โปรแกรมช่วยออกแบบและศึกษาการทำงานของโครงข่ายประสาทเทียม

PROGRAM AIDED DESIGN AND LEARNING NEURAL NETWORKS

นายสมิทธิ์ เอมสมบัติ

MR. SMITDH EMSOMBAT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2539

ISBN 974-621-584-1

PROGRAM AIDED DESIGN AND LEARNING NEURAL NETWORKS

SMITDH EMSOMBAT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF GRAUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

1996

ISBN 974-621-584-1

หัวข้อวิทยานิพนธ์	โปรแกรมช่วยออกแบบและศึกษาการทำงานของโครงข่ายประสาทเทียม
นักศึกษา	นายศมิทธิ์ เอมสมบัติ
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.ชม กิ้มปาน
อาจารย์ผู้ควบคุมวิทยานิพนธ์ร่วม	ดร.บุญธีร์ เครือตราชู
ระดับการศึกษา	วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า
ภาควิชา	คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.	2539

บทคัดย่อ

โปรแกรมช่วยออกแบบและศึกษาการทำงานของโครงข่ายประสาทเทียม (Neural Networks) เป็นโปรแกรม ที่ช่วยในการวิเคราะห์และศึกษา โครงสร้างของระบบโครงข่ายประสาทเทียม โดยผู้ใช้ไม่จำเป็นต้องเขียนโปรแกรมขึ้นมาเอง และมีการติดต่อกับผู้ใช้ที่ง่ายต่อการใช้งาน การเรียนรู้ของโครงข่ายเป็นการเรียนรู้เช่นเดียวกับสมองของมนุษย์ คือต้องมีการฝึกฝน (Training) สำหรับ ข้อมูลแต่ละชุด โดยอาศัยแต่ละอัลกอริทึมที่มีความสามารถเฉพาะตัวที่แตกต่างกันไป ซึ่งผู้ใช้สามารถเลือกใช้อัลกอริทึมที่ สำคัญๆเช่น Perceptrons , Back Propagation , Kohonen Self-Organizing Maps... ในการฝึกฝนโครงข่าย อีกทั้งสามารถเลือกจำนวนชั้นของโครงข่าย และ จำนวนนิวรอลในแต่ละชั้น เพื่อศึกษาหาขนาดที่เหมาะสมสำหรับแต่ละชุดของข้อมูล สามารถเปลี่ยนแปลงค่าผิดพลาดต่ำ สุดที่ใช้ในการเรียนรู้ ปริมาณอัตราการเรียนรู้ (Learning rate) ที่ใช้ในการปรับน้ำหนัก สามารถกำหนดช่วงในการสุ่มค่าน้ำหนัก (Random Weight) ในตอนเริ่มต้นได้ โปรแกรมยังสามารถให้ทำงานทีละขั้นได้ เช่น feed forward , backward propagation , weight adaptation พร้อมทั้งแสดงผลการคำนวณออกมาในรูปของตารางการทำงาน เพื่อให้ผู้ใช้ศึกษาขั้น ตอนการทำงานโดยละเอียด

Thesis Title	PROGRAM AIDED DESIGN AND LEARNING NEURAL NETWORKS
Student	Mr.Smitdh Emsombat
Thesis Advisor	Assoc.Prof.Dr.Chom Kimpan
Thesis CO.Advisor	Dr.Boontee Kerutrachoo
Level of Study	Master of Engineering in Electrical Engineering
Department	Computer, Faculty of Engineering King Mongkut's Institute of TechnologyLadkrabang
Year	1996

Abstract

PROGRAM AIDED DESIGN AND LEARNING NEURAL NETWORKS is a program that helps analysis and study of Neural Networks structure, which simulate human's neural cell. The learning process of simulated neural is base on human's neural cell. Each cell must be trained for each pair of input-output. There are many models of neural network simulation. Each model has special characteristic. The tool allow user to select the neural network model such as Perceptron , Back-Propagation , Kohonen Self-Organizing Maps... with various number of layers and nodes in each layer. So that appropriate size of network can be obtained. User can also change many neural characteristics such as minimum error level and learning rate. This Program can simulate each training phase such as feed forward , back propagation , weight adaptation and display output at each phase.

กิตติกรรมประกาศ

ขอขอบพระคุณ บิดา มารดา ที่สนับสนุนทางด้านการศึกษา และให้กำลังใจมาโดยตลอด

ขอขอบพระคุณ รองศาสตราจารย์ ดร. ชมกิ้มปาน ที่ได้ให้ความอนุเคราะห์คำปรึกษา

ขอขอบพระคุณ ดร. บุญธีร์ เครือตราชู ที่ให้ความรู้และคำปรึกษา ในการจัดทำวิทยานิพนธ์ฉบับนี้จนสำเร็จลุล่วงด้วยดี

ขอขอบพระคุณเจ้าหน้าที่สำนักวิจัยและเพื่อนๆ ที่ให้ความช่วยเหลือทางด้านอุปกรณ์และสถานที่ในการปฏิบัติงาน

ขอขอบพระคุณมูลนิธิเพื่อการศึกษาคอมพิวเตอร์และการสื่อสารที่สนับสนุนทุนวิจัย

สุดท้ายนี้ขอขอบพระคุณ

รศ.ดร.ชม	กิ้มปาน
ดร.บุญธีร์	เครือตราชู
ดร.เอื้อน	ปิ่นเงิน
รศ.ดร.รัตติกร	วรากูลศิริพันธุ์
ผศ.เกษตร์	ศิริสันติสัมฤทธิ์

ที่ให้ความกรุณามาเป็นกรรมการสอบวิทยานิพนธ์ในครั้งนี้

ศมิทธิ์ เอมสมบัติ

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญภาพ VI

สารบัญตาราง IX

บทที่

1. บทนำ 1

แนวความคิดที่ทำให้เกิดโครงข่ายประสาทเทียม 1

ประวัติและความเป็นมาของงานวิจัยในสาขานี้ 4

วัตถุประสงค์ของวิทยานิพนธ์ แนวทางของวิทยานิพนธ์ 7

แนวทางของวิทยานิพนธ์ในแต่ละบทโดยย่อ 7

2. ความรู้ทั่วไปเกี่ยวกับโครงข่ายประสาทเทียม 9

สรีรวิทยาของเซลล์ประสาท 9

แบบจำลองทางคณิตศาสตร์ของหนึ่งหน่วยเซลล์ประสาท 12

อัลกอริทึม ที่ใช้กับโครงข่ายประสาทเทียม 13

Perceptron 13

Adaline & Madaline 15

BackPropagation 17

Kohonen Self-Organize Maps 21

สารบัญ(ต่อ)

บทที่	หน้า
3. โครงสร้างข้อมูล	26
โครงสร้างข้อมูลของโครงข่ายนิวรอล	26
โครงสร้างข้อมูลของน้ำหนัก	28
4. ระบบและการใช้งานโปรแกรม	30
ระบบที่ใช้	30
ขั้นตอนการใช้งานโปรแกรม	31
ตัวอย่างการใช้งานโปรแกรม	45
5. บทสรุป	58
ภาคผนวก	59
ภาคผนวก ก โครงสร้างของข้อมูลอินพุต และ การเตรียมข้อมูลอินพุต	60
ภาคผนวก ข โครงสร้างของหน่วยความจำ และ การจัดการหน่วยความจำ	68
ภาคผนวก ค สรุปขั้นตอนการทำงานของโปรแกรมในแต่ละอัลกอริทึม	72
ภาคผนวก ง SOURCE CODE	71
เอกสารอ้างอิง	206
ประวัติผู้เขียน	207

สารบัญภาพ

หน้า

1. การวิเคราะห์รูปภาพด้วยคอมพิวเตอร์ 1
2. การวิเคราะห์รูปภาพด้วยโครงข่ายประสาทเทียม 2
3. โครงสร้างและส่วนประกอบของเซลล์ประสาท 9
4. รูปแบบของการทรงจำภาพถ่ายของเซลล์ประสาท 11
5. แบบจำลองทางคณิตศาสตร์ของเซลล์ประสาทของ McCulloch และ Pitts 12
6. หน่วยประมวลผลแบบ Perceptron 13
7. หน่วยประมวลผลแบบ Adaline 15
8. โครงสร้างของโครงข่ายแบบ Back Propagation network (BPN) 17
9. โครงสร้างการคำนวณของขั้นตอน forward-propagation 18
10. ลักษณะของ Mexican-hat ฟังก์ชัน 23
11. โครงสร้างทั่วไปของโครงข่ายประสาทเทียม 25
12. โครงสร้างข้อมูลของโครงข่ายประสาทเทียม 26
13. ลักษณะการมองจุดภาพของโครงข่าย 28
14. โครงสร้างข้อมูลของน้ำหนักโดยใช้อาร์เรย์ของพอย์เตอร์ชี้ข้อมูลแต่ละชุด 29
15. ลักษณะการทำงานของโปรแกรม 30
16. แสดงภาพหน้าจอของโปรแกรมที่พัฒนาขึ้น 31
17. การเปิดไฟล์ของชุดข้อมูล Pattern เพื่อนำข้อมูลเข้า 32
18. การ Setup Network 33
19. การฝึกฝนโครงข่าย 36
20. แสดงกราฟ histogram ของน้ำหนักเมื่อผู้ใช้เลือกสุ่มค่าน้ำหนัก 37

สารบัญภาพ (ต่อ)

หน้า

21. การแสดงน้ำหนักด้วยลำดับของสี 37

22. แสดงวิธีการคำนวณออกทางจอภาพในขั้นตอน Forward 38

23. แสดงวิธีการคำนวณออกทางจอภาพในขั้นตอน Backward 39

24. แสดงวิธีการคำนวณออกทางจอภาพในขั้นตอน Adaptive Weight 40

25. การแสดงผลการคำนวณของน้ำหนัก 41

26. การทดสอบการรู้จำของโครงข่ายเมื่อโครงข่ายสามารถ "จำได้" 43

27. การทดสอบการรู้จำของโครงข่ายเมื่อโครงข่ายไม่รู้จัก Pattern นั้น 44

28. ตัวอย่างตัวเลขไทยที่นำมาใช้ฝึกฝนโครงข่าย 45

29. กราฟแสดงค่าผิดพลาดรวม (Sum of square error) กับจำนวนรอบที่ใช้ (epochs) .. 46

30. จอภาพของอัลกอริทึม Kohonen Self-Organizing (SOM) 50

31. การเปลี่ยนแปลงของน้ำหนักที่โหนดของ winner neuron เมื่อวงรอบเท่ากับ 2 51

32. การเปลี่ยนแปลงของน้ำหนักที่โหนดของ winner neuron เมื่อวงรอบเท่ากับ 5 52

33. การเปลี่ยนแปลงของน้ำหนักที่โหนดของ winner neuron เมื่อวงรอบเท่ากับ 20 53

34. แสดงน้ำหนักของ winner neuron และ Pattern ที่จัดอยู่ในกลุ่มเดียวกัน เมื่อให้โครงข่ายมีโหนดเอาต์พุตเท่ากับ 5 54

35. แสดงน้ำหนักของ winner neuron และ Pattern ที่จัดอยู่ในกลุ่มเดียวกัน เมื่อให้โครงข่ายมีโหนดเอาต์พุตเท่ากับ 10 55

36. แสดงน้ำหนักของ winner neuron และ Pattern ที่จัดอยู่ในกลุ่มเดียวกัน เมื่อให้โครงข่ายมีโหนดเอาต์พุตเท่ากับ 20 56

37. ภาพหน้าจอของโปรแกรม Pattern Editor 60

สารบัญภาพ (ต่อ)

หน้า

38. รายการคำสั่งเมื่อเลือกหัวข้อ Pattern 60
39. หน้าต่างการกำหนดค่าต่างๆ ของ Pattern ที่สร้างใหม่ 61
40. แสดง Pattern ที่ต้องการจะแก้ไขพร้อมทั้งเอาต์พุตเป้าหมาย 62
41. รายการคำสั่งเมื่อเลือกหัวข้อ Effect 63
42. ตัวอย่างการสุ่ม noise 64
43. การจัดหน่วยความจำของ MSDOS 67
44. การจัดหน่วยความจำของ MSDOS เมื่อเทียบกับ Linear Address ของ DOS4GW 68

สารบัญตาราง

ตารางที่		หน้า
1.	ข้อแตกต่างพื้นฐานระหว่างดิจิตอลคอมพิวเตอร์ กับ โครงข่ายประสาทเทียม.....	4
2.	แบบจำลองของโครงข่ายประสาทเทียมในช่วงปี ค.ศ. 1950-1986	6
3.	การทดสอบการรู้จำของโครงข่าย	47
4.	ผลการทำงานของโครงข่ายเมื่อเปลี่ยนแปลงอัตราการเรียนรู้	48
5.	ผลการคำนวณของโครงข่ายที่มีจำนวนโหนดของชั้นที่ซ่อนอยู่ขนาดต่างๆ กัน ..	49

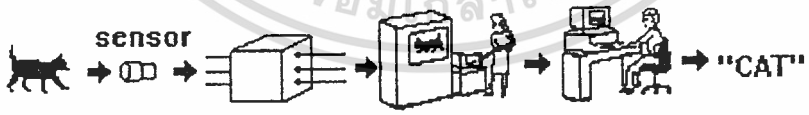
บทที่ 1

บทนำ

1.1 แนวความคิดที่ทำให้เกิดโครงข่ายประสาทเทียม

คอมพิวเตอร์ในปัจจุบันมีประสิทธิภาพในการทำงานสูง สามารถทำการคำนวณได้อย่างรวดเร็วและแม่นยำกว่าสมองของมนุษย์ แต่ก็มีงานบางอย่างที่สมองของมนุษย์ยังคงทำงานได้ดีกว่า เช่น การวิเคราะห์รูปภาพ การแยกลักษณะที่แตกต่างกันของสิ่งของที่มีคุณลักษณะใกล้เคียงกันและความสามารถในการเรียนรู้จากประสบการณ์ เป็นเพราะในการวิเคราะห์ภาพคอมพิวเตอร์มีการทำงานตามลำดับขั้น (Sequential Computers) โดยการวิเคราะห์แต่ละส่วนแยกจากกัน ซึ่งจะไม่ได้ข้อมูลที่มีความหมายของภาพมากนัก ในทางตรงกันข้ามถ้าเราวิเคราะห์ภาพเป็นกลุ่มพร้อมๆกัน คล้ายกับการวิเคราะห์ภาพของสมองมนุษย์จะให้ความหมายและความสัมพันธ์ของภาพมากขึ้น ลักษณะการวิเคราะห์รูปภาพด้วยดิจิตอลคอมพิวเตอร์มีขั้นตอนการทำงาน ดังภาพที่ 1

ภาพที่ 1

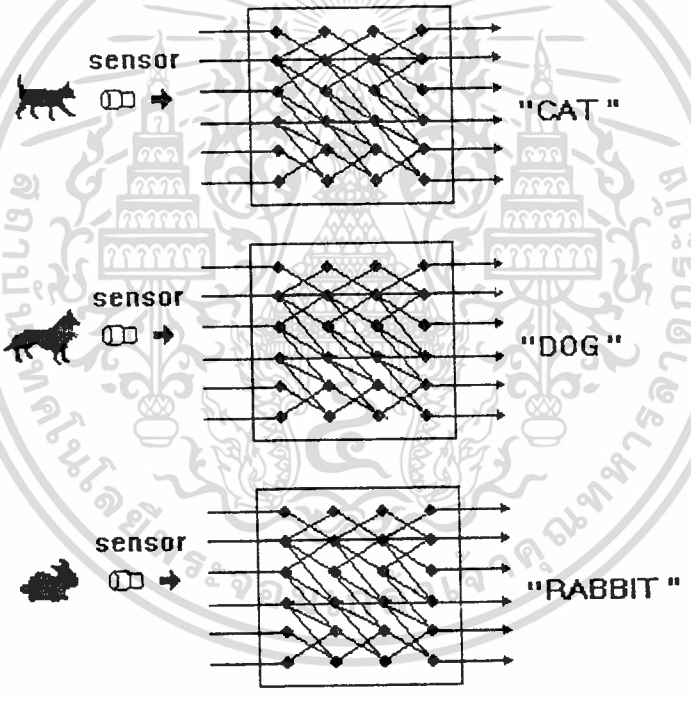


การวิเคราะห์รูปภาพด้วยดิจิตอลคอมพิวเตอร์[3]

อีกประการหนึ่งความรู้และประสบการณ์ที่มนุษย์สะสมมา จะศูนย์หายไปไปพร้อมกับความตาย ถึงแม้จะมีการถ่ายทอดความรู้โดยการจดบันทึกและโดยการสอน ก็ไม่อาจจะถ่ายทอดความรู้ทั้งหมดไว้ได้ ดังนั้นถ้าเราสามารถสร้างคอมพิวเตอร์ให้เลียนแบบการทำงานของสมองของ

มนุษย์เมื่อเราฝึกฝนและสอนคอมพิวเตอร์ให้มีความชำนาญในเรื่องใดเรื่องหนึ่งแล้วจะสามารถเก็บการเรียนรู้และประสบการณ์นั้นไว้ได้ตลอดไป นักวิชาการจึงพยายามที่จะศึกษาโครงสร้างเชิงสถาปัตยกรรมภายในและลักษณะการทำงานของเซลล์สมองของมนุษย์ อาจจะเป็นองค์ประกอบสำคัญที่ทำให้สมองของมนุษย์สามารถทำงานดังกล่าวได้ดีกว่าเครื่องคอมพิวเตอร์ จากข้อสังเกตข้างต้น ทำให้เกิดการออกแบบและสร้างระบบคอมพิวเตอร์ให้มีโครงสร้างเลียนแบบโครงสร้าง

ภาพที่ 2



การวิเคราะห์รูปภาพด้วยโครงข่ายประสาทเทียม[3]

เซลล์สมองของมนุษย์ โดยการประยุกต์ใช้ความรู้ที่ได้มาจากสรีรวิทยาประสาท (Neurophysiology) ในรูปของ Artificial Neural Network (ANN) ซึ่งเป็นการลอกเลียนแบบการทำงานของเซลล์ประสาท (nerve cells) ซึ่งมีชื่ออีกอย่างหนึ่งว่านิวรอน (neurons) รูปแบบ

คอมพิวเตอร์ที่กล่าวถึงนี้มีชื่อโดยอ้างอิงถึง"เซลล์สมอง" หรือ "นิวรอน" งานวิจัยอาจเรียกออกไปได้หลายชื่อเช่น

- ระบบโครงข่ายประสาทเทียม หรือ นิวรอลเน็ตเวอร์ก (Neural Networks)
- คอนเน็คชันนิซึม (Connectionism)
- โมเดลเชื่อมโยง (Connectionnist Models)
- ระบบเซลล์สมองจำลอง (Artificial Neural Systems - ANS)
- ระบบประมวลผลขนานแบบกระจายอำนาจ (Parallel Distributed Processing PDP)

ลักษณะการวิเคราะห์รูปภาพของโครงข่ายประสาทเทียมแสดงไว้ในภาพที่ 2 รูปแบบของโครงข่ายประสาทเทียมเป็นรูปแบบการคำนวณ ที่เขียนแทนด้วยกราฟ ซึ่งประกอบด้วยโหนดของนิวรอลเซลล์ และในแต่ละโหนดจะเชื่อมโยงถึงกันหมดระหว่างนิวรอลเซลล์ของแต่ละชั้น ในชั้นสุดท้ายจะเป็นชั้นเอาต์พุต จำนวนโหนดของชั้นเอาต์พุต ขึ้นอยู่กับจำนวนชนิดของภาพที่จะแบ่งแยก ก่อนที่จะมีการใช้งานโครงข่ายนิวรอลนั้นจะต้องสอนให้โครงข่ายรู้จักภาพแต่ละภาพ (Training) ด้วยการป้อนเอาต์พุตเป้าหมาย (target output) ให้กับโครงข่ายสำหรับรูปภาพหนึ่งๆ โดยกำหนดเป็นลักษณะคู่ข้อมูลอินพุต-เอาต์พุต ลักษณะเช่นนี้เรียกว่า "การเรียนรู้แบบมีผู้ช่วย" (Supervised Learning) ในขั้นตอนการเรียนรู้ของโครงข่ายจะนำเอาต์พุตที่ได้จากโครงข่ายมาเปรียบเทียบกับเอาต์พุตเป้าหมายที่ป้อนไว้ โดยนำปริมาณของค่าผิดพลาดมาใช้ในการปรับน้ำหนักที่จุดเชื่อมโยง ซึ่งรายละเอียดในการปรับน้ำหนักจะแตกต่างกันไปขึ้นอยู่กับชนิดของโครงข่าย การปรับน้ำหนักจะกระทำซ้ำจนกว่าจะได้เอาต์พุตตรงกับเอาต์พุตเป้าหมาย จึงถือว่าโครงข่ายสามารถทำการเรียนรู้ได้สำเร็จ และน้ำหนักที่ถูกปรับครั้งหลังสุดจะถูกเก็บไว้ เพื่อนำมาใช้ในขั้นตอนการทำงาน การฝึกฝนโครงข่ายควรจะนำรูปภาพหลายๆรูปภาพซึ่งอยู่ในคลาสเดียวกัน (มีเอาต์พุตเป้าหมายเดียวกัน) มาฝึกฝนโครงข่ายเพื่อให้โครงข่ายมีความชำนาญ หลังจากที่สอนโครงข่ายให้เรียนรู้ได้แล้ว ในขั้นตอนการทำงานเพียงแต่ป้อนรูปภาพให้กับอินพุตของโครงข่ายถ้าโครงข่ายสามารถ "จำได้" จะให้เอาต์พุตออกมาตรงกับที่สอนไว้ สำหรับโครงข่ายที่สามารถหาคำตอบได้เองโดยไม่ต้องมีเอาต์พุตเป้าหมายเรียกว่า "การเรียนรู้แบบไม่มีผู้ช่วย" (Unsupervised Learning) โครงข่ายจะนำอินพุตเวคเตอร์มาหากลุ่มของเอาต์พุตที่ตอบสนองมากที่สุดและจะใช้อัลกอริทึมการเรียนรู้ จัดกลุ่มและแยกประเภทของอินพุตได้เอง โดยให้คำตอบแทนด้วยเลขที่ของนิวรอนเซลล์และชุดข้อมูลของน้ำหนักของโหนดที่ตอบสนองต่ออินพุตนั้น โดยนิวรอนเซลล์ที่โหนดนี้มีชื่อว่า winner neuron

ตารางที่ 1

ดิจิตอลคอมพิวเตอร์	โครงข่ายประสาทเทียม
ประมวลผลข้อมูลแบบ binary	ประมวลผลกับข้อมูลแบบต่อเนื่อง analog
ให้คำตอบเป็น yes/no หรือ คำตอบทางตรรก	ใช้น้ำหนักที่จุดเชื่อมโยงเป็นตัวตัดสินคำตอบที่มีลักษณะคลุมเคลือ หรือ อยู่ในระหว่างผิดกับถูก
มีการประมวลผลเป็นลำดับขั้นตามที่ได้โปรแกรมไว้	รูปแบบวิธีการดำเนินไปอย่างอิสระ
ให้คำตอบที่จำกัด	ให้คำตอบที่ใกล้เคียงกับปัญหาที่ยุ่งยากซับซ้อน
รูปแบบของข้อมูลไม่มีความเกี่ยวข้องกันทางคณิตศาสตร์	รูปแบบของข้อมูลมีความสัมพันธ์กันในทาง คณิตศาสตร์
การเก็บข้อมูลมีลักษณะเฉพาะของแต่ละข้อมูล	การเก็บข้อมูลแบบเชื่อมโยง

ข้อแตกต่างพื้นฐานระหว่างดิจิตอลคอมพิวเตอร์ กับ โครงข่ายประสาทเทียม

ในตารางที่ 1 แสดงให้เห็นถึงข้อแตกต่างพื้นฐานระหว่างดิจิตอลคอมพิวเตอร์ กับ โครงข่ายประสาทเทียม ในเรื่องของการแก้ปัญหาตลอดจนได้คำตอบออกมา อนึ่งความสำเร็จที่ปรากฏทางด้านโครงข่ายประสาทเทียมยังคงอยู่ในรูปของการใช้ดิจิตอลคอมพิวเตอร์จำลองการทำงานของโครงข่ายประสาทเทียม

1.2 ประวัติและความเป็นมาของงานวิจัยในสาขานี้

โครงข่ายนิวรอลมีพื้นฐานมาจากนิวรอนในสมองมนุษย์ การศึกษาโครง สร้างการทำงานของสมองเพื่อที่จะลอกเลียนลักษณะการทำงานของเซลล์ประสาท (nerve cell) จึงเป็นจุดเริ่มต้นที่มนุษย์ทำการศึกษาการทำงานของนิวรอลในแง่ของสรีรวิทยาประสาท (Neurophysiology) เพื่อที่เป็นความรู้เบื้องต้นที่สำคัญ ก่อนจะก้าวสู่เรื่องของการจำลองการทำงานของเซลล์ประสาทด้วยเครื่องคอมพิวเตอร์ต่อไป William James นักสรีรศาสตร์ชาวอเมริกันได้เขียนหนังสือ "Psychology (Briefer Couse)"Holt,New York,1890. แสดงให้เห็นถึงความสัมพันธ์ของโครงสร้างสมองและลักษณะการทำงานของสมอง โดยได้อธิบายลักษณะของการเรียนรู้กับความสัมพันธ์ของจุดเชื่อมโยงระหว่างนิวรอน ที่ทำให้เกิดความทรงจำ James ได้ให้เหตุผลว่าปริมาณ

การกระตุ้นในแต่ละจุดของเยื่อหุ้มสมอง (braincortex)เกิดจากผลรวมที่ได้รับจากการกระตุ้นของจุดอื่นๆ ในอัตราส่วนที่พอเหมาะ ทั้งนี้ขึ้นอยู่กับ

1). จำนวนครั้งของการกระตุ้นในแต่ละจุด
2). ความเข้มในการกระตุ้น
3). การเปลี่ยนแปลงไปของน้ำหนักที่จุดเชื่อมโยงที่เกิดจากการกระทำครั้งก่อนที่อาจจะกลายเป็นผลในการกระตุ้นหรือเป็นผลในทางขับยั้งจนทำให้จุดเชื่อมโยงนั้นขาดออกจากกัน

อีกประมาณ 50 ปีต่อมา McCulloch and Pitts ได้เสนอบทความที่มีชื่อเสียงเกี่ยวกับทฤษฎีความสัมพันธ์ของระบบประสาทโดยอยู่บนพื้นฐานความรู้เกี่ยวกับโครงสร้างทางกายภาพของสิ่งมีชีวิตในต้นปี 1940 ซึ่งสรุปได้เป็นสมมุติฐาน 5 ประการ

1). การกระตุ้นของเซลล์ประสาทเป็นขบวนการ "all or none"
2). มีจำนวนไซแนพส์ที่แน่นอนที่กระตุ้นในช่วงเวลาอินพุต แทนที่จะได้รับการกระตุ้นโดย ตลอดเวลา ซึ่งปริมาณการกระตุ้นขึ้นอยู่กับการกระตุ้นครั้งก่อน และตำแหน่งของ นิวรอน
3). เวลาที่ใช้ในการส่งข้อมูลของเซลล์ประสาทที่บริเวณไซแนพส์ เรียกว่า synaptic delay
4). กิจกรรมที่เป็นการยับยั้งหรือขัดขวางที่บริเวณไซแนพส์ เป็นการป้องกันการกระตุ้น ของเซลล์ประสาท ณ ช่วงเวลานั้น
5). โครงสร้างของโครงข่ายเชื่อมโยงจะไม่เปลี่ยนไปตามเวลา

ช่วงเวลาอินพุต คือเวลาที่นิวรอนใช้ในการตรวจสอบอินพุตที่ปรากฏบริเวณไซแนพส์ (the period of latent addition) ซึ่งจะใช้เวลาน้อยกว่า 0.25 msec synaptic delay คือเวลาที่ใช้ในการส่งอินพุตตั้งแต่อินพุตเริ่มปรากฏจนกระทั่งส่งออกไปเป็นพัลส์จะใช้เวลา ประมาณ 0.5 msec บทความของ McCulloch and Pitts ยากที่จะทำความเข้าใจและปฏิบัติ ทำให้นักวิจัยทั้งหลายเลิกที่จะติดตาม และหลักการที่ได้เสนอในบทความนี้ได้ถูกนำมาใช้กับ neural network tools ในปัจจุบันเพียงบางส่วนเท่านั้น

โครงสร้างของโครงข่ายและอัลกอริทึม ที่ใช้ในการเรียนรู้ได้ถูกพัฒนาและคิดค้นทฤษฎีขึ้นมาใหม่โดยตลอดในช่วงระยะเวลาที่ผ่านมา โดยแต่ละโครงข่ายจะมีความสามารถเฉพาะตัวในลักษณะที่ต่างกันออกไป พอจะสรุปได้ดังตารางที่ 2

ตารางที่ 2

ชื่อแบบจำลอง	ผู้คิดค้น	ปีที่พัฒนา	การใช้งาน
1.Perceptron	F.Rosenblatt	1957	อ่านตัวพิมพ์ดีด
2.Madaline	B.Widrow	1960-62	adaptive equalizer
3.Avalanche	S.Grossberg	1967	เข้าใจคำพูด
4.Cerebellatron	D.Mar,J.Albus และ A.Pellionez	1969-82	ควบคุมการเคลื่อนไหวของแขนหุ่นยนต์
5.Back Propagation	P.Werbos,D.Parker และ D.Rumelhart	1974-85	เลียนแบบฟังก์ชันทางคณิตศาสตร์,อ่านตัวเขียน,คำพูด,ทำนายหุ้น
6.Brain-State-in-a-box	J.Anderson	1977	สกัดข้อมูลบางอย่างจากฐานข้อมูล
7.Neocognitron	K.Fukushima	1978-84	อ่านตัวเลขและตัวอักษร เขียน
8.AdativeResonance	G.Carpenter และ S.Grossberg	1978-86	pattern recognition
9.Self-Organizing Map	T.Kohonen	1980	จำลองลักษณะพื้นผิวของวัตถุ เช่น ปีกเครื่องบิน
10.Hopfield	J.Hopfield	1982	ทำนายส่วนของข้อมูลที่หายไปเช่นบางส่วนของหน้าคน และทำ optimization สำหรับปัญหาพวก combinatorics
11.Bidirectional	B.Kosko	1985	content-addressable associative memory
12.Boltzman และ Cauchy machine	J.Hinton,T.Sejnowskky, และ H.Szu	1985-86	patternrecognition
13.Counterpropagation	R.Hecht-Neielsen	1986	อัดข้อมูลของภาพให้น้อยลง ,table-lookup

แบบจำลองของโครงข่ายประสาทเทียมในช่วงปี ค.ศ. 1950-1986

1.3 วัตถุประสงค์ของวิทยานิพนธ์ แนวทางของวิทยานิพนธ์

ปัจจุบันโครงข่ายประสาทเทียมได้รับความจนใจเพิ่มขึ้นเนื่องจาก คอมพิวเตอร์มีความเร็วสูงขึ้นและมีราคาถูกลง ทำให้ข้อจำกัดต่างๆในอดีตหมดไป อัลกอริทึมที่ใช้ในการเรียนรู้มีประสิทธิภาพมากขึ้น แต่ยังขาดซอฟท์แวร์ที่ใช้ในการออกแบบและจำลองการทำงานของโครงข่าย ส่วนซอฟท์แวร์ของต่างประเทศมีราคาสูงและยังไม่ค่อยมีแพร่หลายมากนัก วิทยานิพนธ์ฉบับนี้จึงได้เสนอแนวทางการพัฒนาโปรแกรมเพื่อใช้ในการศึกษาและช่วยในการออกแบบโครงข่ายประสาทเทียม โดยใช้เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) เพราะปัจจุบันมีราคาถูกและมีใช้กันอย่างแพร่หลาย ซึ่งผู้ใช้สามารถทำการทดลองได้โดยไม่ต้องเขียนโปรแกรมขึ้นมาเอง เนื่องจากโครงข่ายประสาทเทียมเกี่ยวข้องกับการคำนวณและตัวเลขจำนวนมาก จึงต้องการส่วนที่ติดต่อกับผู้ใช้ (User Interface) ที่ดี มีส่วนแนะนำ (HELP) เป็นแนวทางในการใช้งาน สามารถนำรูปแบบอินพุตที่สร้างขึ้นมา เลือกใช้กับอัลกอลิทึมที่ต้องการ ผู้ใช้สามารถสังเกตขั้นตอนการเรียนรู้และเอาต์พุตที่ได้จากการทดลอง

1.4 แนวทางของวิทยานิพนธ์ในแต่ละบทโดยย่อ

โครงร่างวิทยานิพนธ์ ในแต่ละบท

บทที่ 2 ความรู้ทั่วไปเกี่ยวกับโครงข่ายประสาทเทียม
- สรีรวิทยาของเซลล์ประสาท (Neurophysiology)
- การทำงานและหน้าที่ของเซลล์ประสาท
- โครงสร้างพื้นฐานของ PE (Processing Element)
- อัลกอริทึม ที่ใช้กับโครงข่ายประสาทเทียม

บทที่ 3 โครงสร้างข้อมูล
- โครงสร้างข้อมูลของโครงข่าย
- โครงสร้างข้อมูลอินพุตและการแปลความหมายของจุดภาพที่ใช้ในโปรแกรม
- โครงสร้างข้อมูลเอาต์พุตของนิวรอนและน้ำหนัก
- การจองหน่วยความจำ

บทที่ 4 ระบบและการใช้งานโปรแกรม

- อุปกรณ์และระบบที่ใช้
- ขั้นตอนการทำงานของโปรแกรม
- การใช้งานโปรแกรม
- การทดลองการทำงาน
- ตัวอย่างการใช้งานโปรแกรม
- ผลการทดลอง

บทที่ 5 บทสรุป

- สรุปผลของวิทยานิพนธ์

เอกสารอ้างอิง

ภาคผนวก ก การเตรียมข้อมูลอินพุต และตัวอย่างอินพุตที่ใช้

ภาคผนวก ข โครงสร้างของหน่วยความจำ และ การจัดการหน่วยความจำ

ภาคผนวก ค สรุปขั้นตอนการทำงานของโปรแกรมในแต่ละอัลกอริทึม

ภาคผนวก ง SOURCE CODE

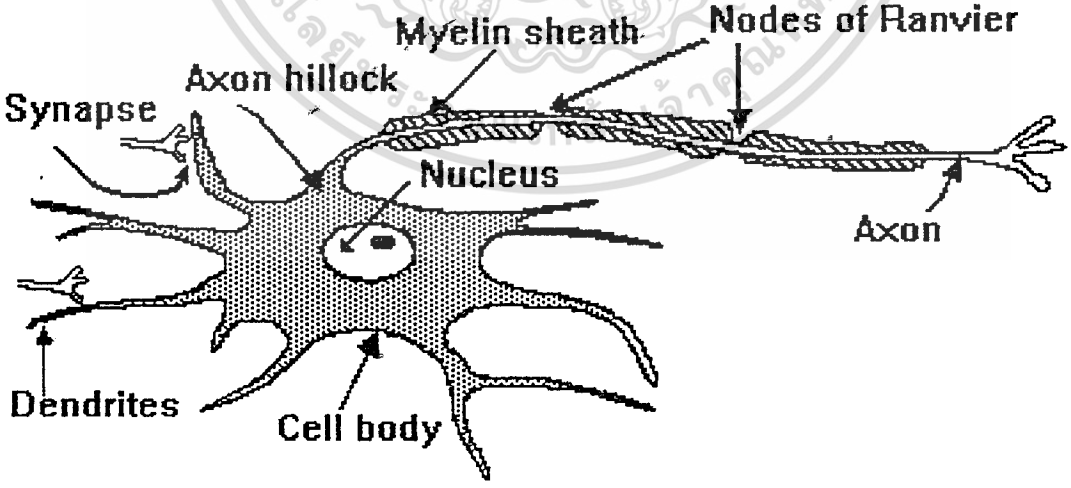
บทที่ 2

ความรู้ทั่วไปเกี่ยวกับโครงข่ายประสาทเทียม

2.1 สรีรวิทยาของเซลล์ประสาท (Neurophysiology)

โครงข่ายประสาทเทียมนั้นได้ศึกษามาจากเซลล์ประสาทของสมองมนุษย์ การศึกษาเซลล์ประสาทก็เพื่อที่จะนำผลที่ได้มาเป็นพื้นฐานในการจำลองการทำงานด้วยเครื่องคอมพิวเตอร์ หน้าที่หลักของเซลล์ประสาทคือการคำนวณและการส่งผลลัพธ์ต่อไปยังเซลล์อื่น โดยอาศัยหลักการทางไฟฟ้า รูปร่างของเซลล์ประสาทหรือนิวรอนมีลักษณะคล้ายกับต้นไม้ที่ไม่มีใบมีแต่กิ่งและรากที่เชื่อมโยงด้วยลำต้น

ภาพที่ 3



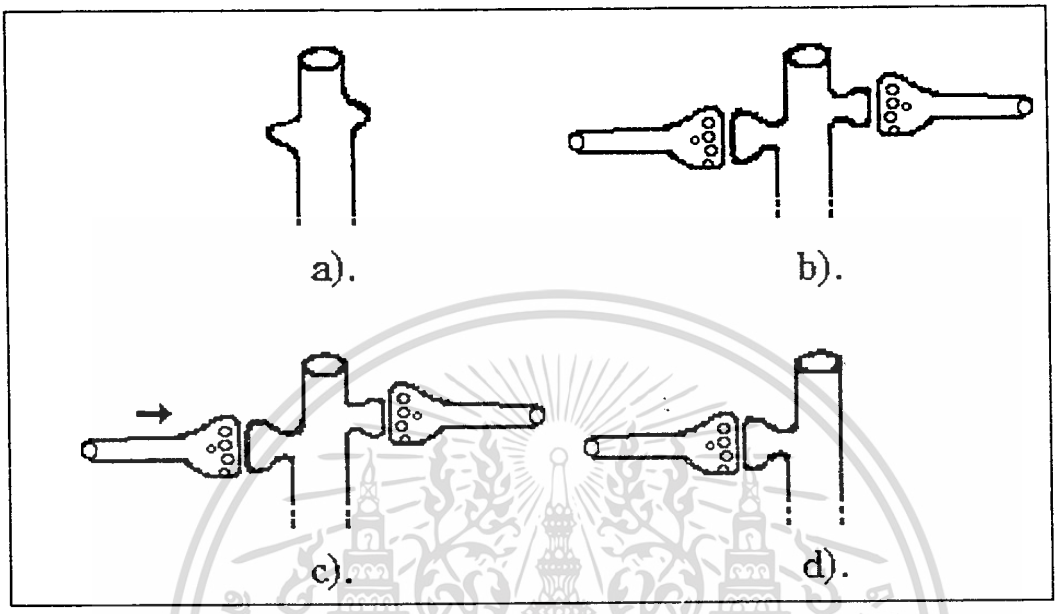
โครงสร้างและส่วนประกอบของเซลล์ประสาท

โครงสร้างและส่วนประกอบของเซลล์ประสาทมีลักษณะดังภาพที่ 3 การทำงานของเซลล์ประสาท จะเริ่มต้นจาก นิวรอนเซลล์รับอินพุตจากเซลล์นิวรอนตัวอื่น ผ่านทางจุดเชื่อมโยงที่เรียกว่า"ไซแนพส์"(Synapse) สัญญาณข้อมูลจากไซแนพส์จะถูกส่งผ่านเข้าทาง "เดนไดรต์" (Dendrite) ซึ่งเป็นส่วนที่ทำหน้าที่เป็นตัวรับข้อมูลอินพุตเข้าสู่ตัวเซลล์ (cell body) สัญญาณข้อมูลอินพุตจะถูกประมวลผลบางประการตามขบวนการที่เกิดขึ้นภายในเซลล์ และจะส่งสัญญาณเอาต์พุตออกทางส่วนของเซลล์ที่เรียกว่า"เอ็กซอน" (Axon) สัญญาณดังกล่าวจะผ่านข้ามไซแนพส์ด้วยเงื่อนไขบางประการซึ่งเป็นส่วนอินพุตของเซลล์อื่นอีกต่อไป โครงสร้างของเซลล์ประสาทแสดงไว้ในภาพที่ 3 การส่งผ่านข้อมูลต่างๆ จะอยู่ในรูปของสัญญาณไฟฟ้า สัญญาณไฟฟ้านี้เรียกว่าอิมพัลส์ (impulse) ที่มีแรงเคลื่อนไฟฟ้าแค่เพียง 0.1 โวลต์ และอยู่ได้นานเพียง 1 ms (1/1000 วินาที) โดยจะวิ่งไปในเอ็กซอนด้วยความเร็วประมาณ 500 กิโลเมตรต่อชั่วโมง การส่งสัญญาณหรืออิมพัลส์ภายในร่างกาย จะอาศัยการต่อเป็นลูกโซ่ของเซลล์ประสาทวิ่งจากเซลล์หนึ่งไปยังอีกเซลล์หนึ่ง ผ่านจุดเชื่อมโยงที่เรียกว่าไซแนพส์ ซึ่งเป็นสารเคมีชนิดหนึ่งที่มีผลต่อการเปลี่ยนแปลงของระดับแรงดันไฟฟ้า ด้วยเหตุนี้การทำงานของยาระงับปวดหรือยานอนหลับจะอาศัยการแทรกแซงกระบวนการทางเคมีที่มีหน้าที่เชื่อมโยงเซลล์ประสาทบริเวณไซแนพส์ด้วยการลดหรือเพิ่มความเข้มในการเชื่อมโยง

เรื่องความทรงจำและการเรียนรู้เป็นสิ่งสำคัญที่สุด การเรียนรู้ของมนุษย์ไม่ได้เกิดจากการเปลี่ยนแปลงทางเคมีเพียงอย่างเดียว แต่พอสังเกตได้ว่าการเปลี่ยนแปลงจะเกิดในด้านกายวิภาค (Anatomical Atterations) ด้วย ซึ่งการเปลี่ยนแปลงทางกายวิภาคนี้ไม่ได้เกิดกับนิวรอนหรือไซแนพส์เพียงตัวเดียวหรือจุดเดียวแต่จะเกิดกับหลายๆ นิวรอนพร้อมกัน นิวรอนตัวใดตัวหนึ่งจะร่วมมือกับนิวรอนตัวอื่นๆ อีกหลายๆ ตัวเพื่อให้เกิดริ้วรอยของความทรงจำสำหรับเรื่องหนึ่งๆขึ้นในสมอง ส่วนที่จุดเชื่อมต่อที่ไม่ได้ผ่านการใช้งานไซแนพส์ที่จุดนั้นก็จะค่อยหลุดออกไปหมายถึงว่าความทรงจำในเรื่องหนึ่งๆ มักเกิดจากการเปลี่ยนแปลงของนิวรอนหลายๆตัวและจะเกิดในย่านใดย่านหนึ่งของสมองที่แต่ละย่านจะถนัดจำในเรื่องที่ต่างกันออกไป

รูปแบบการทรงจำในลักษณะของการจดจำแบบภาพถ่าย ต้องอาศัยตัวนิวรอนเป็นกลไกสำคัญ 4 ขั้นตอนด้วยกันคือ ขั้นแรกนิวรอนจะแตกหน่อออกมา ขั้นที่สอง จะต่อกับนิวรอนที่อยู่ใกล้ ๆโดยไม่ได้อาศัยกฎเกณฑ์ที่แน่นอน (Random Connection) ขั้นที่สาม เป็นการใช้ไซแนพส์บางตัววิเคราะห์เหตุการณ์ในขณะใดขณะหนึ่ง ส่วนขั้นที่สี่ แสดงการหลุดร่วงของไซแนพส์ที่ไม่ได้ใช้ ซึ่งการหลุดร่วงจะเกิดเป็นคราว ๆ ไป คงเหลือแต่ไซแนพส์ที่กำลังใช้งานเท่านั้น

ภาพที่ 4



รูปแบบของการทรงจำภาพถ่ายของเซลล์ประสาท

a). ขั้นแรกนิวรอนจะแตกหน่อออกมา

b). ขั้นที่สองจะต่อกับนิวรอนที่อยู่ใกล้ๆ โดยไม่ได้อาศัยกฎเกณฑ์ที่แน่นอน (Random Connection)

c). ขั้นที่สาม เป็นการใช้ไซแนพส์บางตัววิเคราะห์เหตุการณ์ในขณะใดขณะหนึ่ง

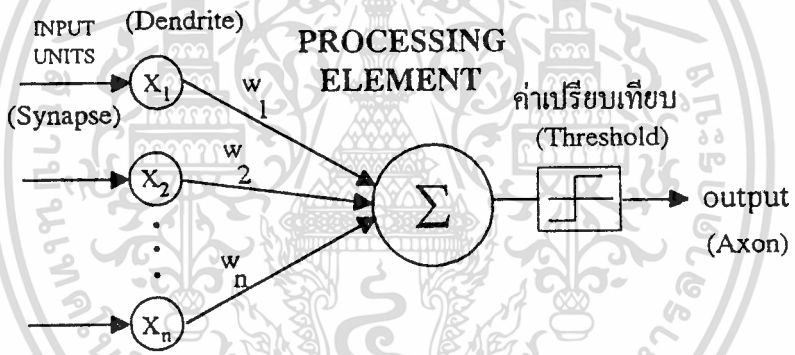
d). ขั้นที่สี่ แสดงการหลุดร่วงของไซแนพส์ที่ไม่ได้ใช้

การส่งสัญญาณระหว่างเซลล์ประสาท กระทำโดยการถ่ายเทสารประกอบของโซเดียมและโปรตัสเซียม Hodgkin และ Huxley ได้รับรางวัลโนเบลทางชีววิทยาได้ค้นพบว่า การไหลของสารประกอบโซเดียมและโปรแตสเซียมของเซลล์ปลาหมึกได้ทำให้เกิดความต่างศักย์จะอยู่ระหว่าง 50 ถึง 70 มิลลิโวลต์ จากผลการศึกษาดังกล่าวทำให้เราสามารถจำลองการทำงานของเซลล์ประสาทโดยอาศัยวงจรอิเล็กทรอนิกส์ได้ และทำให้เกิดแนวทางการพัฒนารูปแบบจำลองในเชิงคณิตศาสตร์ที่นำมาใช้กับคอมพิวเตอร์

2.2 แบบจำลองทางคณิตศาสตร์ของหนึ่งหน่วยเซลล์ประสาท

จากโครงสร้างพื้นฐานของเซลล์ประสาทในภาพที่ 3 ในปี ค.ศ. 1943 McCulloch และ Pitts ได้เสนอแบบจำลองเชิงคณิตศาสตร์สำหรับแต่ละหน่วยเซลล์ประสาท ซึ่งโครงสร้างของเซลล์ประสาทจำลองนี้สามารถแสดงได้ดังภาพ

ภาพที่ 5



แบบจำลองทางคณิตศาสตร์ของเซลล์ประสาทของ McCulloch และ Pitts

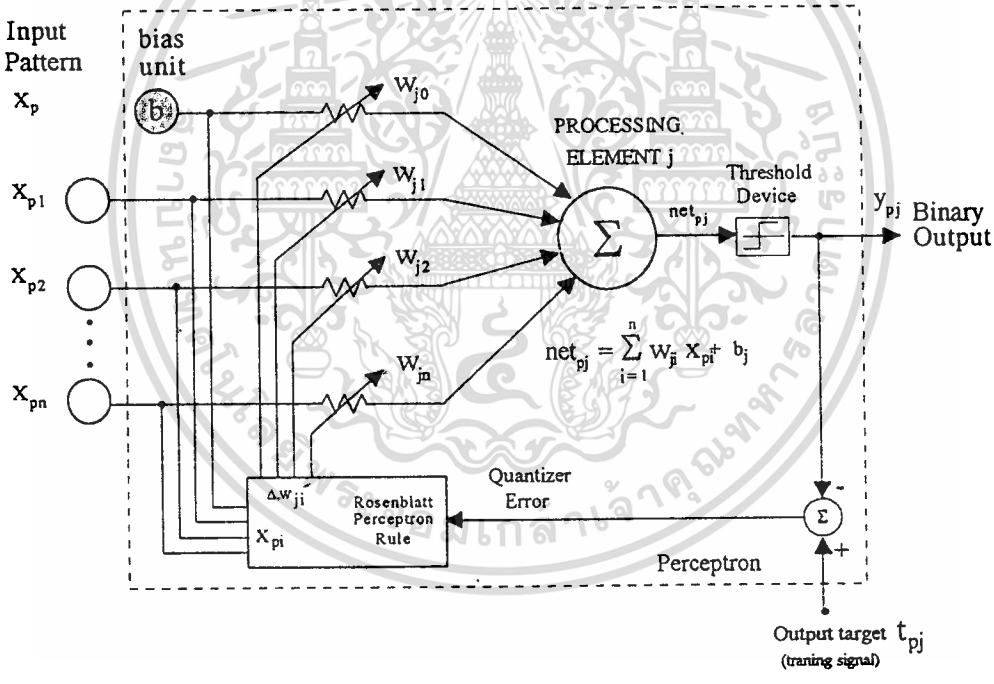
โครงสร้างของเซลล์ประสาทจำลองในภาพที่ 5 ประกอบด้วยชั้นอินพุต(dendrites) มีหน้าที่รับเอาต์พุตที่ได้จากนิวรอนเซลล์ตัวอื่น (axon) ผ่านเข้ามายังตัวเซลล์ โดยที่ปริมาณความเข้มข้นของสารเคมีที่บริเวณจุดเชื่อมโยง (synapse) จะถูกแทนด้วยค่าของเลขทศนิยมซึ่งเรียกว่าน้ำหนัก (weight) อินพุตของเซลล์ประสาทจะเป็นได้ทั้งค่าบวกและลบ อินพุตที่เป็นบวกจะไปกระตุ้นหรือเร้าให้เกิดเอาต์พุตส่วนอินพุตที่เป็นลบจะไปห้ามหรือยับยั้งไม่ให้เกิดเอาต์พุต พลังงานที่เป็นอินพุตบวกและลบจะไปรวมกันถ้ามากเกินกว่าค่าที่กำหนดค่าหนึ่งซึ่งเรียกว่าค่าพิกัด (threshold value) ก็จะทำให้เกิดเอาต์พุต

2.3 อัลกอลิทึมที่ใช้กับโครงข่ายประสาทเทียม

2.3.1 Perceptron

เป็นโครงสร้างพื้นฐานของโครงข่ายประสาทเทียมที่ถูกสร้างขึ้นมาในยุคแรกๆ เป็นอัลกอริทึมที่คิดค้นโดย Frank Rosenblatt ปี ค.ศ.1958 โครงสร้างพื้นฐานของ Perceptron จะประกอบไปด้วยนิวรอนสองชั้นคือชั้นอินพุตและชั้นเอาต์พุต ลักษณะอินพุตเป็นแบบไบนารีคือมีค่าเป็น 0 หรือ 1 ในระหว่างชั้นอินพุตและชั้นเอาต์พุตจะเชื่อมโยงแบบ ต่อถึงกันหมดโดย w_{ji} จะเป็นค่าน้ำหนักที่จุดเชื่อมโยง

ภาพที่ 6



หน่วยประมวลผลแบบ Perceptron[7]

จากแบบจำลองในภาพที่ 6 การหาผลรวมของสัญญาณอินพุตที่เข้าสู่นิวรอนเซลล์โหนด j หาได้จากผลรวมของผลคูณระหว่างสัญญาน อินพุต กับค่าน้ำหนักที่จุดเชื่อมโยง

$$net_{pj} = \sum_{i=1}^{n} w_{ji} x_{pi} + b_j \qquad [2.1]$$

เมื่อ net_{pj} เป็นผลรวมของของสัญญาณอินพุตทั้งหมดของนิวรอลเซลล์ที่โหนด j ของ pattern p. x_{pi} เป็นอินพุตของนิวรอลเซลล์ตัวที่ i และ w_{ji} เป็นน้ำหนักที่จุดเชื่อมโยง b_j เป็น bias unit ของโหนด j ซึ่งจะมีค่าคงที่เท่ากับ 1 โดยจะเชื่อมโยงต่อกับทุกๆโหนดในชั้น j และจะมีค่าน้ำหนักเช่นเดียวกับจุดเชื่อมโยง อื่นๆ ดังนั้นน้ำหนักของ bias unit จะถูกเรียกเป็น w_{j0} เนื่องจาก bias unit จะมีค่าเป็น 1 เสมอ ดังนั้นผลคูณของ bias unit กับน้ำหนักที่จุดเชื่อมโยงจะมีค่าเท่ากับน้ำหนักที่จุดเชื่อมโยงนั้น ผลที่ได้รับจาก bias unit จะทำให้ผลรวมของสัญญาณที่นิวรอลได้รับมีค่าเปลี่ยนไปซึ่งจะเป็นวิธีการปรับค่า Theshold ของโครงข่ายแบบหนึ่ง เอาต์พุตของนิวรอลเซลล์จะเป็นฟังก์ชันของ net_{pj} คือ $f(net_{pj})$ ถ้าหากผลรวมของสัญญาณอินพุตที่เกิดขึ้นภายในนิวรอลมีค่ามากกว่าค่าพิกัดที่กำหนด เอาต์พุตของ Perceptron จะมีค่าเป็น 1 ไม่เช่นนั้นจะได้เอาต์พุตที่เป็น 0 ออกมา ฟังก์ชันของเอาต์พุตอาจเขียนได้เป็น

$$f(net_{pj}) = \begin{cases} 0 & \text{ถ้า } net_{pj} \leq 0 \\ 1 & \text{ถ้า } net_{pj} > 0 \end{cases} \qquad [2.2]$$

โครงข่ายนี้จะต้องอาศัยข้อมูลเอาต์พุตเป้าหมาย (target output) จากภายนอกโดยต้องอาศัยคู่ของข้อมูลอินพุต-เอาต์พุต คือการป้อนคำตอบที่ถูกต้องให้แก่เอาต์พุตนี้ การเรียนรู้ในแบบนี้จึงมีชื่อเรียกทางเทคนิคว่า "การเรียนรู้แบบมีผู้ช่วย" (Supervised Learning) เอาต์พุตของโครงข่ายจะถูกนำมาเปรียบเทียบกับเอาต์พุตเป้าหมาย เพื่อคำนวณหาค่าผิดพลาดของโครงข่าย ค่าผิดพลาดที่ได้จะถูกนำมาใช้วิเคราะห์ในการปรับน้ำหนัก สามารถคำนวณหาความผิดพลาดได้จาก

$$(t_{pj} - y_{pj}) \qquad [2.3]$$

เมื่อ t_{pj} เป็นเอาต์พุตเป้าหมายของโครงข่ายสำหรับ pattern P ของโหนด j

และ y_{pj} เป็นเอาต์พุตของโหนด j ที่ได้จากโครงข่ายเมื่อใช้ pattern P เป็นอินพุต

เมื่อทราบความผิดพลาดของโครงข่ายแล้ว จะนำค่าผิดพลาดที่ได้นั้นมาใช้ในการปรับค่าน้ำหนัก โดยค่าน้ำหนักที่ถูกปรับใหม่หาได้จาก

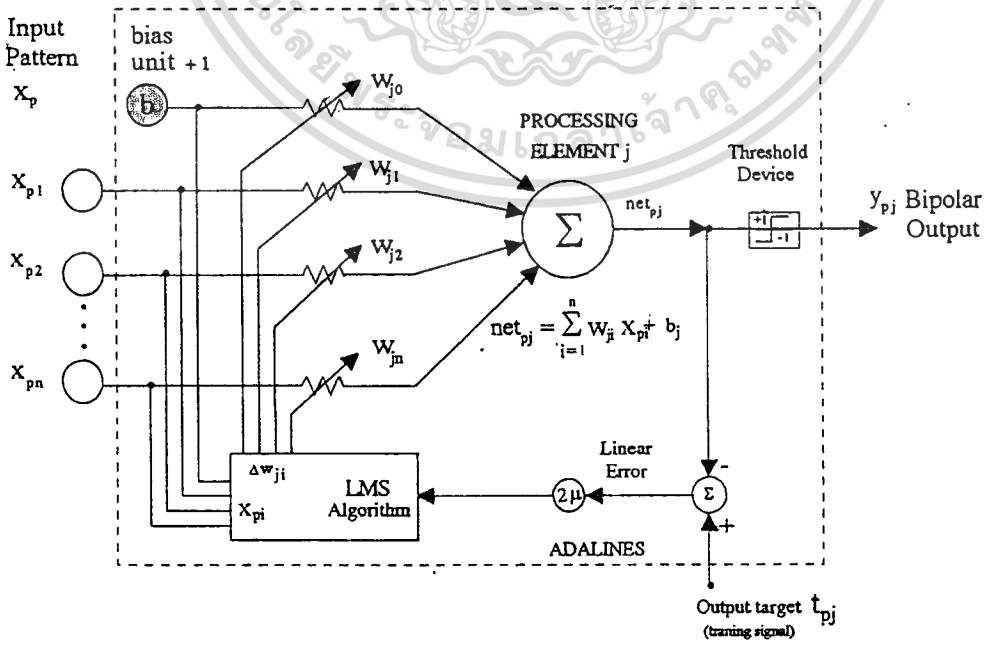
$$W_{ji}(new) = W_{ji}(old) + \eta x_{pi}(t_{pj} - y_{pj}) \qquad [2.4]$$

W_{ji}(new) เป็นค่าน้ำหนักที่ถูกปรับใหม่ W_{ji}(old) เป็นค่าน้ำหนักเดิม η เป็นอัตราการเรียนรู้จะมีค่าอยู่ในช่วง 0-1 มักจะเป็นปริมาณค่าน้อยๆโดยจะมีค่าประมาณ 0.15 ซึ่งจะใช้ค่านี้เป็นแนวทางในตอนเริ่มต้นเท่านั้น เพราะอัตราการเรียนรู้ที่เหมาะสมจะหาได้จากการทดลอง และ x_{pi} เป็นอินพุตที่มากระตุ้น (activate value) ของ pattern p ที่โหนด i การปรับน้ำหนักจะกระทำจนกว่าเอาต์พุตที่ได้มีค่าความผิดพลาดน้อยกว่าค่าผิดพลาดต่ำสุดที่กำหนดไว้ เช่นยอมให้ค่าผิดพลาดที่เกิดขึ้นมีได้ไม่เกิน 0.02 เป็นต้น ผลของเอาต์พุตที่ได้จะเป็นไบนารี และความสามารถในการจำแนกประเภทของข้อมูลอินพุตจะเป็นแบบเชิงเส้น (linear separator)

2.3.2 ADALINES AND MADALINES

ADALINES ชื่อของอัลกอริทึมนี้เป็นคำที่ย่อมาจาก *adaptive linear neuron* ซึ่งถูกคิดขึ้นโดย Bernard Widrow และ Marcian Hoff ในปี ค.ศ. 1960 เป็นโครงข่ายเซลล์ประสาทชั้นเดียว มีอินพุตเป็นค่าจริงอยู่ในช่วง [-1,+1] การเรียนรู้จะใช้หลักการของ ค่าที่น้อยที่สุดของค่าเฉลี่ยกำลังสองของค่าผิดพลาด (minimizing the average square error) เอาต์พุตของ Adaline จะเป็น bipolar {-1,+1}

ภาพที่ 7



หน่วยประมวลผลแบบ Adaline [7]

จากแบบจำลองทางคณิตศาสตร์ในภาพที่ 7 การหาผลรวมของสัญญาณอินพุตที่เข้าสู่นิวรอนเซลล์โหนด j หาได้จากผลรวมของผลคูณระหว่างสัญญาณ อินพุต กับค่าน้ำหนักที่จุดเชื่อมโยง

$$net_{pj} = \sum_{i=1}^{n} W_{ji} X_{pi} + b_j \qquad [2.5]$$

เมื่อ net_{pj} เป็นผลรวมของของสัญญาณอินพุตทั้งหมดของนิวรอลเซลล์ที่โหนด j ของ pattern p x_{pi} เป็นอินพุตของนิวรอลเซลล์ตัวที่ i และ w_{ji} เป็นน้ำหนักที่จุดเชื่อมโยง b_j เป็น bias unit ของโหนด j เอาต์พุตของนิวรอลเซลล์จะเป็นฟังก์ชันของ net_{pj} คือ $f(net_{pj})$ ถ้าหากผลรวมของสัญญาณอินพุตที่เกิดขึ้นภายในนิวรอลมีค่ามากกว่าค่าพิกัดที่กำหนด เอาต์พุตของ Adaline จะมีค่าเป็น +1 ไม่เช่นนั้นจะได้เอาต์พุตที่เป็น -1 ฟังก์ชันของเอาต์พุตอาจเขียนได้เป็น

$$f(net_{pj}) = \begin{cases} +1 & \text{ถ้า } net_{pj} \geq 0 \\ -1 & \text{ถ้า } net_{pj} < 0 \end{cases} \qquad [2.6]$$

Adaline จะใช้ delta rule[2] หรือเป็นที่รู้จักกันดีในชื่อของ lest mean squares (LMS) เป็นสมการที่ใช้ในการเรียนรู้ ค่าผิดพลาดของโครงข่ายหาได้จาก เอาต์พุตของโครงข่ายโดยสมการที่ใช้เรียนรู้จะมีลักษณะเป็น

$$W_{ji}(t+1) = W_{ji}(t) + 2 \mu \varepsilon_{pj} x_{pi} \qquad [2.7]$$

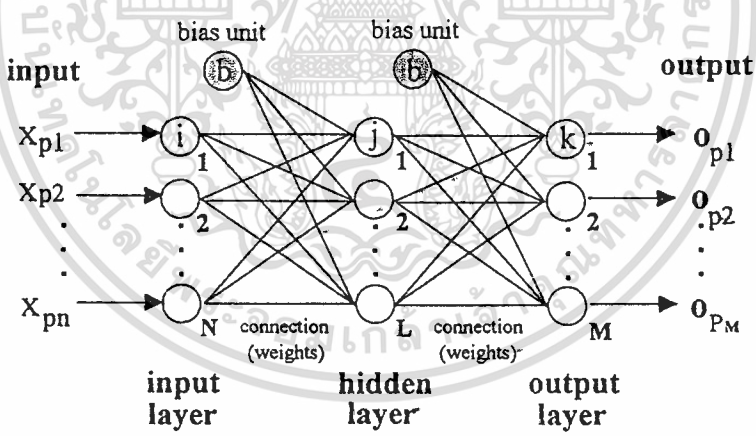
เมื่อ $W_{ji}(t+1)$ เป็นค่าน้ำหนักที่ถูกปรับใหม่ $W_{ji}(t)$ เป็นค่าน้ำหนักเดิม μ เป็นอัตราการเรียนรู้ p หมายถึงPattern ε_{pj} เป็น เทอร์มค่าผิดพลาดที่โหนด j หาได้จากผลต่างของเอาต์พุตเป้าหมายกับเอาต์พุตของโครงข่าย $(t_{pj} - y_{pj})$ และ x_{pi} เป็นอินพุตตัวที่ i การปรับน้ำหนักจะกระทำจนกว่าเอาต์พุตที่ได้มีค่าความผิดพลาดน้อยกว่าค่าผิดพลาดต่ำสุดที่กำหนดไว้

Madaline[6] เป็นโครงข่ายหลายชั้นที่ประกอบขึ้นด้วย Adaline ยูนิต Madaline I (MRI) น้ำหนักของชั้นที่ซ่อนอยู่จะสามารถปรับได้ (ถูกปรับในขั้นตอนการเรียนรู้ตามสมการที่ 2.7) ส่วนน้ำหนักของชั้นเอาต์พุตจะปรับไม่ได้ (ให้น้ำหนักเป็นคงที่หลังจากการสุ่มค่าน้ำหนัก) ส่วน Madaline II (MRII) น้ำหนักของชั้นที่ซ่อนอยู่และชั้นเอาต์พุตสามารถปรับได้ และเอาต์พุตที่โหนดเอาต์พุตจะได้จากการ OR (ทางตรรก)กับผลลัพธ์ที่ได้จากชั้นที่ซ่อนอยู่

2.3.3 Back Propagation Network (BPN)

โครงสร้างพื้นฐานของ BPN ประกอบด้วยนิวรอลเซลล์ที่เรียงกันอยู่อย่างน้อย 3 ชั้นเพื่อทำให้โครงข่ายสามารถจำแนก Pattern แบบไม่เป็นเชิงเส้น (non linear separable) โดยจะใช้เทคนิคสำคัญในการเรียนรู้ที่เรียกว่าการแพร่กลับ ชั้นแรกเป็นชั้นอินพุต ชั้นถัดมาจะเป็นชั้นที่ซ่อนอยู่ ซึ่งชั้นนี้สามารถมีได้หลายชั้น และชั้นสุดท้ายเป็นชั้นเอาต์พุต ในแต่ละชั้นจะมีการเชื่อมโยงแบบต่อถึงกันหมด (full connected) แต่ละนิวรอลเซลล์จะเชื่อมโยงกับทุกๆ หน่วยในชั้นก่อนหน้านี้และชั้นถัดไป แต่ไม่ได้เชื่อมโยงกันระหว่างนิวรอนเซลล์ในชั้นเดียวกัน ส่วน b เป็น bias unit ซึ่งจะมีค่าเป็น +1 เสมอโดยจะเชื่อมโยงกับทุกๆโหนดในแต่ละชั้น โครงสร้างพื้นฐานของโครงข่ายแบบ Back Propagation Network แสดงไว้ในภาพที่ 8

ภาพที่ 8



โครงสร้างของโครงข่ายแบบ Back Propagation network (BPN)

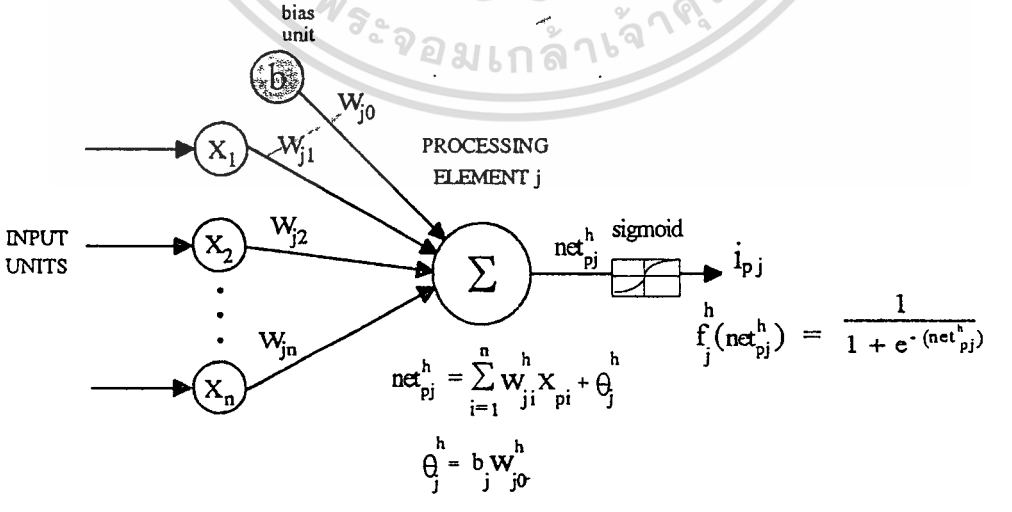
ขั้นตอนการทำงานของ BPN จะเริ่มต้นจากการนำค่าอินพุตที่ได้รับจากชั้นอินพุตมาคำนวณหาผลรวมที่ได้รับในแต่ละโหนด แล้วจึงส่งผลลัพธ์ที่ได้ส่งต่อไปยังชั้นถัดไป ทิศทางการไหลของข้อมูลจะเป็นลักษณะเคลื่อนไปข้างหน้า จากชั้นของอินพุต ผ่านชั้นของนิวรอลเซลล์ภายในไปสู่ชั้นของเอาต์พุต วิธีการคำนวณของโครงข่ายในลักษณะนี้ จึงมีชื่อตามลักษณะดังกล่าวว่า forward-propagation ขั้นต่อไปจะเป็นการคำนวณแบบ backward-propagation สำหรับขั้นตอนนี้จะเป็นการนำผลลัพธ์ที่ได้จากเอาต์พุตของโครงข่าย มาเปรียบเทียบกับเอาต์พุตเป้าหมาย

(target output) ค่าผิดพลาดที่ได้จากโครงข่ายจะถูกส่งถอยหลังกลับมายังชั้นเอาต์พุต และส่งต่อไปยังโหนดต่างๆของชั้นภายใน แต่ละโหนดจะได้รับค่าผิดพลาดเพียงบางส่วนขึ้นอยู่กับว่าโหนดหรือนิวรอลเซลล์นั้นเป็นตัวที่ส่งผลมากหรือน้อยไปสู่เอาต์พุตนั้น ขบวนการของการส่งค่าความผิดพลาดกลับมานั้นจะทำซ้ำกับชั้นถัดลงมาอีกจนกระทั่งทุกโหนดในโครงข่ายได้รับส่วนแบ่งค่าความผิดพลาดนั้น ขั้นตอนสุดท้ายจะนำค่าความผิดพลาดที่ได้มาใช้ในการปรับค่าน้ำหนัก ซึ่งค่าน้ำหนักจะเปลี่ยนไปมากหรือน้อยขึ้นอยู่กับปริมาณของค่าความผิดพลาดที่ได้รับ ขบวนการจะกลับไปทำซ้ำในขั้นตอนแรกจนกระทั่งค่าผิดพลาดที่ได้มีค่าน้อยกว่าค่าผิดพลาดต่ำสุดที่กำหนดไว้ ต่อไปเป็นรายละเอียดของการทำงานในแต่ละขั้นตอน

FORWARD-PROPAGATION

เมื่อข้อมูลอินพุตปรากฏบนชั้นอินพุต สัญญาณจากชั้นอินพุตทั้งหมดจะถูกส่งมาตามสายเชื่อมโยงไปยังนิวรอนเซลล์ในชั้นถัดมา โครงสร้างการคำนวณของ forward-propagation สำหรับแต่ละหน่วยของนิวรอลเซลล์ ดังภาพที่ 9

ภาพที่ 9



โครงสร้างการคำนวณของขั้นตอน forward-propagation

การหาผลรวมสัญญาณอินพุตของนิวรอลเซลล์ของชั้นที่ซ่อนอยู่ (hidden layer) ได้จากสมการ

$$net^h_{pj} = \sum_{i=1}^{n} w^h_{ji} x_{pi} + \theta^h_j \qquad [2.8]$$

เมื่อ h หมายถึง hidden layer , p หมายถึง pattern , net^h_{pj} เป็นผลรวมของนิวรอลเซลล์ที่โหนด j w^h_{ji} เป็นน้ำหนัก x_{pi} เป็นอินพุตที่ i ของโหนด j θ^h_j เป็นผลของ bias unit มีค่าเท่ากับ $b_j w^h_{j0}$ เมื่อ b เป็น bias unit ซึ่งจะมีค่าเป็น +1 w^h_{j0} เป็นน้ำหนักที่เชื่อมโยงระหว่าง bias unit กับโหนด j เอาต์พุตของนิวรอลเซลล์จะเป็นฟังก์ชันของ net^h_{pj} ซึ่งเป็น sigmoid ฟังก์ชันจะมีลักษณะคล้ายรูปตัว S สมการเอาต์พุตของนิวรอลเซลล์จะมีรูปแบบเป็น

$$i_{pj} = f^h_j(net^h_{pj}) \qquad [2.9]$$

$$f^h_j(net^h_{pj}) = \frac{1}{1+e^{-(net^h_{pj})}} \qquad [2.10]$$

เมื่อ i_{pj} เป็นเอาต์พุตของชั้นที่ซ่อนอยู่ เอาต์พุตที่ได้จะส่งต่อไปยังชั้นถัดไป จนกระทั่งไปถึงชั้นเอาต์พุต k ในทำนองเดียวกันการหาผลรวมที่ชั้นเอาต์พุต net^o_{pk} และสมการเอาต์พุตของนิวรอลเซลล์ o_{pk} เป็น

$$net^o_{pk} = \sum_{j=1}^{L} w^o_{kj} i_{pj} + \theta^o_k \qquad [2.11]$$

$$o_{pk} = f^o_k(net^o_{pj}) \qquad [2.12]$$

BACKWARD-PROPAGATION

ค่าผิดพลาดของโครงข่ายจะเป็นผลต่างระหว่างเอาต์พุตเป้าหมายกับเอาต์พุตของโครงข่าย $(t_{pk} - o_{pk})$ ค่าผิดพลาดที่ได้จะถูกส่งกลับเข้ามายังโครงข่ายในรูปของสัญญาณค่าผิดพลาด (error signal) การหาสัญญาณค่าผิดพลาดที่ชั้นเอาต์พุต δ^o_{pk} หาได้จากสมการ

$$\delta^{o}_{pk} = (t_{pk} - o_{pk})\, f^{o'}_{k}(net^{o}_{pj}) \qquad [2.13]$$

เนื่องจากเอาต์พุตของนิวรอลเซลล์จะเป็นฟังก์ชันของ sigmoid ฟังก์ชันและ $f^{o'}_{k}$ เป็นอนุพันธ์อันดับหนึ่งของ f^{o}_{k} จะได้ว่า[1] $f^{o'}_{k} = f^{o}_{k}(1 - f^{o}_{k})$ สมการ [2.13] จึงสามารถเขียนใหม่ได้เป็น

$$\delta^{o}_{pk} = (t_{pk} - o_{pk})\, o_{pk}(1 - o_{pk}) \qquad [2.14]$$

ค่าสัญญาณความผิดพลาดที่ได้จากชั้นเอาต์พุตจะส่งต่อไปยังโหนดต่างๆของชั้นภายใน แต่ละโหนดจะได้รับค่าผิดพลาดเพียงบางส่วนขึ้นอยู่กับว่าโหนดหรือนิวรอลเซลล์นั้นเป็นตัวที่ส่งผลมากหรือน้อยไปสู่เอาต์พุตนั้น สมการหาสัญญาณค่าผิดพลาดของชั้นที่ซ่อนอยู่ δ^{h}_{pj} จะเขียนได้เป็น

$$\delta^{h}_{pj} = f^{h'}_{j}(net^{h}_{pj}) \sum_{k} \delta^{o}_{pk} w^{o}_{kj} \qquad [2.15]$$

$$\delta^{h}_{pj} = i_{pj}(1 - i_{pj}) \sum_{k} \delta^{o}_{pk} w^{o}_{kj} \qquad [2.16]$$

การปรับน้ำหนัก

ถ้าหากมีข้อผิดพลาดเกิดขึ้น ค่าผิดพลาดที่คำนวณได้จะถูกนำมาใช้ในการปรับน้ำหนักเพื่อให้โครงข่ายได้ผลลัพธ์ที่ถูกต้อง การปรับน้ำหนักของชั้นเอาต์พุต (Output layer) คำนวณได้จากสมการ

$$w^{o}_{ji}(t+1) = w^{o}_{ji}(t) + \eta \delta^{o}_{pk} i_{pj} + \alpha\left[\Delta w^{h}_{ji}(old)\right] \qquad [2.17]$$

$w^{o}_{ji}(t+1)$ เป็นน้ำหนักของชั้นเอาต์พุตที่ถูกปรับใหม่ $w^{o}_{ji}(t)$ เป็นค่าน้ำหนักเดิม η เป็นอัตราการเรียนรู้ i_{pj} เป็นเอาต์พุตที่ได้จากสมการ [2.9] δ^{o}_{pk} เป็นสัญญาณค่าผิดพลาดของชั้นเอาต์พุตที่ได้จากสมการ [2.14] α คือโมเมนตัม[1] เป็นค่าที่ควบคุมปริมาณของ $\Delta w^{h}_{ji}(old)$ หรือปริมาณของการปรับน้ำหนักครั้งก่อน จะถูกเก็บไว้เพื่อนำมาใช้ในการปรับน้ำหนักในครั้งถัดไป ค่าของ α จะมีค่าอยู่ในระหว่างช่วง 0 ถึง 1 ถ้า α มีค่าเป็น 0 จะไม่มีการนำปริมาณของการปรับน้ำหนักครั้งก่อนมาคิดร่วมด้วย ปริมาณของ η และ α มักนิยมให้เท่ากับ 0.15 และ 0.75

ถ้าโครงข่ายมีขนาดใหญ่มาก จะกำหนดให้มีค่าเท่ากับ 0.04 และ 0.02 ค่าที่กำหนดมาให้นี้เป็นแนวทางที่ใช้ในตอนเริ่มต้นเท่านั้นสำหรับค่าที่เหมาะสมจะหาได้จากการทดลอง ส่วนการปรับน้ำหนักของชั้นที่ซ่อนอยู่ การทำงานจะเป็นเช่นเดียวกับสมการที่ [2.17] แต่ตัวแปรต่างๆ จะเป็นของชั้นที่ซ่อนอยู่ ซึ่งสามารถหาได้จาก

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \eta\delta_{pj}^{h}x_{pi} + \alpha\left[\Delta w_{ji}^{h}(old)\right] \qquad [2.18]$$

$w_{ji}^{h}(t+1)$ เป็นน้ำหนักของชั้นที่ซ่อนอยู่ที่ถูกปรับใหม่ $w_{ji}^{h}(t)$ เป็นค่าน้ำหนักเดิม x_{pi}เป็นเอาต์พุตที่ได้จากชั้นอินพุต (Input layer) ของ pattern p ตัวที่ i δ_{pj}^{h} เป็นสัญญาณค่าผิดพลาดของชั้นที่ซ่อนอยู่ที่ได้จากสมการ [2.16]

วัฏจักรของการคำนวณหาค่าความผิดพลาดและการปรับน้ำหนักจะคงดำเนินต่อไปจนกระทั่งค่าความผิดพลาดต่ำกว่าที่กำหนดไว้ค่าหนึ่งแล้วจึงหยุด การคำนวณหาความผิดพลาดรวมของ pattern สำหรับทุกโหนดเอาต์พุต หาได้จาก (total sum of square error)ซึ่งมีสมการเป็น

$$E_p = \frac{1}{2}\sum_{k=1}^{M}(t_{pk} - o_{pk})^2 \qquad [2.19]$$

เมื่อ E_p เป็นค่าผิดพลาดรวมของทุกโหนดของ pattern p k หมายถึงแต่ละโหนดในชั้นเอาต์พุต M เป็นจำนวนโหนดของชั้นเอาต์พุต t_{pk} เป็นเอาต์พุตเป้าหมาย o_{pk} เป็นเอาต์พุตที่ได้จากชั้นเอาต์พุตที่โหนด k

2.3.4 KOHONEN SELF-ORGANIZING MAPS

self-organizing Maps เป็นทฤษฎีถูกคิดขึ้นโดย Teuvo Kohonen เป็นโครงข่ายสองชั้นประกอบด้วยชั้นอินพุตและชั้นเอาต์พุต การทำงานของโครงข่ายมีลักษณะที่เรียกว่า topology preserving maps ซึ่งมีลักษณะโครงสร้างการทำงานที่เป็นกลุ่มหรือคลัสเตอร์ (cluster) คล้ายกับแบบอย่างการทำงานในสมองของมนุษย์ โครงสร้างของคลัสเตอร์จะถูกแทนด้วยอาร์เรย์หนึ่งหรือสองมิติ ซึ่งประกอบไปด้วยคลัสเตอร์ของอินพุตเวคเตอร์ และคลัสเตอร์ของเอาต์พุต หรือ

คลัสเตอร์เวคเตอร์ของน้ำหนัก การเรียนรู้ของโครงข่ายนั้นโครงข่ายจะนำคลัสเตอร์ของ อินพุตเวคเตอร์มาหาคลัสเตอร์เอาต์พุตที่มีผลตอบสนองมากที่สุด ซึ่งเอาต์พุตที่โหนดนี้มีชื่อว่า winner neuron คลัสเตอร์เอาต์พุตที่ถูกระบุด้วย winner neuron จะประกอบไปด้วยเวคเตอร์ของ น้ำหนักที่โหนด winner neuron เอง และเวคเตอร์ของน้ำหนักที่อยู่ในรัศมีบริเวณใกล้เคียง (neighborhood) ลักษณะการเรียนรู้ของโครงข่ายนั้นจะเห็นว่าโครงข่ายสามารถหาคำตอบได้เองโดยอัลกอริทึมที่ใช้ในการเรียนรู้ ลักษณะการเรียนรู้เช่นนี้เรียกว่าการเรียนรู้แบบไม่มีผู้ช่วย หรือ Unsupervised learning อินพุตเวคเตอร์และเวคเตอร์ของน้ำหนักที่ได้จากการสุ่ม ก่อนที่จะนำมาใช้นั้นจะต้องทำ normalized[1] เวคเตอร์นั้นเสียก่อน เพราะในขั้นตอนการเรียนรู้นั้นจะต้องนำอินพุตเวคเตอร์ไปลบกับเวคเตอร์ของน้ำหนัก เมื่อเวคเตอร์ $A = a_1 + a_2 + a_3 + ... + a_n$ การหา normalized A' ได้จากสมการ

$$A' = \frac{A}{\left(\sum_{i=0}^{n} (a_i)^2 \right)^{\frac{1}{2}}} \qquad [2.20]$$

เมื่ออินพุตเวคเตอร์ปรากฏที่ชั้นอินพุต โครงข่าย Kohonen จะคำนวณหาผลลัพธ์ที่มีการตอบสนองต่ออินพุตคลัสเตอร์นั้นมากที่สุดเพื่อหา Winner neuron การหาผลรวมของนิวรอลเซลล์ที่โหนดเอาต์พุตหาได้จาก สมการ

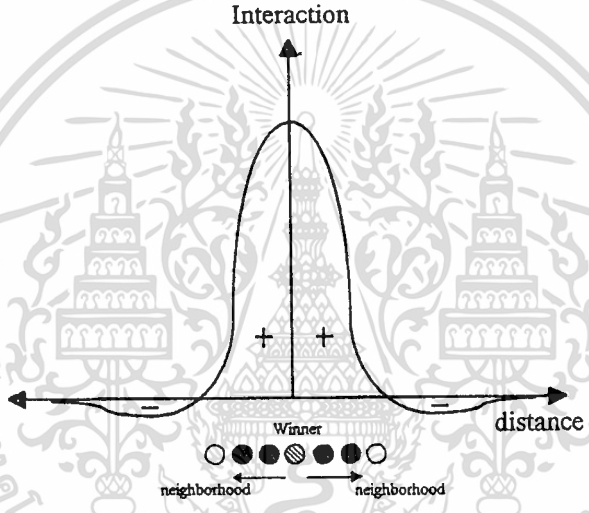
$$net_{pj} = \sum_{i=1}^{n} w_{ji} x_{pi} \qquad [2.21]$$

เมื่อ p หมายถึง pattern , net_{pj} เป็นผลรวมของนิวรอลเซลล์ที่โหนด j w_{ji} เป็นน้ำหนัก x_{pi} เป็นอินพุตที่ i ของโหนด j ผลที่ได้จากการคำนวณเป็น dot product ของผลรวมในการคูณระหว่างอินพุตเวคเตอร์กับเวคเตอร์น้ำหนัก

เอาต์พุตของสมการจะอยู่ในรูปของ Mexican-hat ฟังก์ชัน ซึ่งแสดงให้เห็นถึงความสัมพันธ์ระหว่างความเข้มของการกระตุ้นกับระยะทางจาก Winner neuron ซึ่งเป็นโหนดเอาต์พุตที่มีค่ามากที่สุด ล้อมรอบด้วยรัศมีบริเวณนิวรอนใกล้เคียง (neighborhood) เอาต์พุตที่

โหนด Winner neuron นี้เท่านั้นจะมีค่าเป็น 1 นอกนั้นจะมีค่าเป็น 0 ผลของฟังก์ชันนี้จะนำน้ำหนักของ Winner neuron และ neighborhood เท่านั้นไปใช้ในสมการเรียนรู้ของโครงข่าย

ภาพที่ 10



ลักษณะของ Mexican-hat ฟังก์ชัน

อีกวิธีหนึ่งในการหา Winner neuron คือการหา Euclidean distance ระหว่างอินพุตเวคเตอร์กับเวคเตอร์น้ำหนักที่สัมพันธ์กันกับโหนดเอาต์พุต j โหนดเอาต์พุตที่มี Euclidean distance น้อยที่สุดจะเป็น Winner neuron สมการหา Euclidean distance จะมีสมการเป็น

$$d_j = \sum_{i=0}^{n} \left(x_i - w_{ji} \right)^2 \qquad [2.22]$$

เมื่อ d_j เป็นระยะทางจากนิวรอนโหนด j ของ Pattern p x_i เป็นอินพุตเวคเตอร์ตัวที่ i และ w_{ji} เป็นน้ำหนักของโหนด j ของอินพุต i

เมื่อโครงข่ายหา Winner neuron ได้แล้วการเรียนรู้ของโครงข่ายจะปรับน้ำหนักเฉพาะในคลัสเตอร์เอาต์พุต ที่มี Winner neuron เป็นจุดศูนย์กลางและล้อมรอบด้วย neighborhood เท่านั้น เมื่อ c เป็นโหนดของ Winner neuron และ N_c เป็นโหนดที่อยู่ในรัศมีของ neighborhood สมการการปรับน้ำหนักจะมีสมการเป็น

เมื่อโหนด $j \in N_c$

$$w_{ji}(t+1) = w_{ji}(t) + \eta(t)\left(x_i - w_{ji}(t)\right)$$

$$= \eta x + (1-\eta)w_{ji}(t) \qquad [2.23]$$

$w_{ji}(t+1)$ เป็นน้ำหนักของชั้นเอาต์พุตที่ถูกปรับใหม่ $w_{ji}(t)$ เป็นค่าน้ำหนักเดิม $\eta(t)$ เป็นอัตราการเรียนรู้ที่วงรอบการเรียนรู้ t อัตราการเรียนรู้จะลดค่าลงในขณะที่โครงข่ายกำลังเรียนรู้ เช่นเดียวกับขนาด neighborhood จะมีลดลง และอาจลดลงจนรัศมีเท่ากับศูนย์ หรือเหลือเพียง Winner neuron เท่านั้น การเรียนรู้ของโครงข่ายจะดำเนินไปจนกระทั่งวงรอบการเรียนรู้เท่ากับจำนวนรอบที่กำหนดไว้หรือ อัตราการเรียนรู้มีค่าลดลงจนมีค่าเท่ากับ 0

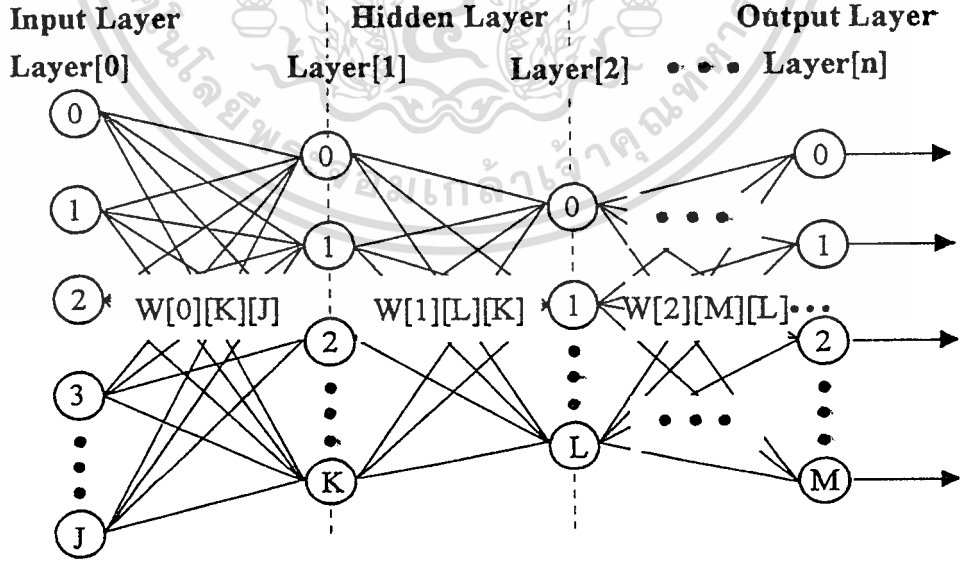
บทที่ 3

โครงสร้างข้อมูล

3.1 โครงสร้างข้อมูลทั่วไปของโครงข่ายประสาทเทียม

โครงสร้างทั่วไปของโครงข่ายประสาทเทียมจะประกอบด้วย ชั้นแรกเป็นชั้นอินพุต ถัดมาเป็นชั้นที่ซ่อนอยู่ (hidden layer) ซึ่งอาจจะมีได้หลายชั้น ในชั้นสุดท้ายเป็นชั้นเอาต์พุตจะเก็บผลลัพธ์ที่ได้จากการคำนวณของโครงข่าย ลักษณะโครงสร้างทั่วไปของโครงข่ายประสาทเทียมแสดงไว้ในภาพที่ 11

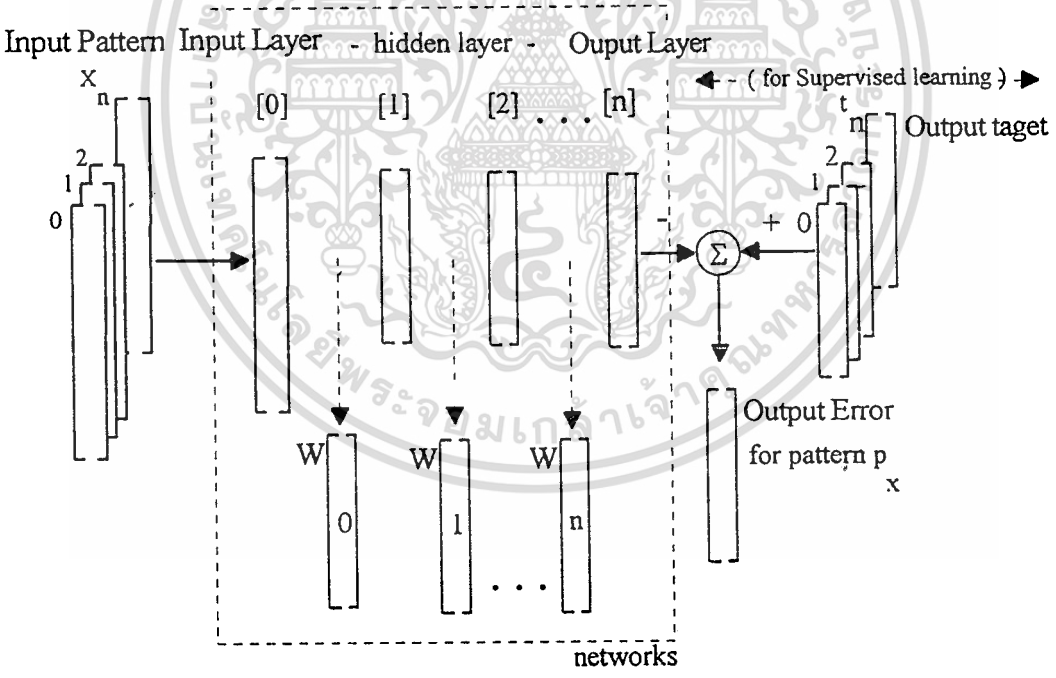
ภาพที่ 11



โครงสร้างทั่วไปของโครงข่ายประสาทเทียม

จากภาพที่ 11 สามารถนำมาเขียนเป็นลักษณะของโครงสร้างข้อมูลของโครงข่ายได้ดังภาพที่12 ทางด้านซ้ายมือเป็นอาร์เรย์ของชุดของข้อมูล Pattern ซึ่งจะถูกนำเข้าสู่โครงข่ายครั้งละหนึ่งชุด โครงสร้างข้อมูลของโครงข่าย(ภายในเส้นประ)จะประกอบด้วย ชั้นแรกเป็นอาร์เรย์ของชั้นอินพุต ถัดมาเป็นอาร์เรย์ของเอาต์พุตที่ได้จากนิวรอลเซลล์ซึ่งจะมีได้หลายชั้น ในระหว่างชั้นจะเก็บน้ำหนักของจุดเชื่อมโยงเป็นอาร์เรย์แบบสามมิติ และชั้นสุดท้ายจะเป็นอาร์เรย์ของชั้นเอาต์พุต ถ้าเป็นโครงข่ายแบบ Supervised learning ทางด้านขวาสุดเป็นอาร์เรย์ของเอาต์พุตเป้าหมายจะได้มาพร้อมกับชุดข้อมูล ซึ่งจะนำมาเปรียบเทียบกับชั้นเอาต์พุตเพื่อหาค่าผิดพลาดของโครงข่าย และจะเก็บไว้ในอาร์เรย์ของค่าผิดพลาด ถ้าโครงข่ายเป็นแบบ Unsupervised learning จะไม่ใช้ส่วนนี้

ภาพที่ 12



โครงสร้างข้อมูลของโครงข่ายประสาทเทียม

โครงสร้างข้อมูลอินพุตถูกจัดเก็บเป็นลักษณะของคู่ข้อมูล อินพุต-เอาต์พุต โดยจะประกอบไปด้วย ความกว้างและความยาวของ Pattern จำนวน Pattern ขนาดของชั้นเอาต์พุต เอาต์พุตเป้าหมายและท้ายสุดตามด้วยข้อมูลของ Pattern ไฟล์ข้อมูลจะมีนามสกุลเป็น .PAT รายละเอียดของไฟล์ข้อมูลอินพุตและการจัดเตรียมข้อมูลอินพุตจะอยู่ในภาคผนวก ก ข้อมูลของ Pattern หลายๆ Pattern จะถูกจัดรวมเป็นกลุ่มที่มีลักษณะของ Pattern ประเภทเดียวกันซึ่งผู้ใช้สามารถจัดกลุ่มได้เอง เช่นกลุ่มของตัวเลขอารบิค กลุ่มของตัวเลขไทย หรือกลุ่มของตัวอักษรไทย เป็นต้น โดยชนิดของไฟล์จะเป็น Text File ที่มีนามสกุลเป็น .GRP ซึ่งจะประกอบไปด้วยส่วนหัว (Header) เป็น "group" จำนวน Pattern ทั้งหมด ตามด้วยชื่อของไฟล์ *.PAT ตัวอย่างกลุ่มของ Pattern ตัวเลขไทยที่มีชื่อไฟล์เป็น TDIGIT.GRP

```
group
10
TDIGIT0.PAT
TDIGIT1.PAT
TDIGIT2.PAT
TDIGIT3.PAT
TDIGIT4.PAT
TDIGIT5.PAT
TDIGIT6.PAT
TDIGIT7.PAT
TDIGIT8.PAT
TDIGIT9.PAT
```

เมื่อโปรแกรมอ่านชุดข้อมูลของ Pattern โปรแกรมจะกำหนดจำนวนโหนดของชั้นอินพุตเท่ากับขนาดของ Pattern และถ้าโครงข่ายเป็นแบบ Supervised learning จะกำหนดโหนดของชั้นเอาต์พุตเท่ากับจำนวนเอาต์พุตเป้าหมาย แต่ถ้าโครงข่ายเป็นแบบ Unsupervised learning ผู้ใช้จะเป็นผู้กำหนดขนาดเอาต์พุตเอง โครงข่ายจะมองข้อมูลของจุดภาพที่มีค่าเป็น 1 เป็นอินพุตที่ก่อให้เกิดการกระตุ้นหรือ +1 และมองจุดภาพที่มีค่าเป็น 0 เป็นอินพุตที่ก่อให้เกิดการยับยั้งหรือมีค่าเป็น -1 ตัวอย่างลักษณะการมองจุดภาพของโครงข่ายแสดงไว้ดังภาพที่ 13

ภาพที่ 13

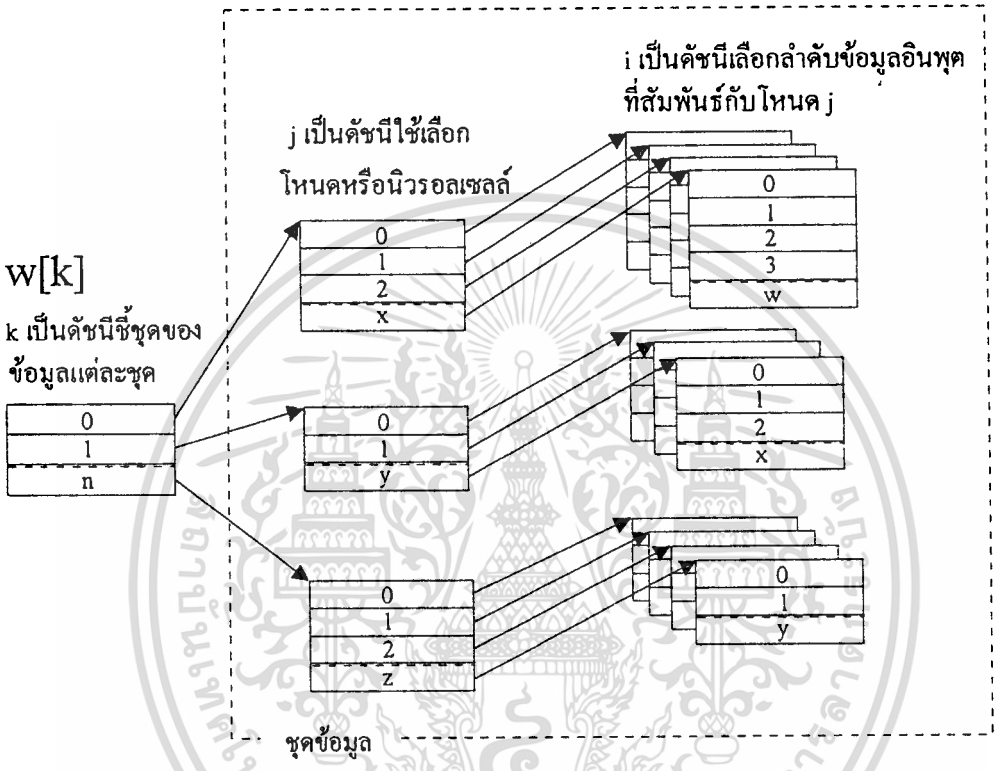
ลักษณะการมองจุดภาพของโครงข่าย

ลักษณะการแปลงจุดภาพให้เป็น bipolar {-1,+1} หรือ binary {0,1} ผู้ใช้สามารถทดลองเปลี่ยนแปลงได้ถ้าไม่ขัดกับกฎของอัลกอริทึมที่เลือกใช้ รายละเอียดจะอยู่ในบทที่ 4

โครงสร้างข้อมูลของน้ำหนัก

ในการใช้งานผู้ใช้โปรแกรมสามารถกำหนดและเปลี่ยนแปลงจำนวนชั้นของโครงข่าย จำนวนเอาต์พุตของนิวรอลเซลล์ และจำนวนน้ำหนักในแต่ละชั้น ซึ่งในแต่ละชั้นอาจมีขนาดที่ไม่เท่ากันได้ ดังนั้นจึงไม่สามารถจองหน่วยความจำแบบอาร์เรย์ปกติ แต่จะเป็นโครงสร้างแบบอาร์เรย์ของพอยน์เตอร์ ซึ่งจะใช้พอยน์เตอร์ชี้ชุดข้อมูลอีกทีหนึ่ง การใช้ w[k][j][i] โดย k แทนชั้นของนิวรอล j กำหนดแต่ละเซลล์และ i เป็นอินพุตที่ i ของโหนด j ดังนั้นเราจึงสามารถใช้วิธีจองหน่วยความจำโดยใช้อาร์เรย์ที่มีขนาดไม่เท่ากันได้ จากภาพที่ 14 ชั้นแรกจะเป็นอาเรย์ของพอยเตอร์ n ตัว ชี้ไปที่ชั้น n ชั้นที่สองจะกำหนดจำนวนโหนดในแต่ละชั้นโดยมีพอยเตอร์ชี้ไปยังชั้นสุดท้ายในชั้นที่ 3 จะเป็นชั้นของเลขทศนิยมที่แทนค่าของน้ำหนัก

ภาพที่ 14



โครงสร้างข้อมูลของน้ำหนักโดยใช้อาร์เรย์ของพอย์นเตอร์ชี้ข้อมูลแต่ละชุด

บทที่ 4

ระบบและการใช้งานโปรแกรม

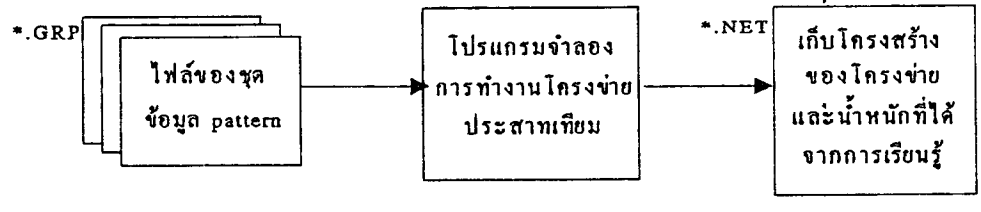
4.1 ระบบที่ใช้

โปรแกรมจำลองการทำงานของโครงข่ายประสาทเทียมถูกพัฒนาขึ้นด้วยภาษา C ทำงานบนสภาวะระบบปฏิบัติการดอส (DOS Environment) โดยสามารถใช้กับเครื่องคอมพิวเตอร์ส่วนบุคคลในตระกูล IBM ทั่วไป โดยมีอุปกรณ์ที่ต้องการสำหรับระบบดังต่อไปนี้

- เครื่องคอมพิวเตอร์ตระกูล IBM ที่ใช้หน่วยประมวลผลกลาง (CPU) 80386 ขึ้นไป
- หน่วยความจำหลักไม่น้อยกว่า 4 Mbytes
- ระบบจัดการหน่วยความจำยืดขยาย (Extended Memory Management ,XMS)
- ฮาร์ดิสก์จำนวน 1 ตัวโดยมีเนื้อที่ว่างอย่างน้อย 5 Mbytes
- เมาส์ พร้อมไดร์เวอร์ Microsoft Mouse Version 8.02 ขึ้นไป

ลักษณะการทำงานของโปรแกรมประกอบด้วยการนำเข้าชุดข้อมูล pattern ที่จะใช้ฝึกฝนโครงข่าย โดยเมื่อผู้ใช้ทำการฝึกฝนโครงข่ายแล้ว โปรแกรมจะสร้างไฟล์ที่มีนามสกุล *.NET เพื่อเก็บโครงสร้างของโครงข่ายและน้ำหนัก ซึ่งเป็นประสบการณ์ที่ได้จากการเรียนรู้

ภาพที่ 15



ลักษณะการทำงานของโปรแกรม

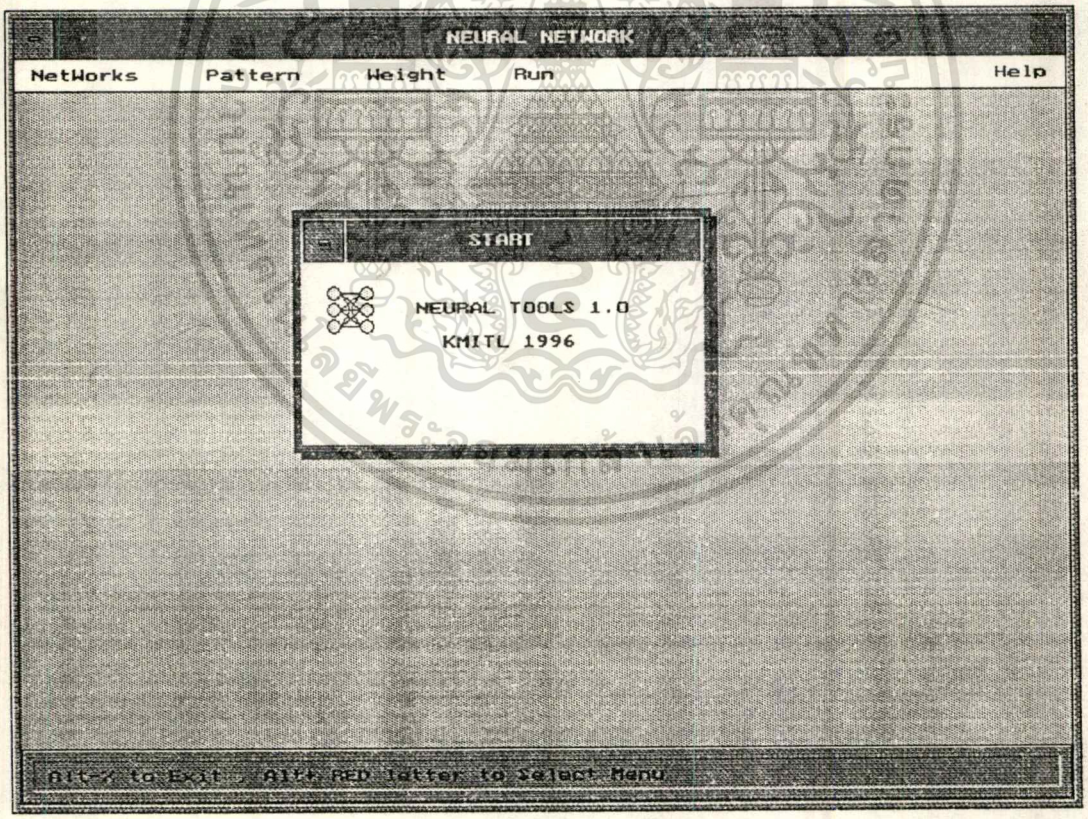
4.2 ขั้นตอนการใช้งานโปรแกรม

4.2.1 การเรียกโปรแกรมทำงาน เมื่ออยู่ใน DOS PROMPT "C:\ >" โดยพิมพ์

C:\NNT <ENTER>

จากนั้นจะปรากฏภาพหน้าจอที่เป็นการทำงานแบบพูลดาวน์เมนู ซึ่งพร้อมที่จะทำงานในขั้นตอนต่อไป

ภาพที่ 16

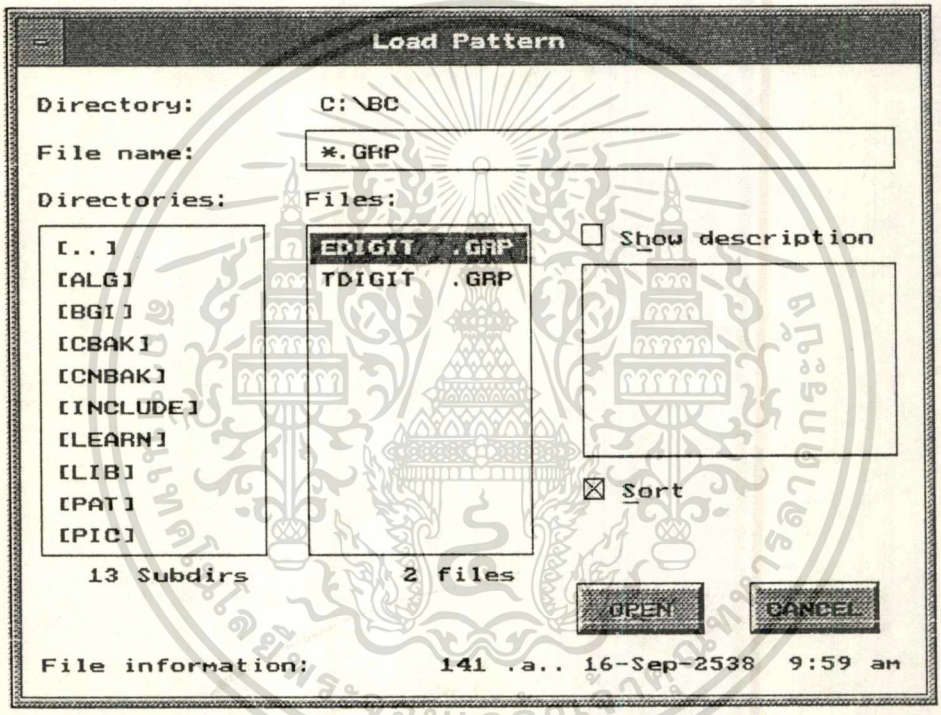


แสดงภาพหน้าจอของโปรแกรมที่พัฒนาขึ้น

4.2.2 การนำเข้าชุดข้อมูล pattern

เลือกชุดข้อมูลที่มีนามสกุล *.GRP โดยเลือกหัวข้อ PATTERN ในพูลดาวน์เมนูแล้วเลือกรายการ LOAD PATTERN จะปรากฏหน้าต่างโต้ตอบดังภาพ

ภาพที่ 17



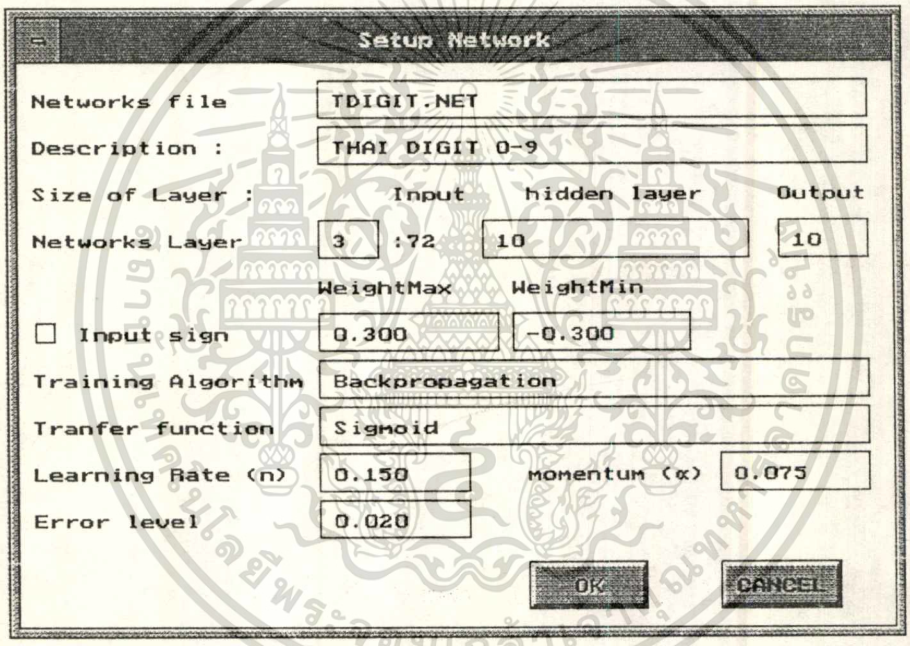
การเปิดไฟล์ของชุดข้อมูล pattern เพื่อนำข้อมูลเข้า

ให้พิมพ์ชื่อชุดข้อมูลที่จะนำเข้าในกรอบ File name: หรือใช้ TAB เลื่อนแถบสว่างไปที่ช่องหน้าต่างย่อย Files: ใช้คีย์ลูกศรเลื่อนแถบสว่างขึ้น-ลง เลือกชื่อของชุดข้อมูลแล้วกด ENTER ถ้าหากชุดข้อมูลอยู่ในไดเรคเทอรี ย่อยอื่น ให้ใช้ TAB เลื่อนไปที่ช่องหน้าต่างย่อย Directoties: เพื่อเปลี่ยนไปไดเรคเทอรีนั้นก่อน แล้วจึงเลือกชื่อของชุดข้อมูล การสับเปลี่ยนไป-มา ระหว่างช่องหน้าต่างย่อย สามารถกระทำได้ผ่านทางคีย์บอร์ดโดยกด TAB และ Shift - TAB หรือใช้เมาส์โดยกดปุ่มซ้ายมือ คำอธิบายรายละเอียดของแต่ละช่องหน้าต่างย่อย จะปรากฏอยู่ในกรอบด้านล่างของ Main Menu เพื่อความสะดวกในการใช้งาน

4.2.3 การกำหนดโครงสร้างของโครงข่าย (Setup Networks)

หลังจากการนำเข้าข้อมูล pattern แล้วจะได้การกำหนดโครงสร้างบางส่วนจากข้อมูล เช่น ขนาดของชั้นอินพุตได้จากขนาดข้อมูล pattern แต่ละตัว ขนาดของชั้นเอาต์พุตได้จากเอาต์พุตเป้าหมาย ข้อกำหนดส่วนที่เหลือผู้ใช้สามารถกำหนดได้โดยการเลือกจากหัวข้อ NETWORK เลือกรายการ Setup Network จะปรากฏหน้าต่าง Setup ดังภาพ

ภาพที่ 18



การ Setup Networks

หน้าต่าง Setup จะแสดงขนาดของอินพุต และเอาต์พุต ที่ได้จากการนำเข้าชุดข้อมูลในขั้นต้น และได้กำหนดค่าต่างๆ ไว้ล่วงหน้าบ้างแล้ว (default) ผู้ใช้สามารถกำหนดเองได้ใหม่ ดังรายละเอียดดังต่อไปนี้

Network file

เป็นชื่อของโครงข่ายนี้และมีนามสกุลเป็น *.NET เช่น TDIGIT.NET

Description

เป็นคำอธิบายสั้นๆ ของโครงข่ายที่ผู้ใช้สามารถใช้เป็นบันทึกช่วยจำ มีขนาดไม่เกิน 30 ตัวอักษร

Networks Layer

เป็นจำนวนชั้นของโครงข่าย โดยชั้นแรกและชั้นสุดท้าย จะเป็นชั้นอินพุตและเอาต์พุตซึ่งจะถูกกำหนดขนาดโดยข้อมูล pattern ที่ได้จากการนำเข้าชุดข้อมูลในขั้นต้นผู้ใช้ไม่สามารถกำหนดใหม่ได้ นอกจากจะนำเข้าข้อมูลมาใหม่ตามขนาดที่ได้สร้างไว้ในชุดข้อมูล แต่ผู้ใช้สามารถกำหนดจำนวนชั้นของโครงข่ายได้ โดยขนาดที่กำหนดจะรวมถึงชั้นอินพุต-เอาต์พุตและชั้นที่ซ่อนอยู่ด้วย เช่นกำหนดจำนวนชั้นของโครงข่ายเท่ากับ 5 ชั้นเมื่อลบด้วยชั้นอินพุตและเอาต์พุตรวม 2 ชั้นแล้วจะได้ชั้นที่ซ่อนอยู่ 3 ชั้น สำหรับจำนวนชั้นนั้นขึ้นอยู่กับอัลกอริทึมที่ใช้ด้วย เช่นถ้าเป็น Back-Propagation โปรแกรมจะไม่ยอมให้กำหนดต่ำกว่า 3 ชั้น

hidden layer

เป็นการกำหนดจำนวนโหนดหรือนิวรอลเซลล์ของชั้นที่ซ่อนอยู่ในแต่ละชั้น โดยเว้นช่องว่าง (space) สำหรับในแต่ละชั้น ตัวอย่างเช่นเมื่อกำหนดจำนวนชั้นของโครงข่ายเท่ากับ 5 ชั้น ดังนั้นจะมีชั้นที่ซ่อนอยู่เท่ากับ 3 ชั้น (5-2 ลบด้วยชั้นอินพุตและเอาต์พุต) เมื่อกำหนดเป็น 20 10 5 (เว้นด้วยช่องว่าง) จะมีความหมายว่าชั้นที่ซ่อนอยู่มีขนาด 20 , 10 และ 5 โหนด ตามลำดับ โปรแกรมจะตัวสอบความถูกต้องด้วย ถ้าผู้ใช้ป้อนจำนวนชั้นไม่ครบหรือเกินโปรแกรมจะกลับมาให้พิมพ์ใหม่

☐ Input Sign

เป็นตัวเลือกชนิด check box ปกติจะถูกกำหนดโดยอัลกอริทึม แต่ผู้ใช้สามารถใช้ในการเลือกวิธีการอ่านค่าอินพุต เมื่อมีค่าเป็น ☒ (ON) โปรแกรมจะมองอินพุตเป็น bipolar คือ {-1,+1} ถ้ามีค่าเป็น ☐ (OFF) โปรแกรมจะมองอินพุตเป็น binary {0,1} โปรแกรมจะเปลี่ยนวิธีตามที่ผู้ใช้กำหนด ถ้าไม่ขัดกับกฎของอัลกอริทึม เช่น ADALINE ไม่สามารถเลือกเป็น OFF ได้ เพราะอัลกอริทึมให้ใช้อินพุตเป็น bipolar

WeightMax WeightMin

ผู้ใช้สามารถเลือกช่วงในการสุ่มค่าน้ำหนักเมื่อตอนเริ่มต้นได้ ในทางปฏิบัติการเริ่มต้นสุ่มค่าน้ำหนัก จะให้อยู่ในช่วง -0.3 ถึง +0.3

Training Algorithm

เลือกอัลกอลิทึมที่ใช้ในการเรียนรู้ ซึ่งโปรแกรมจะเลือกฟังก์ชันที่ใช้ในการหาค่าพิกัดให้โดยอัตโนมัติ

Learning Rate (η)

อัตราการเรียนรู้ของโครงข่าย มักจะกำหนดให้มีปริมาณค่าน้อยๆ โดยทั่วไปมักนิยมกำหนดให้เท่ากับ 0.15 ผู้ใช้สามารถเปลี่ยนแปลงค่าให้มากขึ้นหรือน้อยลงได้ และสังเกตความเร็วในการเรียนรู้ที่เปลี่ยนไป ถ้าผู้ใช้กำหนดให้มีค่ามากเกินไป กราฟแสดงค่าผิดพลาดรวมของโครงข่ายจะสลับขึ้นๆลงๆกลับไปกลับมา (Oscillate) ทำให้โครงข่ายไม่สามารถเรียนรู้ได้

momentum (α)

ตัวแปรนี้จะมีให้เลือกในอัลกอลิทึม Back-Propagation เท่านั้น โดยจะมีค่าเริ่มต้นประมาณ 0.75 ใช้ควบคุม การปรับน้ำหนักโดยร่วมกับค่า $\Delta w(old)$

Error Level

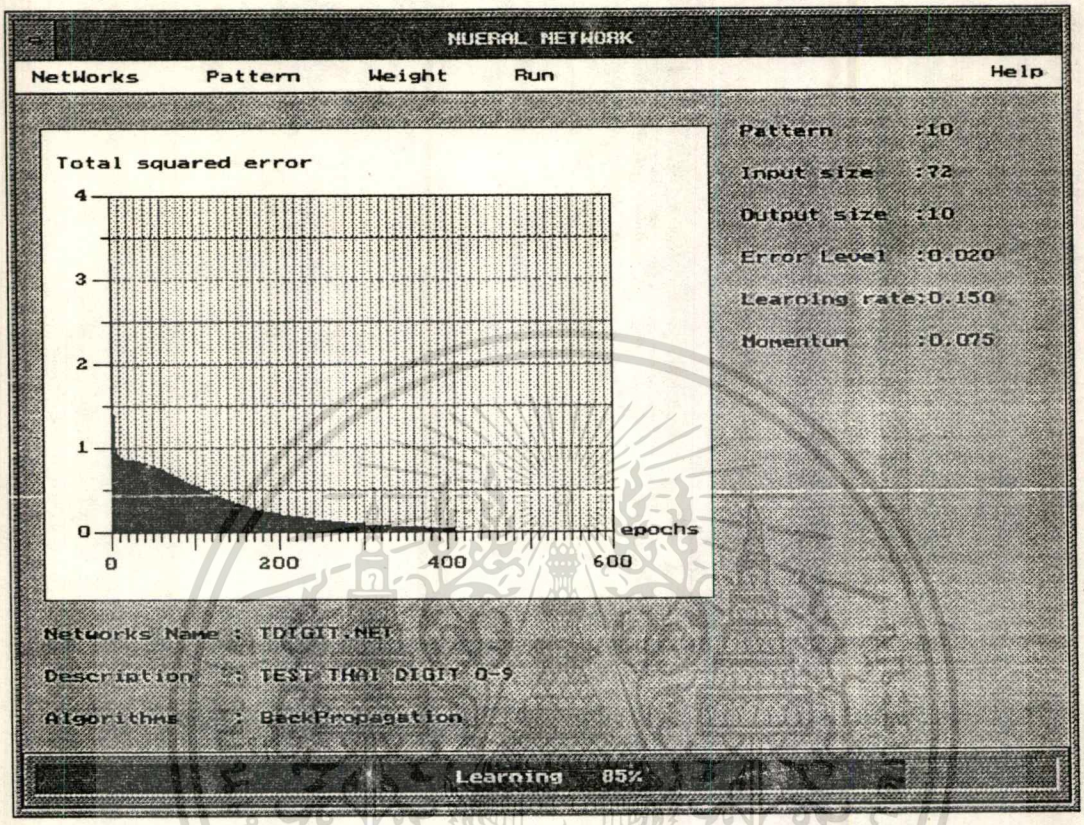
เป็นค่าผิดพลาดต่ำสุดที่กำหนดไว้ในการเรียนรู้ หากค่าผิดพลาดรวมของโครงข่ายต่ำกว่าหรือเท่ากับค่านี้จะทำให้โครงข่ายยุติการทำงาน

เมื่อเริ่มต้นใช้งาน ถ้าต้องการใช้ค่าที่โปรแกรมกำหนดให้ สามารถข้ามขั้นตอนการ Setup นี้ได้

4.2.4 การฝึกฝนโครงข่าย (Training networks)

เลือกหัวข้อ RUN ในพูลดาวน์เมนู เลือกรายการ Training โปรแกรมจะแสดงรายละเอียดต่างๆ ที่ได้กำหนดไว้ แล้วจึงเริ่มทำการฝึกฝนโครงข่าย ภาพหน้าจอของขั้นตอนการฝึกฝนโครงข่ายจะมีลักษณะดังภาพต่อไปนี้

ภาพที่ 19

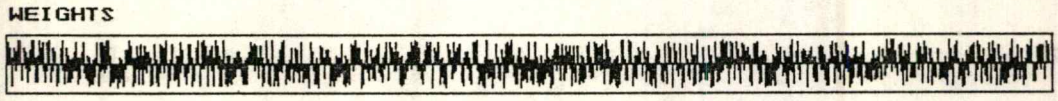


การฝึกฝนโครงข่าย

ขณะที่โปรแกรมกำลังทำงานอยู่นั้น จะแสดงกราฟของค่าผิดพลาดกำลังสองรวม (total sum of squared error) ของโครงข่าย สำหรับทุก pattern ที่ใช้ในการสอน ส่วนทางด้านล่างจะแสดงปริมาณการเปลี่ยนแปลงของค่าผิดพลาด คิดเป็นเปอร์เซนต์เทียบกับค่าผิดพลาดที่เกิดขึ้นครั้งแรกกับค่าผิดพลาดต่ำสุดที่กำหนดไว้ เมื่อค่าผิดพลาดที่คำนวณได้มีค่าน้อยกว่าหรือเท่ากับค่าผิดพลาดต่ำสุดที่กำหนดไว้ โปรแกรมจะหยุดทำงาน และจะแสดงค่าผิดพลาดครั้งหลังสุดนั้นออกมา พร้อมทั้งแสดงจำนวนรอบที่ใช้ในการปรับน้ำหนัก และเวลาที่ใช้ในการคำนวณ เพื่อใช้ในการเปรียบเทียบหาข้อแตกต่างในการออกแบบโครงข่าย

เมื่อผู้ใช้เริ่มฝึกฝนโครงข่าย โปรแกรมจะทำการสุ่มค่าน้ำหนักในตอนเริ่มต้นให้โดยอัตโนมัติ แต่ผู้ใช้ก็สามารถเลือกสุ่มค่าน้ำหนักได้เอง โดยเลือกหัวข้อ Weight และเลือกรายการ Random หรือกด F6 โปรแกรมจะแสดงกราฟแบบ histogram ของน้ำหนักให้เห็นถึงขนาดและลักษณะการกระจายของน้ำหนัก โดยสามารถกำหนดช่วงของการสุ่มได้จากหัวข้อ 4.2.3

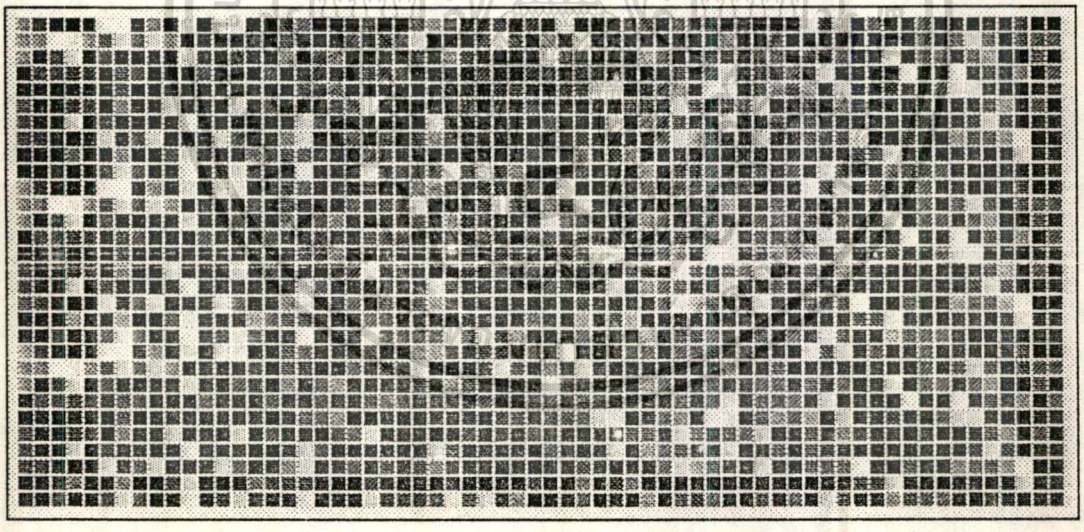
ภาพที่ 20



แสดงกราฟ histogram ของน้ำหนักเมื่อผู้ใช้เลือกสุ่มค่าน้ำหนัก

โปรแกรมสามารถแสดงปริมาณค่าน้ำหนัก ในลักษณะแทนด้วยความแตกต่างของเฉดสี เพื่อให้ผู้ใช้สังเกตได้ง่ายขึ้น โดยกดคีย์ Alt พร้อมกับหมายเลขของชั้นที่ต้องการดู ตัวเลขของชั้นจะเริ่มตั้งแต่ 0 ถึง n โดยน้ำหนักของโหนดทางด้านอินพุตจะอยู่ในแนวแกน X และลำดับของนิวรอลเซลล์ในชั้นนั้นจะอยู่ในแนวแกน Y ตัวอย่างของการแสดงน้ำหนักในชั้นแรกของโครงข่าย (ชั้นที่ 0) โดยกดคีย์ ALT-0 แสดงไว้ในภาพที่ 21

ภาพที่ 21



การแสดงน้ำหนักด้วยลำดับของสี

4.2.5 การให้โปรแกรมทำงานทีละขั้น

ผู้ใช้สามารถให้โปรแกรมทำงานทีละขั้นได้ เมื่อนำชุดข้อมูลเข้าและกำหนดค่าต่างๆ ตามขั้นตอนที่กล่าวมาข้างต้นเรียบร้อยแล้ว โดยเลือกหัวข้อ RUN และเลือกรายการ Make File จะสังเกตได้ว่า คำว่า OFF ทางขวามือจะเปลี่ยนเป็น ON เลือกหัวข้อ RUN อีกครั้งและเลือกรายการ Forward โปรแกรมจะทำการคำนวณ และแสดงการคำนวณออกมาบนจอภาพ

ภาพที่ 22

```
MakeOutFile                                          FORWARD.TMP
FEEDFORWARD
sum each layer sum(wx)
          layer0     layer1
node  0 -0.06297 -0.28306
node  1  2.65136  0.50576
node  2  1.64254 -0.13846
node  3 -1.47810  0.10712
node  4  0.05912  0.30427
node  5  0.36440 -0.73861
node  6  1.52149  0.23617
node  7  2.26664 -0.59680
node  8 -0.96551 -0.22767
node  9  1.83770 -0.67503

Pass Transfer funtion  LayerOut[k][j] = 1.0/(1.0+exp(-sum));
OUTPUT NEURON LAYER
          layer0     layer1
node  0  0.48426  0.42970
node  1  0.93409  0.62381
node  2  0.83788  0.46544
node  3  0.18571  0.52675
node  4  0.51478  0.57549
node  5  0.59011  0.32331
node  6  0.82076  0.55877
node  7  0.90608  0.35508
node  8  0.27578  0.44333
node  9  0.86268  0.33737

Output Error
 (taget  -  output)     Error      Error*(output*(1-output))
0.00000 -  0.42970 = -0.42970 -> -0.10530
0.00000 -  0.62381 = -0.62381 -> -0.14639
0.00000 -  0.46544 = -0.46544 -> -0.11580
0.00000 -  0.52675 = -0.52675 -> -0.13131
0.00000 -  0.57549 = -0.57549 -> -0.14059
0.00000 -  0.32331 = -0.32331 -> -0.07073
0.00000 -  0.55877 = -0.55877 -> -0.13776
0.00000 -  0.35508 = -0.35508 -> -0.08131
0.00000 -  0.44333 = -0.44333 -> -0.10941
1.00000 -  0.33737 =  0.66263 ->  0.14813

Sum Square Error =  1.28876
```

แสดงวิธีการคำนวณออกทางจอภาพในขั้นตอน Forward

ขั้นตอนต่อไปเป็นการทำงานแบบ Backward สำหรับขั้นตอนนี้ใช้ได้กับ Back-Propagation เท่านั้นเพราะเป็นการหาสัญญาณค่าความผิดพลาดของโครงข่ายแล้วแพร่กลับเข้าไปในโครงข่าย สำหรับอัลกอริทึมอื่นจะถูกข้ามไป เลือกหัวข้อ RUN และเลือกรายการ Backward โปรแกรมจะ

ทำการคำนวณหาสัญญาณค่าความผิดพลาดของโครงข่ายแล้วแพร่กลับเข้าไปในโครงข่าย การแสดงผลการคำนวณออกมาทางจอภาพโดยมีลักษณะต่อไปนี้

ภาพที่ 23

```
FeedBack...
Calculate LayerErr[0]
bettween Layer size 10 and Layer size 10
sum += LayerErr[1][0...10] * WEIGHTS[1][0...10][0]
 0.00000 +=  -0.10530  *  0.23566   = -0.02482
-0.02482 +=  -0.14639  *  -0.29239  =  0.01799
 0.01799 +=  -0.11580  *  -0.21823  =  0.04326
 0.04326 +=  -0.13131  *  -0.33140  =  0.08678
 0.08678 +=  -0.14059  *  -0.18494  =  0.11278
 0.11278 +=  -0.07073  *  -0.30454  =  0.13432
 0.13432 +=  -0.13776  *   0.26210  =  0.09821
 0.09821 +=  -0.08131  *  -0.27395  =  0.12049
 0.12049 +=  -0.10941  *  -0.30085  =  0.15340
 0.15340 +=   0.14813  *   0.29017  =  0.19638

*final sum of node0   0.19638
LayerErr[0][0] = sum * LayerOut[0][0]*(1.0-LayerOut[0][0])
 0.04905 =   0.19638 *   0.48426 *(1.0-  0.48426)

sum += LayerErr[1][0...10] * WEIGHTS[1][0...10][1]
 0.00000 +=  -0.10530  *  -0.17784  =  0.01873
 0.01873 +=  -0.14639  *  -0.28012  =  0.05973
 0.05973 +=  -0.11580  *  -0.22941  =  0.08630
 0.08630 +=  -0.13131  *   0.24883  =  0.05363
 0.05363 +=  -0.14059  *  -0.28163  =  0.09322
 0.09322 +=  -0.07073  *  -0.28350  =  0.11327
 0.11327 +=  -0.13776  *   0.27864  =  0.07489
 0.07489 +=  -0.08131  *  -0.24684  =  0.09496
 0.09496 +=  -0.10941  *  -0.26446  =  0.12389
 0.12389 +=   0.14813  *  -0.33236  =  0.07466

*final sum of node1   0.07466
LayerErr[0][1] = sum * LayerOut[0][1]*(1.0-LayerOut[0][1])
 0.00460 =   0.07466 *   0.93409 *(1.0-  0.93409)
 .
 .
 .

sum += LayerErr[1][0...10] * WEIGHTS[1][0...10][9]
 0.00000 +=  -0.10530  *   0.29467  = -0.03103
-0.03103 +=  -0.14639  *   0.19985  = -0.06028
-0.06028 +=  -0.11580  *   0.24602  = -0.08877
-0.08877 +=  -0.13131  *  -0.28332  = -0.05157
-0.05157 +=  -0.14059  *   0.19260  = -0.07865
-0.07865 +=  -0.07073  *  -0.16858  = -0.06673
-0.06673 +=  -0.13776  *  -0.32594  = -0.02182
-0.02182 +=  -0.08131  *   0.18959  = -0.03724
-0.03724 +=  -0.10941  *  -0.29881  = -0.00455
-0.00455 +=   0.14813  *  -0.32770  = -0.05309

*final sum of node9   -0.05309
LayerErr[0][9] = sum * LayerOut[0][9]*(1.0-LayerOut[0][9])
 -0.00629 =   -0.05309 *   0.86268 *(1.0-  0.86268)
```

แสดงวิธีการคำนวณออกทางจอภาพในขั้นตอน Backward

ในขั้นตอน Adaptive Weight เป็นขั้นตอนสุดท้ายของการทำงานเป็นขั้นของโปรแกรม เพื่อให้ผู้ใช้ดูการทำงานของโปรแกรมแบบตาราง เลือกหัวข้อ RUN และเลือกรายการ Adaptive Weight เนื่องจากตารางการปรับน้ำหนักมีความยาวมาก การแสดงในภาพต่อไปนี้จึงเป็นรูปแบบย่อ

ภาพที่ 24

```
ADAPTIVE WEIGHTS
Adaptive WEIGHTS[1][j][i] between (i) size 10 and (j) size 10
node j[0..9]
(j = 0) (i = [0..9])
        delta = LayerErr[1][0]*LayerOut[0][0]
      -0.05099 = -0.10530 *  0.48426
        dw   = (LearnConst * delta)  + (Alpha * deltaW[1][0][0]);
      -0.00765 = ( 0.15000 * -0.05099) + ( 0.07500 *  0.00000)
      WEIGHTS[1][0][0] += dw;
          0.23566 +=   -0.00765  =  0.22801
      deltaW [1][0][0]  =   -0.00765
                .
                .
                .
Adaptive WEIGHTS[0][j][i] between (i) size 72 and (j) size 10
node j[0..9]
(j = 0) (i = [0..71])
        delta   = LayerErr[0][0]*InputLayer[PatternNo][0]
      0.00000 =  0.04905 *  0.00000
        dw   = (LearnConst * delta)  + (Alpha * deltaW[0][0][0]);
      0.00000 = ( 0.15000 *  0.00000) + ( 0.07500 *  0.00000)
      WEIGHTS[0][0][0] += dw;
          -0.27995 +=   0.00000  =  -0.27995
      deltaW [0][0][0]  =   0.00000
                .
                .
                .
```

แสดงวิธีการคำนวณออกทางจอภาพในขั้นตอน Adaptive Weight

เมื่อผ่านขั้นตอนนี้แล้ว ผู้ใช้จะต้องกลับไปเลือก Forward , Backward และ Adaptive Weightใหม่ โดยจะข้ามขั้นตอนไม่ได้ โปรแกรมจะสร้างไฟล์ FORWARD.TMP, BACKWARD.TEMP และ ADAPTW.TMP ไว้เป็นที่เก็บข้อมูลของตาราง และจะเขียนทับไฟล์เดิมอีกเมื่อกลับมาที่ขั้นตอนแรกอีกครั้ง เมื่อต้องการยกเลิกการทำงานแบบทีละขั้น ให้เปลี่ยนในรายการ Make File ให้เป็น OFF แล้วเลือก Training เพื่อให้โปรแกรมทำงานในแบบปกติต่อไปได้

4.2.6 การจัดเก็บน้ำหนักแยกออกเป็นไฟล์

บางครั้งเมื่อต้องการทดสอบเวลาในการเรียนรู้ หรือจำนวนรอบในการปรับน้ำหนักของโครงข่ายจากตัวแปรที่ต่างกัน เช่นเปลี่ยนแปลงอัตราการเรียนรู้ η โมเมนตัม α หรือค่าผิดพลาดต่ำสุด จะต้องเก็บน้ำหนักชุดเดียวกันไว้ใช้ในการทดสอบ หรือหาข้อแตกต่าง ผู้ใช้สามารถแยกเก็บน้ำหนักออกมาเป็นไฟล์ เพื่อโหลดขึ้นมาใช้ใหม่ในภายหลังได้ แต่โครงข่ายต้องมีโครงสร้างเดียวกัน เช่นขนาดของอินพุต-เอาต์พุต และจำนวนชั้นของโครงข่ายจะต้องเหมือนกัน ดังนั้นหลังจากที่สั่งให้โปรแกรมสุ่มค่าน้ำหนักแล้ว จะต้องเก็บค่าน้ำหนักนั้นไว้ โดยการเลือกหัวข้อ Weight และเลือกรายการ Save Weight เมื่อปรากฏหน้าต่างในการบันทึกให้ตั้งชื่อของน้ำหนักโดยมีนามสกุลเป็น *.WHT เมื่อทำการทดสอบโครงข่ายแล้วให้จดบันทึกจำนวนรอบและเวลาที่ใช้เพื่อเก็บไว้เปรียบเทียบ แล้วจึงอ่านค่าน้ำหนักเดิมเข้าสู่โครงข่ายใหม่ จึงเริ่มทำการทดลองใหม่

4.2.7 การแสดงผลการคำนวณของน้ำหนัก

ในกรณีที่ต้องการให้โปรแกรมแสดงผลของน้ำหนัก การแสดงผลจะแสดงลำดับของจุดเชื่อมโยงเรียงไปตามแถว และแต่ละโหนดในชั้นนั้นในแนวคอลัมภ์ โดยจะแสดงน้ำหนักของ bias unit ของชั้นนั้นไว้ท้ายสุดของทุกชั้น การแสดงน้ำหนักสามารถทำได้โดยเลือกหัวข้อ Weight และเลือกรายการ Make Report โปรแกรมจะสร้างไฟล์ที่มีนามสกุล *.RPT ผู้ใช้สามารถใช้ Viewer ของโปรแกรมดูได้โดยกด F10 โปรแกรมจะแสดงน้ำหนักดังภาพ

ภาพที่ 25

WEIGHT (0) between input layer and neuron layer 1

	node0	node1	node2	node3	node4	node5	node6	node7	node8	node9
0	-0.521	-0.090	0.171	-0.401	-0.055	0.507	-0.001	-0.019	-0.060	0.426
1	0.510	-0.131	0.190	-0.182	-0.486	-0.553	0.511	0.089	-0.331	0.623
2	-0.504	-0.842	-0.665	-0.494	-0.584	0.258	-0.644	0.418	0.603	0.552
3	0.424	0.550	-0.279	0.368	0.307	-0.030	-0.452	0.109	0.250	0.387
4	-0.164	0.201	-0.567	0.077	-0.015	0.060	-0.156	-0.086	0.384	-0.324
5	0.199	-0.039	0.065	0.146	0.387	-0.009	0.013	0.587	0.491	0.017
6	0.705	-0.255	-0.447	0.590	0.310	-0.742	-0.320	0.430	-0.181	-0.062
7	-0.286	-0.167	-0.284	0.265	-0.242	-0.299	-0.206	0.193	0.275	-0.222
8	-0.006	0.438	0.268	0.112	-0.492	0.528	0.404	0.388	-0.541	0.013
9	0.275	-1.007	-1.066	0.273	-0.183	0.544	-0.324	-0.148	0.412	0.451
10	0.086	0.535	1.223	0.619	0.034	-1.167	-0.211	-0.430	-0.580	-0.428
11	0.223	0.507	0.130	0.123	-0.495	-0.348	-0.085	-0.560	-0.377	0.472
12	0.088	-0.440	0.454	-0.219	-0.336	0.054	0.367	-0.084	-0.100	0.087
13	0.087	0.113	0.352	0.335	0.185	-0.318	0.286	0.473	0.353	0.095
14	0.258	-0.405	-0.671	0.259	0.223	0.628	-0.230	-0.212	0.322	0.396
15	0.169	0.260	0.206	0.212	-0.265	-0.189	-0.298	0.200	-0.169	-0.325
16	-0.086	-0.144	-0.298	0.138	0.069	0.533	-0.022	0.393	-0.520	-0.102
17	1.012	0.626	-0.559	-0.688	-0.576	-0.354	0.351	0.315	-0.401	-0.522
18	-0.238	-0.288	0.179	0.271	0.274	-0.312	-0.322	-0.239	-0.178	0.292
19	-0.494	0.377	0.192	0.299	-0.187	-0.028	-0.603	-0.369	0.236	0.213

20	-0.181	-0.309	-0.220	-0.172	-0.261	0.222	0.214	0.224	-0.190	-0.267
21	-0.526	0.142	-0.085	0.281	0.124	-0.375	-0.099	0.471	0.444	0.092
22	0.261	-0.501	-0.601	0.196	0.276	0.665	-0.257	0.348	0.446	0.408
23	-0.197	0.285	-0.205	0.288	-0.295	0.186	0.327	0.325	0.197	0.222
24	-0.524	0.359	-0.221	0.091	-0.066	0.547	0.021	0.368	-0.015	0.425
25	1.362	0.238	-0.497	-0.261	-0.316	0.411	0.115	-0.024	-1.159	-0.756
26	-0.194	-0.262	-0.175	-0.293	-0.692	0.427	0.618	0.615	-0.361	0.234
27	-0.432	0.145	0.611	0.035	-0.173	0.255	-0.353	0.155	-0.743	0.630
28	0.124	0.103	0.052	0.045	-1.145	0.139	0.857	0.133	-0.697	1.001
29	0.154	-0.769	-0.326	0.494	-0.969	-0.774	0.751	0.273	0.285	0.750
30	-0.743	0.051	-0.445	-0.053	-0.358	1.016	-0.729	0.148	0.312	-0.277
31	0.191	-0.309	-0.187	-0.199	0.327	0.280	-0.185	0.170	0.246	0.270
32	-0.468	-0.115	-0.308	-0.462	-0.035	0.513	0.543	0.363	-0.054	0.413
33	0.444	-0.155	0.785	0.197	-0.387	-1.395	1.297	-0.181	-0.543	0.215
34	0.870	0.897	-0.300	-0.573	0.022	0.383	-0.105	0.060	-0.088	-1.613
35	0.662	-0.204	0.604	-0.479	-0.355	0.069	-0.543	-0.771	-0.186	0.497
36	0.037	-0.073	-0.248	-0.882	0.285	-0.303	-0.191	-0.523	-0.099	0.423
37	-0.007	-0.184	0.211	-0.357	0.482	-0.174	0.057	0.918	0.178	-1.070
38	0.284	-0.076	0.113	-0.086	-1.353	-0.572	-0.029	-0.241	0.393	0.390
39	-0.204	-0.185	0.290	0.321	0.311	-0.213	0.327	0.274	0.227	0.267
40	-0.470	0.147	0.378	0.360	-0.410	0.105	0.276	-0.081	0.437	-0.409
41	1.186	0.242	0.047	-0.281	-0.173	0.027	0.142	-0.513	-1.309	-1.079
42	-0.097	-0.068	-0.382	0.212	-0.294	-0.584	0.209	-0.012	0.073	0.784
43	0.460	0.352	0.378	0.089	-0.532	-0.804	0.253	-0.805	-0.097	0.436
44	-0.124	0.376	-0.599	0.323	0.715	-0.485	-0.314	0.123	0.381	-0.118
45	-0.502	0.120	0.440	-0.197	-0.386	-0.337	-0.114	0.360	-0.132	-0.036
46	-0.226	0.082	0.345	-0.999	0.002	0.535	0.562	0.217	0.206	0.108
47	0.199	0.296	-0.267	0.203	0.177	-0.330	-0.174	0.217	0.278	-0.238
48	0.044	0.105	0.469	0.309	0.112	-0.386	-0.145	0.418	0.338	-0.383
49	0.956	-0.436	0.221	-0.230	-0.284	-0.445	0.216	-0.359	-1.118	-1.198
50	0.048	0.021	-0.099	-0.152	-0.339	-0.450	0.315	0.508	0.474	-0.536
51	-0.342	0.484	0.483	1.041	-0.250	-0.434	-0.210	0.263	0.301	-0.812
52	-0.041	-0.479	0.261	0.083	-0.258	-0.808	0.322	0.579	0.590	-0.402
53	-0.126	0.463	0.398	-0.421	-0.456	0.049	-0.402	0.151	0.415	-0.421
54	-0.834	-0.915	0.435	-0.707	-0.166	-0.171	0.773	-0.390	-0.643	0.364
55	0.252	0.182	0.257	0.291	0.236	0.309	-0.237	0.283	-0.195	0.272
56	-0.424	0.310	0.272	0.068	-0.512	0.005	0.492	0.331	-0.507	0.434
57	-0.236	-0.980	0.282	-0.577	0.861	0.157	0.487	-0.263	0.036	-0.200
58	-0.655	0.773	-0.438	0.478	0.234	-0.037	0.159	-0.039	0.388	-0.772
59	0.728	0.093	0.009	0.979	-0.455	-0.884	-0.598	0.152	0.202	0.618
60	0.637	0.599	-0.202	-0.493	-0.753	0.237	-0.051	0.578	0.141	0.297
61	-0.707	-0.200	-0.193	0.238	-0.088	-0.231	-0.037	0.580	0.128	-0.164
62	0.122	-0.358	0.361	-0.612	0.856	0.275	0.566	-0.834	-0.065	0.291
63	-0.313	0.205	0.253	-0.183	-0.285	-0.264	-0.275	0.318	-0.218	0.222
64	-0.224	-0.310	-0.175	-0.325	0.321	-0.215	0.223	-0.283	-0.206	-0.220
65	-0.055	1.429	-0.459	0.134	-0.492	0.149	0.135	0.327	0.003	-0.476
66	-0.069	1.011	-0.026	0.207	0.265	-0.009	0.025	-0.516	0.053	-0.244
67	-0.058	0.045	0.366	0.359	-0.217	-0.100	-0.501	-0.249	0.213	0.432
68	-0.978	0.636	-0.082	-0.033	0.692	-0.092	0.272	-0.751	-0.831	0.249
69	-0.413	-0.003	0.326	-0.198	1.122	-0.146	0.572	-0.462	0.262	-0.314
bias	-0.244	-0.129	0.457	0.079	0.256	-0.232	-0.393	0.432	-0.124	0.267

WEIGHT (1) between neuron layer 1 and neuron Output

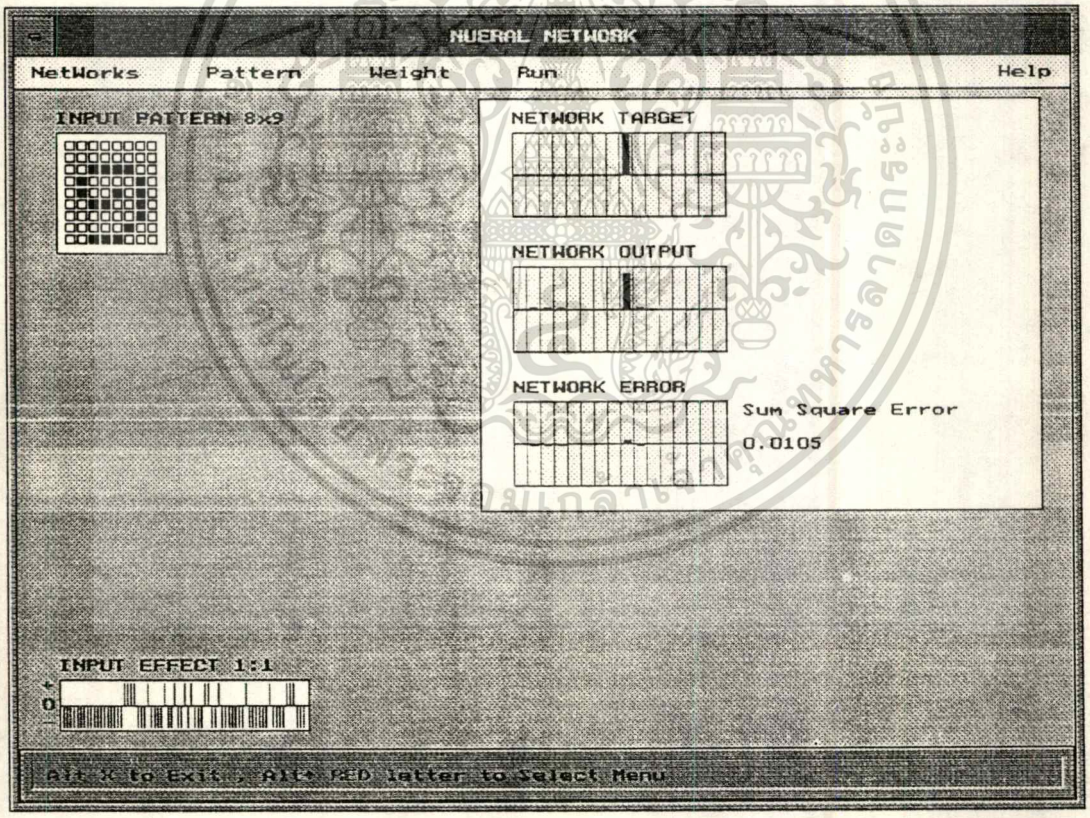
	node0	node1	node2	node3	node4	node5	node6	node7	node8	node9
0	1.166	1.175	0.639	0.765	-2.658	-1.742	-2.574	-2.634	-0.524	1.049
1	1.176	-1.554	-1.984	0.378	0.285	-1.393	-2.126	1.512	1.186	-1.631
2	-1.333	-2.289	-1.984	1.263	-0.726	1.444	-0.698	-2.114	1.153	-1.427
3	-1.982	-1.001	1.246	-1.212	-0.942	0.133	-1.877	1.565	1.485	-0.936
4	-2.517	1.125	-0.695	0.745	-1.804	-1.589	1.371	0.977	-1.138	-2.077
5	0.537	1.781	-1.940	-2.685	1.335	-2.197	0.593	-0.154	-1.667	-0.954
6	-2.083	-0.636	-1.671	0.730	0.416	0.201	-0.982	-1.416	-2.720	1.203
7	0.439	-1.450	1.226	-1.161	1.526	1.791	-1.429	-0.206	-1.936	-2.085
8	1.005	-1.641	0.854	-1.992	-2.630	-0.083	1.324	0.638	-1.180	-1.289
9	-1.172	-2.761	0.943	-2.723	0.778	-2.184	1.436	-1.766	0.398	1.646
bias	-1.283	-0.631	-1.819	-1.214	-1.139	-0.388	-1.199	-1.540	-0.694	-0.664

การแสดงผลการคำนวณของน้ำหนัก

4.2.8 การทดสอบการรู้จำของโครงข่าย

เมื่อได้ฝึกฝนโครงข่ายเป็นที่เรียบร้อยแล้ว ผลของน้ำหนักที่ได้จากการเรียนรู้ยังคงอยู่ในหน่วยความจำ การทดสอบการรู้จำของโครงข่ายเพียงแต่โหลดข้อมูล Pattern เข้ามา โดยเลือกหัวข้อ PATTERN เลือกรายการ Load Pattern โปรแกรมจะแสดงรูปของ Pattern นั้นและจะทำการคำนวณผลลัพธ์ออกทางจอภาพดังภาพที่ 26 ถ้าผลการคำนวณค่าผิดพลาดที่ได้มีค่าน้อยกว่าหรือเท่ากับค่าผิดพลาดที่กำหนดไว้ แสดงว่าโครงข่ายสามารถ "จำได้"

ภาพที่ 26

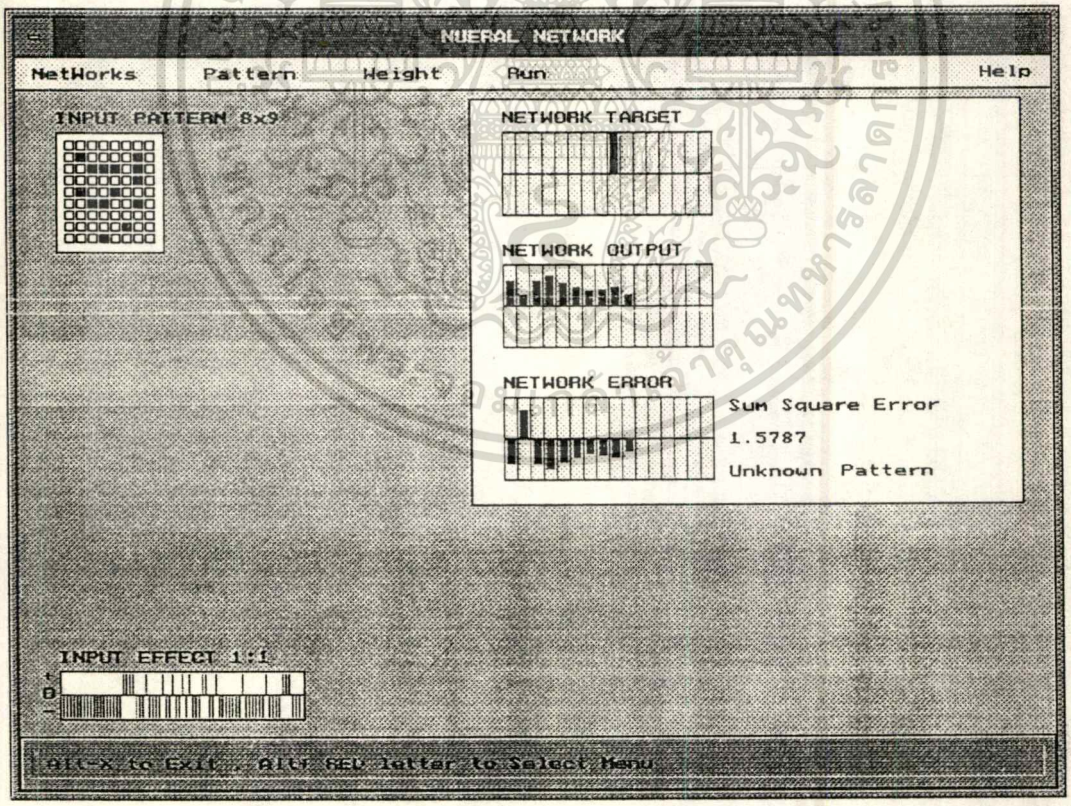


การทดสอบการรู้จำของโครงข่ายเมื่อโครงข่ายสามารถ "จำได้"

ในภาพที่ 26 โปรแกรมจะแสดงภาพของ Pattern ที่ใช้ในการทดสอบการรู้จำ พร้อมทั้งแสดงเอาต์พุตเป้าหมาย เอาต์พุตของโครงข่าย และค่าผิดพลาดที่เกิดขึ้น ในรูปของกราฟแบบ histogram ขนาดของช่องในแนวแกน X จะมีจำนวนเท่ากับขนาดของเอาต์พุตเป้าหมาย และเอาต์พุตของโครงข่าย สำหรับในแนวแกน Y จะแสดงขนาดของผลการคำนวณที่ได้จากโครงข่าย

แต่ถ้าโหลดข้อมูล Pattern เข้ามาแล้ว ผลการคำนวณค่าผิดพลาดที่ได้มีค่าเกินกว่าค่าผิดพลาดที่กำหนดไว้แสดงโครงข่ายไม่รู้จัก Pattern นั้น และแสดงกราฟของเอาต์พุตที่ได้จากโครงข่าย กราฟของความผิดพลาด และข้อความ Unknown Pattern

ภาพที่ 27



การทดสอบการรู้จำของโครงข่ายเมื่อโครงข่ายไม่รู้จัก Pattern นั้น

4.3 ตัวอย่างการใช้งานโปรแกรม

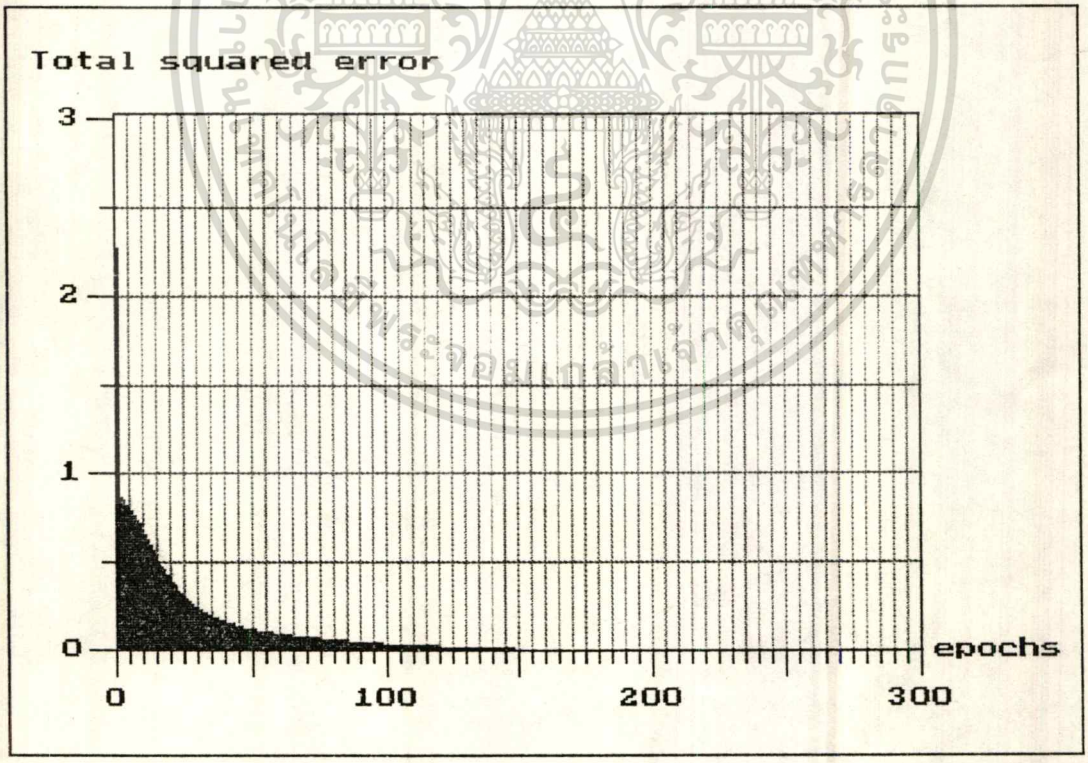
ตัวอย่างการฝึกฝนโครงข่ายด้วยตัวเลขไทยขนาด 8x9 จุดภาพ โดยจัดเตรียมรูปแบบตัวเลขไทย 0–๙ ไว้สามชุดสำหรับฝึกฝนและสำหรับทดสอบอีกหนึ่งชุด ดังภาพ

ภาพที่ 28

ตัวอย่างตัวเลขไทยที่นำมาใช้ฝึกฝนโครงข่าย

โหลดชุดข้อมูล Pattern ชื่อ TDIGIT.GRP เข้าสู่หน่วยความจำ และกำหนดค่าต่างๆให้กับโครงข่าย โดยกำหนดให้โครงข่ายมีขนาด 3 ชั้น และชั้นที่ซ่อนอยู่มีขนาด 50 โหนด กำหนดช่วงในการสุ่มน้ำหนักเท่ากับ -0.3 ถึง 0.3 เลือกอัลกอลิทึม Backpropagation กำหนดอัตราการเรียนรู้เท่ากับ 0.15 โมเมนตัมเท่ากับ 0.075 และค่าผิดพลาดต่ำสุดเท่ากับ 0.02 เลือกหัวข้อ Weight และเลือกรายการ Random Weight (หรือกด F6) จะสังเกตกราฟแบบ histogram ของน้ำหนักปรากฏอยู่ทางด้านล่าง ให้ทำการจัดเก็บน้ำหนักที่ได้จากการสุ่มนี้เพื่อใช้ในการเปรียบเทียบในครั้งต่อไป โดยเลือกหัวข้อ Weight และเลือกรายการ Save Weight โดยกำหนดชื่อของน้ำหนักนี้เป็น TSTART.WHT เสร็จแล้วเลือก RUN/Training โปรแกรมจะเริ่มทำงาน เมื่อโปรแกรมฝึกฝนโครงข่ายเรียบร้อยแล้วจะแสดงกราฟ ดังภาพต่อไปนี้

ภาพที่ 29



Learning rate 0.15 epochs 147 Sum square error 0.20 Time 14:26:92

กราฟแสดงค่าผิดพลาดรวมของโครงข่าย (Total squared error) กับจำนวนรอบที่ใช้ (epochs)

เมื่อฝึกฝนโครงข่ายเป็นที่เรียบร้อยแล้ว ขณะนั้นค่าน้ำหนักและโครงสร้างของโครงข่ายยังอยู่ในหน่วยความจำ เมื่อนำอินพุต Pattern เข้าสู่หน่วยความจำ โครงข่ายจะคำนวณและแสดงผลลัพธ์ออกมาทันที การทดสอบนำ Pattern ชุดที่ยังไม่ได้ผ่านการฝึกฝน (ซึ่งได้มาจากการสุ่ม noise ประมาณ 10% รายละเอียดการเตรียมอินพุตสำหรับการทดสอบจะอยู่ในภาคผนวก ก) มาทดสอบการรู้จำของโครงข่าย ผลการทดลองการรู้จำแสดงไว้ในตารางที่ 3

ตารางที่ 3

ชื่อไฟล์ที่ใช้ทดสอบ	เป็น Pattern ของหมายเลข	ค่าผิดพลาดที่คำนวณได้จากการทดสอบ (Sum of Square error)	ผลของการรู้จำ
T4DIGIT0 .PAT	๐	0.0059	จำได้
T4DIGIT1 .PAT	๑	0.0063	จำได้
T4DIGIT2 .PAT	๒	0.0121	จำได้
T4DIGIT3 .PAT	๓	0.0110	จำได้
T4DIGIT4 .PAT	๔	0.0073	จำได้
T4DIGIT5 .PAT	๕	0.0141	จำได้
T4DIGIT6 .PAT	๖	0.0029	จำได้
T4DIGIT7 .PAT	๗	0.0041	จำได้
T4DIGIT8 .PAT	๘	0.0071	จำได้
T4DIGIT9 .PAT	๙	0.0048	จำได้

การทดสอบการรู้จำของโครงข่าย

ตัวอย่างการหาอัตราการเรียนรู้ที่เหมาะสม โดยใช้น้ำหนักที่ถูกแยกเก็บไว้ในครั้งแรก โหลดเข้าสู่โครงข่าย เพื่อนำมาใช้ในการเปรียบเทียบ และต้องทำการโหลดน้ำหนักใหม่ทุกครั้งเมื่อเปลี่ยนแปลงอัตราการเรียนรู้เพื่อให้โครงข่ายอยู่ในสภาวะแวดล้อมเดียวกัน เปลี่ยนแปลงอัตราการเรียนรู้ให้มีค่าต่างๆกันดังตารางที่ 4 สังเกตการเปลี่ยนแปลงของจำนวนรอบและเวลาที่ใช้ในการฝึกฝนโครงข่าย

ตารางที่ 4

อัตราการเรียนรู้ (h)	จำนวนรอบที่ใช้ (epoch)	ค่าผิดพลาดรวม (Sum of square error)	เวลาที่ใช้ min:sec:1/100
0.15	147	0.0200	14:26:92
0.2	111	0.0199	10:54:78
0.25	90	0.0198	08:51:04
0.30	77	0.0197	07:34:39
0.35	68	0.0198	06:41:37
0.40	63	0.0192	06:11:92
0.45	59	0.0196	05:48:35
0.50	58	0.0180	05:42:47
0.55	57	0.0194	05:36:59
0.60	58	0.0176	05:42:47
0.65	59	0.0171	05:48:40
0.70	60	0.0182	05:54:23
0.75	62	0.0167	06:06:04
0.80	64	0.0162	06:17:85
0.85	66	0.0167	06:29:61
0.90	68	0.0176	06:41:42
0.95	70	0.0195	06:53:24
1.00	73	0.0170	07:10:87
1.50	94	0.0160	09:14:67
2.00	-	-	-

ผลการทำงานของโครงข่ายเมื่อเปลี่ยนแปลงอัตราการเรียนรู้

ผลการทดลองในตารางที่ 4 อัตราการเรียนรู้ที่เหมาะสม จะทำให้โครงข่ายมีจำนวนรอบในการฝึกฝน และเวลาที่ใช้ น้อยที่สุด สำหรับในการทดลองนี้จะได้เท่ากับ 0.55 โครงข่ายที่ต่างกัน หรือชุดข้อมูลอินพุตที่ต่างกันจะได้ขนาดอัตราการเรียนรู้ที่เหมาะไม่เท่ากัน

ตัวอย่างการเปลี่ยนแปลงจำนวนโหนดของนิวรอลเซลล์ในชั้นที่ซ่อนอยู่ สำหรับโครงข่าย 3 ชั้น โดยกำหนดค่าต่างๆ เหมือนกับตัวอย่างที่แล้วเพียงแต่เปลี่ยนจำนวนโหนดของนิวรอลเซลล์ให้มีขนาดต่างๆ กัน ผลการคำนวณของโครงข่ายแสดงให้เห็นได้ตามตารางต่อไปนี้

ตารางที่ 5

จำนวนโหนดของนิวรอลเซลล์ในชั้นที่ซ่อนอยู่	จำนวนรอบที่ใช้ (epoch)	ค่าผิดพลาดรวม (Sum of square error)	เวลาที่ใช้ min:sec:1/100
3	-	-	-
4	377	0.0199	03:13:84
5	229	0.0199	02:24:56
10	**88**	**0.0200**	**01:47:25**
15	74	0.0196	02:13:57
20	59	0.0196	02:21:15
25	50	0.0194	02:29:01
30	46	0.0196	02:44:12
35	43	0.0196	02:58:68
40	39	0.0194	03:05:60
45	37	0.0200	03:17:30
50	37	0.0197	03:39:01

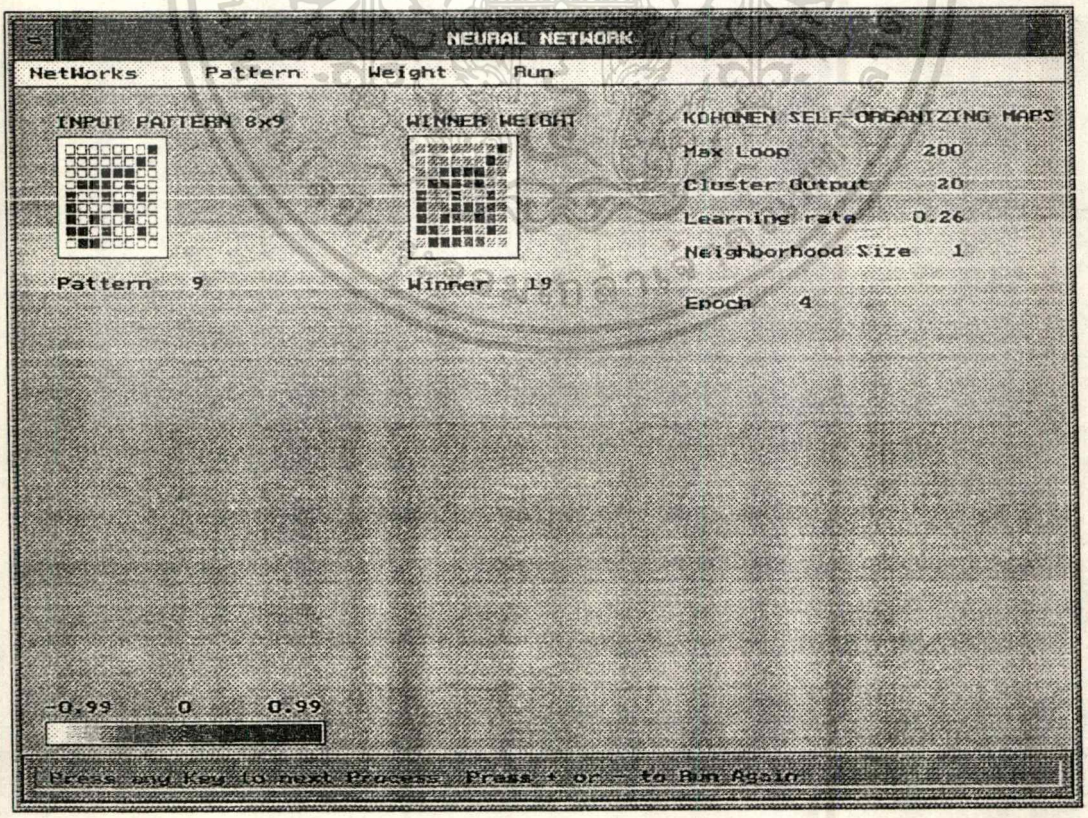
ผลการคำนวณของโครงข่ายที่มีจำนวนโหนดของชั้นที่ซ่อนอยู่ขนาดต่างๆ กัน

จากผลการทดลองในตารางที่ 5 เมื่อขนาดของชั้นที่ซ่อนอยู่มีค่าน้อยกว่า 4 โหนด โครงข่ายไม่สามารถแยกอินพุตได้ (สังเกตจากกราฟแสดงการทำงานจะไม่ลดลง หลังจากที่ใช้เวลา

ในการฝึกฝนโครงข่ายเป็นเวลานานมาก) ขนาดของนิวรอลที่เหมาะสมคือ ขนาดที่น้อยที่สุดที่โครงข่ายสามารถเรียนรู้ได้ ในเวลาที่น้อยที่สุด จากการทดลองจำนวนโหนดที่เหมาะสมสำหรับโครงข่าย 3 ชั้น มีอินพุตที่มีเอาต์พุตเป้าหมายต่างกัน 10 ประเภทจาก 30 อินพุต (0-๙) จะมีขนาดเท่ากับ 10 โหนด จะสังเกตได้ว่าจำนวนประเภทที่ต้องการ เป็นผลโดยตรงกับขนาดของนิวรอลเซลล์ไม่ใช่จำนวนอินพุต ซึ่งถ้าให้มีจำนวนโหนดมากกว่านี้จะทำให้โครงข่ายมีขนาดใหญ่เกินความจำเป็นและเรียนรู้ได้ช้า

ตัวอย่างการใช้อัลกอริทึม Kohonen Self-Organizing Maps (SOM) เลือกหัวข้อ Self-Organizing Maps จากเมนู อินพุตใช้ Pattern ของตัวเลขไทย 0-๙ เช่นเดียวกันกับตัวอย่างที่ผ่านมา กำหนดค่าต่างๆโดยให้ อัตราการเรียนรู้เริ่มต้นที่ 0.5 ขนาดของ neighborhood เท่ากับ 5 จำนวนรอบสูงสุดที่ใช้ เท่ากับ 200 การทดลองจะทำการเปลี่ยนจำนวนเอาต์พุตให้มีขนาดต่างๆกัน เพื่อสังเกตการจับกลุ่มและคลัสเตอร์เอาต์พุตที่ได้จาก winner neuron โดยให้จำนวนเอาต์พุตเท่ากับ 5 10 และ 20 ตามลำดับ เมื่อเลือกคำสั่ง Training จะปรากฏภาพหน้าจอดังภาพต่อไปนี้

ภาพที่ 30



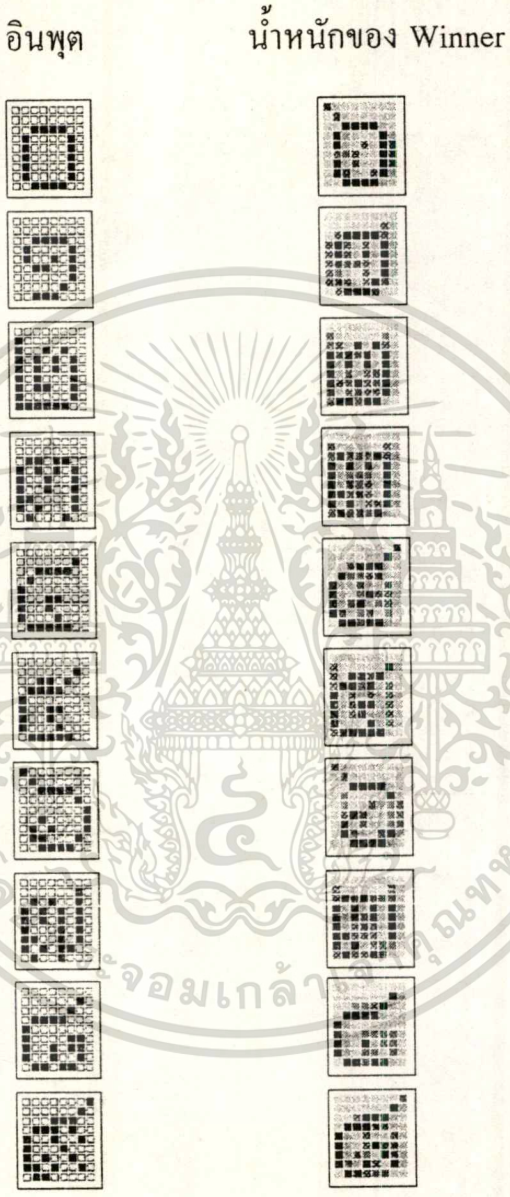
จอภาพของอัลกอริทึม Kohonen Self-Organizing Maps (SOM)

จากภาพที่ 30 เป็นจอภาพของอัลกอริทึม Kohonen Self-Organizing Maps (SOM) ที่โครงข่ายทำการฝึกฝนไปแล้ว 4 รอบ จอภาพจะแสดงค่าต่างๆ ที่กำหนดไว้ พร้อมทั้งแสดงการเปลี่ยนแปลงที่เกิดขึ้น เช่นการเปลี่ยนแปลงอัตราการเรียนรู้ ขนาดของ neightborhood และจำนวนรอบที่ใช้ไป โปรแกรมจะแสดงรูป Pattern ที่โครงข่ายกำลังคำนวณพร้อมทั้งน้ำหนักของ winner neuron ของ Pattern นั้น โดยจะแทนขนาดของน้ำหนักด้วยลำดับความเข้มของสีที่มีมิติเหมือนกับอินพุต(ระนาบ 2 มิติ) ระดับของสีจะแสดงการเปรียบเทียบให้เห็นอยู่ทางด้านล่างของจอภาพ การเปลี่ยนแปลงของน้ำหนักที่วงรอบต่างๆ จะมีลักษณะดังภาพที่ 31

ภาพที่ 31

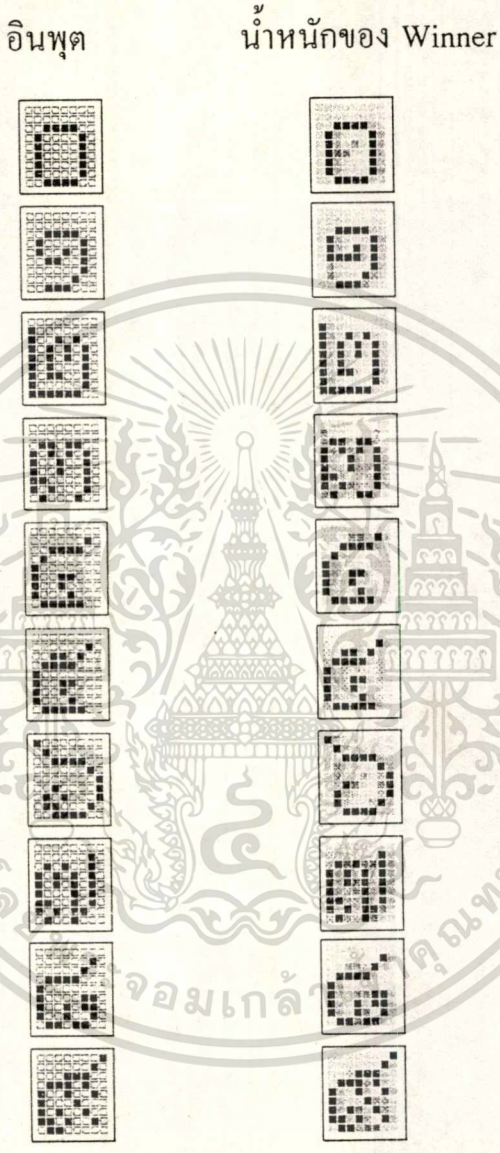
การเปลี่ยนแปลงของน้ำหนักที่โหนดของ winner neuron เมื่อวงรอบ (epoch) เท่ากับ 2

ภาพที่ 32



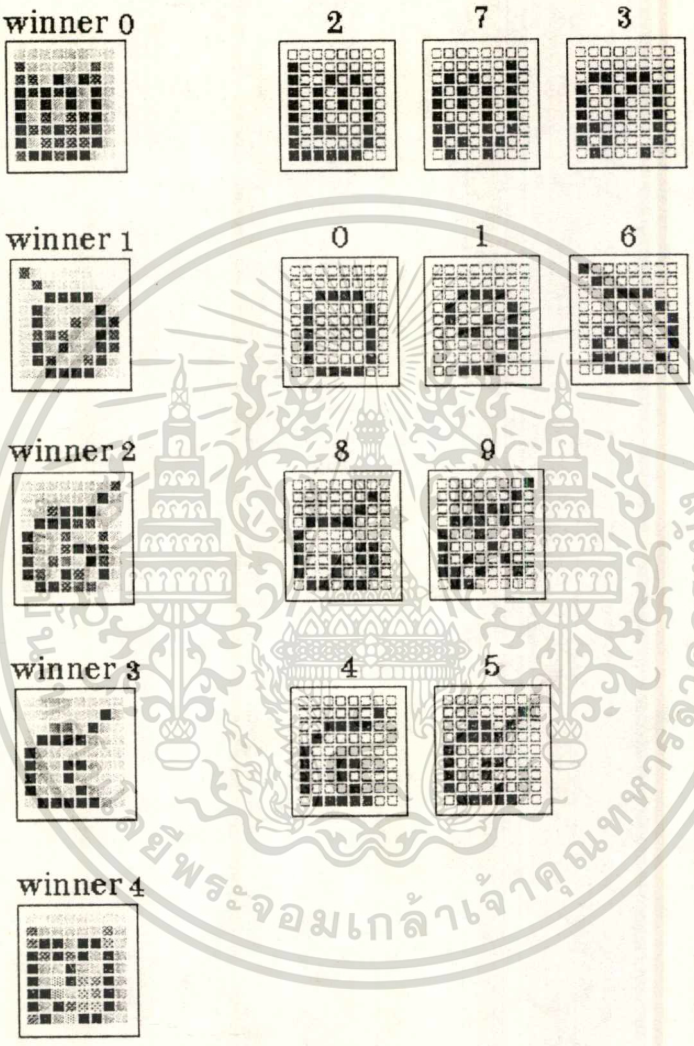
การเปลี่ยนแปลงของน้ำหนักที่โหนดของ winner neuron เมื่อวงรอบ (epoch) เท่ากับ 5

ภาพที่ 33



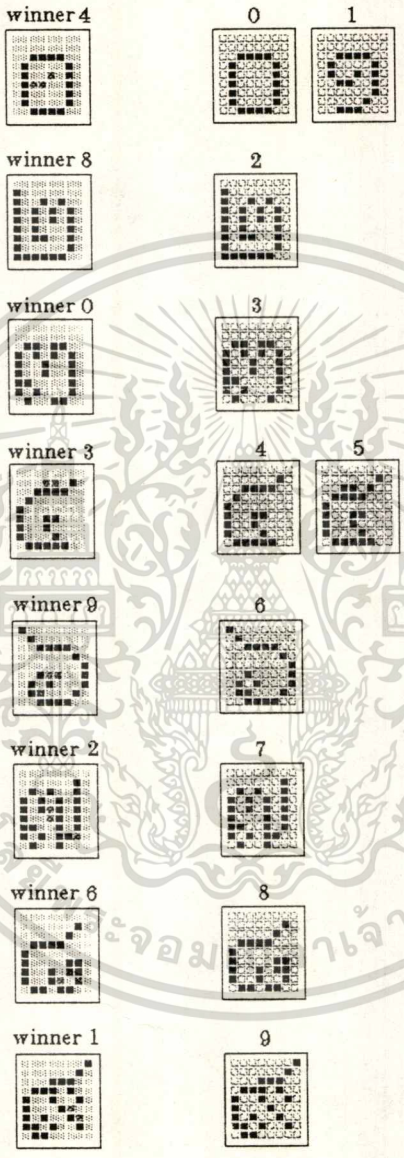
การเปลี่ยนแปลงของน้ำหนักที่โหนดของ winner neuron เมื่อวงรอบ (epoch) เท่ากับ 20

ภาพที่ 34



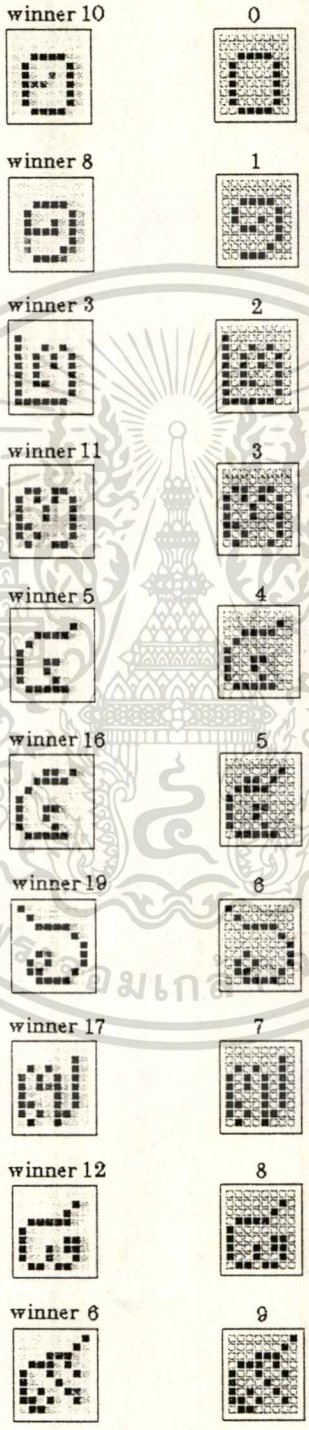
แสดงน้ำหนักของ winner neuron และ Pattern ที่จัดอยู่ในกลุ่มเดียวกัน
เมื่อให้โครงข่ายมีโหนดเอาต์พุตเท่ากับ 5

ภาพที่ 35



แสดงน้ำหนักของ winner neuron และ Pattern ที่จัดอยู่ในกลุ่มเดียวกัน
เมื่อให้โครงข่ายมีโหนดเอาต์พุตเท่ากับ 10

ภาพที่ 36



แสดงน้ำหนักของ winner neuron และ Pattern ที่จัดอยู่ในกลุ่มเดียวกัน
เมื่อให้โครงข่ายมีโหนดเอาต์พุตเท่ากับ 20

การทำงานของโครงข่ายจะนำอินพุต Pattern มาหา winner neuron ที่มี Euclidean distance น้อยที่สุด และจะปรับน้ำหนักเฉพาะ winner neuron และ neighborhood รอบๆ ผู้ใช้จะสามารถสังเกตการเปลี่ยนแปลงของน้ำหนักได้จากจอภาพ ขนาดของน้ำหนักซึ่งแทนด้วยระดับความแตกต่างของสี จะค่อยๆเปลี่ยนแปลงจนมีรูปร่างคล้ายกับอินพุตนั้น อัตราการเรียนรู้และขนาด neighborhood จะค่อยๆลดค่าลงเมื่อวงรอบเพิ่มขึ้น โครงข่ายจะหยุดทำงานเมื่ออัตราการเรียนรู้มีค่าลดลงจนเท่ากับ 0 หรือการทำงานมีวงรอบครบตามจำนวนที่กำหนดขึ้นกับว่าจะพบเงื่อนไขใดก่อน การทำงานของโครงข่ายถูกจำลองการทำงานให้คล้ายกับการทำงานของสมองมนุษย์ ที่พยายามจดจำภาพที่คล้ายกันไว้ในกลุ่มเดียวกันก่อน แล้วจึงนำภาพในกลุ่มมาหาข้อแตกต่างอีกทีหนึ่ง การทำงานของโครงข่ายในขณะเริ่มต้น ในขณะที่อัตราการเรียนรู้ และ neighborhood ยังมีค่ามาก โครงข่ายจะมีการทำงานในลักษณะการจับกลุ่มตามลักษณะของภาพที่มีลักษณะใกล้เคียงกันในขณะเริ่มต้น สังเกตได้จากภาพที่มี winner neuron เหมือนกัน เช่น ๐ ๑ , ๒ ๓ ๗ , ๔ ๕ หลังจากที่โครงข่ายมีจำนวนรอบเพิ่มขึ้น อัตราการเรียนรู้ และ neighborhood จะมีค่าลดลง จะเห็นว่าโครงข่ายพยายามที่จะแบ่งประเภทออกไปอีก จากการปรับน้ำหนักในบริเวณรัศมี neighborhood ทำให้ภาพที่อยู่ในกลุ่มเดียวกันได้ winner ตัวใหม่ที่กระจายออกไปในรัศมี neighborhood ที่อยู่ข้างๆwinner ตัวเก่า การลดลงของ neighborhood เป็นการเฉพาะเจาะจงภาพอินพุตกับ winner neuron ที่ตอบสนองมากยิ่งขึ้น เมื่อ neighborhood มีค่าลดลงจนมีค่าเท่ากับ 1 ในตอนนี้โครงข่ายจะไม่สามารถกระจายภาพอินพุตออกจากกลุ่มได้อีก เป็นการระบุแน่ชัดขึ้นไปอีกว่าอินพุตภาพใดได้ winner neuron ตัวใด ขณะเดียวกันอัตราการเรียนรู้จะมีค่าลดลงเป็นผลให้การปรับน้ำหนักที่โหนด winner neuron ที่เหลือเพียงตัวเดียวนั้นเป็นไปอย่างช้าๆ ทำให้ขนาดของน้ำหนักนั้นค่อยๆ เปลี่ยนไปเหมือนกับอินพุตในที่สุด

จำนวนโหนดของเอาต์พุตเป็นตัวแปรที่สำคัญอีกประการหนึ่ง ที่จะทำให้โครงข่ายสามารถแยกประเภทอินพุตได้ครบหรือไม่ จะสังเกตได้ว่าเมื่อกำหนดโหนดเอาต์พุตมีขนาดน้อยกว่าความแตกต่างของ Pattern โครงข่ายจะจับกลุ่ม Pattern ที่มีลักษณะใกล้เคียงกันไว้ในกลุ่มเดียวกัน

บทที่ 5

บทสรุป

สรุปการทำงานของโปรแกรม

การทำงานของโปรแกรมพอสรุปได้ดังต่อไปนี้ ส่วนแรกเป็นการเตรียมข้อมูลอินพุตซึ่งเป็น pattern ที่สร้างขึ้นมาใช้ฝึกฝนโครงข่าย ข้อมูล Pattern จะเป็นไฟล์ที่มีนามสกุลเป็น *.PAT Pattern จะถูกจัดกลุ่มให้อยู่ในประเภทเดียวกันโดยเป็นไฟล์ที่มีนามสกุล *.GRP โครงสร้างข้อมูลอินพุตถูกจัดเก็บเป็นลักษณะของคู่ข้อมูล อินพุต-เอาต์พุต คือเก็บเอาต์พุตเป้าหมายไว้ในไฟล์เดียวกัน โปรแกรมจะนำข้อมูลอินพุตมาใช้กำหนดขนาดของชั้นอินพุตของโครงข่าย และนำเอาต์พุตเป้าหมายมาใช้กับอัลกอริทึมที่เป็นแบบ Supervise learning เมื่อผู้ใช้กำหนดค่าต่างๆ ที่จำเป็นให้กับโครงข่ายและทำการฝึกฝนโครงข่ายแล้ว โปรแกรมจะสร้างไฟล์ที่มีนามสกุล *.NET เพื่อเก็บโครงสร้างของโครงข่ายและน้ำหนัก ซึ่งเป็นประสบการณ์ที่ได้จากการเรียนรู้

จากแนวทางการใช้งานโปรแกรมในบทที่ 4 ผู้ใช้สามารถศึกษาการทำงานของโครงข่ายประสาทเทียมและทดลองเปลี่ยนแปลงค่าต่างๆ เพื่อสังเกตวิธีการทำงานของโครงข่าย ผู้ใช้สามารถออกแบบอินพุต Pattern ขึ้นมาใช้ในการทดสอบโครงข่ายได้เอง และเลือกศึกษาอัลกอริทึมในการฝึกฝนโครงข่ายได้หลายอัลกอริทึมโดยที่ไม่ต้องเขียนโปรมแกรมขึ้นมาเอง โปรแกรมยังสามารถทำงานทีละขั้นได้ เพื่อที่จะสังเกตขั้นตอนการทำงาน และวิธีการคำนวณของโครงข่ายโดยละเอียด สำหรับในกรณีที่ผู้ใช้พึ่งเริ่มต้นศึกษาการใช้งานโครงข่ายประสาท เทียม จะมีส่วนช่วยเหลืออธิบายขั้นตอนการทำงานของโครงข่ายสำหรับในแต่ละอัลกอริทึม เพื่อให้ผู้ใช้เข้าใจถึงขบวนการทำงานในแต่และขั้นตอนก่อนเริ่มต้นใช้งานโปรแกรม ทำให้ช่วยลดเวลาค้นคว้าหาความรู้เกี่ยวกับโครงข่ายประสาทเทียมจากสื่ออื่นๆ

ภาคผนวก

ภาคผนวก ก
โครงสร้างของข้อมูลอินพุตและการเตรียมข้อมูลอินพุต

โครงสร้างของข้อมูลอินพุต

ลักษณะข้อมูลอินพุตมีโครงสร้างด้วยกันสองแบบคือ แบบที่ใช้โปรแกรม Pattern editor สร้างและแก้ไข Pattern ที่มาพร้อมกับโปรแกรมในวิทยานิพนธ์ฉบับนี้ และแบบที่สร้างขึ้นด้วย Text editor ทั่วไป ไฟล์ที่สร้างด้วย Pattern editor จะจัดการเก็บข้อมูลภาพที่ผู้ใช้ออกแบบไว้บนหน้าจอเป็นไฟล์ข้อมูลอินพุต โดยจะเก็บ 8 จุดภาพต่อหนึ่งไบท์ (bit pack) และบันทึกข้อมูลและรายละเอียดลงไฟล์เป็นแบบไบนารี โครงสร้างส่วนหัวของข้อมูลจะมีลักษณะดังนี้

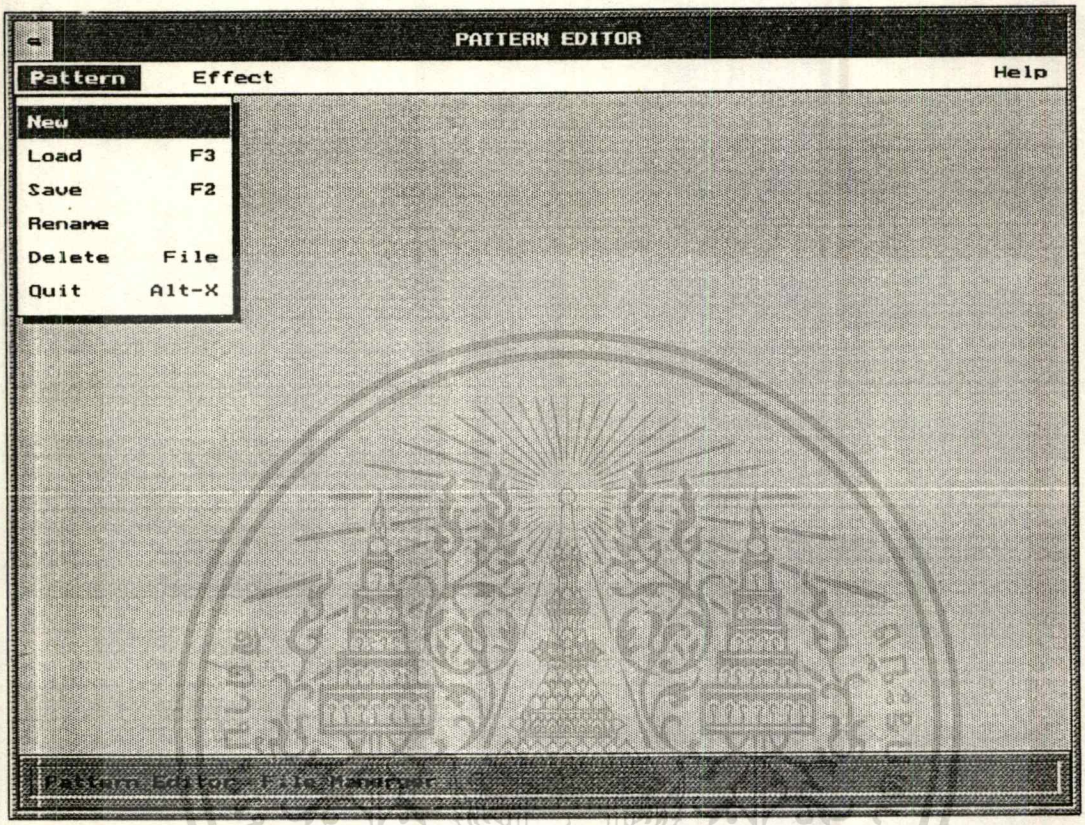
```
struct PatternHeader{
   char code [3];       //รหัสนำหน้า "BP" ซึ่งใช้แยกชนิดของข้อมูล
   char nPat;           //จำนวน Pattern ปกติกำหนดให้เป็น 1
   char Xsize;          //ความกว้างของ Pattern หน่วยเป็นจุดภาพ
   char Ysize;          //ความยาวของ Pattern หน่วยเป็นจุดภาพ
   int  nOutput;        //จำนวนเอาต์พุตโหนด
   int  target;         //เอาต์พุตเป้าหมายของ Pattern นี้
};
```

การใช้งานโปรแกรม Pattern editor เมื่ออยู่ใน DOS PROMPT ให้พิมพ์คำว่า

C:\>PATTEDIT <ENTER>

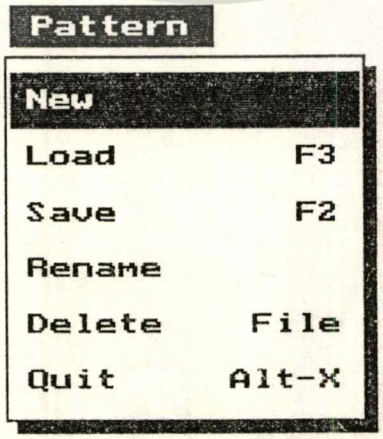
เมื่อโปรแกรมเริ่มทำงานจะปรากฏภาพหน้าจอดังภาพ

ภาพที่ 37



ภาพหน้าจอของโปรแกรม Pattern Editor

ภาพที่ 38



รายการคำสั่งเมื่อเลือกหัวข้อ Pattern

เลือกหัวข้อ Pattern โดยกด Alt-R จะปรากฏพูลดาวน์เมนูซึ่งความหมายของคำสั่งต่างๆ จะมีรายระเอียดดังต่อไปนี้

New

สร้าง Pattern ใหม่ตามขนาดที่ต้องการพร้อมทั้งกำหนดเอาต์พุตเป้าหมาย

Load

นำ Pattern ที่มีอยู่เดิมมาแก้ไข หรือเปลี่ยนแปลงเอาต์พุตเป้าหมาย

Save

จัดเก็บ Pattern เป็นไฟล์ที่มีนามสกุล *.PAT

Rename

เปลี่ยนชื่อไฟล์ใหม่

Delete

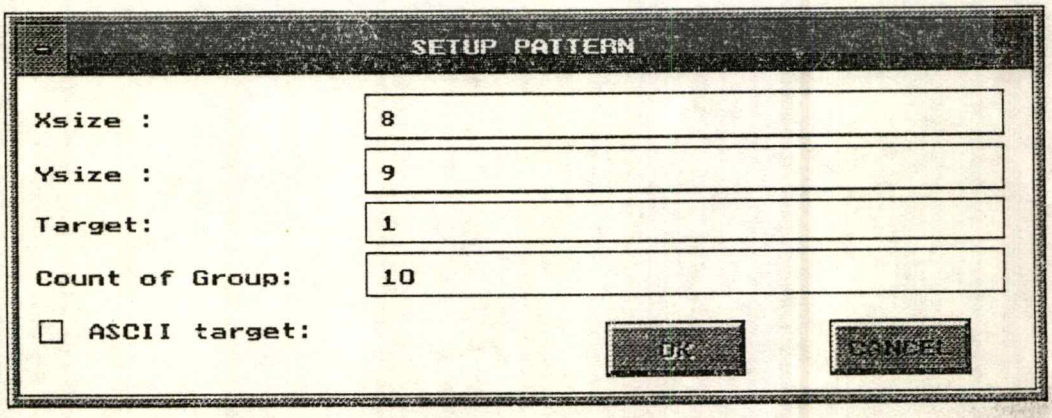
ลบไฟล์ที่ไม่ต้องการ

Quit

ออกจากโปรแกรม

เมื่อเลือกหัวข้อ Pattern และเลือกรายการ New จะปรากฏหน้าต่างสำหรับกำหนดค่าต่างๆ ดังภาพ

ภาพที่ 39



หน้าต่างการกำหนดค่าต่างๆ ของ Pattern ที่สร้างใหม่

Xsize :

กำหนดความกว้างของ Pattern

Ysize :

กำหนดความยาวของ Pattern

Target :

กำหนดเอาต์พุตเป้าหมายของ Pattern นี้

Count of Group:

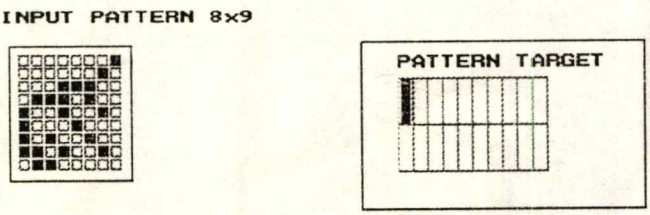
กำหนดขนาดของกลุ่มของ Pattern ที่ใช้ เพื่อใช้ในการกำหนดจำนวนโหนดของเอาต์พุต เมื่อกำหนดเอาต์พุตเป้าหมายเป็นแบบไบนารี เช่นกลุ่มของตัวเลขจะเป็น 10

☐ **ASCII target :**

กำหนดเอาต์พุตเป็นรหัสของตัวอักษร เช่น A เท่ากับ 41 ในเลขฐาน16 จะมีเอาต์พุตเป้าหมายเป็น 01000001 ซึ่งจำนวนเอาต์พุตจะมีขนาดเท่ากับ 8 เสมอทำให้ประหยัดหน่วยความจำเมื่อใช้จำนวนอินพุตที่เอาต์พุตเป้าหมายที่มีค่าต่างกันจำนวนมากๆ ปกติตัวเลือกนี้จะเป็น OFF หรือกำหนดเอาต์พุตเป้าหมายเป็นแบบไบนารี ที่จำนวนโหนดจะเท่ากับ Pattern ที่ใช้แยกประเภท

เมื่อกำหนดค่าต่างๆ เรียบร้อยแล้วโปรแกรมจะแสดงกรอบของ Pattern และขนาดที่กำหนด ผู้ใช้สามารถใช้เมาส์คลิ๊กในช่องสี่เหลี่ยมเพื่อสร้างรูปแบบของ Pattern ตามต้องการ เสร็จแล้วเลือกหัวข้อ Pattern/Save แล้วตั้งชื่อเพื่อบันทึกเก็บลงไฟล์ สำหรับการแก้ไขข้อมูลที่มีอยู่เดิมให้เลือกหัวข้อ Pattern/Load แล้วเลือกชื่อไฟล์ที่ต้องการแก้ไข โปรแกรมจะแสดงภาพ Pattern และเอาต์พุตเป้าหมายดังภาพ

ภาพที่ 40

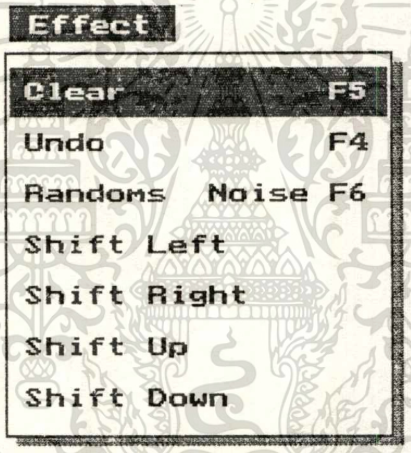


แสดง Pattern ที่ต้องการจะแก้ไข พร้อมทั้งเอาต์พุตเป้าหมาย

หากต้องการแก้ไขเฉพาะเอาต์พุตเป้าหมายให้เลือกหัวข้อ Pattern/New แล้วเปลี่ยนแปลงเฉพาะค่าของเอาต์พุตเป้าหมาย ถ้ามีการเปลี่ยนแปลงขนาดโปรแกรมจะถือว่าต้องการสร้าง Pattern ใหม่ โดยโปรแกรมจะสร้างกรอบว่างเพื่อให้ผู้ใช้เริ่มสร้าง Pattern ใหม่

ในหัวข้อถัดมาเป็นหัวข้อ Effect เป็นเครื่องมือในการดัดแปลง Pattern เพื่อใช้ในการทดสอบการรู้จำของโครงข่าย เช่นการสร้าง noise และการเลื่อนของ Pattern โดยเลือกหัวข้อ Effect หรือกดคีย์ ALT-E โปรแกรมจะแสดงพูลดาวน์เมนูดังภาพ

ภาพที่ 41



รายการคำสั่งเมื่อเลือกหัวข้อ Effect

Clear

ลบจุดภาพทั้งหมดให้เป็นกรอบแบบว่าง

Undo

ยกเลิกการแก้ไข โดยกลับไปใช้ Pattern เดิมก่อนการแก้ไข

Randoms Noise

การสร้าง noise แบบสุ่ม 10% ต่อการเลือก 1 ครั้ง

Shift Left

เลื่อน Pattern ไปทางซ้าย 1 จุดภาพ

Shift Right

เลื่อน Pattern ไปทางขวา 1 จุดภาพ

Shift Up

เลื่อน Pattern ไปข้างบน 1 จุดภาพ

Shift Down

เลื่อน Pattern ลงข้างล่าง 1 จุดภาพ

ตัวอย่างภาพการสุ่ม noise ในการสุ่มในแต่ละครั้งจะมีทั้งแบบจุดเพิ่ม และจุดขาดหาย โดยจะเกิดการเปลี่ยนแปลงประมาณ 10% ต่อจำนวนจุดภาพทั้งหมด ของการสุ่มในแต่ละครั้ง

ภาพที่ 42

a). b).

ตัวอย่างการสุ่ม noise a).ก่อนการสุ่ม b).หลังการสุ่ม noise 10%

ข้อมูลอินพุตอีกแบบหนึ่งเป็นไฟล์ที่ถูกสร้างขึ้นด้วย Text editor ทั่วไป โดยบรรทัดแรกจะต้องขึ้นต้นด้วย TX ตามด้วยจำนวน Pattern, ความกว้างและความยาวของ Pattern , เอาต์พุตเป้าหมาย และท้ายสุดตามด้วยข้อมูลของ Pattern โดยข้อมูลแต่ละตัวจะอยู่คนละบรรทัด ส่วนข้อมูลของ Pattern จะขึ้นบรรทัดใหม่เมื่อครบความกว้างตัวอักษร และใช้เลข 1 แทนจุดภาพที่เป็น 1 ใช้เลข 0 แทนจุดภาพที่เป็น 0 ตัวอย่างการสร้างข้อมูลของตัวเลข ๑ มีขนาด 8x9 และเอาต์พุตเป้าหมายเป็น 0000000001 ที่เขียนด้วย Text editor จะมีลักษณะดังนี้

```
TX
1
8
9
0000000001
00000000
00000000
00111100
01000010
01001010
00110010
00000010
00000100
00111000
```

โครงข่ายที่เป็นแบบ Supervised learning จะนำเอาต์พุตเป้าหมายมาใช้ในการหาข้อผิดพลาด และนำขนาดของเอาต์พุตเป้าหมายมาเป็นขนาดของชั้นเอาต์พุต สำหรับโครงข่ายที่เป็นแบบ Unsupervised learning จะไม่นำเอาต์พุตเป้าหมายมาใช้ ส่วนขนาดของชั้นเอาต์พุตผู้ใช้จะต้องกำหนดในโปรแกรมก่อนการฝึกฝนโครงข่าย

เนื่องจากการสร้างข้อมูลอินพุตจำนวนมากๆ อาจไม่สะดวกนัก ดังนั้นการนำตัวอักษรที่ใช้แสดงบนจอภาพของโปรแกรมประมวลผลคำ เช่นเวิร์ดราชวิถี หรือ เวิร์ดจุฬา มาใช้สร้างเป็น ข้อมูลอินพุตจะทำให้สะดวกขึ้นมาก ไฟล์ตัวอักษรปกตินั้นจะมีนามสกุลเป็น NORMAL.FON ซึ่งจะมีขนาด 8x20 ดั้งนั้นการใช้งานจะต้องเลือกตัดมาใช้ตามขนาดที่ต้องการ โดยใช้โปรแกรม MAKEFONT ซึ่งจะมีรูปแบบการใช้งานดังนี้

MAKEFONT [เริ่มที่ ASCII] [จำนวน Pattern] [เริ่มใช้บิทที่] [จำนวนบิทที่ใช้][ชื่อไฟล์]

ตัวอย่างเช่นต้องการเลขไทยซึ่งมีรหัสตัวอักษรจะเริ่มที่ 240 จำนวน 10 ตัว เริ่มใช้บิทที่ 7 จำนวนบิทที่ใช้ 9 บิท มีชื่อไฟล์เป็น TDIGIT จะได้ตัวอักษรที่มีขนาด 8x9 การเรียกโปรแกรมใช้งานจะเป็น

C:>MAKEFONT 240 10 7 9 TDIGIT /b

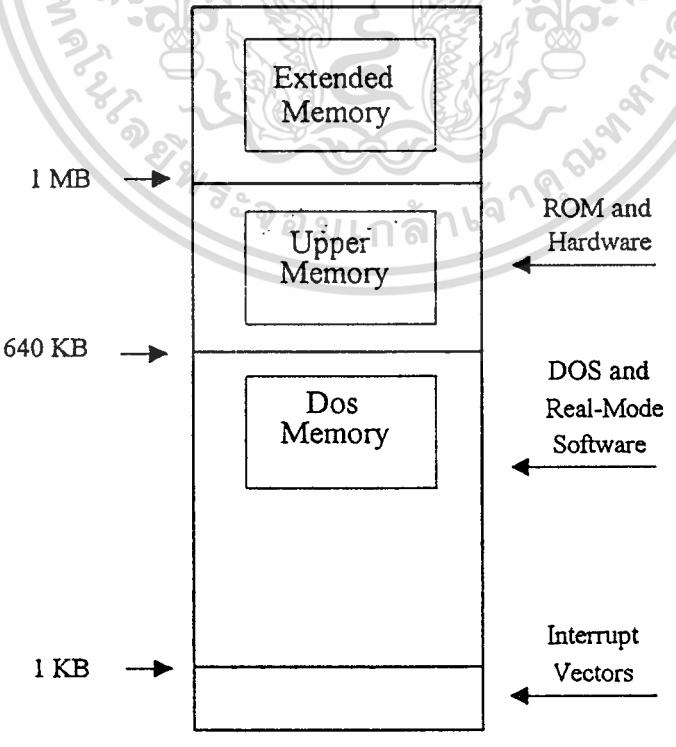
โปรแกรมจะนำตัวอักษรในไฟล์ NORMAL.FON มาตัดตัวอักษรที่ต้องการออกและจะสร้างเป็นข้อมูลอินพุต โดยบันทึกเป็นไฟล์เป็น TDIGIT1.PAT ,TDIGIT2.PAT ,TDIGIT3.PAT,...โดยจะมีหมายเลขของไฟล์เรียงลำดับต่อเนื่องกันไป โปรแกรมจะบรรจุเอาต์พุตเป้าหมายให้โดยอัตโนมัติ โดย /b จะให้โปรแกรมเริ่มนับตั้งแต่ 0000000000 , 0000000001 ,0000000010, ... และเพิ่มขึ้นไปเท่ากับจำนวน Pattern ที่ใช้ แต่ถ้าไม่ใส่ /b โปรแกรมจะนำรหัสตัวอักษรนั้นบรรจุเป็นเอาต์พุตเป้าหมายเช่น รหัสตัวอักษรเป็น F0 ฐาน16 จะมีเอาต์พุตเป้าหมายเป็น 11110000 เมื่อได้ข้อมูล Pattern แล้ว อาจนำ Pattern ที่ได้มาดัดแปลงเป็นรูปแบบอื่นๆ หรือนำมาสุ่มเพื่อใส่สัญญาณรบกวน ด้วยโปรแกรม Pattern editor เพื่อนำมาใช้ในการทดสอบโครงข่าย

ภาคผนวก ข

โครงสร้างของหน่วยความจำ และ การจัดการหน่วยความจำ

เนื่องจากโปรแกรมที่ใช้ในวิทยานิพนธ์ฉบับนี้ ได้ใช้หน่วยความจำทั้ง Real Mode และ Protect Mode การจัดหน่วยความจำทั้งสองแบบจะมีโครงสร้างที่ต่างกัน การจัดหน่วยความจำของ MSDOS บนเครื่อง 386 PC/AT จะประกอบไปด้วยหน่วยความจำของ DOS 640KB และหน่วยความจำส่วนบน (Upper memory) 384KB และต่อด้วยหน่วยความจำยืดขยาย (Extended memory) ซึ่งหน่วยความจำส่วนนี้ DOS ไม่สามารถนำมาใช้ได้ หน่วยความจำของ DOS และหน่วยความจำส่วนบนรวมกันเรียกว่า Real Memory

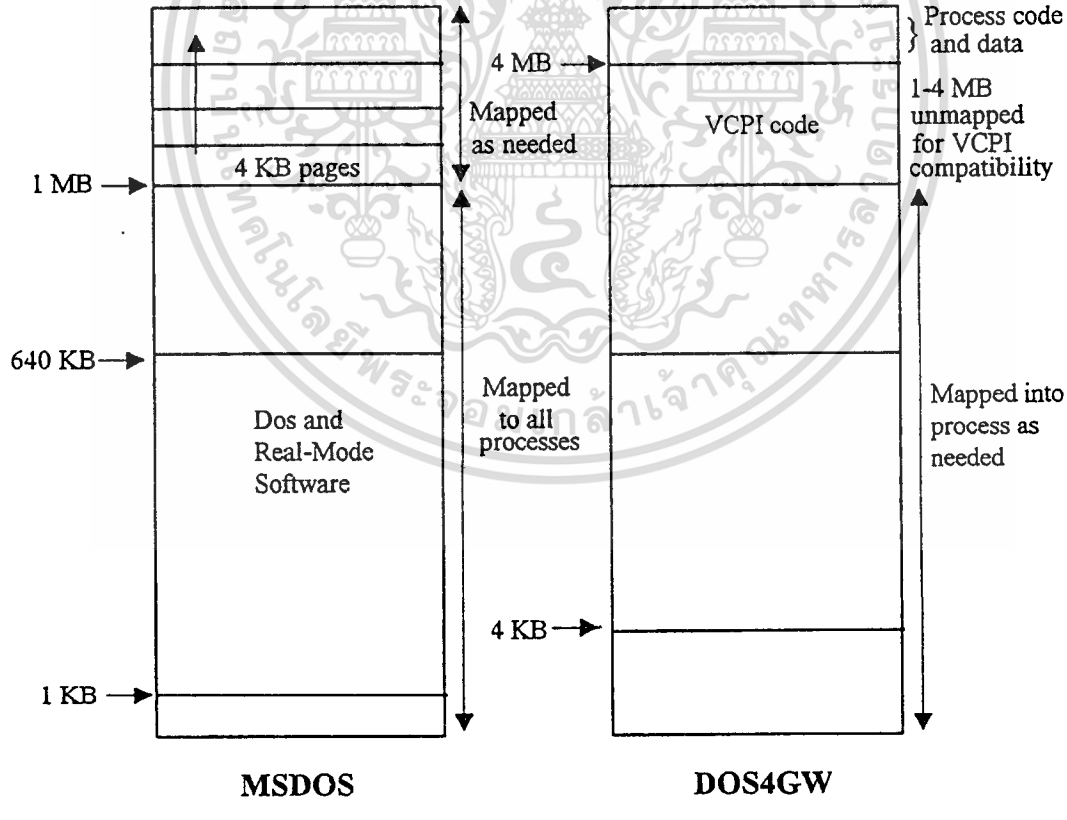
ภาพที่ 43



การจัดหน่วยความจำของ MSDOS

ภายใต้ DOS4GW หน่วยความจำ 1 MB แรก จะเป็น Physical memory และสามารถเข้าถึงหน่วยความจำส่วนนี้ได้แบบ absolute address เช่นการเข้าถึง video RAM หรือ ROM BIOS ส่วน Code และ Data ทั้งหมดจะถูกวางบน linear address เริ่มที่ Page 4 MB แรก โดยจะใช้ Virtual Control Program Interface หรือ VCPI ของ Quarterdeck และ Phar Lap Solft- ware ซึ่งเป็น Memory Interface บนระบบไมโครโปรเซสเซอร์ Intel 80386 หรือสูงกว่า

ภาพที่ 44



การจัดหน่วยความจำของ MSDOS เมื่อเทียบกับ Linear Address ของDOS4GW

การใช้งาน DOS4GW โดยใช้คอมไพเลอร์ WATCOM C/C++ 10.0a การใช้งานจะคล้ายกับคอมไพเลอร์ภาษา C ทั่วไป ยกเว้นฟังชั่นที่เกี่ยวกับ อินพุต-เอาต์พุต และการใช้หน่วยความจำ เช่นการอ่านคีย์บอร์ดในระดับ bioskey และการใช้กราฟิกไลบรารี การใช้งานอินเตอร์รัพรูทีนจะต้องใช้ int386 แทน int86 และตัวแปร intreger จากเดิมมีขนาด 16 บิท หรืออ้างได้ 64 KB จะมีขนาด 32 บิท สามารถอ้างได้ถึง 4GB ทำให้ตัวแปร size_t ที่เป็น type definition ของ unsigned int มีขนาดเปลี่ยนไปด้วยเช่น

```
void *malloc(size_t size);
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);
```

ทำให้ข้อจำกัดของอาเรย์ที่เดิมมีขนาด 64KB ต่อโหนดหมดไป สามารถใช้หน่วยความจำบนเครื่องคอมพิวเตอร์ที่ใช้ CPU 80386 หรือสูงกว่า ได้อย่างเต็มประสิทธิภาพ

ภาคผนวก ค
สรุปขั้นตอนการทำงานของโปรแกรมในแต่ละอัลกอริทึม

สำหรับการทำงานของโปรแกรม ตามทฤษฎีที่กล่าวมาในบทที่ 2 สามารถสรุปได้เป็นขั้นตอนการทำงานในแต่ละอัลกอริทึมดังต่อไปนี้

ตัวแปรและสัญลักษณ์ที่ใช้

p	หมายถึง Pattern
b	เป็น bias unit
nPattern	หมายถึงจำนวน Pattern ทั้งหมดที่นำมาใช้ในการฝึกฝน
nLayer	จำนวนชั้นของโครงข่าย
nOutput	จำนวนโหนดของชั้นเอาต์พุต
net	ผลรวมที่ได้จากนิวรอลเซลล์ (ยังไม่ผ่านฟังก์ชัน threshold)
LayerOut	เก็บผลลัพธ์ของนิวรอลเซลล์ที่ผ่านฟังก์ชัน threshold แล้ว
LayerErr	เก็บค่าผิดพลาดที่ได้จากโครงข่ายของชั้นที่ซ่อนอยู่
OutErr	เก็บค่าผิดพลาดที่ได้จากโครงข่ายของชั้นเอาต์พุต
target	เอาต์พุตเป้าหมาย
x	อินพุตของนิวรอลเซลล์
w	น้ำหนัก
η	อัตราการเรียนรู้
k	เป็นดัชนีของชั้นที่กำลังคำนวณ
j	เป็นดัชนีเลือกโหนดของนิวรอลเซลล์
i	เป็นดัชนีเลือกอินพุตที่เชื่อมโยงกับโหนด j
LearnDec	ขนาดที่ลดลงของอัตราการเรียนรู้
NeightborSize	ขนาดของ neightborhood
NeightborDec	ขนาดที่ลดลงของ neightborhood

ขั้นตอนการทำงานของ อัลกอริทึม Perceptron (PCN)

ขั้นที่ 1 จัดเตรียมข้อมูลอินพุตและเอาต์พุตเป้าหมาย เข้าสู่หน่วยความจำ

ขั้นที่ 2 สุ่มค่าน้ำหนักเป็นค่าเริ่มต้น (อยู่ในช่วงที่ผู้ใช้กำหนด)

ขั้นที่ 3 เมื่อเงื่อนไขเป็นจริง ให้ทำขั้นตอนที่ 4 ถึง 11

ขั้นที่ 4 สำหรับแต่ละคู่ข้อมูลอินพุต-เอาพุต เมื่อ Pattern p=0 ; p<nPattern

ให้ทำขั้นตอนที่ 5 ถึง 10

ขั้นที่ 5 จัดเตรียมข้อมูลอินพุตของ Pattern p เข้าสู่ชั้นอินพุต

ขั้นที่ 6 คำนวณผลรวมของแต่ละนิวรอลเซลล์จากสมการ

$$net_j = \sum_{i=0}^{n} w_{ji} x_i + b_j$$

ขั้นที่ 7 นำเอาต์พุตที่ได้มาหาค่าพิกัด

ถ้า $net_j > 0$ $LayerOut_j = 1$

ถ้า $net_j < 0$ $LayerOut_j = 0$

ขั้นที่ 8 หาค่าผิดพลาดของโครงข่ายในแต่ละโหนดได้จากสมการ

$$OutErr_j = (target_j - LayerOut_j)$$

ขั้นที่ 9 นำค่าผิดพลาดที่ได้มาใช้ในการปรับนำหนักจากสมการ

$$W_{ji}(new) = W_{ji}(old) + \eta X_i OutErr_j$$

ขั้นที่ 10 นำ Pattern ชุดต่อไป (p = p+1)

กลับไปทำขั้นตอนที่ 5 จนครบทุก Pattern

ขั้นที่ 11 ตรวจสอบค่าผิดพลาดรวมของโครงข่ายของทุก Pattern

$$totalErr = \sum_{p}^{nPattern} \left(\frac{1}{2} \sum_{j=0}^{nOutput} (OutErr_j)^2 \right)$$

ถ้าค่าผิดพลาดรวมมากกว่าค่าผิดพลาดต่ำสุดที่กำหนดไว้

กลับไปทำขั้นตอนที่ 3

ถ้าค่าผิดพลาดรวมน้อยกว่าหรือเท่ากับค่าผิดพลาดต่ำสุดที่กำหนดไว้

จบการทำงาน

ขั้นตอนการทำงานของ อัลกอริทึม ADALINE (ADN)

ขั้นที่ 1 จัดเตรียมข้อมูลอินพุตและเอาต์พุตเป้าหมาย เข้าสู่หน่วยความจำ โดยกำนดให้ จุดภาพที่เป็น 1 มีค่าเท่ากับ +1 จุดภาพที่เป็น 0 มีค่าเท่ากับ -1

ขั้นที่ 2 สุ่มค่าน้ำหนักเป็นค่าเริ่มต้น (อยู่ในช่วงที่ผู้ใช้กำหนด)

ขั้นที่ 3 เมื่อเงื่อนไขเป็นจริง ให้ทำขั้นตอนที่ 4 ถึง 11

ขั้นที่ 4 สำหรับแต่ละคู่ข้อมูลอินพุต-เอาพุต เมื่อ p=0 ; p<nPattern

ให้ทำขั้นตอนที่ 5 ถึง 10

ขั้นที่ 5 จัดเตรียมข้อมูลอินพุตของ Pattern p เข้าสู่ชั้นอินพุต

ขั้นที่ 6 คำนวณผลรวมของแต่ละนิวรอลเซลล์จากสมการ

$$net_j = \sum_{i=0}^{n} w_{ji} x_i + b_j$$

ขั้นที่ 7 นำเอาต์พุตที่ได้มาหาค่าพิกัด

ถ้า $net_j >= 0$ $LayerOut_j = +1$

ถ้า $net_j < 0$ $LayerOut_j = -1$

ขั้นที่ 8 หาค่าผิดพลาดของโครงข่ายในแต่ละโหนดได้จากสมการ

$$OutErr_j = (target_j - net_j)$$

ขั้นที่ 9 นำค่าผิดพลาดที่ได้มาใช้ในการปรับนำหนักจากสมการ

$$W_{ji}(new) = W_{ji}(old) + \eta x_i OutErr_j$$

ขั้นที่ 10 นำ Pattern ชุดต่อไป (p = p+1)

กลับไปทำขั้นตอนที่ 5 จนครบทุก Pattern

ขั้นที่ 11 ตรวจสอบค่าผิดพลาดรวมของโครงข่ายของทุก Pattern จากสมการ

$$totalErr = \sum_{p}^{nPattern} \left(\frac{1}{2} \sum_{j=0}^{nOutput} (OutErr_j)^2 \right)$$

ถ้าค่าผิดพลาดรวมมากกว่าค่าผิดพลาดต่ำสุดที่กำหนดไว้

กลับไปทำขั้นตอนที่ 3

ถ้าค่าผิดพลาดรวมน้อยกว่าหรือเท่ากับค่าผิดพลาดต่ำสุดที่กำหนดไว้

จบการทำงาน

ข้อสังเกต ADALINE ต่างจาก Preceptron คือกำหนดให้อินพุตเป็น Bipolar {-1,+1} และในขั้นที่ 8 การหาค่าความผิดพลาดจะใช้เอาต์พุตของนิวรอลเซลล์ โดยไม่ได้ผ่านฟังก์ชัน threshold หากพิจารณา จากภาพที่ 6 เปรียบเทียบกับ ภาพที่ 7 ในบทที่ 2 จะสังเกตได้ชัด

ขั้นตอนการทำงานของ อัลกอริทึม Back Propagation (BPN)

ขั้นที่ 1 จัดเตรียมข้อมูลอินพุตและเอาต์พุตเป้าหมาย เข้าสู่หน่วยความจำ

ขั้นที่ 2 สุ่มค่าน้ำหนักเป็นค่าเริ่มต้น (อยู่ในช่วงที่ผู้ใช้กำหนด)

ขั้นที่ 3 เมื่อเงื่อนไขเป็นจริง ให้ทำขั้นตอนที่ 4 ถึง 18

ขั้นที่ 4 สำหรับแต่ละคู่ข้อมูลอินพุต-เอาพุต เมื่อ Pattern p=0 ; p<nPattern

ให้ทำขั้นตอนที่ 5 ถึง

ขั้นที่ 5 จัดเตรียมข้อมูลอินพุตของ Pattern p เข้าสู่ชั้นอินพุต

Forward-Propagation:

ขั้นที่ 6 คำนวณผลรวมของแต่ละนิวรอลเซลล์ของทุกชั้น k

เมื่อเงื่อนไขเป็นจริง (k<nLayer) ให้ทำขั้นตอนที่ 7 ถึง 9

ขั้นที่ 7 คำนวณผลรวมของแต่ละนิวรอลเซลล์ของชั้นที่ k ของทุกๆ

โหนด j ที่เชื่อมโยงกับอินพุต i ได้จากสมการ

$$net_{kj} = \sum_{i=0}^{n} w_{kji}\, x_i + b_{kj}$$

ขั้นที่ 8 นำเอาต์พุตที่ได้มาหาค่าพิกัดของแต่ละโหนด j จากสมการ

$$LayerOut_{kj} = \frac{1}{1+e^{-(net_{kj})}}$$

ขั้นที่ 9 เลื่อนมาคำนวณชั้นถัดไป (k = k+1)

ไปขั้นที่ 6

Backward-Propagation:

ขั้นที่ 10 หาสัญญาณค่าผิดพลาดของโครงข่ายที่ชั้นเอาต์พุตในแต่ละโหนด

ให้ k เป็นชั้นเอาต์พุต j เป็นโหนดของชั้นเอาต์พุต

$$LayerErr_{kj} = (target_j - LayerOut_{kj})\, LayerOut_{kj}(1 - LayerOut_{kj})$$

ขั้นที่ 11 หาสัญญาณค่าผิดพลาดของชั้นที่ซ่อนอยู่เริ่มจากชั้นก่อนหน้าชั้นเอาต์พุต

(nLayer-1) ไปจนถึงหลังชั้นอินพุต

เมื่อเงื่อนไขเป็นจริง (k >=0)ให้ทำขั้นตอนที่ 12 ถึง 13

ขั้นที่ 12 คำนวณหาค่าผิดพลาดของแต่ละชั้นเมื่อ k = nLayer-1 ,k >=0

เมื่อ k เป็นชั้นที่จะหาค่าผิดพลาด k+1 เป็นชั้นถัดมา

ค่าผิดพลาดของชั้น k หาได้จาก

$$LayerErr_{ki} = \sum_{j} (LayerErr_{(k+1)j} * W_{(K+1)ji}) * LayerOut_{kj} * (1 - LayerOut_{kj})$$

ขั้นที่ 13 ลดชั้น k ลง 1 (k = k-1)

กลับไปทำขั้นตอนที่ 11 จนครบทุกชั้น

Adaptive Weights:

ขั้นที่ 14 นำค่าผิดพลาดที่ได้มาใช้ในการปรับน้ำหนักทุกชั้น

เริ่มจากชั้นก่อนหน้าชั้นเอาต์พุต (nLayer-1) ไปจนถึงหลังชั้นอินพุต

เมื่อเงื่อนไขเป็นจริง (k >=0)ใช้ทำขั้นตอนที่ 15 ถึง 16

ขั้นที่ 15 ทำการปรับน้ำหนักของแต่ละชั้นเมื่อ k = nLayer-1 ,k >=0

หาขนาดของน้ำหนักที่จะปรับ

$deltaW = \eta * LayerErr_{kj} * LayerOut_{(k-1)i}$

($LayerOut_{(k-1)i}$ เป็นเอาต์พุตของชั้นก่อนหน้านี้(k-1) จะถูกใช้เป็นอินพุตของชั้น k นี้)

ทำการปรับน้ำหนักที่ชั้น k ของทุกจุดเชื่อมโยง i ที่ต่อกับทุกโหนด j ในชั้นนั้น

$W_{kji}(t+1) = W_{kji}(t) + deltaW_{kji} + \alpha\, deltaWold_{kji}$

เก็บขนาดของน้ำหนักที่เปลี่ยนไป

$deltaWold_{kji} = deltaW$

ขั้นที่ 16 ลดชั้น k ลง 1 (k = k-1)

กลับไปทำขั้นตอนที่ 15 จนครบทุกชั้น

ขั้นที่ 17 นำ Pattern ชุดต่อไป (p = p+1)

กลับไปทำขั้นตอนที่ 5 จนครบทุก Pattern

ขั้นที่ 18 ตรวจสอบค่าผิดพลาดรวมของโครงข่ายของทุก Pattern จากสมการ

$$totalErr = \sum_{p}^{nPattern} \left(\frac{1}{2} \sum_{j=0}^{nOutput} (OutErr_j)^2 \right)$$

ถ้าค่าผิดพลาดรวมมากกว่าค่าผิดพลาดต่ำสุดที่กำหนดไว้

กลับไปทำขั้นตอนที่ 3

ถ้าค่าผิดพลาดรวมน้อยกว่าหรือเท่ากับค่าผิดพลาดต่ำสุดที่กำหนดไว้

จบการทำงาน

ขั้นตอนการทำงานของ อัลกอริทึม SELF-ORGANIZING MAPS (SOM)

ขั้นที่ 1 จัดเตรียมข้อมูลอินพุตเข้าสู่หน่วยความจำ

พร้อมทั้งทำ normalize อินพุต(จากสมการ [2.20])

ขั้นที่ 2 สุ่มค่าน้ำหนักเป็นค่าเริ่มต้น (อยู่ในช่วงที่ผู้ใช้กำหนด)

พร้อมทั้งทำ normalize น้ำหนัก

ขั้นที่ 3 เมื่อเงื่อนไขเป็นจริง (อัตราการเรียนรู้ (η) มีค่ามากกว่า 0

และ ยังไม่เท่ากับจำนวนรอบที่กำหนดไว้) ให้ทำขั้นตอนที่ 4 ถึง 10

อัตราการเรียนรู้ (η) มีค่ามากกว่า 0 *และ* ยังไม่เท่ากับจำนวนรอบที่กำหนดไว้

ขั้นที่ 4 สำหรับแต่ละอินพุต เมื่อ Pattern p=0 ; p<nPattern

เมื่อเงื่อนไขเป็นจริง (p<nPattern)ให้ทำขั้นตอนที่ 5 ถึง 8

ขั้นที่ 5 จัดเตรียมข้อมูลอินพุตของ Pattern p เข้าสู่ชั้นอินพุต

ขั้นที่ 6 หา winner neuron จากโหนดเอาต์พุตที่มี Euclidean distane น้อยที่สุด

จากสมการ

$$d_j = \sum_{i=0}^{n} \left(x_i - w_{ji} \right)^2$$

ขั้นที่ 7 ทำการปรับน้ำหนักของทุกๆจุดเชื่อมโยง i ของโหนด j ที่อยู่ในรัศมี

ของ neighborhood รอบๆโหนด j ($j \in N_c$) จากสมการ

$$W_{ji}(t+1) = W_{ji}(t) + \eta(t)\left(x_i - W_{ji}(t) \right)$$

ขั้นที่ 8 นำ Pattern ชุดต่อไป (p = p+1)

กลับไปทำขั้นตอนที่ 5 จนครบทุก Pattern

ขั้นที่ 9 ลดอัตราการเรียนรู้ (η) ลง (นำอัตราการเรียนรู้ลบกับค่าที่กำหนดไว้)

LearnConts = LearnConts - LearnDec

ลด neighborhood ลง

NeighborSize = NeighborSize - NeighborDec

กลับไปทำขั้นตอนที่ 4

ขั้นที่ 10 ตรวจสอบการทำงาน

ถ้าอัตราการเรียนรู้มีค่ามากกว่า 0 *และ* ยังไม่เท่ากับจำนวณรอบที่ตั้งไว้

กลับไปทำขั้นตอนที่ 3

ถ้าอัตราการเรียนรู้มีค่าน้อยกว่าหรือเท่ากับ 0 *หรือ* เท่ากับจำนวณรอบที่ตั้งไว้

จบการทำงาน

ภาคผนวก ง
SOURCE CODE

```
/*************************************************/
/*                                               */
/*              NEURAL NETWORK TOOLS             */
/*                   MAIN PROGRAM                */
/*                   NEUMAIN.C                   */
/*                                               */
/*************************************************/
#ifdef __BORLAND__
 #ifndef __HUGE__
 #error This Programe must Compile in Huge model
 #endif
#endif
#include <alloc.h>
#include <bios.h>
#include <conio.h>
#include <ctype.h>
#include <dir.h>
#include <dos.h>
#include <io.h>
#include <math.h>
#include <process.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <stdarg.h>

 #define  MaxSubDir              25
 #define  MaxFiles               170
 //$ count of pulldown & max frame (pulldown+popup)
 #define  MAX_PULLDOWN           4
 #define  MAX_FRAME              6
 #define  ALG_PCN                0
 #define  ALG_ADN                1
 #define  ALG_BPN                2
 #define  ALG_SOM                3
```

```
#define  MaxLayer          20
#define  MaxX              639
#define  MaxY              479
//return value
#define  Cancel            -27
#define   ExitProg         -24
#define  ChangeLeft        -10
#define  ChangeRight       -11
#define  EndofProg         -4
#define  ResultTo          -3
#define  KeyBoard          0xFE00
#define  Mouse             0xFF00
//button kind
#define  Kind_CANCEL    0
#define  Kind_OK        1
#define  Kind_OK_CANCEL 2
#define  K_FRAME        0
#define  K_BUTTON       1
#define  K_CHECKBOX     2
//return button
#define  NO             0
#define  OK             1
#define  CANCEL         2
#define  MouseDB        5

#define  TEMP_BANK       0
#define  NETWORK_BANK    2
#define  WEIGHT_BANK     3
#define  READ_MODE       0
#define  WRITE_MODE      1

struct PatternHeader{
 char code[3];              //code BP
 char nPat;                 //n of pattern
 char Xsize;                //Xsize
 char Ysize;                //Ysize
 char nOut;                 //n Output
 int  target;               //Output target
};
struct FileDetail{
  char name[13];                    //found file name
  long fsize;                //file size
  char attrib;               //attribute found
  unsigned int  fdate;       //file date
```

```
  unsigned int  ftime;          //file time
};
struct NetWorksCommand {
  char NetName[36];                  //NetWork Name.NET
  char Describtion[36];         //Describtion
  char Inputfile[36];           //name of Input Pattern File *.GRP
  short NetLayer;               //Size of NetWork Layer
  char hiddenSize[36];          //Size of each hidden Layer in string
  short InputXsize;             //Size of input x size
  short InputYsize;             //Size of input y size
  short nOutput;                //Size of Output Layer (UnSupervised)
  short InputSign;              //Use Input Signed
  float Wmin;                   //Random Weight min
  float Wmax;                   //Random Weight max
  short AlgNo;                  //Algrorithm number
  short TranferFn;              //Type of tranfer function
  float LearnConst;             //Learning Constant
  float LearnDec;               //Learning Constant decrease
  float Momentum;               //momentum of BPN
  float Errlevel;               //min Error Level
  short Neighbor;               //Neighborhood Size
  short NeigDec;                //Neighborhood decrease
  short MaxLoop;                //Max loop to procress
};
//function to use.....
unsigned int ReadInput(struct Win_frame *Win);
unsigned int GetMouseItem(struct Win_frame *Win);
int GetMenu (struct Win_frame *Win);
void SaveArea(struct frame_Board *Area,char *ptr);
void RestorArea(struct frame_Board *Area,char *ptr);
int FileMan(char *Header,char *filename,char *Spec,char mode);
int GgetString(int ID,int Tx,int Ty,int MAXstring,
               struct BoxRect *Rect,char *StrPtr,char HisStr[5][36],
               struct Win_frame *Win,char Status);
void DisplayMem(void);
void InsertComma(char *Str);
int sort_function( const void *a, const void *b);
void FileInfrom(struct FileDetail *itemPtr,char *StrInfrom);
void ErrMessage(int num);
void Beep(unsigned Frequency,unsigned Delay);
//Neural.....`
void InitialNetwork(void);
void SetupNetworks1(void);
void SetupNetworks2(void);
void Working(char *Str,int percent);
```

```
int Str2intArray(char *Str,int *data);
void ArrayInt2Str(char *Str,int *arr,int count);
void Int2BinAry(int scr,int ndigit,char *Str);
int NetErr(int PatLoad);
short ReadPatternInform(char *name);
short ReadPattern(char *filename);

//File.....`
unsigned long SearchFile(char *filename);
unsigned long  ReadBFile(char *name,char *obj,long offset);
short FileExist(char *filename,short Mode);
int WriteBFile(char *name,char *obj,unsigned long size,char *mode);
int SaveXMemBank(char *filename,int Bank);
int LoadXMemBank(char *filename,int bank);
int IsCreateTMP(char *filename,int bank);
short WriteNetCmd(void);
short ReadNetCmd(void);

void ShowNetInform1(short Sx,short Sy);
void ShowNetInform2(short Sx,short Sy);
void gprintfxy(short x,short y,char *fmt, ...);
short KillChar(char *SourceStr,char *TargetStr,char Kill);

#include "gmouse.h"
#include "grapfix.h"
#include "keycode.h"
#include "xmem.h"

//define struct...
struct BoxRect{
  int ID;
  int Sx,Sy,Ex,Ey;
  unsigned Kind;
  char *Str,Status;
};
struct Win_frame{
  int ID;                          // Window ID
  char Type;                       // Type of Windows
  char *name;                      // Name of Windows
  struct frame_Board *Frames;      // frame pointer
  char EdgeWidth,HelpBar,MenusBar;     // edge & status
  int startx, endx, starty, endy;      // Position of Windows
  int frame_No, count;             // menu_No. to active
  int ClnColor;                    // fill Client color
  int ClientSx,ClientSy,ClientEx,ClientEy;
```

```
  int MbarSx,MbarSy,MbarEx,MbarEy;
  int HelpBarSx,HelpBarSy,HelpBarEx,HelpBarEy;
  int active;
  int Component;
  struct BoxRect *table;
};
struct frame_Board {
  char *name;                  //name of this board
  char HotKey;                  //position of Hot Key in name array
  int startx,starty,endx,endy;   //position of frame
  int namex,namey,namel;          //position of the frame name & lenght
  char datakind;                //Kind of String data
  char **menu;                   //point to member string          (kind 0)
  char * StrP;                  //point to frist address of data  (Kind 1)
  int  Strlen;                  //lenght off String array (if have Kind 1)
  int  first;                  //first item to show
  int  index;                   //position of hightlight in frame
  int item_No;                   //item_No to select
  unsigned Key;                        //hot key of this frame
  char *keys;                   //hot key of items
  int count;                   //count of member
  int Maxline;                  //Max line to show in frame
  int LineSpc;                  //line space
  int active;                  //status
};
struct WinMessage{
  int frame;
  int item;
  int ID;
  char DBclick;
};
//define function...
void CreateWindows(
int ID,
char Type,
char *Name,
int count,
int Sx,int Sy,int Ex,int Ey,
int color,char EdgeWidth,
struct Win_frame *Win,
char MenusBar,
char HelpBar,
struct frame_Board *board,
struct BoxRect *Rect,
int RectCount
```

```
);
void ShowWindow(struct Win_frame *Win,char *HBstr);
void ClearClient(struct Win_frame *Win,int color);
void HeadTitle(struct Win_frame *Win,char Status);
void HelpBar(struct Win_frame *Win,char *Str);
void HightLight(struct frame_Board *board,char paint);
void WriteHelpBar(struct Win_frame *Win,char *Str);
int MakeMenu(int PdNo,int x,int y,char *name,char HotKey,char **menu,
                    unsigned Key,char *keys,int linesize);
void MakePopup(struct frame_Board *board,int x,int y,char **menu,char *keys,
              int PdNo,int item_No,int linesize);
void MakeFixFrame(int ID,int Sx,int Sy,char *name,char HotKey,int MaxView,
    int MaxChar,char *Str,int Strlen,struct frame_Board *Fxboard,
    struct BoxRect *Rect);
void WriteFrame(struct frame_Board *board,char name,char item,char shadow);
void WriteName(struct frame_Board *board,char mode);
void WriteItem(struct frame_Board *board);
void DrawFrame0(struct Win_frame *Win);
void DrawFrame1(int Sx,int Sy,int Ex,int Ey,char *header);
int Popup_Menu(struct Win_frame *Win,int Popframe);
//Component.....
void button(int Sx,int Sy,char *Str,char press,char Select);
int AskCompo(int RectNo,struct BoxRect *Rects,struct Win_frame *Win);
int SelectCompo(struct Win_frame *Win,int passto);
void CreateCompo(int ID,char Kind,int Sx,int Sy,struct BoxRect *Rect,
                    char *Str, char status);
void CheckBox(struct BoxRect *Rects,int Sx,int Sy,char status);
int MessageWin(char *Header,char *Mesg1,char *Mesg2,char button,
              char pic,char BakScreen);
int Askframe(struct Win_frame *Win,char Inform);
void PicNeural(int Sx,int Sy);
void PicInform(int Sx,int Sy);
//start Gobal...
//$
char *NetWorks[]= {
  "Load  Inputpattern\x0\x1",
  "Setup NetWorks\x0\x2",
  "Delete  File\x0\x3",
  "Quit          Alt-X\x0\x4"
};
char *ALGorithms[]= {
  "Perceptron\x0\x5",
  "Adaline\x0\x6",
  "Backpropagation\x0\x7",
  "Self Organize Maps\x0\x8"
```

```
};
char *Run[]= {
  "Training\x0\x9",
  "Networks config\x0\xF"
};
char *Help[]={
  "Help\x0\xA",
  "About...\x0\xB"
};
char *Algorithms[]= {
  "Perceptron\x0\x0",
  "Adaline\x0\x1",
  "Backpropagation\x0\x2",
  "Self Organize Maps\x0\x3"
};
struct Win_frame Windows;
struct frame_Board frame[MAX_FRAME];
struct WinMessage MESG;
char HelpStr[80];
unsigned MaxFrameSize;

#include "thais.h"
//Gobal.....
struct FileDetail SubDir[MaxSubDir],Fdptr[MaxFiles];
int Mflag=0;

struct NetWorksCommand NetCmd;
int neuronSize[MaxLayer];
int StartUp,Networkflag,Patflag,Inputflag;
int nPattern,PatternNo,neuronLayer;
void main(int argc,char *argv[])
{
int Sx,Sy,Ex,Ey,Axis;
int MenuSelect,x,y,CanDo,CanWrite;
char filename[MAXPATH],buff[30];
unsigned long nbyte;
//for test
char *scrptr;

  Inputflag = OFF;
  InitXMem();
  InitialNetwork();
  nbyte = ReadBFile("normal.fon",Tfont,0);
  mouse_present();
```

```
sw_graph();
setbkcolor(BLACK);                    // Real Blackground
  define_graphics_cursor((unsigned char*) dotplus,dotplus[32],dotplus[33]);
set_mouse_cursor(620,427);
strcpy(HelpStr,"<Alt-X> to Exit , Alt+ <RED> letter to Select Menu");
CreateWindows(0,0,"NUERAL NETWORK MAIN",MAX_PULLDOWN,0,0,MaxX,MaxY,
                LIGHTGRAY,EdgeSize,&Windows,ON,ON,frame,NULL,NULL);
// initial PullDown menu in Windows with Auto Place
  x=0;
  y = Windows.ClientSy; // last of FrameBar
            //$                                    //$
  x = MakeMenu(0, x,y,"NetWorks"   ,0,NetWorks   ,AltN,"LSDQ"  ,20);
  x = MakeMenu(1, x,y,"Algorithms",0,ALGorithms,AltA,"PABS"  ,20);
  x = MakeMenu(2, x,y,"Run"        ,0,Run        ,AltR,"TN"    ,20);
  x = MakeMenu(3,-1,y,"Help"       ,0,Help       ,AltH,"AH"    ,20);

// Show Main Windows & detail  ? Where HelpStr
 ShowWindow(&Windows,HelpStr);

 mouse_cursor_on();
if(argc<2)
 MessageWin("START","NEURAL TOOLS 1.0    ","KMITL 1996      ",0,0,ON);
while(MenuSelect != ExitProg)
{
 MenuSelect = GetMenu (&Windows);
   switch(MenuSelect)
    {
//menu
     case 1: //load pattern
          CanDo = FileMan("Select Pattern Name",
                  filename,"*.GRP",READ_MODE);
         if(CanDo==OK)
           {
            CanDo = ReadPattern(filename);
            if(CanDo>0)
             {
              KillChar(filename,NetCmd.Inputfile,' ');
              Inputflag = ON;
             }
            else
              Inputflag = OFF;
           }
         if(NetCmd.AlgNo==ALG_SOM)
            ShowNetInform2(20,70);
         else
```

```
                ShowNetInform1(20,70);
                break;
//[1] select algorithms
  case 5:
  case 6:
  case 7:
  case 8:
          NetCmd.AlgNo = MenuSelect-5;
//[2] select Input Pattern
          if(Inputflag==OFF)
          {
              CanDo = FileMan("Select Pattern Name",
                      filename,"*.GRP",READ_MODE);
              if(CanDo==OK)
              {
                CanDo = ReadPattern(filename);
                if(CanDo>0)
                {
                 KillChar(filename,NetCmd.Inputfile,' ');
                 Inputflag = ON;
                }
                else
                 Inputflag = OFF;
              }
              else
                break;
          }
  case 2: //setup
//[3] setup Layer
          if(NetCmd.AlgNo!=ALG_SOM)
              { SetupNetworks1(); ShowNetInform1(20,70); }
          else
              { SetupNetworks2(); ShowNetInform2(20,70); }
          break;
    case 3: //delete file
          CanDo = FileMan("Delete File",filename,"*.*",READ_MODE);
          if(CanDo==OK)
              remove(filename);
          break;
    case 4://Exit Program
          //closegraph();
          exit(0);
    case 9://training
          if(Inputflag == OFF)
          {
```

```
                MessageWin("ERROR","NO FILE SELECT   ",NULL,1;0,ON);
                break;
            }
            CanDo = WriteNetCmd();
            if(!CanDo)
                {MessageWin("WRITE ERROR","WRITE FILE ERROR",NULL,1,0,ON);
                break;}

                switch(NetCmd.AlgNo)
                {
                case ALG_PCN:
                            strcpy(filename,"NEUPCN.EXE");
                            break;
                case ALG_ADN:
                            strcpy(filename,"NEUADN.EXE");
                            break;
                case ALG_BPN:
                            strcpy(filename,"NEUBPN.EXE");
                            break;;
                case ALG_SOM:
                            strcpy(filename,"NEUSOM.EXE");
                            break;;
                }

                CanDo = FileExist(filename,READ_MODE);
                if(CanDo==OK)
                {
                MessageWin("INITIAL","WAIT...   ",
                        "INITTAIL SYSTEM  ",0,0,OFF);
                closegraph();
                execl("NEULINK.COM","NEULINK.COM",filename,NULL);
                }
//out to run overlay.....
            break;
        case 0xA0: //shift help
        case 0x0A: //help
            if(MenuSelect==0x0A)
                strcpy(filename,"NEURAL.HLP");
            if(MenuSelect==0xA0)
                strcpy(filename,"KEY.HLP");

            CanDo = FileExist(filename,READ_MODE);
            if(CanDo!=OK)
                break;
            XMem2File("N01.SWP");          //~
```

```
        scrptr = SaveScr(Windows.ClientSx,Windows.ClientSy,
                         Windows.ClientEx,Windows.ClientEy);
        XMem2File("SCREEN.SWP");          //~

        define_graphics_cursor((unsigned char*)arrow_cursor,
                    arrow_cursor[32],arrow_cursor[33]);
        VeiwTxtThai(filename,3,2,72,15,1,WHITE,BLUE);

        File2XMem("SCREEN.SWP");          //~
        RestoreScr(scrptr);
        remove("SCREEN.SWP");
        File2XMem("N01.SWP");             //~
        remove("N01.SWP");
        WriteHelpBar(&Windows,HelpStr);   //Restore Old Help
        define_graphics_cursor((unsigned char*) dotplus,dotplus[32],
                                                dotplus[33]);
        break;
case 0x0B: //About...
        MessageWin("About","NEURAL TOOLS 1.0",NULL,1,0,ON);
        break;
case 0x0C://Information
        MessageWin("TEST","TEST Hello World",
                   "This is Neral World",2,1,ON);
        break;
case 0x0F: //network config
        if(NetCmd.AlgNo==ALG_SOM)
            ShowNetInform2(20,70);
        else
            ShowNetInform1(20,70);
        break;
case 0xEE://view Pattern group
        XMem2File("N01.SWP");             //~
        scrptr = SaveScr(Windows.ClientSx,Windows.ClientSy,
                         Windows.ClientEx,Windows.ClientEy);
        XMem2File("SCREEN.SWP");          //~
        CanDo = FileMan("View File",filename,"*.GRP",READ_MODE);
        if(CanDo==OK)
        {
            WriteHelpBar(&Windows,"Press <ESC> Back to Main Menu");
            VeiwTxtThai(filename,3,2,72,15,1,WHITE,BLUE);
        }
        File2XMem("SCREEN.SWP");          //~
        RestoreScr(scrptr);
        remove("SCREEN.SWP");
        File2XMem("N01.SWP");             //~
```

```
                remove("N01.SWP");
                WriteHelpBar(&Windows,HelpStr);      //Restore Old Help
                break;
        case 0xF0:
                CanDo = ReadNetCmd();
                if(NetCmd.AlgNo==ALG_SOM)
                    ShowNetInform2(20,70);
                else
                    ShowNetInform1(20,70);
                break;
      }

 }

closegraph();
}//end main

//[*]
int SaveXMemBank(char *filename,int bank)
{
FILE *fp;
int ReturnMesg,CanWrite;
char far *AdsPtr;

    ReturnMesg = 0;
    fp = fopen(filename,"wb"); // wb = write binary
    if(fp!=NULL)
     {
      //save lastmem
       AdsPtr = (char far *)&lastmem[bank];
           fwrite(AdsPtr,sizeof(char far*),1,fp);
      //save PreViousPtr
       AdsPtr = (char far *)&PreViousPtr[bank];
           fwrite(AdsPtr,sizeof(char far*),1,fp);
       fclose(fp);
      //save data in Bank
       CanWrite = WriteBFile(filename,MemBank[bank],ONE_MEM_BANK+1,"ab");
       if(CanWrite!=0)
            ReturnMesg = 1;
     }
  //reset bank
   lastmem[bank]     = MemBank[bank];
   PreViousPtr[bank] = NULL;
   return(ReturnMesg);
}//end
```

```
//[*]
int LoadXMemBank(char *filename,int bank)
{
FILE *fp;
int ReturnMesg;
char far *AdsPtr;
unsigned long nbyte;

  ReturnMesg = 0;
  fp = fopen(filename,"rb");
  if(fp!=NULL)
   {
   //restore lastmem
     AdsPtr = (char far *)&lastmem[bank];
     fread(AdsPtr,sizeof(char far*),1,fp);
   //restore PreViousPtr
     AdsPtr = (char far *)&PreViousPtr[bank];
     fread(AdsPtr,sizeof(char far*),1,fp);
     fclose(fp);
   //read data
     nbyte = ReadBFile(filename,MemBank[bank],sizeof(char far *)*2);
     if(nbyte!=0)
       ReturnMesg = 1;
    }
  remove(filename);          //clear it
  return(ReturnMesg);
}//end
//[*]
int IsCreateTMP(char *filename,int bank)
{
int createTMP;

     createTMP = IsBankUse(bank);    //this bank is use
     if(createTMP != 0)              //if used bank
       SaveXMemBank(filename,bank);          //swap

 return(createTMP);
}//end
//[*]
void Working(char *Str,int percent)
{
char StrBuff[85];
int Sx,Sy,Ex,Ey,toEx,midx,midy;
float Buff;
```

```
if(percent>100)
   percent = 100;
if(percent>0)
{
 Sx   = Windows.HelpBarSx;
 Sy   = Windows.HelpBarSy;
 Ex   = Windows.HelpBarEx;
 Ey   = Windows.HelpBarEy;
 Buff = (float)percent;
 Buff /= 100;
 Buff *= (Ex-Sx);
 toEx = (int)Buff;

 setfillstyle(SOLID_FILL,LIGHTGRAY);
 bar(Sx,Sy,Ex,Ey);
 setfillstyle(SOLID_FILL,RED);
 bar(Sx,Sy,Sx+toEx,Ey);

 midx = (Ex+Sx)/2;              // find center
 midy = (Ey+Sy)/2;
 sprintf(StrBuff,"%s %3d%%",Str,percent);
 setcolor(WHITE);
 settextjustify(CENTER_TEXT,CENTER_TEXT );
 outtextxy(midx,midy,StrBuff);
 }
}//end
//[*] !Err
void ErrMessage(int num)
{
char StrHead[]="Message";

 switch(num)
 {
  case 1:
     MessageWin(StrHead,"Can't Open File",NULL,1,1,ON);
     break;
  case 2:
     MessageWin(StrHead,"No Pattern Loaded",NULL,1,1,ON);
     break;
  case 3:
     MessageWin(StrHead,"Weight not Exist",NULL,1,1,ON);
     break;
  case 4:
     MessageWin(StrHead,"Data lenght Error",
                        "Can't use in Network",1,1,ON);
```

```
    break;
case 5:
    MessageWin(StrHead,"Not any Data to Save",NULL,1,1,ON);
    break;
case 6:
    MessageWin(StrHead,"Run FeedForward before",
                        " choose Run Backward",1,1,ON);
    break;
case 7:
    MessageWin(StrHead,"Run Backward before",
                        "Run Adaptive Weight",1,1,ON);
    break;
case 8:
    MessageWin(StrHead,"Not Enough Memory",NULL,1,1,ON);
    break;
case 9:
    MessageWin(StrHead,"File  Format  Error",NULL,1,1,ON);
    break;
case 10:
    MessageWin(StrHead,"Read File Error",NULL,1,1,ON);
    break;
default :
    MessageWin(StrHead,"Unkown Error",NULL,1,1,ON);

}
}//end

//[*] Iconv
//split String seperate by bank char to int array[k]
//return the number to seperate
//if data is NULL it only count the number can seperate
int Str2intArray(char *StrSource,int *data)
{
int len,i,k,t,flag;
char *p,Str[36],buff[15][15];

    strcpy(Str,StrSource);
    len = strlen(Str);
    k=0;
    flag=0;
    for(i=0;i<len+1;i++)
    {
     t = isdigit(Str[i]);
     if(t>0 && flag==0)        //find start digit
     {
```

```
            p=Str+i;                    //start digit
            flag =1;
        }
        if(t==0 && flag==1)     //find end digit
        {
            Str[i]=0;                   //end digit
            strcpy(buff[k],p);
            k++;
            flag=0;
        }
      }
      if(data != NULL)
      {
       for(i=0;i<k;i++)                 //converts data
            data[i] = atoi(buff[i]);
      }
    return(k);
}//end
//[*] Iconv
void ArrayInt2Str(char *Str,int *arr,int count)
{
int i,len;
char buff[15];

    buff[0] = 0;
    Str[0]  = 0;
    for(i=0;i<count;i++)
     {
      sprintf(buff,"%d ",arr[i]);
      strcat(Str,buff);
     }
    len = strlen(Str);
    Str[len-1] = 0;
}//end
//[*] Iconv
void Int2BinAry(int scr,int ndigit,char *Str)
{
int i,j;
unsigned Mark,result;

      j = ndigit-1;
      Mark = 1;
      for(i=0;i<ndigit;i++,j--)
      {
            result = scr&Mark;
```

```
            if(result>0)
              Str[j] = 1;
            else
              Str[j] = 0;
        Mark <<= 1;
       }
}//end
//[*]
//default in setup Network
void InitialNetwork(void)
{
  StartUp      = 0;
  Patflag      = 0;    //test Pattern loaded
  nPattern     = 1;    //use for allocate memory
  PatternNo    = 0;    //use in Calulate networks

   strcpy(NetCmd.NetName,"TDIGIT.NET");
   strcpy(NetCmd.Describtion,"Test Thai digit 0-9");
   NetCmd.Inputfile[0]  = 0;
   NetCmd.NetLayer      = 3;
   strcpy(NetCmd.hiddenSize,"10");    //conv //!
   NetCmd.nOutput       = 10;
   NetCmd.InputSign     = 0;
   NetCmd.Wmin          = -0.3;
   NetCmd.Wmax          = 0.3;
   NetCmd.AlgNo         = 2;
   NetCmd.TranferFn     = 0;
   NetCmd.LearnConst    = 0.15;
   NetCmd.LearnDec      = 0.01;
   NetCmd.Momentum      = 0.075;
   NetCmd.Errlevel      = 0.02;
   NetCmd.Neighbor      = 5;
   NetCmd.NeigDec       = 1;
   NetCmd.MaxLoop       = 150;

   neuronSize[1]        = 10;
   neuronLayer          = NetCmd.NetLayer-1;

}//end
//Read KeyBoard 16 Bit & Mouse Area
unsigned int ReadInput(struct Win_frame *Win)
{
unsigned int key,item,KeyReturn=0;
static long Dbstart,Dbend,Timetest;
static char Dbflag;
```

```
/* mouse tool*/
 int x,y,lx=800,ly=800;
 char buff[15],buff2[15];

    MESG.frame   =  -1;
    MESG.item    =  -1;
    MESG.ID      =  -1;
    MESG.DBclick =   OFF;

//if Double click time is long reset
    Timetest = biostime(0, 0L);
    if(Dbflag ==1 && (Timetest - Dbstart) > MouseDB )
      Dbflag = 0;

      if(bioskey(1))              // When key press
           KeyReturn=bioskey(0);   // What a Key
           //return (key);
// Read Mouse Botton
          if(Mflag==1){ //button press
            while(button_status()>0); // wait until release button
            Mflag = 0;
          }

 switch(button_status())
 {
   case 1:
// mouse tool1 left button Show Position
           if(Dbflag==0){
              Dbstart = biostime(0, 0L);
              Dbflag  = 1;}
           else{
              Dbend   = biostime(0, 0L);
              Dbflag  = 0;}
           if((Dbend - Dbstart)<MouseDB && (Dbend - Dbstart) > 0)
             {MESG.DBclick = ON;
              Beep(500,10);}
          count_left_press();//clear buffer
          mouse_cursor_off();
          setcolor(WHITE);
          setfillstyle(SOLID_FILL,LIGHTBLUE);
          bar(545,6,625,28);
          mouse_position(&(int)x,&(int)y);
          sprintf(buff,"%d",x);
          sprintf(buff2,", %d",y);
          outtextxy(550,21,buff);
```

```
            outtextxy(580,21,buff2);
            Mflag=1;
            mouse_cursor_on();
//end mouse tool1
            if(Win!=NULL)
              KeyReturn = GetMouseItem(Win);
            break;
// press right button
   case 2:
//mouse tool2 Big Cross
            mouse_cursor_off();
            setcolor(WHITE);
            setwritemode(XOR_PUT);
            if(lx<640 || ly<480){
             line(lx,0,lx,479);
             line(0,ly,639,ly);
            }
            mouse_position(&(int)x,&(int)y);
            line(x,0,x,479);
            line(0,y,639,y);
            lx = x;
            ly = y;
            setwritemode(COPY_PUT);
            Mflag=1;
            mouse_cursor_on();
 //end mouse tools2
            //KeyReturn = ESC;
            break;
   }//sw

   return(KeyReturn);
 }//end

 void SaveArea(struct frame_Board *Area,char *ptr)
 {
 int Sx,Sy,Ex,Ey;
 unsigned size;//@

     mouse_cursor_off();
     Sx   = Area->startx;
     Sy   = Area->starty;
     Ex   = Area->endx;
     Ey   = Area->endy;
     size = imagesize(Sx,Sy,Ex+4,Ey+4); //@
     getimage(Sx,Sy,Ex+4,Ey+4,ptr);     //Save screen
```

```
    if((int)*ptr != Ex+4-Sx || size>MaxFrameSize){     //@
            MessageWin("EXIT PROGRAM","DATA ERROR IN GETIMAGE",NULL,1,1,OFF);
            closegraph();
            exit(1);
     }
    mouse_cursor_on();
}
void RestorArea(struct frame_Board *Area,char *ptr)
{
int Sx,Sy;

    mouse_cursor_off();
    Sx   = Area->startx;
    Sy   = Area->starty;
    putimage(Sx,Sy,ptr,COPY_PUT);      //Restore
    mouse_cursor_on();
}



int GetMenu (struct Win_frame *Win)
{
int i,ReturnID,Pd_item,len,popmenu,PopupPtr;
unsigned key16,size,mdf;
struct frame_Board *board,*Isframe;
char draw,Scrflag;
char far *ScrSave;

//save for Pd_item over on

    XMem(&(void far *)ScrSave,MaxFrameSize,NONE);
    if(MaxFrameSize > ONE_MEM_BANK || ScrSave == NULL)
     {
      MessageWin("EXIT PROGRAM","NOT ENOUGH HEAP SPACE",
                    "FOR SAVE SCREEN ",1,1,OFF);
      closegraph();
      exit(1);
     }
   draw      = ON;
   Scrflag   = OFF;
   Pd_item   = NONE;
   ReturnID  = NONE;
   board     = Win->Frames;
   while(ReturnID == NONE)    //not value to return
    {
```

```
key16 = ReadInput(Win);
//Scan Hot Key for Alt-Key
  if(key16>0 && key16 != Mouse )
  {
   for(i=0;i < Win->count;i++)
   {
        if(key16 == board[i].Key)
        {
          key16 = KeyBoard; // Hold frame .No in i
          break;
        }
   }
  }
switch(key16)
{
  case AltX : //Exit Program
                ReturnID = ExitProg;
                break;
  case AltL : //Load Command
                ReturnID = 0xF0;
                break;
  case 0x5400://shift right and left F1
                ReturnID = 0xA0;
                 break;
  case FnF1 : //Help
                 ReturnID = 0x0A;
                 break;
  case FnF7 : //Veiw Weight
                ReturnID = 0x0B;
                 break;
  case FnF9 : //Train Network
                ReturnID = 9;
                 break;
  case AltV : //Veiw Pattern Group
                ReturnID = 0xEE;
                 break;

  case Mouse:
  case KeyBoard:
          switch(key16)
          {
            case Mouse:   Win->frame_No = MESG.frame;
                          break;
            case KeyBoard :
                          Win->frame_No = i;
```

```
                    break;
        }
//PullDown Loop
        while(Win->frame_No != NONE)
        {
         Isframe  = &(board[Win->frame_No]);
         Isframe->active =1;
         if(Scrflag==OFF)
          {
           SaveArea(Isframe,ScrSave);
           Scrflag =ON;            //set screen flag
          }
         if(draw == ON)
            WriteFrame(Isframe,2,2,ON);
         Pd_item =  Askframe(Win,OFF);
         if(draw == OFF)
           draw =  ON;
         switch(Pd_item)
         {

           case ExitProg   : ReturnID = ExitProg;
           case Cancel     :       Win->frame_No = NONE;
                                   Pd_item = NONE;
                                   break;
           case Mouse      : Win->frame_No = MESG.frame;
                                   break;
           case ChangeLeft :
                                (Win->frame_No)--;
                                if(Win->frame_No<0)
                                    Win->frame_No =   MAX_PULLDOWN-1;
                                break;
           case ChangeRight:
                                (Win->frame_No)++;
                                if(Win->frame_No  >= MAX_PULLDOWN)
                                    Win->frame_No =   0;
                                break;
           default  :
                  len = strlen(Isframe->menu[Pd_item]);
                  len--;       //point to last char
           //IF POPUP MENU
                  if(*(Isframe->menu[Pd_item]+len) == 0x10)   //the " char
                   {
                  //get the frame to point
                    PopupPtr =  (int)*(Isframe->menu[Pd_item]+len+2);
                    popmenu = Popup_Menu(Win,PopupPtr);
```

```
                    if(popmenu==Cancel)
                     {
                      Pd_item =  NONE;
                      draw     = OFF; //off draw frame
                     }
                    else
                     {
                      ReturnID = popmenu;
                      Win->frame_No = NONE;
                     }
                  }
               //NO is normal item
                     else
                     {
               //get ID form the tail of string
                       ReturnID  = (int)*(Isframe->menu[Pd_item]+len+2);
                       Win->frame_No = NONE;
                     }
               }//switch(Pd_item)
               if(draw==ON)
               {
                WriteName(Isframe,OFF);
                RestorArea(Isframe,ScrSave);
                Scrflag = OFF;                    //clear screen flag
                Isframe->active =0;
               }

             }//Win->frame_No != NONE
            break;
      }//switch(key16)

    }//while
   FreeXMem(ScrSave);
   return(ReturnID);
  }//end GetMenu

  /* GgetString */
  //return  0  when only draw frame
  //return  1  OK result reference in StrPtr
  //return -1  when press tab
  //return -27 when press ESC
  int GgetString(
  int ID,
  int Tx,
  int Ty,
```

```
int MAXstring,
struct BoxRect *Rect,
char *StrPtr,
char HisStr[5][36],
struct Win_frame *Win,
char Status
)
{
int i,j,index,hisnum=0,ReturnMesg;
unsigned int key16,keyOld=Notthing;
unsigned int len,insx;
unsigned char key8,count,startin=0;
char Str[2] =" ";
unsigned char InSert = 1,InsOld=0;

        ReturnMesg = -1;
/* Draw String Frame */
        mouse_cursor_off();
        setcolor(BLACK);
        settextjustify(LEFT_TEXT,LEFT_TEXT );
        setfillstyle(SOLID_FILL,BoardColor);
        bar3d(Tx,Ty,Tx+(MAXstring*8)+16,Ty+20,0,1);
        if(Status==OFF)    //Draw only frame
        {
          if(Rect != NULL)
           {
             Rect->ID     = ID;
             Rect->Sx     = Tx;
             Rect->Sy     = Ty;
             Rect->Ex     = Tx+(MAXstring*8)+16;
             Rect->Ey     = Ty+20;
             Rect->Kind   = 0;
             Rect->Str    = NULL;
             Rect->Status = NULL;
           }
          setcolor(BLACK);
          if(*(HisStr[0]) != NULL)
            outtextxy(Tx+8,Ty+16,HisStr[0]);
          mouse_cursor_on();
          return(0);
         }
        len=strlen(HisStr[0]);
        index=len;
        count=len;
  //paint file spec
```

```
        setfillstyle(SOLID_FILL,HightLightBar);
        bar(Tx+4,Ty+2,Tx+(len*8)+8,Ty+18);
        strcpy(StrPtr,HisStr[0]);
        setcolor(HightLightText);
        outtextxy(Tx+8,Ty+16,HisStr[0]);
        //setcolor(NormTextColor);
        //outtextxy(Tx+8+(index*8),Ty+18,"_");

// HightLight file spec
        mouse_cursor_on();
        key16=Notthing;
          while(key16==Notthing)
                key16=ReadInput(Win);
          switch(key16){
          case UP:
          case DOWN:
          case LEFT:
          case RIGHT:
          case BACKSPACE:
          case RIGHTTAB :
          case BACKTAB  :
                        break;
          case Mouse:
                        if(MESG.ID == ID || MESG.ID == 0)
                          break;
          case ENTER:
                        mouse_cursor_off();
                        setcolor(BLACK);
                        setfillstyle(SOLID_FILL,BoardColor);
                        bar(Tx+1,Ty+1,Tx+(MAXstring*8)+15,Ty+19);
                        outtextxy(Tx+8,Ty+16,HisStr[0]);
                        mouse_cursor_on();
                        if(key16==Mouse)
                          return(Mouse); //@ out of range
                        if(key16==ENTER)
                          return(OK);

          case ESC:
                      return(Cancel);
          default: keyOld=key16;      //keep key press
                     key16=CTRL_BKSPC;
          }
         mouse_cursor_off();
         setcolor(BLACK);
         setfillstyle(SOLID_FILL,BoardColor);
```

```
        bar(Tx+4,Ty+2,Tx+(len*8)+8,Ty+18);
        outtextxy(Tx+8,Ty+16,HisStr[0]);
        outtextxy(Tx+8+(index*8),Ty+18,"_");

        mouse_cursor_on();
        Tx+=8; /* change original Tx Ty to Text line */
        Ty+=16;
        while(key16!=ESC && key16!=ENTER)
        {
         if(startin==0)          //wait for kbhit in first
           startin=1;
         else
           do
           {
            key16=ReadInput(Win);
           }while(key16==NULL);

        key8  =(char) (key16 & 0xff);

mouse_cursor_off();
switch(key16)
 {
 case INSERT:  if(InSert)
                 InSert=0;
               else
                 InSert=1;
               InsOld=1;
               break;
 case Mouse:
               if(MESG.ID == ID || MESG.ID == 0)
                 break;
 case ENTER :  StrPtr[count]=0x0;
               //test notthing & the first char is space
               if(count==0 || StrPtr[0]==0x20)
                 {
                  key16=CTRL_BKSPC;      //clear it and start again
                  startin=0;             //loop again
                  break;
                 }
               else
                 {
                 //find redundancy
                  j=(-1);
                  for(i=0;i<5;i++)       // @ if Str>5
                   {
```

```
                    if(!strcmp(HisStr[i],StrPtr))
                     {
                         j=i;                //found it at j
                         break;
                     }
                    }
                   if(j==(-1))              //if not found j still -1
                   //rotate it down
                     for(i=0;i<4;i++)
                          strcpy(HisStr[4-i],HisStr[3-i]);
                   else
                   //rotate down and swap it first
                     for(i=0;i<j;i++)
                          strcpy(HisStr[j-i],HisStr[j-1-i]);

                   if(j!=0)      //if it already in [0] don't copy
                     strcpy(HisStr[0],StrPtr); //copy it to frist
                  }
                 setcolor(WHITE);
                 outtextxy(Tx+(index*8),Ty+2,"_");
                 mouse_cursor_on();
                 if(key16 == ENTER)
                   return(OK);
                 if(key16 == Mouse)
                   return(Mouse); //@ out of range

case ESC :
case BACKTAB  :
case RIGHTTAB :   setfillstyle(SOLID_FILL,BoardColor);
                  bar(Tx-7,Ty-15,Tx+(MAXstring*8)+7,Ty+3);
                  outtextxy(Tx,Ty,HisStr[0]);
                  setcolor(WHITE);
                  outtextxy(Tx+(index*8),Ty+2,"_");
                  mouse_cursor_on();
                  switch(key16)
                  {
                    case RIGHTTAB :return (ChangeRight);
                    case BACKTAB  :return(ChangeLeft);
                    case ESC      :return(Cancel);
                  }
case CTRL_BKSPC:
                  index=0;
                  count=0;
                  StrPtr[0]=0;
                  setcolor(BLACK);
```

```
                    setfillstyle(SOLID_FILL,BoardColor);
                    bar3d(Tx-8,Ty-16,Tx-8+(MAXstring*8)+16,Ty+4,0,1);
                    outtextxy(Tx,Ty+2,"_");
                    if(keyOld!=Notthing)
                     {
                      startin=0;        //key16 is first char to press
                      key16=keyOld;
                      keyOld=Notthing;
                     }
                      break;
case UP :       hisnum--;
                  if(hisnum<=0)
                          hisnum=0;
                  len=strlen(HisStr[hisnum]);
                  if(len==0){
                          hisnum++;
                          break;}
                  index=len;
                  count=len;
                  strcpy(StrPtr,HisStr[hisnum]);
                  setcolor(BLACK);
                  setfillstyle(SOLID_FILL,BoardColor);
                  bar3d(Tx-8,Ty-16,Tx-8+(MAXstring*8)+16,Ty+4,0,1);
                  outtextxy(Tx,Ty,HisStr[hisnum]);
                  outtextxy(Tx+(index*8),Ty+2,"_");
                  break;
case DOWN  :    hisnum++;
                  if(hisnum>4)
                          hisnum=4;
                  len=strlen(HisStr[hisnum]);
                  if(len==0){
                       hisnum--;
                       break;}
                  index=len;
                  count=len;
                  strcpy(StrPtr,HisStr[hisnum]);
                  setcolor(BLACK);
                  setfillstyle(SOLID_FILL,BoardColor);
                  bar3d(Tx-8,Ty-16,Tx-8+(MAXstring*8)+16,
                                                  Ty+4,0,1);
                  outtextxy(Tx,Ty,HisStr[hisnum]);
                  outtextxy(Tx+(index*8),Ty+2,"_");
                  break;
case LEFT  :  if(index==0)
                          break;
```

```
                setcolor(WHITE);
                outtextxy(Tx+(index*8),Ty+2,"_");
                setcolor(BLACK);
                index--;
                outtextxy(Tx+(index*8),Ty+2,"_");
                break;
case RIGHT : if(index==count || index==MAXstring )
                 break;
             setcolor(WHITE);
             outtextxy(Tx+(index*8),Ty+2,"_");
             setcolor(BLACK);
             index++;
             outtextxy(Tx+(index*8),Ty+2,"_");
             break;
case BACKSPACE : if(index==0)
                 break;
             setcolor(WHITE);
                 outtextxy(Tx+(index*8),Ty+2,"_");
             for(i=index-1;i<count;i++){
                      Str[0]=StrPtr[i];
                      outtextxy(Tx+(i*8),Ty,Str);}
        /* move data */
             for(i=0;i<=(count-index);i++)
                      StrPtr[index+i-1]=StrPtr[index+i];
             count--;
             index--;
        /* Write data */
             setcolor(BLACK);
             for(i=index;i<count;i++){
                      Str[0]=StrPtr[i];
                      outtextxy(Tx+(i*8),Ty,Str);}
             outtextxy(Tx+(index*8),Ty+2,"_");
                      break;
case DELETE : if(index==count)
                      break;
             setcolor(WHITE);
             for(i=index;i<count;i++){
                      Str[0]=StrPtr[i];
                      outtextxy(Tx+(i*8),Ty,Str);}
        /* move data */
             for(i=0;i<=(count-index);i++)
                      StrPtr[index+i]=StrPtr[index+i+1];
             count--;
        /* Write data */
             setcolor(BLACK);
```

```
                for(i=index;i<count;i++){
                        Str[0]=StrPtr[i];
                        outtextxy(Tx+(i*8),Ty,Str);}
                          break;
case HOME :
                setcolor(WHITE);
                outtextxy(Tx+(index*8),Ty+2,"_");
                index=0;
                setcolor(BLACK);
                outtextxy(Tx+(index*8),Ty+2,"_");
                          break;
case END :
                setcolor(WHITE);
                outtextxy(Tx+(index*8),Ty+2,"_");
                index=count;
                setcolor(BLACK);
                outtextxy(Tx+(index*8),Ty+2,"_");
                          break;
default:
          key8=toupper(key8);
          if(isprint(key8))
          {
/* write normal */
                if(index==count && count<MAXstring){
                  StrPtr[index] = key8;
                  Str[0] =key8;
                  setcolor(WHITE);
                  outtextxy(Tx+(index*8),Ty+2,"_");
                  setcolor(BLACK);
                  outtextxy(Tx+(index*8),Ty,Str);
                  outtextxy(Tx+(index*8)+8,Ty+2,"_");
                  index++;
                  count++;
                 }
/* Insert */
          if(index<count && InSert==ON && count<MAXstring){
          /* delete */
                setcolor(WHITE);
                outtextxy(Tx,Ty+2,"_");
                for(i=index;i<count;i++){
                        Str[0]=StrPtr[i];
                        outtextxy(Tx+(i*8),Ty,Str);
                }
          /* move data */
                StrPtr[count+1]=NULL;
```

```
                for(i=0;index<=(count-i);i++)
                        StrPtr[count-i]=StrPtr[count-i-1];
                StrPtr[index]=key8;
                count++;
        /* Write data */
                setcolor(BLACK);
                for(i=index;i<count;i++){
                        Str[0]=StrPtr[i];
                        outtextxy(Tx+(i*8),Ty,Str);
                }
        /* move to next index */
                setcolor(WHITE);
                outtextxy(Tx+(index*8),Ty+2,"_");
                index++;
                setcolor(BLACK);
                outtextxy(Tx+(index*8),Ty+2,"_");
        }
/* overwrite */
        if(index<count && InSert==OFF)
        {
        /* delete */
                setcolor(WHITE);
                outtextxy(Tx+(index*8),Ty+2,"_");
                Str[0]=StrPtr[index];
                outtextxy(Tx+(index*8),Ty,Str);
         /* write */
                setcolor(BLACK);
                StrPtr[index]=key8;
                Str[0]=StrPtr[index];
                outtextxy(Tx+(index*8),Ty,Str);
                index++;
                outtextxy(Tx+(index*8),Ty+2,"_");
        }

         } /*if(isprint(key8)) */
        } /* switch (key16)*/
        mouse_cursor_on();
        }/*while*/

        return(NULL);

}//end GgetString

void InsertComma(char *Str)
```

```
{
int len,i,j;

        len=strlen(Str);
// insert xxx,xxx
        if(len>3&&len<7)
        {
         Str[len+1]=0;
         for(i=0;i<3;i++)
             Str[len-i]=Str[len-(i+1)];
         Str[len-3]=',';
        }
// insert x,xxx,xxx
        if(len>6)
        {
          Str[len+2]=0;
          for(i=0,j=0;i<6;i++,j++)
          {
          if(i==3)
             j=4;
          Str[len+(1-j)]=Str[len-(i+1)];
          }
          Str[len-2]=',';
          Str[len-6]=',';
        }
}//end
void DisplayMem(void)
{
int i,x,use;
unsigned long mem;
char Buffer[15];

        mem = farcoreleft();
        sprintf(Buffer,"%ld",mem);
        InsertComma(Buffer);
        setcolor(WHITE);
        settextjustify(LEFT_TEXT,LEFT_TEXT );
        setfillstyle(SOLID_FILL,LIGHTBLUE);
        bar(40,10,200,25);
        outtextxy(45,23,Buffer);
        x=137;
        for(i=0;i<MEMORY_BANK;i++)
        {
         use = IsBankUse(i);
        if(use == 0)
```

```
            {
             setcolor(WHITE);
             outtextxy(x,23,"*");
            }
           else
            {
             setcolor(RED);
             outtextxy(x,23,"*");
            }
           x+=8;
          }
}
//[*]
int sort_function( const void *a, const void *b)
{
  return( strcmp((char *)a,(char *)b) );
}
//[*]
//Where is mouse press in ?
unsigned int GetMouseItem(struct Win_frame *Win)
{
int x1,x2,i,j,is,Mx,My,Sx,Sy,Ex,Ey,ES,slide;
int LH,board_No,RectSize;
unsigned int Mesg,MesgMouse;
struct frame_Board *board;

  Mesg      = 0;
  MesgMouse = 0;
  Mx        = horizontal_position();
  My        = vertical_position();

  MESG.frame  = -1;
  MESG.item   = -1;
  MESG.ID     = -1;

  RectSize  = sizeof(struct BoxRect);
  board     = Win->Frames;
  if(Win->frame_No != -1)
   {
    board_No  = Win->frame_No;
    LH        = Win->Frames[board_No].LineSpc;
    slide     = (LH-TextHight)/2;
   }

// check in MenuBar  return the name off Pulldown
```

```
if(Win->MenusBar==ON)
{
   Sy = Win->ClientSy - MenuBar;
   Ey = Win->ClientSy;
   //if in Y range
 if(My >= Sy &&  My<= Ey )
  {
    for(i=0;i< (Win->count);i++)
    {
   //load X name position
      x1 = board[i].namex - TextHight;
      x2 = board[i].namex + board[i].namel + TextHight;
   //Scan X position
       if(Mx>=x1&&Mx<=x2)
       {                          //is same and it active
            if(board_No==i && board[board_No].active==1)
               break;
            else
               {
                MESG.frame = i;          //change frame
                MesgMouse  = Mouse;      //message from mouse
                break;
               }
       }
    }
  }
}//if(Win->MenusBar==ON)

//check in PullDown Menu when it active
if(Win->frame_No != (-1) ) //&
 {
   Sx    = board[board_No].startx + EdgeSpace;
   Sy    = board[board_No].starty + slide;
   Ex    = board[board_No].endx   - EdgeSpace;
   Ey    = Sy+LH-1;
    if(board[board_No].active==1)
    {
     if(board[board_No].count < board[board_No].Maxline)
       x2 = board[board_No].count;
     else
       x2 = board[board_No].Maxline;
     //Region X
     if( Mx>=Sx && Mx<=Ex){
     //Scan Y position
           for(i=0,j=0;i < x2;i++,j+=LH)
```

```
            if(My>= Sy+j && My<= Ey+j )
             {
               MESG.item  = i;
               MesgMouse  = Mouse;
               break;
             }
       }
      }
    }
  //Scan it Rect table
    if(Win->table != NULL)
     {
      for(i=(Win->Component -1);i > -1;i--)
       {
        if(My >= Win->table[i].Sy  && My <= Win->table[i].Ey)
            {
             if(Mx >= Win->table[i].Sx && Mx <= Win->table[i].Ex)
             {
              MESG.ID    = Win->table[i].ID;
              MesgMouse  = Mouse;
              break;
             }
            }
       }
       is = i;
       switch(Win->table[is].Kind)
        {
           case K_CHECKBOX:
            if(Win->table[is].Status == ON)
              CheckBox(&(Win->table[is]),Win->table[is].Sx,Win->table[is].Sy,
                       OFF);
            else
              CheckBox(&(Win->table[is]),Win->table[is].Sx,Win->table[is].Sy,
                       ON);

            while(left_button_down());
            break;
         case K_BUTTON:
         //find other button is select
            for(i=0;i < Win->Component;i++)
             {
               if(Win->table[i].Kind == 1)
                {
                 if(Win->table[i].Status == ON) //if the other select
                  {
```

```
                Win->table[i].Status = OFF; // off it
                button(Win->table[i].Sx,Win->table[i].Sy,
                        Win->table[i].Str,0,0);
              }
             }
           }
          Win->table[is].Status = ON;
          button(Win->table[is].Sx,Win->table[is].Sy,
                  Win->table[is].Str,1,1);
          while(left_button_down());
          button(Win->table[is].Sx,Win->table[is].Sy,
                  Win->table[is].Str,0,1);
          break;
      }
    }//if(Win->table != NULL)

//Ctrl Block [-]
    ES = Win->EdgeWidth;
    Sx = Win->startx+ES;
    Sy = Win->starty+ES;
    Ex = Sx+24;
    Ey = Sy+24;

    if( Mx>Sx && Mx<Ex)
    if( My>Sy && My<Ey)
     {
        if(MESG.DBclick == ON && Win->ID == 0)
         {
          Mesg=MessageWin("EXIT PROGRAM","EXIT NEURAL TOOLS!",NULL,2,0,ON);
          if(Mesg==OK)
           { closegraph(); exit(0);}
         }
        else
          MesgMouse = 0;
      }
    if(MesgMouse)
     {
        if(MESG.item != NONE)
          MESG.ID = NONE;
        return (MesgMouse);
     }
    else
        return (ESC);       //return  When nothing
}//end
```

```
void FileInfrom(struct FileDetail *itemPtr,char *StrInfrom)
{
char AorP[3],Attrib[5],StrBuffer[11];
char *months[] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun",
                   "Jul", "Aug", "Sep", "Oct", "Nov", "Dec",};
int  junk, month, day, year, hour, minute;
int  len,i,j;
// fill attrib
        strcpy(Attrib,"....");
        if(itemPtr->attrib & FA_RDONLY)
                Attrib[0]='r';
        if(itemPtr->attrib & FA_ARCH)
                Attrib[1]='a';
        if(itemPtr->attrib & FA_SYSTEM)
                Attrib[2]='s';
        if(itemPtr->attrib & FA_HIDDEN)
                Attrib[3]='h';
// calculate date and time
        junk   = itemPtr->fdate;
        day    = junk % 32;
        junk   = junk / 32;
        month  = junk % 16;
        year   = junk / 16 + 1980+543;
        junk   = itemPtr->ftime / 32;
        minute = junk % 64;
        hour   = junk / 64;
// AM or PM ?
        strcpy(AorP, "am");
        if (hour == 12)
          strcpy(AorP, "pm");
        if (hour == 0)
           hour = 12;
        if (hour > 12){
           hour = hour - 12;
           strcpy(AorP,"pm");}

        sprintf(StrBuffer,"%ld",itemPtr->fsize);
        len=strlen(StrBuffer);
// insert xxx,xxx
        if(len>3&&len<7)
        {
         StrBuffer[len+1]=0;
         for(i=0;i<3;i++)
           StrBuffer[len-i]=StrBuffer[len-(i+1)];
         StrBuffer[len-3]=',';
```

```
        }
// insert x,xxx,xxx
        if(len>6)
        {
          StrBuffer[len+2]=0;
          for(i=0,j=0;i<6;i++,j++)
          {
           if(i==3)
              j=4;
           StrBuffer[len+(1-j)]=StrBuffer[len-(i+1)];
          }
          StrBuffer[len-2]=',';
          StrBuffer[len-6]=',';
        }
//Add all data to inform
        sprintf(StrInfrom,"%10s %4s %2d-%3s-%4d %2d:%02d %s",
                StrBuffer,Attrib,day,months[month-1],
                year,hour,minute,AorP);

}
//end FileInfrom

//[*] get file name and Spec
//return 1 when Success
//return 0 when notthing
int FileMan(char *Header,char *filename,char *Spec,char mode)
{
//struct FileDetail Fdptr[?],SubDir[?]; //in Gobal
struct Win_frame   FileWin;
struct frame_Board board[2];
struct ffblk       file_data;
int Sx,Sy,Ex,Ey,key,result,passto,prev,next,i,j,x,y;
int countfn,countsd,done,attrib,ReturnMesg;
char *p,Sort=1,descrip=0,ReadSubDir;
char FileSpec[36],SubSpec[36],StrBuffer[13];
char HisStr[5][36]={"*.C","*.*","*.EXE","*.COM","*.BAK"};
struct BoxRect Rects[8];
char far *ScrSave;

// for fnsplit();
int  flags,chd;
char drive[MAXDRIVE],dir[MAXDIR],file[MAXFILE],ext[MAXEXT];
char curpath[MAXPATH],Oldpath[MAXPATH],DefaultSpec[MAXEXT];
      ReturnMesg =0;
```

```
    if(Spec != NULL)
     {
      strcpy(HisStr[0],Spec);
      flags=fnsplit(Spec,drive,dir,file,ext);   //Extract path
      strcpy(DefaultSpec,ext);
     }
    HeadTitle(&Windows,OFF);        //OFF Main Windows
    Windows.active= 0;
    set_mouse_cursor(290,230);

    CreateWindows(2,0,Header,2,85,50,550,400,WHITE,EdgeSize,&FileWin,
                  OFF,OFF,board,Rects,8);
//for test
    Sx = FileWin.startx;
    Sy = FileWin.starty;
    Ex = FileWin.endx;
    Ey = FileWin.endy;
    ScrSave = SaveScr(Sx,Sy,Ex+4,Ey+4);

    ShowWindow(&FileWin,NULL);

// Save Oldpath
    getcwd(curpath,MAXPATH);
    strcpy(Oldpath,curpath);

    Sx = FileWin.ClientSx;
    Sy = FileWin.ClientSy;
    Ex = FileWin.ClientEx;
    Ey = FileWin.ClientEy;

    WriteHelpBar(&Windows,"<Reading File...>");
//..... Initial consist of frame.....
    GgetString  (1,Sx+144,Sy+31,35,&Rects[1],FileSpec,HisStr,&FileWin,OFF);
    MakeFixFrame(2,Sx+144,Sy+79,"Files:",0,10,12,(char*)Fdptr,
                        sizeof(Fdptr[0]),&board[0],&Rects[2]);
    MakeFixFrame(3,Sx+10,Sy+79,"Directories:",0,10,12,(char*)SubDir,
                        sizeof(SubDir[0]),&board[1],&Rects[3]);
    CreateCompo (4,K_CHECKBOX,Sx+282,Sy+79 ,&Rects[4], NULL   ,0);
    CreateCompo (5,K_CHECKBOX,Sx+282,Sy+206,&Rects[5], NULL   ,1);
    CreateCompo (6,K_BUTTON  ,Sx+278,Sy+259,&Rects[6]," OPEN ",1);
    CreateCompo (7,K_BUTTON  ,Sx+366,Sy+259,&Rects[7],"CANCEL",0);
//write the name of frame
    setcolor(BLACK);
    settextjustify(LEFT_TEXT,LEFT_TEXT );
    outtextxy(Sx+10,Sy+23,"Directory:");
```

```
        outtextxy(Sx+10,Sy+47,"File name:");
        outtextxy(Sx+10,Ey-10,"File information:");
        outtextxy(Sx+302,Sy+91,"Show description");
        outtextxy(Sx+310,Sy+93,"_");
        outtextxy(Sx+302,Sy+217,"Sort");
        outtextxy(Sx+302,Sy+219,"_");
        rectangle(Sx+282,Sy+99,Sx+440,Sy+195);
        WriteFrame(&board[0],ON,OFF,OFF);
        WriteFrame(&board[1],ON,OFF,OFF);

prev = 1;
next = 0; //start to frame 1 GgetString()
ReadSubDir = 0;
do{
  WriteHelpBar(&Windows,"<Reading File...>");
  strcpy(FileSpec,HisStr[0]);                  //important only when start
//Extract FileSpec to drive,dir,file,ext
  flags=fnsplit(FileSpec,drive,dir,file,ext);   //Extract path

//have a drive change or have path Dir
  if((flags & DRIVE) || (flags & DIRECTORY))
    {
      if(flags & DRIVE)           //have a drive change
          setdisk(drive[0]-'A');   //change drive
      if(flags & DIRECTORY)        //have a path Dir
          {
           x=strlen(dir);
           if(x>1 && !(dir[0]=='.'))//don't cut it when dir = '..'
              dir[x-1]=0;          //cut '\\' it the tail one
           chd=chdir(dir);         //change path
          }
    }

//if not filename  make it to *.*
  if(!(flags & FILENAME))
    strcpy(file,"*");
  if(!(flags & EXTENSION))        //if not extension append default
    strcpy(ext,DefaultSpec);

  sprintf(HisStr[0],"%s%s",file,ext); //save only file & ext
  sprintf(FileSpec,"%s%s",file,ext);
//SubSpec alway *.*
  strcpy(SubSpec,"*.*");
//read path change and write on screen
  getcwd(curpath,MAXPATH);
```

```
  setcolor(BLACK);
  setfillstyle(SOLID_FILL,BoardColor);
  bar(Sx+144,Sy+13,Sx+440,Sy+23);
  outtextxy(Sx+152,Sy+23,curpath);

//---------------------------
// Read Sub directory name
//---------------------------
// constand to Read file attributes
        attrib = FA_RDONLY+FA_HIDDEN+FA_SYSTEM+FA_DIREC+FA_ARCH;
        done = findfirst(SubSpec,&file_data,attrib);
// constand to test
        attrib = FA_HIDDEN+FA_SYSTEM;

 countsd=0;
 while(!done)
  {
// if hidden or system change to lower case
  if(file_data.ff_attrib & attrib)
        strlwr(file_data.ff_name);
//Sub Dir
  if(file_data.ff_attrib & FA_DIREC)
   {
    if( strcmp(file_data.ff_name,"."))
     {
        strncpy(SubDir[countsd].name,file_data.ff_name,
        strlen(file_data.ff_name)+1);
        SubDir[countsd].fsize=file_data.ff_fsize;
        SubDir[countsd].attrib=file_data.ff_attrib;
        SubDir[countsd].fdate=file_data.ff_fdate;
        SubDir[countsd].ftime=file_data.ff_ftime;
        countsd++;
     }
   }
  done = findnext(&file_data);      // read next file
  if(countsd==MaxSubDir)
    done =1;
 }// end while
//----------------
// Read File name
//----------------
//constand to Read file attributes
   attrib = FA_RDONLY+FA_HIDDEN+FA_SYSTEM+FA_DIREC+FA_ARCH;
   done = findfirst(FileSpec,&file_data,attrib);
// constand to test
```

```
   attrib = FA_HIDDEN+FA_SYSTEM;
 countfn=0;
 while(!done)
  {
// if hidden or system change to lower case
   if(file_data.ff_attrib & attrib)
            strlwr(file_data.ff_name);

   if(!(file_data.ff_attrib & FA_DIREC))   // Not Sub Dir
    {
// find '.' position
      p = strchr(file_data.ff_name,'.');
      strcpy(StrBuffer,file_data.ff_name);
      StrBuffer[p-file_data.ff_name]=0;
      sprintf(Fdptr[countfn].name,"%s        ",StrBuffer);
// cut until to '.'
      Fdptr[countfn].name[8] = 0x0;
      if(p != NULL)
           strcat(Fdptr[countfn].name,p);      //add .ext
      Fdptr[countfn].fsize=file_data.ff_fsize;
      Fdptr[countfn].attrib=file_data.ff_attrib;
      Fdptr[countfn].fdate=file_data.ff_fdate;
      Fdptr[countfn].ftime=file_data.ff_ftime;
      countfn++;
    } // if
   done = findnext(&file_data);              //read next file
   if(countfn==MaxFiles)
      done = 1;
  }// end while
// Sort
           if(Sort){
           qsort((void *)Fdptr,(size_t)countfn,sizeof(Fdptr[0]),
                sort_function);
           qsort((void *)SubDir,(size_t)countsd,sizeof(SubDir[0]),
                sort_function);
           }
 //put [] in subdirectory
           if(SubDir[0].name[0]!='['){
             for(i=0;i<countsd ;i++){
                      sprintf(StrBuffer,"[%-s]",SubDir[i].name);
                      strcpy(SubDir[i].name,StrBuffer);
                      // if subdirectory > 8 cut it
                      if(strlen(SubDir[i].name)>10){
                                SubDir[i].name[9] =0;
                                strcat(SubDir[i].name,"]");}
```

```
        }
      }

      //file_No  = 0;
      settextjustify(LEFT_TEXT,LEFT_TEXT );

      board[0].count = countfn;
      board[1].count = countsd;
//Write sub dir frame & a number of Subdirs
      board[1].active  = 1;
      board[1].item_No = 0;              //clear item to frist
      board[1].index   = 0;
      board[1].first   = 0;
      WriteFrame(&board[1],OFF,ON,OFF);
      //Write count of SubDir in the buttom of frame size
        bar(Sx+18,Sy+249,Sx+54,Sy+259);
        outtextxy(Sx+58,Sy+259,"Subdirs");
        if(strcmp(SubDir[0].name,"[..]")==0)
          sprintf(StrBuffer,"%3d",countsd-1);
        else
          sprintf(StrBuffer,"%3d",countsd);
        outtextxy(Sx+26,Sy+259,StrBuffer);
//Write file name  frame & a number of files
      board[0].active  = 1;
      board[0].item_No = 0;              //clear item to frist
      board[0].index   = 0;
      board[0].first   = 0;
      WriteFrame(&board[0],OFF,ON,OFF);
      //Write count of file in the buttom of frame size
        bar(Sx+168,Sy+249,Sx+204,Sy+259);
        outtextxy(Sx+208,Sy+259,"files");
        sprintf(StrBuffer,"%3d",countfn);
        outtextxy(Sx+176,Sy+259,StrBuffer);

//...Read KeyBoard.....................................
if(next)   { passto=next;next =0;} //next is next board to go
else       passto=1;               //key  is No. of board default 1

  if(countfn == 1)
  {
   p = strchr(FileSpec,'*');     //scan *
   if(p == NULL)
    {
     sprintf(filename,"%s\\%s",curpath,FileSpec);
     passto     = ESC;
```

```
    ReturnMesg = OK;                    //Output
   }
  }
while(passto!=ESC && passto != 0)
 {
  switch(passto)
  {
// get string from edit line
 case 1 :
          prev = 1;
          WriteHelpBar(&Windows,"Enter <File name> and Press <Enter> ,\
 <Press Tab> to Change ");
          result =GgetString(1,Sx+144,Sy+31,35,NULL,FileSpec,HisStr,
                             &FileWin,ON);
          switch(result)
          {
            case ChangeRight:
                  passto=2;             //tab change to file board
                  break;
            case ChangeLeft:
                  passto=3;             //tab change to SubDir board
                  break;

            case OK:
                switch(mode)
                {
                case READ_MODE :
                        passto = 0;         //loop up to read Dir
                        next   = 2;
                        break;
              case WRITE_MODE:
                        flags=fnsplit(HisStr[0],drive,dir,file,ext);
                        if(!(flags & EXTENSION))
                          strcpy(ext,DefaultSpec);
                        sprintf(filename,"%s%s",file,ext);
                        passto=ESC;
                        ReturnMesg = OK;  //Output
                        break;
                }
                break;
            case -27:passto=ESC;            //exit fileman.
                  break;
            case Mouse:
                  passto = MESG.ID;         //Mouse change next frame
                  break;
```

```
            }//switch(result)
            break;
//a frame of file name
 case 2 : prev =2;
            WriteHelpBar(&Windows,"Press <Frist letter> to Search ,\
   Press <Enter> When you Select  ");
            FileWin.frame_No = 0;
            board[0].active  = 1;
            HightLight(&board[0],ON);
            result = Askframe(&FileWin,ON);
            board[0].active  = 0;

            switch(result)
            {
              case ChangeRight:
                    passto=3;           //tab change to SubDir board
                    break;
              case ChangeLeft:
                    passto=1;           //tab change to GgetString
                    break;
              case -27:passto=ESC;      //exit fileman
                    break;
              case Mouse:
                    passto = MESG.ID;   //Mouse change next frame
                    break;
               default:
                 if(result>=  0 )
                     {
                      sprintf(filename,"%s\\%s",curpath,Fdptr[result].name);
                      passto=ESC;
                      ReturnMesg = OK;  //Output
                     }
            }//switch(result)
            break;
//SubDir frame
 case 3 : prev =3;
            WriteHelpBar(&Windows,"Press <Frist letter> to Search ,\
   Press <Enter> When you Select  ");
            FileWin.frame_No = 1;
            board[1].active  = 1;
            HightLight(&board[1],ON);
            result = Askframe(&FileWin,ON);
            board[1].active  = 0;
```

```
        switch(result)
        {
          case ChangeRight:
                    passto=1;              //tab change to GgetString
                    break;
          case ChangeLeft:
                    passto=2;              //tab change to file board
                    break;
          case -27:passto=ESC;             //exit fileman
                    break;
          case Mouse:
                    passto = MESG.ID;      //Mouse change next frame
                    break;
          default :
          //change SubDir
            ReadSubDir = ON;
            passto=0;                      //loop up to read new Dir
            next=3;                        //and back to read SubDir again
        }//switch(result)
        break;
//describtion CheckBox
 case 4: descrip = Rects[4].Status;
        passto = prev;
        break;
//Sort CheckBox
 case 5: passto = prev;
        Sort = Rects[5].Status;
        break;
//OK Button
 case 6:
        switch(prev)
         {
          case 1: passto = 0;
                  next   = 1;
                  break;
          case 2: strcpy(filename,Fdptr[board[0].item_No].name);
                  passto     = ESC;
                  ReturnMesg = OK;
                  break;
          case 3: ReadSubDir = ON;
                  result = board[1].item_No;
                  passto = 0;
                  next   = 3;
                  break;
         }
```

```
            break;
//Cancel Button
 case 7: *filename = NULL;
            passto=ESC;
            break;
  }//switch(passto)
//Read subdir again
  if(ReadSubDir == ON)
   {
     strcpy(StrBuffer,SubDir[result].name);
     x=strlen(StrBuffer);
     StrBuffer[0]=32;
     StrBuffer[x-1]=0;
     x=strlen(curpath);
     if(x>3)
          sprintf(FileSpec,"%s\\%s",curpath,StrBuffer+1);
     else
          sprintf(FileSpec,"%s%s",curpath,StrBuffer+1);
     chd=chdir(FileSpec);
     strcat(FileSpec,"\\");
     ReadSubDir = OFF;
   }

 }//while
//...End Read KeyBoard.....

}while(passto!=ESC);// do-while loop
//-------------------------------------------------------------------------

//Restore Old PATH
     flags=fnsplit(Oldpath,drive,dir,file,ext);
     setdisk(drive[0]-'A');
     chd=chdir(Oldpath);

     RestoreScr(ScrSave);
     HeadTitle(&Windows,ON);
     WriteHelpBar(&Windows,HelpStr);       //Restore Old Help
     return (ReturnMesg);
}//end
//[*]
void Beep(unsigned Frequency,unsigned Delay)
{
     sound(Frequency);
     delay(Delay);
     nosound();
```

```
}
//[*]
// return
// File Size   When File name exist
// NULL        When File name not exist
unsigned long SearchFile(char *filename)
{
FILE *fp;
unsigned long filesize;

        fp = fopen(filename,"rb"); // rb = read binary
        if(fp==NULL)                    //New file name or file not found
          filesize =NULL;
        else
          filesize =filelength(fileno(fp));

        fclose(fp);
        return((unsigned long)filesize);
}//end
//[*]
unsigned long  ReadBFile(char *name,char *obj,long offset)
{
FILE *fp;
char huge *ptr;
int done;
unsigned rsize;
unsigned long nbyte,filesize, ReturnMesg;

        done =0;
        ReturnMesg = 0;
        ptr=(char huge *)obj;
        filesize = SearchFile(name);
        fp = fopen(name,"rb"); // rb = read binary
        if(fp==NULL)
          ReturnMesg = 0;
        else
          {
           if(offset != 0L)
            {
             fseek(fp,offset,SEEK_SET);
             filesize -= offset;
          }
            while(!done)
            {
                if(filesize>65535) {filesize -= 65535;    rsize = 65535;}
```

```
                else            {rsize    = (unsigned)filesize; done=1;}
                nbyte = fread(ptr,sizeof(char),rsize,fp);
                if(nbyte != rsize)
                    {
                     ReturnMesg = 0;
                     done = 1;
                    }
                    else
                      ReturnMesg += rsize;
                    ptr+=rsize;
                }
               }
 fclose(fp);
 return(ReturnMesg);
}
//[*]
int WriteBFile(char *name,char *obj,unsigned long size,char *mode)
{
FILE *fp;
char huge *ptr;
int done,ReturnMesg;
unsigned wsize;
unsigned long nbyte,filesize;

        ptr=(char huge *)obj;
        fp = fopen(name,mode);
        if(fp==NULL)
          ReturnMesg = 0;
        else
          {
           done = 0;
           while(!done)
            {
             if(size>65535) {size -= 65535;          wsize = 65535;}
             else           {wsize = (unsigned)size; done  = 1;}

                 fwrite(ptr,sizeof(char),wsize,fp);
                 ptr+=wsize;
            }
           ReturnMesg = 1;
          }
 fclose(fp);
 return(ReturnMesg);
}//end
//[*]
```

```
//Suspect  Value to test  in neuronSize
// PatLoad = 0  use when do not load pattern    don't cheak I/O size
// PatLoad = 1  use pattern loaded                 cheak I/O size
int NetErr(int PatLoad)
{
int i,ErrNo,done,start,end;

  if(PatLoad==0)
   { start = 1; end = NetCmd.NetLayer-1; }
  else
   { start = 0; end = NetCmd.NetLayer;   }
  ErrNo   = 0;
  done    = NO;
  while(!done)
  {
  if(NetCmd.NetLayer<2 || NetCmd.NetLayer>MaxLayer)
    { ErrNo = 1; break; }
  if(neuronLayer != (NetCmd.NetLayer-1))
   { ErrNo = 2; break; }
  for(i=start;i<end;i++)
   {
    if(neuronSize[i] <= 0 || neuronSize[i] > 2000)  //is Suspect Value
     { ErrNo = 2; break;}
   }
   done = OK;
  }//while(!done)

 return(ErrNo);
}//end
//start function...
void CreateWindows(
int ID,                    //Window ID
char Type,                  //Type of Window
char *Name,                  //name of Window
int count,                  //count of frame
int Sx,int Sy,int Ex,int Ey,  //position
int color,                  //color of client area
char Edge,                  //sizeof edge
struct Win_frame *Win,        //struct of Window
char MenusBar,                //flag of menu bar
char HelpBar,                 //flag of help bar
struct frame_Board *board,    //hold struct of frame
struct BoxRect Rect[],        //hold the table of pos component
int RectCount                 //count of component
)
```

```
{
//initial constant in windows number
    Win->ID         = ID;
    Win->Type       = Type;
    Win->name       = Name;
    Win->EdgeWidth = Edge;
    Win->startx     = Sx;
    Win->endx       = Ex;
    Win->starty     = Sy;
    Win->endy       = Ey;
    Win->frame_No = (-1);                    //no pulldown to active
    Win->count      = count;          //number of frame

    Win->ClnColor  = color;
    Win->ClientSx  = Sx + Edge+1;
    Win->ClientSy  = Sy + Edge+HeadHight+1;
    Win->ClientEx  = Ex - Edge-1;
    Win->ClientEy  = Ey - Edge-1;
    Win->MenusBar  = MenusBar;
    Win->HelpBar   = HelpBar;
    Win->Frames    = board;
    Win->table     = Rect;
    Win->Component = RectCount;
    if(Rect != NULL)
      {
          Win->table[0].ID   = 0;
          Win->table[0].Sx   = Sx;
          Win->table[0].Ex   = Ex;
          Win->table[0].Sy   = Sy;
          Win->table[0].Ey   = Ey;
          Win->table[0].Kind = NULL;
          Win->table[0].Str  = NULL;
      }
    if(MenusBar==ON)
      {
       Win->MbarSx = Win->ClientSx-1;
       Win->MbarSy = Win->ClientSy-1;
       Win->MbarEx = Win->ClientEx+1;
       Win->MbarEy = Win->ClientSy +MenuBar+1;

       Win->ClientSy  += MenuBar+2;          //extend ClientSy  below frameBar
      }
    else
      {
       Win->MbarSx = 0;
```

```
      Win->MbarSy = 0;
      Win->MbarEx = 0;
      Win->MbarEy = 0;
     }
   if(HelpBar==ON)
      Win->ClientEy -= (HelpBarSize+1);

}//end CreateWindows
void ShowWindow(struct Win_frame *Win,char *HBstr)
{
int Sx,Sy,Ex,Ey,ES,midx,midy,count,i,j;
struct frame_Board *board;
   mouse_cursor_off();
   Sx = Win->startx;
   Sy = Win->starty;
   Ex = Win->endx  ;
   Ey = Win->endy  ;

   switch(Win->Type)
   {
     case 0: DrawFrame0(Win);
             break;
     case 1: DrawFrame1(Sx,Sy,Ex,Ey,Win->name);
             break;
   }

   if((Win->HelpBar)&& HBstr != NULL)
     HelpBar(Win,HBstr);

   mouse_cursor_on();
}//end ShowWindow
//[*] Clear Client Area in Windows
void ClearClient(struct Win_frame *Win,int color)
{
   mouse_cursor_off();
   setfillstyle(SOLID_FILL,color);
   bar(Win->ClientSx,Win->ClientSy,Win->ClientEx,Win->ClientEy);
   mouse_cursor_on();
}//end
void HeadTitle(struct Win_frame *Win,char Status)
{
int Sx,Sy,Ex,Ey,ES,midx,midy;
   mouse_cursor_off();
   Sx = Win->startx;
   Sy = Win->starty;
```

```
    Ex = Win->endx  ;
    Ey = Win->endy  ;
    ES = Win->EdgeWidth;
    settextjustify(CENTER_TEXT,CENTER_TEXT );
    setcolor(BLACK);
    line(Sx+ES+1,Sy+ES+HeadHight,Ex-ES-1,Sy+ES+HeadHight);
    if(Status===0){
      setfillstyle(SOLID_FILL,WHITE);
      setcolor(BLACK);}
    else{
      setfillstyle(SOLID_FILL,LIGHTBLUE);
      setcolor(WHITE);}

    bar(Sx+ES+HeadHight+1,Sy+ES+1,Ex-ES-1,Sy+ES+HeadHight-1);
    midx = (Ex+Sx)/2;              // find center
    midy = Sy+ES+(HeadHight/2)+1;   //+1 pixel
       outtextxy(midx,midy,Win->name);
    Win->active    = Status;       //@
    mouse_cursor_on();
}//end
void HelpBar(struct Win_frame *Win,char *Str)
{
int Sx,Sy,Ex,Ey,midx,midy,Ecy;

//space between head and bottom
          Ecy = (HelpBarSize-HelpLine)/2;
// position of Help Bar
          Sx = Win->ClientSx;
          Ex = Win->ClientEx;
          Sy = (Win->ClientEy)+1;
          Ey = Sy+HelpBarSize;
     setcolor(BLACK);
          setfillstyle(SOLID_FILL,LIGHTGRAY);
          bar3d(Sx,Sy,Ex,Ey,0,1);
     setcolor(WHITE);
          line(Sx+1,Sy+1,Ex-1,Sy+1);
          line(Sx+1,Sy+1,Sx+1,Ey-1);
//Change position of Help Bar to Help Box Area in size
          Sx += 8;
          Ex -= 8;
          Ey -= Ecy;
          Sy  = Ey-HelpLine;
     setcolor(DARKGRAY);
          line(Sx-1,Ey+1,Sx-1,Sy-1);
          line(Sx-1,Sy-1,Ex+1,Sy-1);
```

```
    setcolor(WHITE);
        line(Ex+1,Sy-1,Ex+1,Ey+1);
        line(Ex+1,Ey+1,Sx-1,Ey+1);

        Win->HelpBarSx = Sx;
        Win->HelpBarSy = Sy;
        Win->HelpBarEx = Ex;
        Win->HelpBarEy = Ey;
        if(Str!=NULL)
          WriteHelpBar(Win,Str);       //write String
}//end
// Draw HightLight Bar and text insize Borad
void HightLight(struct frame_Board *board,char paint)
{
int Sx,Sy,Ex,Ey,slide,Item,LH,first,n;
char **m,*Str;

if(board->count >0)
 {
    mouse_cursor_off();
    LH    = (board->LineSpc);          //load space of line
    Item  = (board->index);            //Item select in frame
    Item *= LH;                        //find position of line
    slide = (LH-TextHight)/2;
//initial position
    Sx    = board->startx+EdgeSpace;
    Sy    = board->starty+Item+slide;
    Ex    = board->endx-EdgeSpace;
    Ey    = Sy+LH-1;
//paint HightLight
    if(paint == ON) {setfillstyle(SOLID_FILL,HightLightBar);
                     setcolor(HightLightText);}
    else            {setfillstyle(SOLID_FILL,BoardColor);
                     setcolor(NormTextColor);}
    bar(Sx,Sy,Ex,Ey);
 //Write String
    Sx    = board->startx;
    Sy    = board->starty;
    settextjustify(LEFT_TEXT,LEFT_TEXT );
  switch(board->datakind)
  {
   case 0:
          m      = board->menu;
          outtextxy(Sx+BoardSpace,Sy+LH+Item,m[board->item_No]);
          break;
```

```
    case 1:
            Str     = board->StrP;
            n       = board->Strlen;
            first   = board->first;
            outtextxy(Sx+BoardSpace,Sy+LH+Item,Str+(board->item_No)*n);
            break;
    }
     mouse_cursor_on();
   }//if(board->count >0)
  }//end
  //[*]
  void WriteHelpBar(struct Win_frame *Win,char *Str)
  {
  int Sx,Sy,Ex,Ey,len,i;
  char StrBuff[2];

          Sx = Win->HelpBarSx;
          Sy = Win->HelpBarSy;
          Ex = Win->HelpBarEx;
          Ey = Win->HelpBarEy;

          StrBuff[1] = 0;

      setfillstyle(SOLID_FILL,LIGHTGRAY);    //clear HelpBar
          bar(Sx,Sy,Ex,Ey);
          len = strlen(Str);
      setcolor(BLACK);
      settextjustify(LEFT_TEXT,LEFT_TEXT );
          moveto(Sx+TextWidth,(Ey+Sy)/2+(TextHight)/2+1);
          for(i=0;i<=len;i++){
            if(Str[i]=='<'){
               setcolor(RED);
            continue;
            }
            if(Str[i]=='>'){
            setcolor(BLACK);
            continue;
           }
        StrBuff[0]=Str[i];
        outtext(StrBuff);
      }
  }//end
  int MakeMenu(
  int FrameNo,     // menu number
  int x,           // first =0,next = next position of x
```

```
int y,          // starty of frame
char *name,     // Name this menu
char HotKey,    // position off Alt hot key in Name
char **menu,    // menu text
unsigned Key,   // hot key of this frame
char *keys ,    // hot keys of item
int linesize    // space between each line
)
{
 int i,Maxlen,Nlen,count;
 int Sx,Sy,Ex,Ey,Lx,Rx;
 unsigned size;
 char Gap;
  Gap=40;   // space between each name of pulldown menu
  Lx = Windows.ClientSx;  //left  menu
  Rx = Windows.ClientEx;  //right menu
  count = strlen(keys);
  if(FrameNo>MAX_PULLDOWN-1)
    printf("Too many menus\n");
  // compute the String Witdh Max
  Maxlen = 0;
  for(i=0; i<count; i++)
   if(strlen(menu[i]) > Maxlen)
          Maxlen = strlen(menu[i]);

  Maxlen *= TextWidth; //change to pixel lenght
  frame[FrameNo].name  = name;
  Nlen = strlen(frame[FrameNo].name);
  Nlen *= TextWidth;

  switch(x)
  {
   case 0:                          //When Start first in left menu
          x = Lx+EdgeSpace+TextWidth;
          frame[FrameNo].namex = x;
          Sx  = x-TextWidth-EdgeSpace;
          Ex  = Sx+(BoardSpace*2)+Maxlen;
          x += Nlen+Gap;
          break;
   case -1:                         //right menu
          frame[FrameNo].namex = Rx-EdgeSpace-TextWidth-Nlen;
          Sx  = Rx-(BoardSpace*2)-Maxlen;
          Ex  = Rx;
          break;
   default :
```

```
            frame[FrameNo].namex = x; //left menu
            Sx  = x-TextWidth-EdgeSpace;
            Ex  = Sx+(BoardSpace*2)+Maxlen;
            x += Nlen+Gap;
    }//switch(x)

    frame[FrameNo].namey = y-(MenuBar/2)+(TextHight/2)-1;
    Sy  = y;
    Ey  = Sy+(count*LineHight)+(LineHight/2);

  // construct the frame leftover
    frame[FrameNo].datakind= 0;      //use **menu to point data
    frame[FrameNo].namel   = Nlen;   //length in pixel
    frame[FrameNo].startx  = Sx;
    frame[FrameNo].starty  = Sy;
    frame[FrameNo].endx    = Ex;
    frame[FrameNo].endy    = Ey;
    frame[FrameNo].menu    = (char **)menu;
    frame[FrameNo].StrP    = NULL;
    frame[FrameNo].Strlen  = NULL;
    frame[FrameNo].first   = 0;      //first item to show
    frame[FrameNo].index   = 0;      //point to frist
    frame[FrameNo].item_No = 0;      //initial frist item
    frame[FrameNo].Key     = Key;
    frame[FrameNo].HotKey  = HotKey;
    frame[FrameNo].keys    = keys;
    frame[FrameNo].count   = count;
    frame[FrameNo].Maxline = count;
    frame[FrameNo].LineSpc = linesize;
    frame[FrameNo].active  = 0;

    size = imagesize(Sx,Sy,Ex+4,Ey+4); //@
    if(size > MaxFrameSize)
       MaxFrameSize = size;
    return (x); //next position
  }//end MakeMenu
  void MakePopup(
  struct frame_Board *board, //frame pointer
  int x,int y,               //position if x=0 & y=0 refer to pulldown
  char **menu,               //menu text
  char *keys,                //hot keys of item
  int PdNo ,                 //child of PdNo
  int ItemNo,                //use item in pulldown create Sy
  int linesize               //space between each line
  )
```

```
{
 int i, Maxlen,slide,count;
 int Sx,Sy,Ex,Ey,Lx,Rx;
 unsigned size;

  count = strlen(keys);
  // compute the String Witdh Max
  Maxlen = 0;
  for(i=0; i<count; i++)
   if(strlen(menu[i]) > Maxlen)
          Maxlen = strlen(menu[i]);

  Maxlen *= TextWidth; //change to pixel lenght
  slide = (LineHight-TextHight)/2;

  if(x == 0 && y == 0)  //refer to pulldown
   {
    ItemNo++;
    ItemNo *= LineHight;
    Sx = frame[PdNo].startx+EdgeSpace;
    Sy = frame[PdNo].starty+ItemNo+slide;
   }
  else                  //get start Sx,Sy
   {
    Sx = x;
    Sy = y;
   }
  Ex = Sx+(BoardSpace*2)+Maxlen;
  Ey = Sy+(count*LineHight)+(LineHight/2);

// construct the frame leftover
  board->datakind= 0;
  board->name    = NULL;
  board->namex   = 0;
  board->namey   = 0;
  board->namel   = 0;
  board->startx  = Sx;
  board->starty  = Sy;
  board->endx    = Ex;
  board->endy    = Ey;
  board->menu    = (char **) menu;
  board->StrP    = NULL;
  board->Strlen  = NULL;
  board->first   = 0;      //first item to show
  board->index   = 0;      //point to frist
```

```
  board->item_No = 0;      //initial frist item
  board->Key     = NULL;
  board->keys    = keys;
  board->count   = count;
  board->Maxline = count;
  board->LineSpc = linesize;
  board->active  = 0;

}//end
//Make fix frame
void MakeFixFrame(
int ID,                    // ID of this frame
int Sx,                    // X position to start
int Sy,                    // Y position to start
char *name,                      // point to Name this menu
char HotKey,               // Hot Key of this frame
int MaxView,               // No. of Max item in frame
int MaxChar,               // No. of Max Char in line
char *Str,                       // point to list of string
int  StrCut,               // lenght of each string (if it have)
struct frame_Board *Fxboard,    // hold the struct format
struct BoxRect *Rect
)
{
int i,Maxlen,Nlen,count;
int Ex,Ey;


  Ex = Sx+(MaxChar*TextWidth) +16;
  Ey = Sy+(MaxView*TextHight*2)+5;

  Rect->ID   = ID;
  Rect->Sx   = Sx;
  Rect->Sy   = Sy;
  Rect->Ex   = Ex;
  Rect->Ey   = Ey;
  Rect->Kind = 0;
  Rect->Str  =NULL;
  Fxboard->datakind= 1;                    //use *StrP to point data
  Fxboard->startx  = Sx;           //position & name
  Fxboard->starty  = Sy;
  Fxboard->endx    = Ex;
  Fxboard->endy    = Ey;
  Fxboard->name    = name;
  Fxboard->namex   = Sx;
```

```
  Fxboard->namey   = Sy-8;
  Fxboard->namel   = strlen(name);
  Fxboard->StrP    = Str;
  Fxboard->Strlen  = StrCut;
  Fxboard->first   = 0;                  //first item to show
  Fxboard->index   = 0;                        //point to frist
  Fxboard->item_No = 0;                        //initial frist item
  Fxboard->HotKey  = HotKey;
  Fxboard->count   = NULL;
  Fxboard->LineSpc = 16;
  Fxboard->Maxline = MaxView;          //Max line to show in frame
  Fxboard->active  = 0;
//not use in fix frame
  Fxboard->menu    = NULL;
  Fxboard->Key     = NULL;
  Fxboard->keys    = NULL;
}//end
//         x     on   hightlight
// name    0     1      2
// item    0     1      2
// shadow  0     1
void WriteFrame(board,name,item,shadow)
struct frame_Board *board;
char name,item,shadow;
{
int Sx,Sy,Ex,Ey;

    mouse_cursor_off();
    Sx   = board->startx;
    Sy   = board->starty;
    Ex   = board->endx;
    Ey   = board->endy;
    if(board->name != NULL)
    {
     switch(name)
     {
       case 1:  WriteName(board,OFF);
                    break;
       case 2:  WriteName(board,ON);
                    break;
     }
    }
 //write shadow
    if(shadow)
    {
```

```
      setcolor(Shadow);
      setfillstyle(SOLID_FILL,Shadow);
            bar3d(Sx+4,Sy+4,Ex+4,Ey+4,0,1);
    }
//write board or clear board
    setcolor(BLACK);
    setfillstyle(SOLID_FILL,BoardColor);
      bar3d(Sx,Sy,Ex,Ey,0,1);
    switch(item)
    {
      case 1 : WriteItem(board);
                break;
      case 2 :  WriteItem(board);
                HightLight(board,ON);
                break;
    }
    mouse_cursor_on();
}//end WriteFrame
//[*]
void WriteName(struct frame_Board *board,char mode)
{
int namex,namey,namel,p;
char *name,StrBuff[]=" ";

    mouse_cursor_off();
    settextjustify(LEFT_TEXT,LEFT_TEXT );
    name  = board->name;
    namex = board->namex;
    namey = board->namey;
    namel = board->namel;
    switch(mode)
    {
      case ON : setcolor(BLACK);            // frame of HightLight Bar
                setfillstyle(SOLID_FILL,HightLightBar);
                  break;
      case OFF: setcolor(BoardColor);     // Clear Color frame
                setfillstyle(SOLID_FILL,BoardColor);
                  break;
    }
      bar3d(namex-TextWidth,namey-TextHight-(TextHight/2),
            namex+namel+TextWidth,namey+(TextHight/2)-2 , 0,1);
 //set text color
    if(mode==ON)
      setcolor(BoardColor);
    else
```

```
      setcolor(BLACK);
    outtextxy(namex,namey,name);
//Write Alt-Hot Key
    if(mode==OFF)
    {
     StrBuff[0] = board->name[board->HotKey];          //get HotKey letter
     p = board->HotKey*TextWidth;                      //get position
     setcolor(RED);
       outtextxy(board->namex+p,board->namey,StrBuff); //Write Hot Key
    }
     mouse_cursor_on();
}//end
//[*]
void WriteItem(struct frame_Board *board)
{
int Sx,Sy,x,y,end,first,Maxcount,Maxline,line,n;
char **m,*Str;
    mouse_cursor_off();
    settextjustify(LEFT_TEXT,LEFT_TEXT );
    Sx       = board->startx;
    Sy       = board->starty;
    Maxcount = board->count;
    Maxline  = board->Maxline;
    first    = board->first;
    line     = board->LineSpc;

    if(Maxcount<Maxline)
          end=Maxcount;
    else
          end =Maxline;
 switch(board->datakind)
  {
   case 0:
          m       = board->menu;
          for(x=0,y=line;(x<end && first <Maxcount);x++,y+=line)
             outtextxy(Sx+BoardSpace,Sy+y,m[x]);
          break;
   case 1:
          Str     = board->StrP;
          n       = board->Strlen;
          for(x=0,y=line;(x<end && first <Maxcount);x++,y+=line)
             outtextxy(Sx+BoardSpace,Sy+y,Str+((first+x)*n) );
          break;
   }
   if(Maxcount==0)
```

```
  {
   setcolor(RED);
   outtextxy(Sx+BoardSpace,Sy+line," not found");
  }
 mouse_cursor_on();
}//end
//[*] Draw Window Type 0  send struct of Windows
void DrawFrame0(struct Win_frame *Win)
{
int Sx,Sy,Ex,Ey,ES,midx,midy,count,i,j;
struct frame_Board *board;

    mouse_cursor_off();
    Sx = Win->startx;
    Sy = Win->starty;
    Ex = Win->endx  ;
    Ey = Win->endy  ;
    ES = Win->EdgeWidth;
    board = Win->Frames;
    setfillstyle(SOLID_FILL,BlackGround);   // Visual Blackground
    bar(Sx,Sy,Ex,Ey);
    setfillstyle(SOLID_FILL,Win->ClnColor); // insize board color
    bar(Win->ClientSx,Win->ClientSy,Win->ClientEx,Win->ClientEy);

// ****** LIGHT ******
 setcolor(WHITE);
// out frame
 moveto(Sx+1,Ey-1);
    lineto(Sx+1,Sy+1);
    lineto(Ex-1,Sy+1);
 moveto(Ex-ES+1,Sy+ES);
    lineto(Ex-ES+1,Ey-ES+1);
    lineto(Sx+ES,Ey-ES+1);
 /****** Dark  ******/
 setcolor(BLACK);
 /* out frame */
 rectangle( Sx,Sy,Ex,Ey);
 rectangle( Sx+ES,Sy+ES,Ex-ES,Ey-ES);
// [-]
    setfillstyle(SOLID_FILL,LIGHTGRAY);
          bar3d(Sx+ES,Sy+ES,Sx+ES+HeadHight,Sy+ES+HeadHight,0,1);

          midx = Sx+ES+(HeadHight/2);
          midy = Sy+ES+(HeadHight/2) ;
```

```
        bar3d(midx-3,midy-1,midx+3,midy+1,0,1);
    setcolor(WHITE);
        line(midx-2,midy,midx+2,midy);
// -
    setcolor(DARKGRAY);
        line(midx-2,midy+2,midx+4,midy+2);
        line(midx+4,midy+2,midx+3,midy);
    HeadTitle(Win,1);           //Wrtie Head Title and it to active

    if(Win->MenusBar)
      {
      setcolor(BLACK);
      setfillstyle(SOLID_FILL,BoardColor);
      bar3d(Win->MbarSx,Win->MbarSy,Win->MbarEx,Win->MbarEy, 0,1);
      count = Win->count;
      settextjustify(LEFT_TEXT,LEFT_TEXT );
     //Write name of frame
      for(i=0;i<count;i++)
          WriteName(&board[i],OFF);
      }
    mouse_cursor_on();
}//end DrawFrame0();
//[*] Draw framw Type 1
void DrawFrame1(int Sx,int Sy,int Ex,int Ey,char *header)
{
int midx,midy,ES=5;

//Draw Box
        setcolor(BLACK);
        setfillstyle(SOLID_FILL,LIGHTBLUE);
        bar3d(Sx,Sy,Ex,Ey,0,1);

        setfillstyle(SOLID_FILL,BoardColor);
        bar(Sx+ES,Sy+ES,Ex-ES,Ey-ES);
        setcolor(BLACK);

/* [-] */
    setfillstyle(SOLID_FILL,LIGHTGRAY);
        bar3d(Sx+ES+1,Sy+ES+1,Sx+ES+HeadHight+1,Sy+ES+HeadHight+1,0,1);

        midx = Sx+ES+1+(HeadHight/2);
        midy = Sy+ES+1+(HeadHight/2);

        bar3d(midx-3,midy-1,midx+3,midy+1,0,1);
    setcolor(WHITE);
```

```
            line(midx-2,midy,midx+2,midy);
            /* - */
    setcolor(DARKGRAY);
            line(midx-2,midy+2,midx+4,midy+2);
            line(midx+4,midy+2,midx+3,midy);
/* Head  Title */
    setfillstyle(SOLID_FILL,LIGHTBLUE);
       bar3d(Sx+ES+HeadHight+3,Sy+ES+1,Ex-ES-1,Sy+ES+HeadHight+1,0,1);
    settextjustify(CENTER_TEXT,CENTER_TEXT );
       midx = (Ex+Sx)/2;                  // find center
       midy = Sy+ES+(HeadHight/2)+1;   //+1 pixel
    setcolor(WHITE);
       outtextxy(midx,midy,header);

}//end DrawFrame1
//[*]
int Popup_Menu(struct Win_frame *Win,int Popframe)
{
int Sx,Sy,Ex,Ey;
int item,len,PopupNo,popmenu,oldframe,ReturnID;
unsigned size;
struct frame_Board *board,*Isframe;
char far *ScrSave,draw;

  board          =  Win->Frames;
  Isframe        =  &(board[Popframe]);
  Isframe->active =  1;
  draw       .   =  ON;
  ReturnID       =  NONE;

  Sx   = Isframe->startx;
  Sy   = Isframe->starty;
  Ex   = Isframe->endx;
  Ey   = Isframe->endy;
  size = imagesize(Sx,Sy,Ex+4,Ey+4);
//save for popup over on
   XMem(&(void far *)ScrSave,size,NONE);
   if(ScrSave  == NULL)
    {
     MessageWin("EXIT PROGRAM","NOT ENOUGH HEAP SPACE",
                    "FOR SAVE SCREEN ",1,0,OFF);
     closegraph();
     exit(1);
    }
   mouse_cursor_off();
```

```
getimage(Sx,Sy,Ex+4,Ey+4,ScrSave);      //Save screen
mouse_cursor_on();
oldframe     = Win->frame_No;          //hold parent frame
Win->frame_No =  Popframe;             //change to current popup frame
while(ReturnID ==  NONE)
{
 if(draw == ON)
    WriteFrame(Isframe,2,2,ON);
 item = Askframe(Win,OFF);
 if(draw == OFF)
   draw = ON;

 if(item < 0)
  ReturnID = item;
 else
  {
    len = strlen(Isframe->menu[item]);
    len--;       //point to last char
    if(*(Isframe->menu[item]+len) == 0x10)   //the " char
       {
    //get the frame to point
         PopupNo =  (int)*(Isframe->menu[item]+len+2);
         popmenu = Popup_Menu(Win,PopupNo);

         if(popmenu==Cancel)
           draw = OFF;
         else
           ReturnID = popmenu;
       }
     else
         ReturnID = (int)*(Isframe->menu[item]+len+2);
    }//else
 }//ReturnID ==  NONE

 Win->frame_No   = oldframe;          //restore parent frame
 mouse_cursor_off();
 putimage(Sx,Sy,ScrSave,COPY_PUT);     //Restore
 Isframe->active =  0;
 mouse_cursor_on();

FreeXMem(ScrSave);
return(ReturnID);
}//end
//[*]
//Ask All button & check box
```

```
//Return ID  OK    CANCEL    Cancel
//          1     2        -27
int SelectCompo(struct Win_frame *Win,int passto)
{
int nCompo,done,result;

   done = 0;
   nCompo = Win->Component;
   while(!done)
   {
    result = AskCompo(passto,Win->table,Win);
     switch(result)
     {
          case Cancel:
               done = 1;
               break;
          case ChangeRight:
               passto++;
               if(passto>nCompo-1)
                    passto = 1;
               break;
          case ChangeLeft:
               passto--;
               if(passto<1)
                    passto = nCompo-1;
               break;
          default:
               done = 1;
     }
   }//while(!done)

 return(result);
}//end
//[*]
// Return result > 0 is ID
//            result < 0 is Ctrl
int AskCompo(int RectNo,struct BoxRect *Rects,struct Win_frame *Win)
{
int i,count,ReturnMesg,done;
unsigned Mesg;

 count = Win->Component;

 switch(Rects[RectNo].Kind)
 {
```

```
case K_BUTTON:
    for(i=0;i<count;i++)
    {
        if(Rects[i].Kind == K_BUTTON && Rects[i].Status == ON)
         {
          Rects[i].Status = OFF;   //OFF button
          button(Rects[i].Sx,Rects[i].Sy,Rects[i].Str,0,0);
         }
    }
    Rects[RectNo].Status = ON;        //ON button
    button(Rects[RectNo].Sx,Rects[RectNo].Sy,Rects[RectNo].Str,0,1);
   break;
}
done = 0;
while(!done)
{
  Mesg = ReadInput(Win);
  switch(Mesg)
  {
   case ENTER:
                ReturnMesg = RectNo;
                done = 1;
                break;
   case Mouse:
                if(MESG.ID == 0)    //press in win
                  break;
                ReturnMesg = MESG.ID;
                done = 1;
                break;
   case SPACE:
            //process check box
                if(Rects[RectNo].Kind == K_CHECKBOX)
                 {
                   if(Rects[RectNo].Status == ON)
                      CheckBox(&Rects[RectNo],Rects[RectNo].Sx,
                               Rects[RectNo].Sy,OFF);
                   else
                      CheckBox(&Rects[RectNo],Rects[RectNo].Sx,
                               Rects[RectNo].Sy,ON);
                 }
                break;
   case RIGHT:
   case RIGHTTAB:
                ReturnMesg = ChangeRight;
                done = 1;
```

```
                    break;
    case LEFT:
    case BACKTAB:
                    ReturnMesg = ChangeLeft;
                    done = 1;
                    break;
    case ESC:
                    ReturnMesg = Cancel;
                    done = 1;
                    break;
   }
  }//while(!done)
 button(Rects[RectNo].Sx,Rects[RectNo].Sy,Rects[RectNo].Str,0,0);
 return(ReturnMesg);
}//end
//[*]
void button(Sx,Sy,Str,press,Select)
int Sx,Sy;
char *Str,press,Select;
{
int len,midx,midy,Ex,Ey;

   mouse_cursor_off();
   len = strlen(Str);
   Ex  = Sx+((len+2)*TextWidth);
   Ey  = Sy+LineHight+5;

   setcolor(BLACK);
   setfillstyle(SOLID_FILL,LIGHTGRAY);
// not press
 if(press==0)
 {
   bar3d(Sx,Sy,Ex,Ey,0,1);
   midx =  (Sx+Ex) /2;
   midy =  (Sy+Ey) /2;

   if(Select)                // select button
    {
     setcolor(WHITE);
     settextjustify(CENTER_TEXT,CENTER_TEXT );
     outtextxy(midx+1,midy+3,Str);
     setcolor(RED);
    }
   else
     setcolor(BLACK);
```

```
  settextjustify(CENTER_TEXT,CENTER_TEXT );
  outtextxy(midx,midy+2,Str);

  /* Light & Shardow */
  setcolor(WHITE);
     moveto(Sx+1,Ey-2); lineto(Sx+1,Sy+1); lineto(Ex-2,Sy+1);
     moveto(Sx+2,Ey-3); lineto(Sx+2,Sy+2); lineto(Ex-3,Sy+2);
  setcolor(DARKGRAY);
     moveto(Ex-2,Sy+2); lineto(Ex-2,Ey-2); lineto(Sx+2,Ey-2);
     moveto(Sx+1,Ey-1); lineto(Ex-1,Ey-1); lineto(Ex-1,Sy+1);
 }
// button press
 else
 {
  bar3d(Sx,Sy,Ex,Ey,0,1);
  midx =  (Sx+Ex) /2;
  midy =  (Sy+Ey) /2;
  setcolor(RED);
  settextjustify(CENTER_TEXT,CENTER_TEXT );
      outtextxy(midx+1,midy+3,Str);
 }
   settextjustify(LEFT_TEXT,LEFT_TEXT );
   mouse_cursor_on();
}//end
//Draw & initial component in struct Rect
void CreateCompo(ID,Kind,Sx,Sy,Rect,Str,status)
int ID,Sx,Sy;
char Kind,*Str,status;
struct BoxRect *Rect;
{
int len,Ex,Ey;

    if(Str != NULL)
         len = strlen(Str);

   switch(Kind)
   {
    case K_BUTTON   : button(Sx,Sy,Str,0,status);
                      Ex  = Sx+((len+2)*TextWidth);
                      Ey  = Sy+LineHight+5;
                      break;
    case K_CHECKBOX : CheckBox(Rect,Sx,Sy,status);
                      Ex  = Sx+10;
                      Ey  = Sy+10;
```

```
                    break;
  }
//fill RectBox...
  if(Rect != NULL)
  {
    Rect->ID     = ID;      //ID of this box
    Rect->Sx     = Sx;      //Position of Component
    Rect->Sy     = Sy;      // ...
    Rect->Ex     = Ex;      // ...
    Rect->Ey     = Ey;      // ...
    Rect->Kind   = Kind;    //Kind   of Component
    Rect->Str    = Str;     //String of Component
    Rect->Status = status;  //Status active, not active
  }
}//end
//[*]
void CheckBox(Rects,Sx,Sy,status)
struct BoxRect *Rects;
int Sx,Sy;
char status;
{
    mouse_cursor_off();
    setcolor(BLACK);
    setfillstyle(SOLID_FILL,BoardColor);
    bar3d(Sx,Sy,Sx+10,Sy+10,0,1);
    if(status==ON)
    {
     line(Sx,Sy,Sx+10,Sy+10);
     line(Sx,Sy+10,Sx+10,Sy);
    }
    if(Rects != NULL)
        Rects->Status = status;
    mouse_cursor_on();
}//end
//\
//ask frame whit select in Win->frame_No
//ask frame let you select a list of item after WriteFrame();
//ask frame don't write any item
//when you want to get infomation form HightLight item
//  you can set Inform ON
//  Infromation type    NULL   file information
//                Inform    0        1
int Askframe(struct Win_frame *Win,char Inform)
{
unsigned key16,Mesg;
```

```
int  Maxcount,MaxView,i,len,pos,n;
char key8,RDinform,informer[38];
struct frame_Board *Isframe;
struct FileDetail *Strinform;

   RDinform = Inform;
   Isframe  = &(Win->Frames[Win->frame_No]);
   if(Isframe->count == 0)
     return(ChangeRight);
   Maxcount = Isframe->count;
   MaxView  = Isframe->Maxline;
   n        = Isframe->Strlen;
   if(Maxcount==0)
     return(NONE);

while (key16!=ESC)
 {
  key16=ReadInput(Win);

//...Read Information.....
if(RDinform!=0 && Inform == 1 && Isframe->datakind == 1)
{
      setfillstyle(SOLID_FILL,BoardColor);
      bar(239,374,535,384);
      //strcpy(
      Strinform = (struct FileDetail *)
                        ((Isframe->StrP)+(Isframe->item_No)*(Isframe->Strlen));
      FileInfrom(Strinform,informer);
//Write information on screen
      setcolor(BLACK);
      outtextxy(245,384,informer);
      RDinform=0;
}
//...end Read Information.....

  if(key16 != 0 && Isframe->datakind== 0)              //Scan hot keys
  {
     key8 = toupper((char)key16&0xFF);
     len = strlen(Isframe->keys);
     for(i=0;i<len;i++)
     {
      if(key8==Isframe->keys[i])
       {
          key16 = KeyBoard;
          Isframe->item_No=i;
```

```
            Isframe->index=i;
            break;
        }
      }
    }

    switch(key16)
    {
    case AltX     : return(ExitProg);
    case KeyBoard : return(Isframe->item_No);

    case Mouse    :
  //select out of frame but still in this Window
      if(MESG.ID == 0 )
            break;
      if(MESG.DBclick == ON)
            return (Isframe->index+Isframe->first);
  //select the other frame
      if(MESG.ID > 0 && MESG.item == -1)
            {
             HightLight(Isframe,OFF);
             return(Mouse);
            }
  //convert item in frame to real item
            MESG.item = (Isframe->first+MESG.item);
            if(MESG.item < 0)
               MESG.item = 0;
            if(MESG.item > Maxcount-1)
               MESG.item = Maxcount-1;
  //still in frame
      if(MESG.frame == -1)
       {
        if(Isframe->item_No != MESG.item && MESG.item != -1)
            {
              HightLight(Isframe,OFF);
              Isframe->index   =  MESG.item - Isframe->first;
              Isframe->item_No =  MESG.item;
              HightLight(Isframe,ON);
            }
       }//if(MESG.frame == -1)
  //change to the new frame
      else
        if(Isframe->datakind== 0)
            return(Mouse);
  //press button and drag
```

```
        while(left_button_down())
        {
            Mesg = GetMouseItem(Win);
//protect drag button on frame bar && MESG.frame == Win->frame_No
            if(Mesg == Mouse && MESG.frame == -1 && MESG.item != -1)
            {
//drag change item
             MESG.item = (Isframe->first+MESG.item);
              if(Isframe->item_No != MESG.item)
              {
                HightLight(Isframe,OFF);
                Isframe->index   =  MESG.item - Isframe->first;
                Isframe->item_No =  MESG.item;
                HightLight(Isframe,ON);
              }
            }
            if(MESG.item == -1)
             {
              if(Isframe->index == 0 && Isframe->first > 0)
              {
                Isframe->first--;
                Isframe->item_No--;
                delay(50);
                WriteFrame(Isframe,0,2,0);
              }
              if(Isframe->index == MaxView-1 &&
                Isframe->first < Maxcount-MaxView )
              {
                Isframe->first++;
                Isframe->item_No++;
                delay(50);
                WriteFrame(Isframe,0,2,0);
              }
             }
          }//while(left_button_down())

             RDinform=1;
             if(Isframe->datakind== 0)
             {
            //get position when button release
              Mesg = GetMouseItem(Win);
            //protect release button on frame bar
              if(Mesg == Mouse && MESG.frame == -1)
                return(Isframe->item_No);
              else
```

```
            return(-27);
        }
        break;
case ENTER    : return (Isframe->index+Isframe->first);

case LEFT     :
case BACKTAB  : HightLight(Isframe,OFF);
                return (ChangeLeft);

case RIGHT    :
case RIGHTTAB : HightLight(Isframe,OFF);
                return (ChangeRight);

case ESC      : return (-27);

case PAGEUP   :
case PAGEDOWN :
                if(key16==PAGEUP)
                {
                if(Maxcount<=MaxView|| Isframe->first==0)
                  break;
                Isframe->first   -= MaxView-1;
                Isframe->item_No -= MaxView-1;
                if(Isframe->first<0)
                  {
                   Isframe->first   = 0;
                   Isframe->item_No = 0;
                   Isframe->index   = 0;
                  }
                }
                if(key16==PAGEDOWN)
                {
                if(Maxcount<=MaxView || Isframe->first == Maxcount-MaxView)
                   break;
                Isframe->first   += MaxView-1;
                Isframe->item_No += MaxView-1;
                if(Isframe->first>Maxcount-MaxView)
                  {
                   Isframe->first   = Maxcount-MaxView;
                   Isframe->item_No = Maxcount-MaxView;;
                   Isframe->index   = 0;
                  }
                }
                WriteFrame(Isframe,0,2,0);
                RDinform=1;
```

```
                    break;
case HOME :
case END  :  if(Maxcount==0)
                    break;
               if(key16==HOME)
               {
                    if( (Isframe->index+Isframe->first) == 0)
                      break;
                    Isframe->first=0;
                    Isframe->index=0;
                    Isframe->item_No=0;
               }
               if(key16==END)
               {
                    if((Isframe->index+Isframe->first)==Maxcount-1)
                      break;
                    Isframe->first=Maxcount-MaxView;
                    if(Isframe->first<0)
                     {
                     Isframe->first=0;
                     if(Isframe->index==Maxcount-1)
                       break;
                     Isframe->index=Maxcount-1;
                     }
                    else
                     Isframe->index = MaxView-1;
                    Isframe->item_No = Maxcount-1;
               }
               WriteFrame(Isframe,0,2,0);
               RDinform=1;
                    break;
case SPACE :
case DOWN  : if ((Isframe->first == Maxcount-MaxView &&
                      Isframe->index == MaxView-1) || Maxcount==0)
                    break;
               if(Maxcount<MaxView && Isframe->index==Maxcount-1 )
                    break;
          //index > MaxView Scroll down
               if(Isframe->index == MaxView-1)
               {
                    (Isframe->first)++;
                    (Isframe->item_No)++;
                    if((Isframe->index+Isframe->first)>=Maxcount)
                      Isframe->first=Maxcount-MaxView;
                    WriteFrame(Isframe,0,2,0);   //clear
```

```
                RDinform=1;
                break;
            }
            HightLight(Isframe,OFF);
            (Isframe->index)++;
            (Isframe->item_No)++;
            HightLight(Isframe,ON);
            RDinform=1;
                break;
 case UP    : if ((Isframe->first==0 && Isframe->index==0)||Maxcount==0)
                    break;
            if(Isframe->index ==0)
            {
                (Isframe->first)--;
                (Isframe->item_No)--;
                if((Isframe->index+Isframe->first)<=0)
                  Isframe->first=0;
                WriteFrame(Isframe,0,2,0);
                RDinform=1;
                break;
            }
            HightLight(Isframe,OFF);
            (Isframe->index)--;
            (Isframe->item_No)--;
            if(Isframe->index < 0)
                Isframe->index=0;
            HightLight(Isframe,ON);
            RDinform=1;
                break;
//... Search a first of char in file name...
 default    :

   if(Isframe->datakind == 1 && key16 != NULL)
    {
     key8  =(char) (key16 & 0xff);
     if(isprint(key8))
     {
          key8=toupper(key8);
    //file or SubDir
          if(*(Isframe->StrP+Isframe->item_No*n) =='[' )
            pos=1;  //if SubDir is a next char
          else
            pos=0;  //if file name is a first char
    //scan first char if  found position in i
    //                not found i = Maxcount
```

```
        for(i=Isframe->item_No+1;i<Maxcount;i++)
            if( *((Isframe->StrP)+(i*n)+pos) == key8)
                break;
//found it
        if(i!=Maxcount)
        {
         HightLight(Isframe,OFF);
         //1). item < frame
          if(Maxcount<=MaxView)
          {
           Isframe->index   = i;
           Isframe->item_No = i;
          }
         //2). item > frame
          else
          {
           //3.)can display it frame don't scroll
           if((i-Isframe->first)<MaxView)
            {
             Isframe->index   = i-Isframe->first;
             Isframe->item_No = i;
            }
           //4).no it must scroll
           else
            {
             //5).it can to scroll
             if(i<= (Maxcount-MaxView))
              {
                  Isframe->first   = i;
                  Isframe->item_No = i;
                  Isframe->index   = 0;
              }
              //6).it can't to scroll move cursor to it
              else
              {
                  Isframe->first   = Maxcount-MaxView;
                  Isframe->item_No = i;
                  Isframe->index   = i-Isframe->first;
              }
            }
          }//else (Maxcount<=MaxView)
         WriteFrame(Isframe,0,2,0);
         RDinform=1;
        }//if(i!=Maxcount)
    }//if(isprint(key8))
```

```
    }//if(Isframe->datakind == 1 && key16 != NULL)
  }//switch
 }//while
}//end Askframe
//[*]
void PicNeural(int Sx,int Sy)
{
int x,y;
 setcolor(BLACK);
  setfillstyle(SOLID_FILL,YELLOW);
//circle
      for(x = -10;x<11;x+=20)
          for(y = -10;y<11;y+=10)
           fillellipse(Sx+x,Sy+y,4,4);
//line
      for(x = -10;x<11;x+=10)
          for(y = -10;y<11;y+=10)
           line(Sx-6,Sy+x,Sx+6,Sy+y);
}//end
//[*]...(i)...
void PicInform(int Sx,int Sy)
{
   setcolor(BLACK);
   setfillstyle(SOLID_FILL,BLUE);
      fillellipse(Sx,Sy,20,20);
   setfillstyle(SOLID_FILL,WHITE);
      fillellipse(Sx,Sy-13,5,5);
   bar(Sx-7,Sy-6,Sx+8,Sy+13);
   setfillstyle(SOLID_FILL,BLUE);
   bar(Sx-7,Sy-4,Sx-4,Sy+10);
   bar(Sx+5,Sy-6,Sx+8,Sy+10);
}//end
//[*]
int MessageWin(char *Header,char *Mesg1,char *Mesg2,char button,
               char pic,char BakScreen)
{
struct Win_frame  MesgWin;
struct BoxRect Rects[3];
int Sx,Sy,Ex,Ey,ES,midx,midy,result;
unsigned size;
char far *ScrPtr;

   Sx=170;
   Sy=120;
   Ex=420;
```

```
Ey=264;
ES=5;
mouse_cursor_off();
size = imagesize(Sx,Sy,Ex+4,Ey+4);
if(BakScreen == ON)
{
 XMem(&(void far *)ScrPtr,size,NONE);
 getimage(Sx,Sy,Ex+4,Ey+4,ScrPtr);        //Save screen
}
setfillstyle(SOLID_FILL,DARKGRAY);
bar(Sx+4,Sy+4,Ex+4,Ey+4);
CreateWindows(1,1,Header,0,Sx,Sy,Ex,Ey,WHITE,EdgeSize,
                  &MesgWin,OFF,OFF,NULL,Rects,button+1);
ShowWindow(&MesgWin,NULL);
switch(pic)
{
   case 0: PicNeural(204,179);
             break;
   case 1: PicInform(204,179);          //...(i)...
             break;
}

midx = (Sx+54+Ex-ES)/2;
settextjustify(CENTER_TEXT,CENTER_TEXT);
if(Mesg1 != NULL)
   outtextxy(midx,Sy+59,Mesg1);
if(Mesg2 != NULL)
   outtextxy(midx,Sy+79,Mesg2);

switch(button)
{
 case 1:
          CreateCompo (1,K_BUTTON,(Sx+Ex)/2-32,Ey-40,
                       &Rects[1]," OK ",1);
            break;
 case 2:
          CreateCompo (1,K_BUTTON,(Sx+Ex)/2-74,Ey-40,
                       &Rects[1]," OK ",1);
          CreateCompo (2,K_BUTTON,(Sx+Ex)/2+10,Ey-40,
                       &Rects[2],"CANCEL",0);
            break;
}

mouse_cursor_on();
```

```
if(button)
 { result = SelectCompo(&MesgWin,1); }
else
 { delay(1000); result = 0;}

mouse_cursor_off();
if(BakScreen == ON)
 {
  putimage(Sx,Sy,ScrPtr,COPY_PUT);        //Restore
  FreeXMem(ScrPtr);
 }
mouse_cursor_on();
return(result);
}//end
//[*]
//check file is exist
short FileExist(char *filename,short Mode)
{
char buff[80];
short CanDo,Mesg;
unsigned nbyte;

  CanDo = 0;
  nbyte = SearchFile(filename);
  switch(Mode)
  {
   case READ_MODE:
                    if(nbyte==0)    //file not exist
                     {
                      sprintf(buff,"%s   ",filename);
                      Mesg = MessageWin("MESSAGE","File not found  ",
                                                    buff,1,1,ON);
                     }
                    else            //new file
                      CanDo=OK;
                    break;
    case WRITE_MODE :
                    if(nbyte!=0)    //file exist
                    {
                     Mesg = MessageWin("MESSAGE","File name Exist",
                                                  "OverWrite   ",2,1,ON);
                     if(Mesg==OK)
                         CanDo=OK;
                    }
                    else            //new file
```

```
                    CanDo=OK;
                   break;
  }
 return(CanDo);
}//end

//print format string and variable with current setcolor
//on clear Area with current fill color
//setcolor & solidfill befor use
void gprintfxy(short x,short y,char *fmt, ...)
{
va_list argptr;
short cnt,len,Th;
char Str[80];

    va_start(argptr, fmt);
    cnt = vsprintf(Str, fmt, argptr);
    va_end(argptr);

    Th   = textheight("N");
    len  = textwidth(Str);
    bar(x-2,y-Th-2,x+len+2,y+1);
    outtextxy(x,y,Str);

}//end
short ReadPatternInform(char *name)
{
FILE *fp;
char obj[8];
short Err,nInputbyte,Equal,CanDo;
unsigned long nbyte,filesize;
struct PatternHeader Pathead;

        Err    = 0;
        fp = fopen(name,"rb"); // rb = read binary
        if(fp==NULL)
          Err = 1;
        else
        {
          nbyte= fread(&Pathead,sizeof(struct PatternHeader),1,fp);
          if(nbyte == 1)
           {
            if(strcmp(Pathead.code,"BP") == 0)          //BP Bitpack Pattern
            {
              NetCmd.InputXsize = Pathead.Xsize;
```

```
            NetCmd.InputYsize = Pathead.Ysize;
            if(NetCmd.AlgNo!=ALG_SOM)
                NetCmd.nOutput    = Pathead.nOut;
            nInputbyte        = NetCmd.InputXsize*NetCmd.InputYsize;
          }
          else
           Err = 9;
         }
         else
           Err = 10;

         fclose(fp);
        }

 if(Err!=0)
   ErrMessage(Err);

 return(nInputbyte);
}//end
short ReadPattern(char *filename)
{
FILE *fpg;
size_t nbyte,filesize;
char *InputPack,Str[30];
short i,done,CanDo,Err,ReturnMesg,target;
fpos_t filepos;

   Err  = 0;
   done = 0;
   while(!done)
   {
//[1] read header file
    fpg = fopen(filename,"rb");
    if(fpg==NULL)
     { strcpy(Str,filename); Err = 1; break; }
    fscanf(fpg,"%s",Str);
    fscanf(fpg,"%d",&nPattern);
  //test header is group
    if(strcmp(Str,"group")!=0)
      {Err = 9; break;}
  //read first patern to get size
     fgetpos(fpg, &filepos);          //save file position of String first
     fscanf(fpg,"%s",Str);
 //[2] read information form pattern
     neuronSize[0] = ReadPatternInform(Str);
```

```
    done = 1;
   }//while(!done)

if(Err!=0)
   ErrMessage(Err);
if(fpg!=NULL)
   fclose(fpg);
return(nPattern);
}//end
//[*]
//read file in to struct
short WriteNetCmd(void)
{
FILE *fp;
short CanDo;
unsigned long nbyte;

          CanDo = 0;

          fp = fopen("NEURON.DEF","wb");
          if(fp!=NULL)
          {
            nbyte = fwrite(&NetCmd,sizeof(NetCmd),1,fp);
            if(nbyte != NULL)
               CanDo=1;
            fclose(fp);
          }

  return(CanDo);
}//end
//[*]
short ReadNetCmd(void)
{
FILE *fp;
short CanDo;
unsigned long nbyte;

          CanDo = 0;

          fp = fopen("NEURON.DEF","rb"); // rb = read binary
          if(fp!=NULL)
          {
            nbyte= fread(&NetCmd,sizeof(NetCmd),1,fp);
            if(nbyte != NULL)
```

```
            CanDo=1;
           fclose(fp);
          }

 return(CanDo);
}//end
short KillChar(char *SourceStr,char *TargetStr,char Kill)
{
 short i,j,len;
 char buff[2];

 buff[1]=0;
 len = strlen(SourceStr);
 j=0;
 for(i=0;i<len;i++)
 {
  if(SourceStr[i] != Kill)
   {
    TargetStr[j] = SourceStr[i];
    j++;
   }
 }
 TargetStr[j]=0;

 return(j);
}//end

//[*]
void SetupNetworks1(void)
{
//setupNet   MaxRect = count of frame +Windows + count of button
#define MaxRects 13
#define LINE_EDIT 9
int Sx,Sy,Ex,Ey,x,y,linspc,SF,i,j,k,passto,prev,result,SetupLable,Algm,Tranf;
int neuron[20],hnode,nNet,isNetErr;
struct Win_frame SetupNetWin;
char far *ScrSave,Str[36],StrResult[36];
struct BoxRect Rects[MaxRects];
struct frame_Board popframe[2];
static char HisStr[LINE_EDIT][5][36];
   //setupNet count  SetupStr
static char *SetupStr[]=
{
"Networks file",
"Description :",
```

```
"Size of Layer :        Input    hidden layer     Output",
"Networks Layer         :",
"                 WeightMax    WeightMin",
"   Input sign",
"Training Algorithms:",
"Learning Rate (n)                momentum (l)",
"Error level"
};
int nStr[]={35,35,2,16,10,10,8,8,8}; //count of string
int xPos[]={0,0,0,88,0,104,0,216,0};
int yinc[]={0,1,2,0,2,0,2,0,1};

    linspc     = 25;
    //setupNet fill it
    SetupLable = 9;        //(skip Windows frame & 2 button)+input
    Algm       = 2;
    Tranf      = 1;
    HeadTitle(&Windows,OFF);        //OFF Main Windows
    //note Setup Windows is Rects[0]
    CreateWindows(2,0,"Setup Network",2,79,50,560,367,WHITE,EdgeSize,
                  &SetupNetWin,OFF,OFF,NULL,Rects,MaxRects);
    mouse_cursor_off();
    Sx = SetupNetWin.startx;
    Sy = SetupNetWin.starty;
    Ex = SetupNetWin.endx;
    Ey = SetupNetWin.endy;
    SF = Sx+166;                    //start frame
    ScrSave = SaveScr(Sx,Sy,Ex+8,Ey+8);
    setfillstyle(SOLID_FILL,DARKGRAY);
    bar(Sx+8,Sy+8,Ex+8,Ey+8);
    ShowWindow(&SetupNetWin,NULL);
    setcolor(BLACK);
    settextjustify(LEFT_TEXT,LEFT_TEXT);
    x  = SetupNetWin.ClientSx + 8;
    y  = SetupNetWin.ClientSy + linspc;
    Ex = SetupNetWin.ClientEx;
    Ey = SetupNetWin.ClientEy;
    Str[0] = 0;
    if(NetCmd.InputXsize==0 || NetCmd.InputYsize==0)
      strcpy(Str,"---");
    else
      sprintf(Str,"%d",NetCmd.InputXsize*NetCmd.InputYsize);
    outtextxy(Sx+214,Sy+130,Str);
    if(NetCmd.nOutput == 0)
        strcpy(Str,"---");
```

```
    else
          sprintf(Str,"%d",NetCmd.nOutput);
    outtextxy(Sx+414,Sy+130,Str);
    setcolor(BLUE);
    outtextxy(Sx+190,Sy+205,Algorithms[NetCmd.AlgNo]);
    setcolor(BLACK);
    strcpy(HisStr[0][0],NetCmd.NetName);
    strcpy(HisStr[1][0],NetCmd.Describtion);
  //two layer networks
    if(NetCmd.AlgNo==ALG_PCN || NetCmd.AlgNo==ALG_ADN)
     {
      NetCmd.NetLayer = 2;
      itoa(NetCmd.NetLayer,HisStr[2][0],10);
      strcpy(HisStr[3][0],"NONE");
     }
  //three or more layer networks
    else
     {
      if(NetCmd.NetLayer<3)
            NetCmd.NetLayer = 3;
      itoa(NetCmd.NetLayer,HisStr[2][0],10);
      if(HisStr[3][0][0]=='N' || HisStr[3][0][0]==0)
            strcpy(HisStr[3][0],"10");
      //ArrayInt2Str(HisStr[3][0],&(neuronSize[1]),neuronLayer-1);
     }
    sprintf(HisStr[4][0],"%5.3f",NetCmd.Wmax);
    sprintf(HisStr[5][0],"%6.3f",NetCmd.Wmin);
    sprintf(HisStr[6][0],"%5.3f",NetCmd.LearnConst);
    sprintf(HisStr[7][0],"%5.3f",NetCmd.Momentum );
    sprintf(HisStr[8][0],"%5.3f",NetCmd.Errlevel);


//initial string frame
for(i=0;i<SetupLable;i++)          //frame  count
 {
  outtextxy(x,y,SetupStr[i]);
  y+= linspc;
 }
     mouse_cursor_on();
//initial GgetString frame
y  = SetupNetWin.ClientSy + linspc -16;
for(i=0,j=1,k=0;i<LINE_EDIT;i++,j++,k++)  //frame  count
 {
     y += linspc*yinc[i];
     GgetString(j,SF+xPos[i],y,nStr[i],&Rects[j],StrResult,HisStr[k],
```

```
                &SetupNetWin,OFF);
}

CreateCompo(MaxRects-3,K_BUTTON,Ex-197,Ey-38,&Rects[MaxRects-3]," OK ",0);
CreateCompo(MaxRects-2,K_BUTTON,Ex-93 ,Ey-38,&Rects[MaxRects-2],"CANCEL",0);
CreateCompo(20,K_CHECKBOX,Sx+15,Sy+169,&Rects[MaxRects-1],NULL,
            NetCmd.InputSign);
passto=1;           //key  is No. of board default 1
while(passto!=ESC && passto != 0)
 {
  switch(passto)
  {
   case 20:
           passto = prev;
           break;

   //BUTTON
   case MaxRects-3:       //OK
   case MaxRects-2:       //CANCEL
           //from Mouse
           if(result == -256)
           {
           if(MESG.ID == MaxRects-3)
             passto  = EndofProg;
           if(MESG.ID ==MaxRects-2)
             passto  = ESC;
           result = 0;
           break;
           }
           //form Keyboard Tab
           prev = passto;
           result = AskCompo(passto,Rects,&SetupNetWin);
           switch(result)
            {
              case MaxRects-3:
                      passto  = EndofProg;
                      break;
              case MaxRects-2:
                      passto  = ESC;
                      break;
              case Cancel:
                      passto = ESC;
                      break;
              case ChangeRight:
                      passto++;
```

```
                if(passto>MaxRects-2)
                  passto = 1;
                break;
            case ChangeLeft:
                passto--;
                if(passto<1)
                  passto = MaxRects-2;
                break;

            default:
                passto = result;
                break;
          }
          break;

case EndofProg: //store net
                Beep(1000,50);
                //checknet again
                strcpy(NetCmd.NetName,    HisStr[0][0]);
                strcpy(NetCmd.Describtion,HisStr[1][0]);
                NetCmd.NetLayer =  atoi(HisStr[2][0]);
                neuronLayer  =  NetCmd.NetLayer-1;
                nNet         = atoi(HisStr[2][0]);
                if(nNet<2)
                 { Beep(100,50); passto =3; break; }
                hnode = Str2intArray(HisStr[3][0],neuron);
                if(hnode != (nNet-2))
                 { Beep(100,50); passto = 4 ; break; }
                else
                  strcpy(NetCmd.hiddenSize,HisStr[3][0]);
                isNetErr = 0;
                for(i=0;i<hnode;i++) //find have node=0
                  {
                   if(neuron[i]==0)
                    { Beep(100,50); passto=4; isNetErr = 1; break;}
                  }
                if(isNetErr==1)
                  break;
                for(i=1;i<MaxLayer;i++)
                   neuronSize[i] = 0;
                for(i=1;i<neuronLayer;i++)
                   neuronSize[i]  = neuron[i-1];
                NetCmd.Wmax      =  atof(HisStr[4][0]);
                NetCmd.Wmin      =  atof(HisStr[5][0]);
                NetCmd.LearnConst =  atof(HisStr[6][0]);
```

```
                NetCmd.Momentum   =  atof(HisStr[7][0]);
                NetCmd.Errlevel   =  atof(HisStr[8][0]);
                NetCmd.InputSign  =  Rects[MaxRects-1].Status;

                //AlgNo
                //TranferFn
                isNetErr = NetErr(Patflag);
                if(isNetErr!=0)
                  passto = isNetErr+2;
                else
                  {
                    Networkflag = 0;
                    passto      = 0;
                  }
                break;
default:
  prev   = passto;
  result = GgetString(passto,Rects[passto].Sx,Rects[passto].Sy,
                      nStr[passto-1],NULL,StrResult,HisStr[passto-1],
                      &SetupNetWin,ON);
  switch(result)
      {
       case 1:
          //[*] Netwrork Layer
              if(passto==3)
              {
               nNet  = atoi(HisStr[2][0]);
               if(nNet<2)
                { Beep(100,50); break; }
               if(nNet==2)
                {
                 strcpy(HisStr[3][0],"NONE");
                 GgetString(4,Rects[4].Sx,Rects[4].Sy,nStr[3],NULL,
                            NULL,HisStr[3],&SetupNetWin,OFF);
                 passto = 5;
                 break;
                }
              }//if(passto==3)
          //[*] Hidden
              if(passto==4)
              {
               nNet  = atoi(HisStr[2][0]);
               hnode = Str2intArray(HisStr[3][0],neuron);
               if(hnode != (nNet-2))
                { Beep(100,50); break; }
```

```
                for(i=0;i<hnode;i++) //find have node=0
                 {
                  if(neuron[i]==0)
                   { Beep(100,50); passto--; break;}
                 }
                }//if(passto==4)

        case ChangeRight:
                passto++;
                if(passto>MaxRects-2)
                   passto = 1;
                break;
        case ChangeLeft:
                passto--;
                if(passto<1)
                   passto = MaxRects-2;
                break;
        case -27:passto=ESC;
                break;
        case Mouse:
                passto = MESG.ID;
                break;
       }

//two layer network
  if(NetCmd.AlgNo==ALG_ADN ||NetCmd.AlgNo==ALG_PCN)
  {
   if(passto==3 || passto==4)
    {
        if(result==-256)
         { Beep(100,150); passto = prev; }
        else
          {
           if(prev==2) passto = 5;
           if(prev==5) passto = 2;
          }
    }
  }

  }//switch(passto)
}//while(passto!=ESC && passto != 0)
RestoreScr(ScrSave);
HeadTitle(&Windows,ON);
}//end
//[*]
```

```
void SetupNetworks2(void)
{
//setupNet   MaxRect = count of frame +Windows + count of button
#define MaxRects2 13
#define LINE_EDIT2  10
int Sx,Sy,Ex,Ey,x,y,linspc,SF,i,j,k,passto,prev,result,SetupLable;
int neuron[20],hnode,nNet,isNetErr;
struct Win_frame SetupNetWin;
char far *ScrSave,Str[36],StrResult[36];
struct BoxRect Rects[MaxRects2];
struct frame_Board popframe[2];
static char HisStr[LINE_EDIT2 ][5][36];
  //setupNet count  SetupStr
static char *SetupStr[]=
{
"Networks file",
"Description :",
"Size of Layer :        Input                    Output",
"Networks Layer :2",
"                WeightMax    WeightMin",
"Randoms Weight :",
"Training Algorithms:",
"Learning Rate (n)            decrease with",
"Neightborhood Size           decrease with",
"MaxLoop"
};
int nStr[]={35,35,4,10,10,8,8,8,8,8}; //count of string
int xPos[]={0,0,248,0,104,0,216,0,216,0};
int yinc[]={0,1,2,2,0,2,0,1,0,1};

    linspc    = 25;
    //setupNet fill it
    SetupLable = 10;        //(skip Windows frame & 2 button)+input
    HeadTitle(&Windows,OFF);        //OFF Main Windows
    //note Setup Windows is Rects[0]
    CreateWindows(2,0,"Setup Network",2,79,50,560,377,WHITE,EdgeSize,
                  &SetupNetWin,OFF,OFF,NULL,Rects,MaxRects2);
    mouse_cursor_off();
    Sx = SetupNetWin.startx;
    Sy = SetupNetWin.starty;
    Ex = SetupNetWin.endx;
    Ey = SetupNetWin.endy;
    SF = Sx+166;                    //start frame
    ScrSave = SaveScr(Sx,Sy,Ex+8,Ey+8);
    setfillstyle(SOLID_FILL,DARKGRAY);
```

```
    bar(Sx+8,Sy+8,Ex+8,Ey+8);
    ShowWindow(&SetupNetWin,NULL);
    setcolor(BLACK);
    settextjustify(LEFT_TEXT,LEFT_TEXT);
    x  = SetupNetWin.ClientSx + 8;
    y  = SetupNetWin.ClientSy + linspc;
    Ex = SetupNetWin.ClientEx;
    Ey = SetupNetWin.ClientEy;
    Str[0] = 0;

    if(NetCmd.InputXsize==0 || NetCmd.InputYsize==0)
      strcpy(Str,"---");
    else
      sprintf(Str,"%d",NetCmd.InputXsize*NetCmd.InputYsize);
    outtextxy(Sx+214,Sy+130,Str);

    setcolor(BLUE);
    outtextxy(Sx+190,Sy+205,Algorithms[NetCmd.AlgNo]);
    setcolor(BLACK);
    strcpy(HisStr[0][0],NetCmd.NetName);
    strcpy(HisStr[1][0],NetCmd.Describtion);
    if(NetCmd.nOutput == 0)
          strcpy(HisStr[2][0],"---");
    else
          sprintf(HisStr[2][0],"%d",NetCmd.nOutput);
    NetCmd.NetLayer = 2;
    sprintf(HisStr[3][0],"%5.3f",NetCmd.Wmax);
    sprintf(HisStr[4][0],"%6.3f",NetCmd.Wmin);
    sprintf(HisStr[5][0],"%5.3f",NetCmd.LearnConst);
    sprintf(HisStr[6][0],"%5.3f",NetCmd.LearnDec );
    sprintf(HisStr[7][0],"%d"   ,NetCmd.Neighbor);
    sprintf(HisStr[8][0],"%d"   ,NetCmd.NeigDec);
    sprintf(HisStr[9][0],"%d"   ,NetCmd.MaxLoop);


//initial string frame
for(i=0;i<SetupLable;i++)         //frame  count
 {
  if(SetupStr[i]!=NULL);
     outtextxy(x,y,SetupStr[i]);
  y+= linspc;
 }
     mouse_cursor_on();
 //initial GgetString frame
 y = SetupNetWin.ClientSy + linspc -16;
```

```
for(i=0,j=1,k=0;i<LINE_EDIT2 ;i++,j++,k++)  //frame  count
 {
    y += linspc*yinc[i];
    GgetString(j,SF+xPos[i],y,nStr[i],&Rects[j],StrResult,HisStr[k],
                    &SetupNetWin,OFF);
 }

CreateCompo(MaxRects2-2,K_BUTTON,Ex-197,Ey-38,&Rects[MaxRects2-2],"  OK  ",0);
CreateCompo(MaxRects2-1,K_BUTTON,Ex-93 ,Ey-38,&Rects[MaxRects2-1],"CANCEL",0);

passto=1;              //key  is No. of board default 1
while(passto!=ESC && passto != 0)
 {
   switch(passto)
   {

    //BUTTON
    case MaxRects2-2:       //OK
    case MaxRects2-1:       //CANCEL
              //from Mouse
              if(result == -256)
              {
               if(MESG.ID == MaxRects2-2)
                 passto  = EndofProg;
               if(MESG.ID ==MaxRects2-1)
                  passto  = ESC;
               result = 0;
               break;
              }
              //form Keyboard Tab
              prev = passto;
              result = AskCompo(passto,Rects,&SetupNetWin);
              switch(result)
               {
                 case MaxRects2-2:
                         passto  = EndofProg;
                         break;
                 case MaxRects2-1:
                         passto  = ESC;
                         break;
                 case Cancel:
                         passto = ESC;
                         break;
                 case ChangeRight:
                         passto++;
```

```
                if(passto>MaxRects2-1)
                  passto = 1;
                break;
            case ChangeLeft:
                passto--;
                if(passto<1)
                  passto = MaxRects2-1;
                break;

            default:
                passto = result;
                break;
          }
          break;

case EndofProg: //store net
                Beep(1000,50);
                //checknet again
                strcpy(NetCmd.NetName,    HisStr[0][0]);
                strcpy(NetCmd.Describtion,HisStr[1][0]);

                NetCmd.nOutput    =  atoi(HisStr[2][0]);
                NetCmd.Wmax       =  atof(HisStr[3][0]);
                NetCmd.Wmin       =  atof(HisStr[4][0]);
                NetCmd.LearnConst =  atof(HisStr[5][0]);
                NetCmd.LearnDec   =  atof(HisStr[6][0]);
                NetCmd.Neighbor   =  atoi(HisStr[7][0]);
                NetCmd.NeigDec    =  atoi(HisStr[8][0]);
                NetCmd.MaxLoop    =  atoi(HisStr[9][0]);
                passto      = 0;
                break;
default:
  prev   = passto;
  result = GgetString(passto,Rects[passto].Sx,Rects[passto].Sy,
                      nStr[passto-1],NULL,StrResult,HisStr[passto-1],
                      &SetupNetWin,ON);
  switch(result)
     {
       case OK:
       case ChangeRight:
                passto++;
                if(passto>MaxRects2-1)
                  passto = 1;
                break;
       case ChangeLeft:
```

```
                passto--;
                if(passto<1)
                  passto = MaxRects2-1;
                break;
        case -27:passto=ESC;
                break;
        case Mouse:
                passto = MESG.ID;
                break;
      }

  }//switch(passto)
 }//while(passto!=ESC && passto != 0)
 RestoreScr(ScrSave);
 HeadTitle(&Windows,ON);
}//end
void ShowNetInform1(short Sx,short Sy)
{
short Tx,Ty,ln;

   mouse_cursor_off();
   Tx     = Sx+20;
   Ty     = Sy;
   ln     = 25;
   setcolor(BLACK);
   settextjustify(LEFT_TEXT,LEFT_TEXT);
   setfillstyle(SOLID_FILL,WHITE);
   bar3d(Sx,Sy,Sx+400,Sy+340,0,1);
   setcolor(BLUE);

   gprintfxy(Tx,Ty+=ln,"Networks Name   : %s",NetCmd.NetName);
   gprintfxy(Tx,Ty+=ln,"Describtion     : %s",NetCmd.Describtion);

   if(NetCmd.Inputfile[0] != NULL)
   gprintfxy(Tx,Ty+=ln,"Input file      : %s",NetCmd.Inputfile);
   else
   gprintfxy(Tx,Ty+=ln,"Input file      : *** NOT DECLARE ***");

   gprintfxy(Tx,Ty+=ln,"Networks Layers : %d",NetCmd.NetLayer);

   if(NetCmd.InputXsize==0 || NetCmd.InputYsize==0)
   gprintfxy(Tx,Ty+=ln,"Input Layer     : *** NOT DECLARE ***",
                       NetCmd.InputXsize*NetCmd.InputYsize);
   else
   gprintfxy(Tx,Ty+=ln,"Input Layer     : %d",
```

```
                        NetCmd.InputXsize*NetCmd.InputYsize);
    gprintfxy(Tx,Ty+=ln,"hidden          : %s",NetCmd.hiddenSize);

    if(NetCmd.nOutput==0)
    gprintfxy(Tx,Ty+=ln,"Output Layer    : *** NOT DECLARE ***");
    else
    gprintfxy(Tx,Ty+=ln,"Output Layer    : %d",NetCmd.nOutput);

    gprintfxy(Tx,Ty+=ln,"Input Signed    : %d",NetCmd.InputSign);
    gprintfxy(Tx,Ty+=ln,"Weight Max-Min  : %5.3f,%5.3f",
                        NetCmd.Wmax,NetCmd.Wmin);

    setcolor(LIGHTBLUE);
    gprintfxy(Tx,Ty+=ln,"Algrolithms     : %s",Algorithms[NetCmd.AlgNo]);
    setcolor(BLUE);

    gprintfxy(Tx,Ty+=ln,"Learning rate   : %5.3f",NetCmd.LearnConst);
    gprintfxy(Tx,Ty+=ln,"Momentum        : %5.3f",NetCmd.Momentum);
    gprintfxy(Tx,Ty+=ln,"Error level     : %5.3f",NetCmd.Errlevel);

    mouse_cursor_on();
}//end
void ShowNetInform2(short Sx,short Sy)
{
short Tx,Ty,ln;

    mouse_cursor_off();
    Tx     = Sx+20;
    Ty     = Sy;
    ln     = 25;
    setcolor(BLACK);
    settextjustify(LEFT_TEXT,LEFT_TEXT);
    setfillstyle(SOLID_FILL,WHITE);
    bar3d(Sx,Sy,Sx+400,Sy+340,0,1);
    setcolor(BLUE);

    gprintfxy(Tx,Ty+=ln,"Networks Name    : %s",NetCmd.NetName);
    gprintfxy(Tx,Ty+=ln,"Describtion      : %s",NetCmd.Describtion);

    if(NetCmd.Inputfile[0] != NULL)
    gprintfxy(Tx,Ty+=ln,"Input file       : %s",NetCmd.Inputfile);
    else
    {
    setcolor(RED);
    gprintfxy(Tx,Ty+=ln,"Input file       : *** NOT DECLARE ***");
```

```
setcolor(BLUE);
}

gprintfxy(Tx,Ty+=ln,"Networks Layers  : %d",NetCmd.NetLayer);

if(NetCmd.InputXsize==0 || NetCmd.InputYsize==0)
{
setcolor(RED);
gprintfxy(Tx,Ty+=ln,"Input Layer      : *** NOT DECLARE ***",
                    NetCmd.InputXsize*NetCmd.InputYsize);
setcolor(BLUE);
}
else
gprintfxy(Tx,Ty+=ln,"Input Layer      : %d",
                    NetCmd.InputXsize*NetCmd.InputYsize);

if(NetCmd.nOutput==0)
{
setcolor(RED);
gprintfxy(Tx,Ty+=ln,"Output Layer     : *** NOT DECLARE ***");
setcolor(BLACK);
}
else
gprintfxy(Tx,Ty+=ln,"Output Layer     : %d",NetCmd.nOutput);

gprintfxy(Tx,Ty+=ln,"Weight Max-Min   : %5.3f,%5.3f",
                    NetCmd.Wmax,NetCmd.Wmin);

setcolor(LIGHTBLUE);
gprintfxy(Tx,Ty+=ln,"Algrolithms      : %s",Algorithms[NetCmd.AlgNo]);
setcolor(BLUE);
gprintfxy(Tx,Ty+=ln,"Learning rate    : %5.3f",NetCmd.LearnConst);
gprintfxy(Tx,Ty+=ln,"Learning dec     : %5.3f",NetCmd.LearnDec);
gprintfxy(Tx,Ty+=ln,"Neightborhood    : %d",NetCmd.Neighbor);
gprintfxy(Tx,Ty+=ln,"Neightborhood dec: %d",NetCmd.NeigDec);
gprintfxy(Tx,Ty+=ln,"MaxLoop          : %d",NetCmd.MaxLoop);

mouse_cursor_on();
}//end
```

```
/**********************************/
/*          NEUBPN.C              */
/**********************************/
```

เนื่องจากโปรแกรมมีความยาวมาก จึงนำมาเฉพาะส่วนที่เห็นว่าเป็นประโยชน์เท่านั้น รายละเอียดดูเพิ่มเติมได้จากแผ่น DISK

```
struct frame_Board Wframe[MAX_FRAMES];
struct Win_frame Windows;
short nPattern,PatternNo,AutoRun;
short Networkflag,Patflag,Forwardflag,Backwardflag,InputSign,WExist;
float **InputLayer,***WEIGHTS,***deltaW,**LayerOut,**LayerErr,**OutputErr;
float SumSqErr;
signed char **Outputtarget;
short neuronLayer ,neuronSize[5];
struct RectArea BoxPos;
struct MemoryMessage MemMesg;
unsigned long epoch;
//0 ..:.....[MAIN]
void main(int argc,char *argv[])
{
short i,done,MesgID,CanDo;
char buff[30],filename[80];
char tttt[]={"HELLO"};
char HisStr[5][36]={"*.C","*.*","*.EXE","*.COM","*.BAK"};


   strcpy(NetCmd.NetName,"TDIGIT.NET");
   strcpy(NetCmd.Describtion,"Test Thai digit 0-9");
   strcpy(NetCmd.Inputfile,"TDIGIT.GRP");
   NetCmd.NetLayer      = 3;
   strcpy(NetCmd.hiddenSize,"10");    //conv //!
   NetCmd.nOutput       = 10;
   NetCmd.InputSign     = 0;
   NetCmd.Wmin          = -0.3;
   NetCmd.Wmax          = 0.3;
   NetCmd.AlgNo         = 1;
   NetCmd.TranferFn     = 0;
   NetCmd.LearnConst    = 0.15;
   NetCmd.LearnDec      = 0.01;
   NetCmd.Momentum      = 0.075;
   NetCmd.Errlevel      = 0.02;
   NetCmd.Neighbor      = 5;
   NetCmd.NeigDec       = 1;
   NetCmd.MaxLoop       = 150;
```

```
neuronSize[1]       = 10;

MemMesg.InputLF     = NULL;
MemMesg.NetF        = NULL;
MemMesg.WeightPtrF  = NULL;
WExist              = OFF;
neuronLayer         = NetCmd.NetLayer-1;
MemMesg.neuronLayer = neuronLayer;
Networkflag         = OFF;
Patflag             = OFF;
Forwardflag         = OFF;
Backwardflag        = OFF;
InputSign           = OFF;
AutoRun             = OFF;

if(argc>=2)
  InitCommandLine(argc,argv);
else
 {
  CanDo = ReadCommand("NEURON.DEF",&NetCmd,sizeof(NetCmd));
  if(CanDo==OK)
        AutoRun=ON;
 }
#ifdef __BORLAND__
InitXMem();
#endif
sw_graph(16);
InitWindows();


//test

 if(AutoRun==ON)
 {
  CanDo= FileExist(NetCmd.Inputfile,READ_MODE);
  if(CanDo==OK)
   {
    nPattern = ReadPattern(NetCmd.Inputfile);
    if(nPattern>0)
         TrainNetwork();
   }
 }
 //sel
 done = 0;
```

```
while(!done)
{
 MesgID = GetMesgID(&Windows);
 setcolor(WHITE);
 setfillstyle(SOLID_FILL,LIGHTGRAY);
 gprintfxy(590,462,"%4d",MesgID);
 switch(MesgID)
  {
  case MAX_FRAMES:
            break;
  case 100: //load networks
            strcpy(filename,"*.NET");
            CanDo = FileMan("LOAD NETWORKS",filename,READ_MODE);
            if(CanDo==OK)
                 LoadNetwork(filename);//LoadNetInform(filename);
            break;
  case 101: //save networks
            if(WExist==0)
             {
                 MessageWin("MESSAGE","WEIGHT NOT EXIST",
                                  "NETWORKS IS EMPTY",1,2,ON);
                 break;
             }
            strcpy(filename,"*.NET");
            CanDo = FileMan("SAVE NETWORKS",filename,WRITE_MODE);
            if(CanDo==OK)
                 SaveNetwork(filename);
            break;
  case 102:
            break;
  case 103:
            break;
  case 105: //Quit
  case ExitProg : done=1;
            break;
  case 107: //Load Group
            strcpy(filename,"*.GRP");
            CanDo = FileMan("OPEN FILE",filename,READ_MODE);
            if(CanDo==OK)
            {
             nPattern = ReadPattern(filename);
             if(nPattern>0)
                 {
                  setcolor(BLUE);
                 sprintf(buff,"PATTERN %dx%d",
```

```
                    NetCmd.InputXsize,NetCmd.InputYsize);
                outtextxy(30,70,buff);
                ShowPattern(30,75,0);
                }
            }
            break;

    case 106://test
            ClearClient(&Windows,LIGHTGRAY);
            TestPattern();
            break;

    case 108: //Randoms
            if(Patflag==0)
             {ErrMessage(NO_PATTERN_LOAD,ON); break;}
            CanDo = RandDomWeight();
            if(CanDo==OK)
                PlotW(20,372,0,LIGHTRED,ON);
            break;

    case 109:
            LoadWeight();
            break;
    case 110:  //save weight
            SaveWeight();
            break;

    case 111: //view weight
            strcpy(filename,"*.WHT");
            CanDo = FileMan("OPEN FILE",filename,READ_MODE);
            if(CanDo==OK)
                ViewTxt(filename,75,22,BLACK,LIGHTGRAY);
            break;

    case 113:  //training
            TrainNetwork();
            break;

    case 118:
            break;

    case 119:
            MemoryInform();
            break;
    case 120:
```

```
            MessageWin("About","NEURAL TOOLS 1.0",
                                "   KMITL 1996   ",1,0,ON);
            break;
      case 200:
            strcpy(filename,"*.TXT");
            CanDo = FileMan("OPEN FILE",filename,READ_MODE);
            if(CanDo==OK)
                ViewTxt(filename,75,22,BLACK,LIGHTGRAY);
            break;
      default:
            MesgID = NONE;
     }

  }//while

}//end main
//[*]
short TrainNetwork(void)
{
char key8,Str[80],buff[80],buff2[20];
short Sx,Sy,i,j,y,p,done,CanDo,Err,percent,oldpercent,StartTrain,ReturnMesg;
short MaxSCY,MaxSCX,Yscale,Errbuff[300],dy,n,nc,r,init,AutoLn,Select;
short OrgX,OrgY,epCount,div;
unsigned short Key16;
float WorkRatio,fpercent,Dy;
double totalErr,GetErr,OldErr;
time_t first, second;
clock_t start, end;

  Err    = 0;
  init   = 0;
  OldErr = 99999;
  Select = 0;
  AutoLn = OFF;
  while(!init)
  {
    if(Patflag==0)
    {
      ErrMessage(NO_PATTERN_LOAD,ON);
      Err = ON;
      break;
    }
     if(Networkflag==OFF)//1
     {
       CanDo= InitNetPtr();
```

```
        if(!CanDo)
          { Err = ON; break; }
      CanDo= InitWeightPtr();
      if(!CanDo)
          { Err = ON; break; }
     Networkflag = ON;
   }
   if(MemMesg.WeightPtrF==ON && WExist==OFF)  //empty weight
    {
      CanDo = RandDomWeight();
      if(!CanDo)
          { Err = ON; break; }
    }
  init = 1;
 }

 if(Err==0)
 {
   mouse_cursor_off();
   Err = 0; r = 1; nc = 0; p = 0; epoch = 0; epCount=0;
   StartTrain = 0; oldpercent = 0; OrgX  = 60;  OrgY =310;
   MaxSCX =300; MaxSCY =200;

   ClearClient(&Windows,BlackGround);
   setcolor(BLUE);
   sprintf(buff,"Networks Name : %s",NetCmd.NetName);
   outtextxy(20,375,buff);
   sprintf(buff,"Description   : %s",NetCmd.Describtion);
   outtextxy(20,400,buff);

   NetCmd.AlgNo = 1;   //test
   switch(NetCmd.AlgNo)
   {
     case 0 : sprintf(buff2,"Perceptron");
              break;
     case 1 : sprintf(buff2,"BackPropagation");
              break;
   }

   sprintf(buff,"Algorithms    : %s",buff2);
   outtextxy(20,425,buff);

   y = 78;
   sprintf(buff,"Pattern       :%d",nPattern);
   outtextxy(440,y,buff);
```

```
    sprintf(buff,"Input Layer  :%d",neuronSize[0]);
    outtextxy(440,y+=25,buff);
    sprintf(buff,"hidden Layer :%s",NetCmd.hiddenSize);
    outtextxy(440,y+=25,buff);
    sprintf(buff,"Output Layer :%d",NetCmd.nOutput);
    outtextxy(440,y+=25,buff);
    sprintf(buff,"Error Level  :%5.3f",NetCmd.Errlevel);
    outtextxy(440,y+=25,buff);
    sprintf(buff,"Learning rate:%5.3f",NetCmd.LearnConst);
    outtextxy(440,y+=25,buff);
    sprintf(buff,"Momentum     :%5.3f",NetCmd.Momentum);
    outtextxy(440,y+=25,buff);
    outtextxyOn(427,178,"",LIGHTGRAY);      //POINT
//    sprintf(buff,"Tranfer function: %s",buff2);
//    outtextxy(440,78,buff);
    y     = 178;     //""
    start = clock();
    done  = 0;
    while(!done)
    {
     //Calulate ratio or how working
     //find total ERR
          totalErr = 0.0;
          for(j=0;j<nPattern;j++)
           {
            PatternNo=j;
            CanDo = Feedforward();
            if(!CanDo)
             { Err = 1; done = 1; break;}
            for(i=0;i<NetCmd.nOutput;i++)
               totalErr += OutputErr[j][i] * OutputErr[j][i];
           }
           totalErr /= nPattern;
           if(AutoLn==ON)
           {
            if(totalErr > OldErr)
              {
               NetCmd.LearnConst -= 0.05;
               Beep(300,50);
              }
            if(totalErr < OldErr)
              {
               NetCmd.LearnConst += 0.05;
               Beep(800,50);
              }
```

```
        if(totalErr!=OldErr)
         {
          setcolor(BLUE);
          settextjustify(LEFT_TEXT,LEFT_TEXT);
          sprintf(buff,"Learning rate:%5.3f",NetCmd.LearnConst);
          outtextxyOn(440,203,buff,LIGHTGRAY);
         }
        OldErr = totalErr;
       }
//plot Err.....
       if(StartTrain==0)
        {
          Yscale = (short)totalErr; //find max scale of Y
          Yscale++;
          ErrBoard(20,70,Yscale,r);
          Yscale *=10;
        }
       if(nc==0)
       {
         GetErr = totalErr;
         GetErr *= 10;              //keep floating point 2 digit
         Dy     = (GetErr/(float)Yscale)*(float)MaxSCY;
         dy     = (short)Dy;
         Errbuff[epCount] = dy;
         setcolor(RED);
         moveto(OrgX+epCount+1,OrgY-dy);
         lineto(OrgX+epCount+1,OrgY-1);
         epCount++;
       }
       nc++;
       if(nc==r)
          nc=0;

       if(epCount>=MaxSCX)
        {
         r*=2;
         ErrBoard(20,70,(Yscale/10),r);
         epCount=0;

         for(n=0;n<(MaxSCX/2);n++)              //fill old value
          {
           div  =  n*2;
           Errbuff[n] = Errbuff[div];
          }
```

```
            setcolor(RED);
            for(n=0;n<(MaxSCX/2);n++)
              {
               moveto(OrgX+epCount+1,OrgY-Errbuff[n]);
               lineto(OrgX+epCount+1,OrgY-1);
               epCount++;
              }
           }
// end plot Err.....................................
            if(totalErr <= NetCmd.Errlevel)
            {
             end       = clock();
             Ck2Str((end-start),buff2);
             sprintf(Str,
             "Loop %lu   NetWork Error %5.4f   Time:%s    Learning ",
             epoch,totalErr,buff2);
             Working(Str,100);
             Beep(2000,200);
             delay(10);
             Beep(2000,200);
             mouse_cursor_on();
             MessageWin("Message"," Learning complete!",NULL,1,1,ON);

             setcolor(DARKGRAY);
             settextjustify(LEFT_TEXT,LEFT_TEXT);
             sprintf(buff,"epoch        :%lu",epoch);
             outtextxy(360,375,buff);
             sprintf(buff,"Total Error  :%5.3f",totalErr);
             outtextxy(360,400,buff);
             sprintf(buff,"Training time %s",buff2);
             outtextxy(360,425,buff);
             done = 1;
             break;
            }
           if(StartTrain==0)
            {
             WorkRatio  = 100/(totalErr-NetCmd.Errlevel);
             StartTrain = 1;
            }
           else
            {
             fpercent  = 100 - ((totalErr-NetCmd.Errlevel)*WorkRatio);
             percent   = (int)fpercent;
             if(percent!= oldpercent)
```

```
            Working("Learning ",percent);
           oldpercent= percent;
         }

for(p=0;p<nPattern;p++)
{
 //train each pattern
     PatternNo = p;
     Feedforward();
     FeedBack();
     AdaptW();
     if(bioskey(1))
      {
        Key16  = bioskey(0);
        key8   = (char) Key16 & 0xff;
         switch(Key16)
         {
           case DOWN:
                        setcolor(LIGHTGRAY);
                        settextjustify(LEFT_TEXT,LEFT_TEXT);
                        outtextxyOn(427,y,"",LIGHTGRAY);
                        Select++; y+=25;
                        if(Select>=3)
                          {Select=0; y = 178;}
                        setcolor(BLUE);
                        outtextxyOn(427,y,"",LIGHTGRAY);
                        break;
           case UP:
                        setcolor(LIGHTGRAY);
                        settextjustify(LEFT_TEXT,LEFT_TEXT);
                        outtextxyOn(427,y,"",LIGHTGRAY);
                        Select--; y-=25;
                        if(Select<0)
                           {Select=2; y = 228;}
                        setcolor(BLUE);
                        outtextxyOn(427,y,"",LIGHTGRAY);
                        break;
           case 0x1E41:   //A
           case 0x1E61:   //a
                     AutoLn = ON;
                     break;
           case 0x3042:  //B
           case 0x3062:  //b
                     AutoLn = OFF;
```

```
            break;

case 0x4E2B:  //+

    Beep(3000,50);
    setcolor(BLUE);
    settextjustify(LEFT_TEXT,LEFT_TEXT);

    switch(Select)
    {
     case 0:
        NetCmd.Errlevel += 0.01;
        sprintf(buff,"Error Level  :%5.3f",NetCmd.Errlevel);
        break;

     case 1:
        NetCmd.LearnConst += NetCmd.LearnDec;
        sprintf(buff,"Learning rate:%5.3f",NetCmd.LearnConst);
        break;

     case 2:
        NetCmd.Momentum += 0.01;
        sprintf(buff,"Momentum     :%5.3f",NetCmd.Momentum);
        break;

    }
    outtextxyOn(440,y,buff,LIGHTGRAY);
    break;
case 0x4A2D:  //-
    Beep(3000,50);
    setcolor(BLUE);
    settextjustify(LEFT_TEXT,LEFT_TEXT);

    switch(Select)
    {
     case 0:
        NetCmd.Errlevel -= 0.01;
        sprintf(buff,"Error Level  :%5.3f",NetCmd.Errlevel);
        break;

     case 1:
        NetCmd.LearnConst -= NetCmd.LearnDec;
        sprintf(buff,"Learning rate:%5.3f",NetCmd.LearnConst);
        break;
```

```
                case 2:
                  NetCmd.Momentum -= 0.01;
                  sprintf(buff,"Momentum     :%5.3f",NetCmd.Momentum);
                  break;

                }
                  outtextxyOn(440,y,buff,LIGHTGRAY);
                  break;
            default:
                mouse_cursor_on();
                CanDo = MessageWin("Message","Exist Training ",NULL,2,1,ON);
                if(CanDo==OK)
                  { done =  OK; break; }
                mouse_cursor_off();
           }//switch(key8)

          if(done==1)
           break;
        }
     }
     epoch++;
   }//while(!done)
 }//if(Err== 0)

 if(Err==0)
   ReturnMesg = 1;
 else
   ReturnMesg = 0;

 PatternNo    = 0;
 Forwardflag  = 0;
 Backwardflag = 0;
 WriteHelpBar(&Windows,HelpStr);       //Restore Old Help
 mouse_cursor_on();
 return(ReturnMesg);
}//end
//[*]
//        Sucess    Error (NOT_ENOUGH_MEMORY)
// return    1         0
short RandDomWeight(void)
{
short i,j,k,CanDo,done,Err,Mesg;
time_t t;
srand((unsigned) time(&t));
```

```
    done = 0;
    Err  = 0;
    Mesg = 0;
    while(!done)
    {
     if(Networkflag==OFF)
      {
           CanDo= InitNetPtr();
            if(!CanDo)
              { Err = NOT_ENOUGH_MEMORY; break; }
           CanDo= InitWeightPtr();
           if(!CanDo)
              { Err = NOT_ENOUGH_MEMORY; break; }
          Networkflag = ON;
     }
     if(Err==0)
     {
          for(k=0;k<neuronLayer;k++)
           for(j=0;j<neuronSize[k+1];j++)
            for(i=0;i<(neuronSize[k]+1);i++)     //+1 for bias
              WEIGHTS[k][j][i] = Randoms(NetCmd.Wmax,NetCmd.Wmin);

           WExist = ON;
           Beep(2000,100);
           done = 1;
           Mesg = OK;
     }
    }
  return(Mesg);
}//end
//[*]
float Randoms(float max ,float min)
{
float w,w2;
      w = ((float)rand()); //rand return 0->32767
      if(w<16384.0)        //if w > 32767/2
       {
           w += 16384.0;      //up it
           w *= -1;           //make it negative
       }
      w /= 32767.0;        //make it < 0
      if(w>0)
           w2 = (w/0.9)*max;
      if(w<0)
           w2 = (w/0.9)*min*(-1);
```

```
 return(w2);
}
//[*]
int Feedforward(void)
{
int i,j,k,percent,CanDo,Err,done,InputSize;
float buff;
double sum;
float fpercent,temp;
static float WorkRatio;

 InputSize = neuronSize[0];
 Err  = 0;
 done = 0;
 while(!done)
 {
//sum input & hidden1 f(s)-> LayerOut[0][j]
   for(j=0;j<neuronSize[1];j++)
   {
     sum = (double)WEIGHTS[0][j][InputSize];    //bias unit
     for(i=0;i<InputSize;i++)
      {
       if(InputSign)                          //all
            {
             if(InputLayer[PatternNo][i]>0)
               sum += WEIGHTS[0][j][i];
             else
               sum += WEIGHTS[0][j][i]*(-1.0);
            }
       else
            sum += WEIGHTS[0][j][i]*((float)InputLayer[PatternNo][i]);

      }
     //CalBuff[0][j]  = sum;
     LayerOut[0][j] = 1.0/(1.0+exp(-sum));
     temp           = LayerOut[0][j];
   }
 //sum each neron until to output
   for(k=1;k<neuronLayer;k++)                 //each between layer
    {
     for(j=0;j<neuronSize[k+1];j++)           //each neuron node
     {
       sum = WEIGHTS[k][j][neuronSize[k]];   //bias unit
       for(i=0;i<neuronSize[k];i++)           //eack input node
            sum += WEIGHTS[k][j][i]*LayerOut[k-1][i];
```

```
      //CalBuff[k][j]  = sum;
      LayerOut[k][j] = 1.0/(1.0+exp(-sum));
      temp  = LayerOut[k][j];
     }
    }
//Output Error
    SumSqErr = 0.0;          //use for out file
    for(j=0;j<NetCmd.nOutput;j++)
     {
      buff = (float)Outputtarget[PatternNo][j];
      OutputErr[PatternNo][j]    =  buff-LayerOut[neuronLayer-1][j];
      LayerErr[neuronLayer-1][j]  =  OutputErr[PatternNo][j]*
            (LayerOut[neuronLayer-1][j]*(1-LayerOut[neuronLayer-1][j]));
      SumSqErr += OutputErr[PatternNo][j] * OutputErr[PatternNo][j];
     }
     SumSqErr /= 2;

  //MakeForwardFile();
  Forwardflag = 1;
  done        = 1;
 }//while(!done)
  if(Err!=0)
   ErrMessage(Err,ON);

  return(Forwardflag);
}//end
//calculate Error in LayerErr
void FeedBack(void)
{
int i,j,k;
double sum;
float tmp;

 if(Forwardflag==1)
 {
    for(k=(neuronLayer-2);k>=0;k--)          //k = LayerErr#k
     for(i=0;i<neuronSize[k+1];i++)
      {
          sum =0.0;
          for(j=0;j<neuronSize[k+2];j++)              //neuron right hand
            sum += LayerErr[k+1][j] * WEIGHTS[k+1][j][i];

          LayerErr[k][i] = sum * LayerOut[k][i] * (1.0-LayerOut[k][i]);
          tmp = LayerErr[k][i];
      }
```

```
   Backwardflag = 1;
 }//if(Forwardflag==1)
 else
   ErrMessage(6,ON);
}//end
//[*]
int AdaptW(void)
{
int i,j,k;
float delta,dw;

if(Backwardflag == 1)
 {
//Adaptive WEIGHTS Output:hidden and the other hidden:hidden
     for(k=(neuronLayer-1);k>=0;k--)
     {
      for(j=0;j<neuronSize[k+1];j++)
      {
          //bias unit
          delta = LayerErr[k][j];
          dw = (NetCmd.LearnConst * delta) +
              (NetCmd.Momentum * deltaW[k][j][neuronSize[k]]);
          WEIGHTS[k][j][neuronSize[k]] += dw;
          deltaW [k][j][neuronSize[k]]  = dw;
          for(i=0;i<neuronSize[k];i++)
          {
           if(k!=0)
             delta = LayerErr[k][j]*LayerOut[k-1][i];
           else
             {
              if(InputSign)
               {
                if(InputLayer[PatternNo][i]>0)       //all
                      delta = LayerErr[k][j];
                else
                      delta = LayerErr[k][j]* (-1);
               }
              else
                delta = LayerErr[k][j]*((float)InputLayer[PatternNo][i]);
             }

          dw  = (NetCmd.LearnConst * delta) +
              (NetCmd.Momentum * deltaW[k][j][i]);
          WEIGHTS[k][j][i] += dw;
          deltaW [k][j][i]  = dw;
```

```
            if(bioskey(1))              // When key press
              return(0);
            }//i
        }//j
       }//k
    Forwardflag  = 0;
    Backwardflag = 0;
  }//if(Backwardflag == 1)
  else
     ErrMessage(7,ON);
     return(0);
}//end
```

```
/*********************************/
/*          NEUSOM.C             */
/*********************************/
```

เนื่องจากโปรแกรมมีความยาวมาก จึงนำมาเฉพาะส่วนที่เห็นว่าเป็นประโยชน์เท่านั้น รายละเอียดดูเพิ่มเติมได้จากแผ่น DISK

```
struct frame_Board Wframe[MAX_FRAMES];
struct Win_frame Windows;
short nPattern,PatternNo,AutoRun;
short Networkflag,Patflag,Forwardflag,Backwardflag,WExist,ViewWeight;
float **InputLayer,***WEIGHTS,**LayerOut,**LayerErr,**OutputErr;
float SumSqErr;
signed char **Outputtarget;
short neuronLayer ,neuronSize[5];
struct RectArea BoxPos;
struct MemoryMessage MemMesg;
unsigned long epoch;
int colorShd[]={15,12,13,11,14,10,2,3,7,5,4,6,9,1,8,0};
//0 ..........................................................[MAIN]
void main(int argc,char *argv[])
{
short i,done,MesgID,CanDo;
char buff[30],filename[80];
char tttt[]={"HELLO"};
char HisStr[5][36]={"*.C","*.*","*.EXE","*.COM","*.BAK"};


   strcpy(NetCmd.NetName,"TDIGIT.NET");
   strcpy(NetCmd.Describtion,"Test Thai digit 0-9");
   strcpy(NetCmd.Inputfile,"TDIGIT.GRP");
   NetCmd.NetLayer      = 2;
   NetCmd.hiddenSize[0] = 0;
   NetCmd.nOutput       = 20;
   NetCmd.InputSign     = 0;
   NetCmd.Wmin          = -0.3;
   NetCmd.Wmax          = 0.3;
   NetCmd.AlgNo         = ALG_SOM;
   NetCmd.TranferFn     = 0;
   NetCmd.LearnConst    = 0.55;
   NetCmd.LearnDec      = 0.01;
   NetCmd.Momentum      = 0.075;
   NetCmd.Errlevel      = 0.02;
   NetCmd.Neighbor      = 5;
   NetCmd.NeigDec       = 1;
```

```
  NetCmd.MaxLoop       = 150;

//  neuronSize[1]       = 10;

  MemMesg.InputLF     = NULL;
  MemMesg.NetF        = NULL;
  MemMesg.WeightPtrF  = NULL;
  WExist              = OFF;
  neuronLayer         = NetCmd.NetLayer-1;
  MemMesg.neuronLayer = neuronLayer;
  Networkflag         = OFF;
  Patflag             = OFF;
  Forwardflag         = OFF;
  Backwardflag        = OFF;
  AutoRun             = OFF;
  ViewWeight          = OFF;

  if(argc>=2)
    InitCommandLine(argc,argv);
  else
   {
    CanDo = ReadCommand("NEURON.DEF",&NetCmd,sizeof(NetCmd));
    if(CanDo==OK)
        AutoRun=ON;
   }
  #ifdef __BORLAND__
  InitXMem();
  #endif
  sw_graph(16);
  InitWindows();


//test

  if(AutoRun==ON)
  {
    CanDo= FileExist(NetCmd.Inputfile,READ_MODE);
    if(CanDo==OK)
     {
      nPattern = ReadPattern(NetCmd.Inputfile);
      if(nPattern>0)
          TrainNetwork();
     }
  }
  //sel
```

```
done = 0;
while(!done)
{
  MesgID = GetMesgID(&Windows);
  setcolor(WHITE);
  setfillstyle(SOLID_FILL,LIGHTGRAY);
  gprintfxy(590,462,"%4d",MesgID);
  switch(MesgID)
   {
    case MAX_FRAMES:
            break;
    case 100: //load networks
            strcpy(filename,"*.NET");
            CanDo = FileMan("LOAD NETWORKS",filename,READ_MODE);
            if(CanDo==OK)
                 LoadNetwork(filename);//LoadNetInform(filename);
            break;
    case 101: //save networks
            if(WExist==0)
             {
                 MessageWin("MESSAGE","WEIGHT NOT EXIST",
                                  "NETWORKS IS EMPTY",1,2,ON);
                 break;
             }
            strcpy(filename,"*.NET");
            CanDo = FileMan("SAVE NETWORKS",filename,WRITE_MODE);
            if(CanDo==OK)
                 SaveNetwork(filename);
            break;
    case 102:
            break;
    case 103:
            break;
    case 105: //Quit
    case ExitProg : done=1;
            break;
    case 107: //Load Group
            strcpy(filename,"*.GRP");
            CanDo = FileMan("OPEN FILE",filename,READ_MODE);
            if(CanDo==OK)
            {
             nPattern = ReadPattern(filename);
             if(nPattern>0)
                 {
                  setcolor(BLUE);
```

```
            sprintf(buff,"PATTERN %dx%d",
                    NetCmd.InputXsize,NetCmd.InputYsize);
            outtextxy(30,71,buff);
            ShowPattern(30,75,0);
           }
        }
        break;

case 106://test
        ClearClient(&Windows,LIGHTGRAY);
        TestPattern();
        break;

case 108: //Randoms
        if(Patflag==0)
         {ErrMessage(NO_PATTERN_LOAD,ON); break;}
        CanDo = RandDomWeight();
        if(CanDo==OK)
            PlotW(20,372,0,LIGHTRED,ON);
        break;

case 109:
        LoadWeight();
        break;
case 110:  //save weight
        SaveWeight();
        break;

case 111: //view weight ON OFF
        setcolor(BLACK);
        settextjustify(LEFT_TEXT,LEFT_TEXT);
        setfillstyle(SOLID_FILL,LIGHTGRAY);
        bar(6,406,342,440);
        if(ViewWeight==ON)
         {
            outtextxy(25,439,"View Weights OFF");
            ViewWeight = OFF;
         }
        else
         {
            outtextxy(25,439,"View Weights  ON");
            ViewWeight = ON;
         }
        break;
```

```
    case 113:  //training
            TrainNetwork();
            break;

    case 118:
            break;

    case 119:
            MemoryInform();
            break;
    case 120:
            MessageWin("About","NEURAL TOOLS 1.0",
                                "   KMITL 1996   ",1,0,ON);
            break;
    case 200:
            strcpy(filename,"*.TXT");
            CanDo = FileMan("OPEN FILE",filename,READ_MODE);
            if(CanDo==OK)
                ViewTxt(filename,75,22,BLACK,LIGHTGRAY);
            break;
    default:
            MesgID = NONE;
    }

  }//while

  closegraph();
}//end main
short TrainNetwork(void)
{
char WAITE,STOP,StopCondition;
short Sx,Sy,Ex,Ey,i,p,CanDo,mx,my,done,MesgID,winner,Tx,Ty,TxtLn,epoch;
short NeighB,init,Err,ReturnMesg,Select,LineSpc;
unsigned short Key16,KeyState;
float LearnC;

//Training
  Err    = 0;
  init   = 0;

  normPatV();
  normWeightV();

  while(!init)
  {
```

```
 if(Patflag==0)
 {
   ErrMessage(NO_PATTERN_LOAD,ON);
   Err = ON;
   break;
 }
  if(Networkflag==OFF)
  {
    CanDo= InitNetPtr();
        if(!CanDo)
          { Err = ON; break; }
    CanDo= InitWeightPtr();
    if(!CanDo)
          { Err = ON; break; }
   Networkflag = ON;
 }
 if(MemMesg.WeightPtrF==ON && WExist==OFF)  //empty weight
  {
    CanDo = RandDomWeight();
    if(!CanDo)
          { Err = ON; break; }
  }
 init = 1;
}
if(Err==0)
{
 LearnC    = NetCmd.LearnConst;
 NeighB    = NetCmd.Neighbor;
 ClearClient(&Windows,BlackGround);
 mouse_cursor_off();
 WriteHelpBar(&Windows,"Press <+> for Quick Process  "
 "Press <0> for Stop");
 //lable
 setcolor(BLUE);
 setfillstyle(SOLID_FILL,BlackGround);
 outtextxy(20,421,"-0.99");
 outtextxy(152,421,"0.99");
 ColorShade(20,425);

 setcolor(BLUE);
 setfillstyle(SOLID_FILL,BlackGround);
 gprintfxy(30,71,"PATTERN %dx%d ",
           NetCmd.InputXsize,NetCmd.InputYsize);
 outtextxy(218,71,"WINNER WEIGHT"); //Sy-4
```

```
//left lable
setcolor(BLUE);
Tx    = 380;

outtextxy(Tx,70,"KOHONEN SELF-ORGANIZING MAPS");
gprintfxy(Tx,90,"Cluster Output   %4d",NetCmd.nOutput);
outtextxy(Tx,110 ,"Max Loop");
gprintfxy(516,110,"%4d",NetCmd.MaxLoop);
outtextxy(Tx,130,"Learning rate");
outtextxy(Tx,150,"Neighborhood Size");
outtextxy(Tx,170,"Epoch ");
Ty = 110;
Select  = 0;
LineSpc = 20;
outtextxyOn(365,Ty,"",LIGHTGRAY);       //POINT

setcolor(BLACK);
settextjustify(LEFT_TEXT,LEFT_TEXT);
if(ViewWeight==ON)
 outtextxyOn(195,439,"View Weights  ON",LIGHTGRAY);
else
 outtextxyOn(195,439,"View Weights OFF",LIGHTGRAY);

if(ViewWeight==ON)
{
 ShowPattern(30,75,0);    //draw first to set BoxPos.Ey
 CanDo = PlotWShade(20,BoxPos.Ey+36,0,ON);
 if(!CanDo)
  {
   setcolor(RED);
   outtextxyOn(195,439,"Weights too BIG ",LIGHTGRAY);
   ViewWeight = OFF;
   WAITE      = ON;
  }
 else
   WAITE = OFF;
}
else
  WAITE = ON;


STOP = OFF; StopCondition = 0; epoch = 0;

done = 0;
while(!done)
```

```
{
 for(i=0;i<=NetCmd.MaxLoop;i++)
  {
   for(p=0;p<nPattern;p++)
   {
        ShowPattern(30,75,p);
        setcolor(BLUE);
        setfillstyle(SOLID_FILL,BlackGround);
        gprintfxy(BoxPos.Sx,BoxPos.Ey+20,"Pattern %3d",p);
        winner = GetWinner(p);
        AdjustWeight(winner,NeighB,InputLayer[p]);
        ShowWeight(218,75,winner);
        if(ViewWeight==ON)
          PlotWShade(20,BoxPos.Ey+36,0,OFF);
        setcolor(BLUE);
        setfillstyle(SOLID_FILL,BlackGround);
        gprintfxy(BoxPos.Sx,BoxPos.Ey+20,"Winner %4d",winner);

        gprintfxy(516,130,"%4.2f",LearnC);
        gprintfxy(524,150,"%3d",NeighB);
        gprintfxy(424,170,"%4u",epoch);

        if(WAITE==ON)
           delay(500);

        if(STOP==ON)
          while(!bioskey(1));

        if(bioskey(1))
        {
          Key16 = bioskey(0);
          switch(Key16)
          {
           case DOWN:
                  setcolor(LIGHTGRAY);
                  settextjustify(LEFT_TEXT,LEFT_TEXT);
                  outtextxyOn(365,Ty,"",LIGHTGRAY);
                  Select++; Ty+=LineSpc;
                  if(Select>=3)
                     {Select=0; Ty = 110;}
                  setcolor(BLUE);
                  outtextxyOn(365,Ty,"",LIGHTGRAY);
                  break;
           case UP:
                  setcolor(LIGHTGRAY);
```

```
                    case 1:
                      LearnC -= 0.1;
                      gprintfxy(516,130,"%4.2f",LearnC);
                      break;
                    case 2:
                      NeighB-=1;
                      gprintfxy(524,150,"%3d",NeighB);
                      break;
                   }
                   break;
            case 0x372A: //(*)
                   WAITE=OFF;STOP=OFF;
                   WriteHelpBar(&Windows,"Press </> to Slow Process  "
                   "Press <0> to Stop");
                    break;
#ifdef __BORLAND__
            case 0x352F: //(/)
#endif
#ifdef __WATCOM__
            case 0xE02F: //(/)
#endif
                   WAITE=ON;STOP=OFF;
                   WriteHelpBar(&Windows,"Press <*> for Quick Process  "
                   "Press <0> for Stop");
                    break;

            default:
                   if(STOP==OFF || Key16==ESC)
                   {
                    mouse_cursor_on();
                    CanDo = MessageWin("Message","Exist Training ",NULL,2,1,ON);
                    if(CanDo==OK)
                     { done =  OK;
                       StopCondition = 1;
                       done = 1;
                     }
                    mouse_cursor_off();
                   }
            }//switch
            if(done)
              break;
          }
      }
      epoch++;
      if(NeighB>0)
```

```
            NeighB--;
        LearnC -= NetCmd.LearnDec;
        if(LearnC<=0)
            LearnC = 0.3;//done=1;
        if(done)
            break;
       }
       done=1;
     }//while(!done)
     mouse_cursor_on();

     if(StopCondition == 0)
     {
      Beep(2000,200);
      delay(10);
      Beep(2000,200);
      mouse_cursor_on();
      MessageWin("Message","Learning done",NULL,1,1,ON);
     }
    }
    if(Err==0)
      ReturnMesg = 1;
    else
      ReturnMesg = 0;

    PatternNo    = 0;
    WriteHelpBar(&Windows,HelpStr);      //Restore Old Help
    mouse_cursor_on();
    return(ReturnMesg);

}//end
//[*]
//        Sucess    Error (NOT_ENOUGH_MEMORY)
// return    1         0
short RandDomWeight(void)
{
short i,j,k,CanDo,done,Err,Mesg;
time_t t;
srand((unsigned) time(&t));

    done = 0;
    Err  = 0;
    Mesg = 0;
    while(!done)
    {
```

```
    if(Networkflag==OFF)
     {
          CanDo= InitNetPtr();
           if(!CanDo)
             { Err = NOT_ENOUGH_MEMORY; break; }
          CanDo= InitWeightPtr();
          if(!CanDo)
             { Err = NOT_ENOUGH_MEMORY; break; }
         Networkflag = ON;
    }
    if(Err==0)
    {
         for(k=0;k<neuronLayer;k++)
          for(j=0;j<neuronSize[k+1];j++)
           for(i=0;i<(neuronSize[k]+1);i++)     //+1 for bias
             WEIGHTS[k][j][i] = Randoms(NetCmd.Wmax,NetCmd.Wmin);

          WExist = ON;
          Beep(2000,100);
          done = 1;
          Mesg = OK;
    }
   }
  return(Mesg);
}//end
//[*]
float Randoms(float max ,float min)
{
float w,w2;
     w = ((float)rand()); //rand return 0->32767
     if(w<16384.0)        //if w > 32767/2
      {
          w += 16384.0;       //up it
          w *= -1;            //make it negative
      }
     w /= 32767.0;         //make it < 0
     if(w>0)
          w2 = (w/0.9)*max;
     if(w<0)
          w2 = (w/0.9)*min*(-1);
  return(w2);
}
short GetWinner(short p)
{
short i,j,winner;
```

```
float dis,Eucldis,mindis;

          for(j=0;j<NetCmd.nOutput;j++)
           {
            Eucldis = 0;
            for(i=0;i<neuronSize[0];i++)
            {
             dis = InputLayer[p][i] - WEIGHTS[0][j][i];
             Eucldis += dis*dis;
            }
            if(j==0 || Eucldis <= mindis)
             {
              mindis = Eucldis;
              winner = j;
             }
           }
 return(winner);
}//end
void AdjustWeight(short winner,short Radius,float *pat)
{
short i,j,Leftneuron,Rightneuron;
float temp;

     Leftneuron = winner - Radius;
     Rightneuron= winner + Radius;

     if(Leftneuron<0)
          Leftneuron  =.0;
     if(Rightneuron > NetCmd.nOutput-1)
          Rightneuron = NetCmd.nQutput-1;

     for(i=0;i<neuronSize[0];i++)
     {
      for(j=Leftneuron;j<=Rightneuron;j++)
      {
          //Weight[j][i] += (LearnConst * (pat[i]-Weight[j][i]));
          temp = (NetCmd.LearnConst * (pat[i]-WEIGHTS[0][j][i]));
          WEIGHTS[0][j][i] += temp;
      }
     }

}//end
//normalize the weight vectors
void normWeightV(void)
{
```

```
short i,j,k;
float sum,normw;

  for(j=0;j<nOutput;j++)
   {
    sum=0;
    for (i=0; i<nInput; i++)
        sum+=Weight[j][i]*Weight[j][i];

    normw = (float)sqrt((double)sum);

   //adapt to normalize
    for(i=0;i<nInput;i++)
        Weight[j][i] /= normw;
   }
}//end
//normalize the Input vectors
void normPatV(void)
{
short i,p;
float norml;

   for(p=0;p<nPattern;p++)
   {
    norml= 0;
    for(i=0;i<nInput;i++)
    {
     norml+= InputPattern[p][i]*InputPattern[p][i];
    }
    norml = 1/(float)sqrt((double)norml);
    for(i=0;i<nInput;i++)
    {
     InputPattern[p][i] *= norml;
    }
   }
}//end
```

เอกสารอ้างอิง

1. Russell C. Eberhart and Roy W. Dobbins "Neural Network PC Tools A Practical Guide" Academic Press, Inc. 1990

2. James A Freeman David M. Skapura "Neural Networks Algorithms, Application and Programming Techniques" Addison-Weslev Publishing Company,Inc. October 1991

3. Judith E. Dayhoff "Neural Network Architecture An Introduction" VAN NOSTRAND REINHOLD New York 1990

4. Computer Sciences Departmant University of Wisconsin,Madison USA "Neural Network Models in Artificial Intelligence" ELLIS HORWOOD 1989

5. Maureen Caudill and Charles Butler "Understanding Neural NetworksComputer Exploration"
   A Bradford Book 1993

6. Laurene Fausett "Fundamentals of Neural Networks Architectures,Algorithm, And Application" Prentice-Hall International, Inc 1994

7. Bernard Widrow and Michael A. Lehr "30 Year Adaptive Neural Networks: Perceptron, Madaline and Backpropagation" IEEE ,pp 1415-1441 ,september 1990

8. ศมิทธิ์ เอมสมบัติ ดร.บุญธีร์ เครือตราชู รศ.ดร.ชม กิ้มปาน "โปรแกรมช่วยออกแบบโครงข่ายประสาทเทียม" วารสารคอมพิวเตอร์ สมาคมคอมพิวเตอร์แห่งประเทศไทยในพระบรมราชูปถัมภ์

ประวัติผู้เขียน

นายศมิทธ์ เอมสมบัติ เกิดวันที่ 2 มกราคม 2504 ที่จังหวัดกรุงเทพฯ สำเร็จการศึกษาวิทยาศาสตร์บัณฑิต (ฟิสิกส์) จากมหาวิทยาลัยรามคำแหง ปีการศึกษา 2530 ผลงานวิจัยที่ผ่านมา

ศมิทธ์ เอมสมบัติ ดร.บุญธีร์ เครือตราชู รศ.ดร.ชม คิ้มปาน "โปรแกรมช่วยออกแบบโครงข่ายประสาทเทียม" วารสารคอมพิวเตอร์ สมาคมคอมพิวเตอร์แห่งประเทศไทยในพระบรมราชูปถัมภ์