไป หลังจากที่ทำการทดสอบแล้วว่าโครงข่ายของเราสามารถจดจำแพทเทิร์นเดิมได้ โดยไม่มีการผิดโหนด เราก็จะทำกระบวนการแยกแยะตัวอักษรต่อไป

4.2 การแยกแยะตัวอักษร

ในการแยกแยะตัวอักษรนี้ เป็นการทดสอบว่าโครงข่ายที่เราสร้างขึ้นมานี้สามารถรู้จำตัวอักษรได้ดีเพียงไร ขั้นตอนในการแยกแยะสามารถทำได้โดยนำ Pattern ตัวใหม่สำหรับตัวพิมพ์อักษรภาษาอังกฤษใช้ Font CordiaUPC ขนาด 20 พอยท์ เป็น Input ที่เข้ามาทำการทดสอบ
กำหนดค่าที่ใช้ในส่วนของ Orienting Subsystem ดังนี้

กำหนดค่า Viligance Parameter ของภาษาอังกฤษ = 0.55

กำหนดค่า Viligance Parameter ของภาษาไทย = 0.50

ขั้นตอนการทำงานในการแยกแยะตัวอักษรมีดังนี้

1. ทำการเทียบค่าความเหมือนกันระหว่าง Input Pattern ตัวใหม่กับ Pattern ที่โครงข่ายได้จดจำเอาไว้โดยการนำมา AND กัน แสดงดังภาพที่ 4.1

0000010000	0000010000	0000010000
0000011000	0000111000	0000011000
0000011000	0000111000	0000011000
0001111000	0000111000	0000111000
0001111000	0001001000	0001001000
0001001100	0001000100	0001001000
0001001100	0001000100	0001001000
0011001100	0011000100	0001001000
0010000100	0010000100	0010000100
0011111100	0011111110	0011111100
0010001100	0010000010	0010000100
0100000010	0100000010	0100000010
0100000010	0100000010	0100000010
0100000010	0100000010	0100000010
1110000111	0100000010	0100000010
(a)	(b)	(c)

ภาพที่ 4.1 แสดงถึงผลของการแยกแยะตัวอักษรระหว่าง (a) ตัวอักษรที่เป็นต้นแบบกับ (b) ตัวอักษรที่นำเข้ามาแยกแยะ ซึ่งจะได้ผลคือ (c) เป็นตัวอักษรที่เกิดจากการ Intersection ของ (a) และ (b)

2. นำผลที่ได้จากขั้นตอนที่ 1 มาทำการหาผลรวมของค่าที่เป็น 1 จากผลที่ได้จากขั้นตอนที่ 1

3. นำค่าที่ได้จากขั้นตอนที่ 2 มาเปรียบเทียบค่าความคล้ายกันตัวใดที่มีค่าสูงสุดจะถือว่าเป็นโหนดที่ชนะการแข่งขันหลังจากนั้นทำการ Set ค่าของโหนดที่ชนะให้มีค่าเท่ากับ 1 ส่วนโหนดอื่น ๆ จะทำการ Set ค่าให้เป็น 0

4. ทดสอบว่าเกิด Resonance หรือไม่โดยนำค่าที่ได้จากข้อ 2 ของโหนดที่เป็นผู้ชนะ นำมาหารด้วยค่าที่เป็น 1 ทั้งหมดใน Pattern ต้นแบบเทียบกับค่า Viligance Parameter ค่า 0.55 สำหรับตัวอักษรภาษาอังกฤษ ค่า 0.50 สำหรับตัวอักษรภาษาไทย ถ้ามีค่ามากกว่าจะ Set ค่า $a^0 = 0$

5. ค่า $a^0 = 0$ ทำงานต่อในขั้นตอนที่ 6

ถ้าค่า $a^0 = 1$ กลับไปทำงานในขั้นตอนที่ 1

6. ทำการปรับค่าคอลัมน์ของ $W^{2:1}$ ที่ตรงกับโหนดที่ชนะด้วยค่าที่มาจากขั้นตอนที่ 1

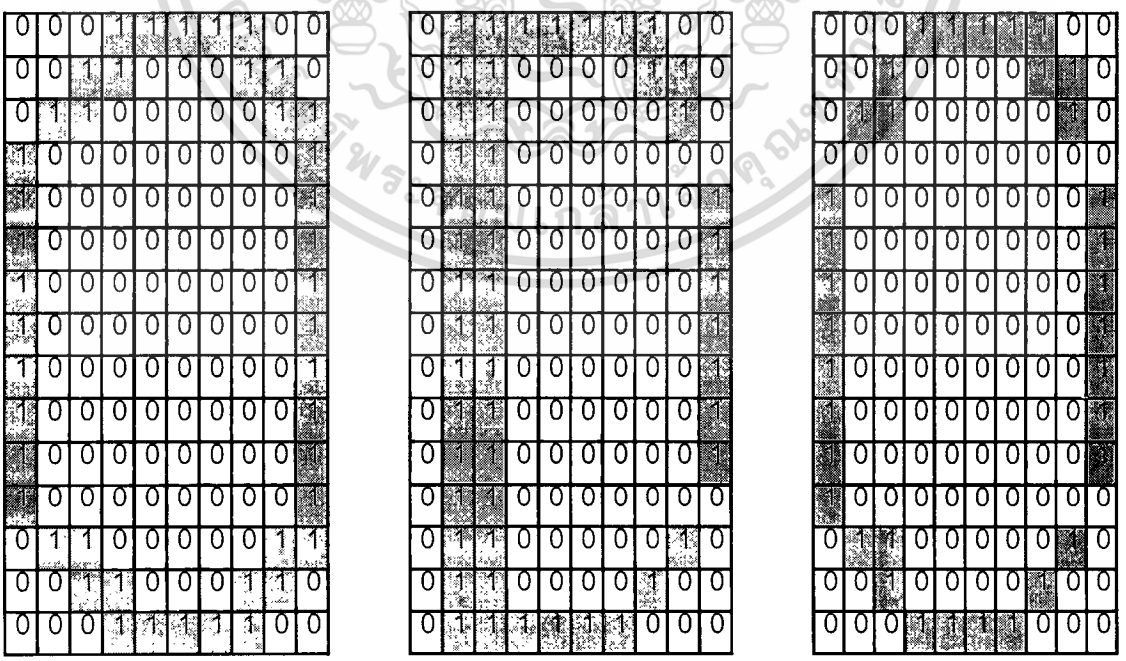
ต่อไปในบทที่ 5 จะเป็นการสรุปถึงผลจากการประยุกต์ใช้ Adaptive Resonance Theory 1 กับการจดจำตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษ

บทที่ 5

บทสรุปและข้อเสนอแนะ

5.1 บทสรุปในการพัฒนาโปรแกรม

การพัฒนาโปรแกรมนี้เป็นการพัฒนาโปรแกรมการจดจำตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษโดยใช้โครงข่ายประสาทเทียม ซึ่งในการศึกษานี้ใช้แบบจำลอง Adaptive Resonance Theory 1 เป็นตัวแบบในการศึกษา ขั้นตอนในการพัฒนาโปรแกรมนี้ประกอบด้วย ขั้นตอนของการเรียนรู้ของโครงข่ายเพื่อจดจำตัวอักษรที่นำเข้ามาฝึกสอน รวมทั้งขั้นตอนในการแยกแยะตัวอักษรใน Pattern ที่ต่างกัน การพัฒนาโปรแกรมได้ทำการพัฒนาตามอัลกอริทึมของ Adaptive Resonance Theory 1 จากการทดสอบในขั้นตอนของการเรียนรู้ตัวอักษรพบว่าสามารถจดจำตัวอักษรได้ครบโดยไม่ผิดโหนด แต่สำหรับการแยกแยะตัวอักษรนั้นพบว่ามีเป็นส่วนน้อยที่พบว่าเป็นการจำผิดโหนด สำหรับตัวพิมพ์อักษรภาษาอังกฤษพบว่ามีการเรียนผิดโหนด 1 ตัวคือ อักษรตัว O เดิมคือโหนดที่ 15 ตกผิดโหนดไปอยู่โหนดที่ 4 แสดงดังภาพ 5.1

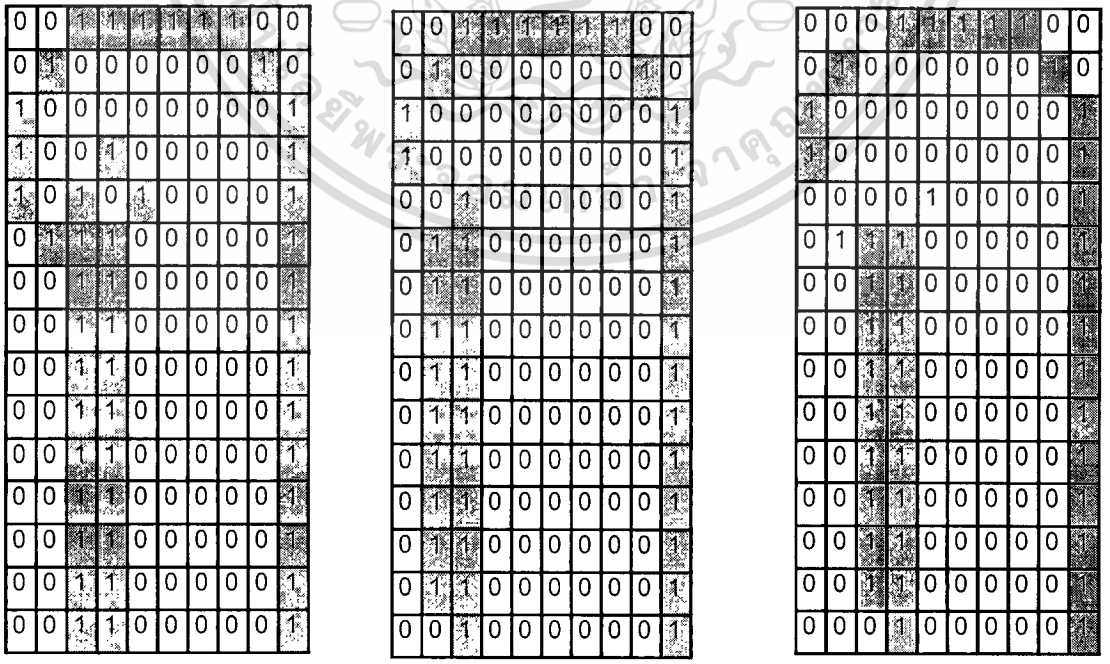


(a) Pattern ที่ใช้ทดสอบ (b) Pattern ต้นแบบ (c) ผลจากการตกผิดโหนด

ภาพที่ 5.1 แสดงถึงผลจากการแยกแยะตัวพิมพ์อักษรภาษาอังกฤษ

สำหรับตัวพิมพ์อักษรภาษาไทยพบว่ามีการเรียนผิดโหนด 2 ตัวคือ อักษรตัว ก เดิมคือโหนดที่ 1 ตกผิดโหนดไปอยู่โหนดที่ 38 อักษรตัว ค เดิมคือโหนดที่ 4 ตกผิดโหนดไปอยู่โหนดที่ 1 แสดงดังภาพ 5.2

(a) Pattern ที่ใช้ทดสอบ (b) Pattern ต้นแบบ (c) ผลจากการตกผิดโหนด



ภาพที่ 5.2 แสดงถึงผลจากการแยกแยะตัวพิมพ์อักษรภาษาไทย

5.2 ข้อเสนอแนะ

ในการพัฒนาโปรแกรมการจดจำตัวพิมพ์อักษรภาษาไทยและภาษาอังกฤษนั้น เพื่อต้องการศึกษาถึงการนำเอาโครงข่ายประสาทเทียมมาใช้ในงานด้านต่าง ๆ ซึ่งในที่นี้คือการจดจำตัวอักษร จากการศึกษาโครงข่ายประสาทเทียมนี้มีข้อจำกัดอยู่บ้างในเรื่องของประสิทธิภาพของเครื่องในการทำงาน ถ้าเรามีจำนวนโหนดเป็นจำนวนมากจะพบปัญหามากขึ้นทั้งในเรื่องของความเร็วและเนื้อที่ที่ใช้งาน ในการพัฒนาโปรแกรมนี้ได้ศึกษาตัวอักษรเพียงขนาดเดียว โครงข่ายควรสามารถยืดหยุ่นได้มากกว่านี้

บรรณานุกรม

Clidchanok Lursinsap, 1992. *Artificial Neural Network* : Semiconductor Electronics, October.

David M. Skapura, 1996. *Building Neural Networks*. Masschusetts: Addition-Wesley.

James A. Freeman and David,1991. *Neural Networks Algorithm Application and Programming Techniques*. Masschusetts: Addition-Wesley .

Martin T. Hagan and Howard B. Demuth, 1996. *Neural Network Design*: PWS Publishing Company.

ประวัติผู้เขียน

ชื่อ-นามสกุล	นางสาวจรรยา สุภาพ
วันเดือนปีเกิด	5 กุมภาพันธ์ 2517
สถานที่เกิด	จังหวัดนครราชสีมา
ประวัติการศึกษา	วิทยาศาสตรบัณฑิต (วทบ.) สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยรามคำแหง
ประวัติการทำงาน	เข้ารับราชการเมื่อปี พ.ศ. 2541 ถึงปัจจุบัน ดำรงตำแหน่ง นักวิชาการคอมพิวเตอร์ 4 สำนักงานสรรพากรภาค 5 ชลบุรี