รายงานวิจัยเรื่อง

เทคนิคการจดจำรหัสเลขโดยเปอเซพตรอนแบบหลายชั้น

โดย ผศ. ดร. กิตติ ไพฑูรย์วัฒนกิจ
นาย เศรฐพล สิ้นปราชญา
นาย วันฉัตร ผดุงรัตน์
นาย สาธิต อินทจักร์

โครงการนี้ได้รับการสนับสนุนจากสำนักงาน คณะกรรมการวิจัยแห่งชาติ
ประจำปี ๒๕๓๖

เทคนิคการจดจำรหัสเลขโดยเปอเซพตรอนแบบหลายชั้น

บทคัดย่อ

ในปัจจุบันเริ่มมีคนสนใจนำนิวรอลเน็ตเวิร์ค (Neural Networks) ซึ่งเป็นศาสตร์ที่จำลองแบบความสามารถของมนุษย์มาช่วยงานประมวลผลข้อมูลด้านการเรียนรู้ การจดจำและแยกแยะรูปแบบหรือแพทเทิร์น (Pattern) เนื่องจากมีความยืดหยุ่นในการทำงานสูงและสามารถปรับตัวเองให้รับรู้สภาพที่เปลี่ยนแปลงได้ โดยรายงานนี้จะเสนอหลักการของเปอเซพตรอนแบบหลายชั้นที่เป็นโครงสร้างชนิดหนึ่งของนิวรอลเน็ตเวิร์ค และวิธีการของแบ็คพร็อบพาเกชั่น (Back Propagation) เพื่อใช้ในการเรียนรู้ที่จะการจดจำและแยกแยะแพทเทิร์นตัวอักษรแบบตัวพิมพ์ได้ สามารถนำไปประยุกต์ใช้กับงานทางด้าน OCR(Optical Character Reader) ได้เป็นอย่างดี ผลจากการทดสอบพบว่า วิธีการของแบ็คพร็อบพาเกชั่น-นิวรอลเน็ตเวิร์ค สามารถจดจำและแยกแยะแพทเทิร์นได้ดี ถึงแม้ว่าข้อมูลจะมีความผิดเพี้ยน (Distortion) เนื่องจาก สัญญาณรบกวน (Noise) หรือแพทเทิร์นตัวอักษรมีคุณภาพไม่ดี

ABSTRACT

Today many researchers use neural networks which simulate the human ability to help in data processing about learning, recognition and pattern classification. Because it is flexible and can adjust itself to learn the changed environment. This paper employs the back propagation principle in learning to recognize the pattern of characters with multilayer perceptron structure, one kind of neural networks. It can apply for OCR to achieve formore accurately outcome. The preliminary result of neural network by back propagation has shown the ability to recognize well although the acquired input-characters have been distorted due to noise or bad quality.

คำนำ

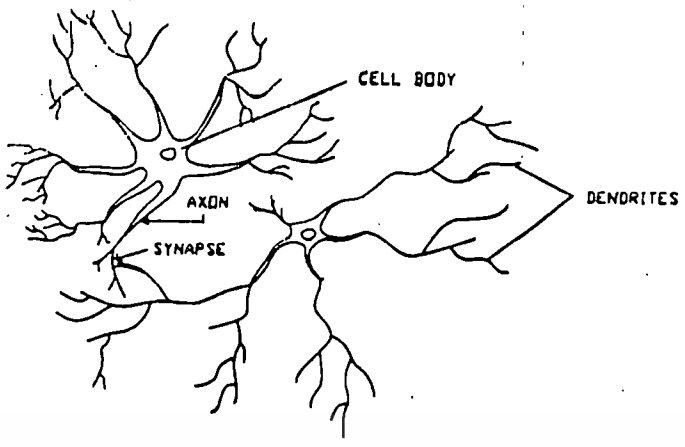
การจดจำแพทเทิร์นต่างๆโดยสมองมนุษย์ ก่อให้เกิดความรู้ ความเข้าใจ และนำไปใช้สร้างภูมิปัญญา โดยมนุษย์เองสามารถจดจำสิ่งที่พบเห็นในลักษณะของแพทเทิร์น และบอกได้ว่าเคยพบเห็นมาแล้วหรือไม่ และสิ่งที่เห็นคืออะไร ซึ่งสิ่งที่พบเห็นได้แก่ ภาษา เสียงพูดและรูปวาดของภาพต่างๆ เพื่อทำให้เกิดการสื่อความหมายของสิ่งที่พบเห็นอยู่ภายในตัวแพทเทิร์น โดยอาศัยประสบการณ์จากการเรียนรู้ และการฝึกฝนที่สะสมมาตั้งแต่ยังเป็นเด็ก

ปัจจุบันเมื่อวิทยาการเจริญมากขึ้น มนุษย์เริ่มนำหลักการพื้นฐานที่มีลักษณะใกล้เคียงกับการทำงานของสมองมนุษย์ในรูปแบบของ อาร์ติฟิเชียลนิวรอลเน็ตเวิร์ค (Artificial Neural Networks) มาช่วยในการตัดสินใจกับงานทางด้านอุตสาหกรรม คมนาคม การสื่อสาร และอื่นๆมากขึ้น

นิวรอล (Neural) ในสมองมนุษย์

ระบบเส้นประสาทของมนุษย์ประกอบด้วยเซลเล็กๆมากกมาย ที่เรียกว่า นิวรอล หรือเซลประสาทภายในสมองมนุษย์ ซึ่งรวมตัวกันอยู่ในรูปโครงสร้างที่ซับซ้อน มีอยู่ประมาณ 10^{11} นิวรอลในสมองมนุษย์ โดยนิวรอลแต่ละตัวประกอบด้วยส่วนสำคัญ 3 ส่วน คือ

1. เด็นไดร์ท (Dendrite) ตัวรับสัญญาณ อินพุท (input)
2. แอ็คซอน (Axon) ตัวส่งสัญญาณ เอาท์พุท (output)
3. ซินแนปส์ (Synapse) จุดเชื่อมต่อ (connection point)

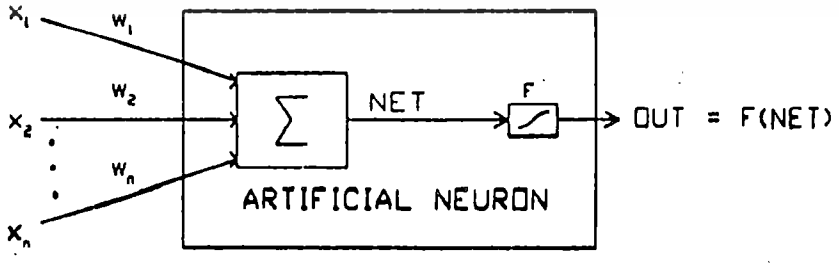


รูปที่ 1 แสดงนิวรอลทางชีววิทยา

แต่ละนิวรอลจะเชื่อมต่อกับนิวรอลอื่นอีกหลายนิวรอล โดยเด็นไดร์ทจะทำหน้าที่รับสัญญาณที่จุดเชื่อมต่อที่เรียกว่า ซินแนปส์ มีผลทำให้เกิดการลดหรือเพิ่มความเข้มของสัญญาณ ซึ่งบางสัญญาณมีผลกระตุ้นตัวเซล บางสัญญาณมีผลยับยั้งตัวเซล เมื่อผลรวมของสัญญาณทั้งหมดมากกว่าค่าเทรดโฮล (Threshold) นิวรอลจะส่งสัญญาณโดยผ่านแอกซอนไปให้นิวรอลอื่นในทันที กล่าวโดยสรุปว่า นิวรอลหรือเซลประสาทจะทำหน้าที่ 2 ประการคือ ทำการคำนวณ และส่งผลการคำนวณผ่านปลายหนึ่งของเซลไปให้เซลอื่นต่อไป

<u>คอมพิวเตอร์นิวรอลเน็ตเวิร์ค</u>

นิวรอลเน็ตเวิร์คในแง่ของคอมพิวเตอร์ประกอบด้วยหน่วยหรือส่วนประมวลผล (Processing Elements) เชื่อมโยงกับหลายๆตัวในลักษณะขนาน คล้ายกับนิวรอลในสมองมนุษย์ ซึ่งมีลักษณะเป็นสถาปัตยกรรมแบบขนาน (Parallel Architecture) เพื่อแปลงข้อมูลจากรูปหนึ่งไปเป็นอีกรูปหนึ่ง การใช้งานนิวรอลเน็ตเวิร์คจะเป็นไปในรูปการสอน (Training)



รูปที่ 2 อาร์ติฟิเชียลนิวรอลกับแอคติเวชั่นฟังก์ชั่น

จากรูปที่ 2 $$net = \Sigma w_i x_i \quad \text{..... (1)}$$

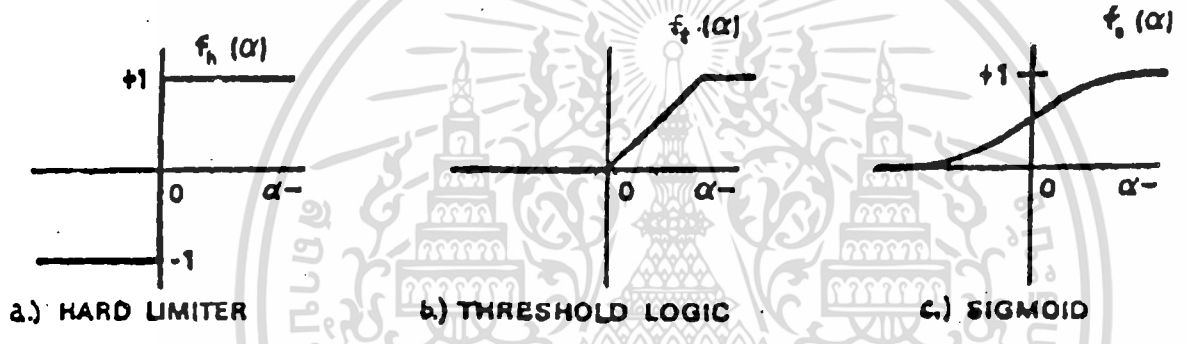
$$= x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$$

โดย $x_1, x_2, \ldots, x_n$ เป็นกลุ่มของข้อมูลอินพุท

$w_1, w_2, \ldots, w_n$ เป็นกลุ่มของตัวเลขน้ำหนัก

net เป็นผลรวมสุทธิที่เกิดจากค่าอินพุท (x) คูณกับ ตัวเลขน้ำหนัก (w)

จากรูปที่ 2 นิวรอล 1 โหนดจะมีค่าอินพุทหลายค่า แต่จะมีค่าเอาท์พุทเพียงค่าเดียว โดยเอาท์พุทในแต่ละนิวรอล เกิดจากผลรวมของผลคูณระหว่างอินพุท และตัวเลขน้ำหนัก โดยส่งผ่านให้กับแอคติเวชั่นฟังก์ชั่น (Activation function) ดังในรูปที่ 3



รูปที่ 3 แสดงชนิดของแอคติเวชั่นฟังก์ชั่น

โดยทั่วไปจะเลือกใช้ Sigmoid ฟังก์ชั่น เป็นแอคติเวชั่นฟังก์ชั่น ซึ่งมีรูปร่างคล้ายตัวอักษร "S" ในภาษาอังกฤษ เพื่อช่วยขยายสัญญาณ net ในช่วงแคบๆได้ ทำให้ได้ค่าจำกัดในขอบเขตที่ต้องการ โดยได้ผลลัพธ์เพียงค่าเดียวคือ out ซึ่งมีค่าอยู่ในช่วง 0 ถึง 1

$$out = f(net) = \frac{1}{1 + e^{-(net+\theta)/\theta_0}} \quad \text{..... (2)}$$

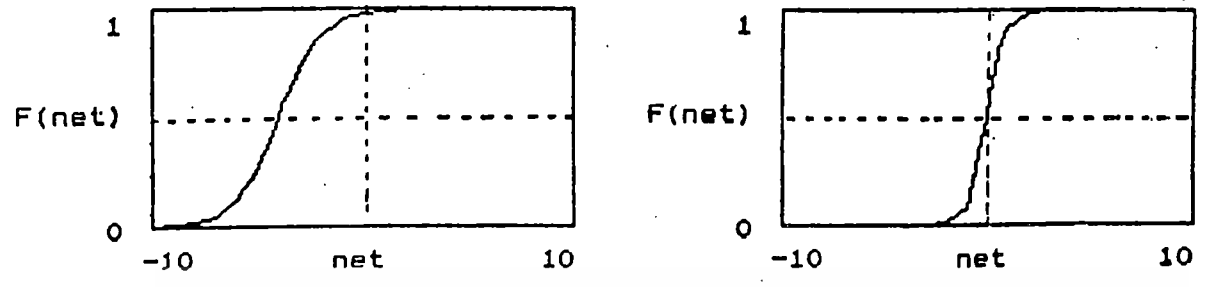
โดย out เป็นค่าสัญญาณเอาท์พุท

f เป็นแอคติเวชั่นฟังก์ชั่น ดังในรูปที่ 3

θ เป็นค่าเทรดโฮล (Threshold) หรือ ค่าไบอัส (Bias)

θ_0 เป็นค่าที่ใช้ปรับรูปร่างของ Sigmoid ฟังก์ชั่น

ถ้า θ มีค่าเป็นบวก รูป Sigmoid ฟังก์ชั่นจะเลื่อนไปทางซ้ายตามแนวแกนนอน และถ้า θ_0 มีค่าน้อยมาก จะทำให้รูป Sigmoid ฟังก์ชั่นมีลักษณะคล้ายรูปฟังก์ชั่นขั้นบันได



a) θ_0 มีค่ามากกว่า 0 b) θ_0 มีค่าน้อยๆ

รูปที่ 4 แสดงผลของไบอัส และค่าที่ใช้ปรับรูปร่างของ Sigmoid ฟังก์ชั่น

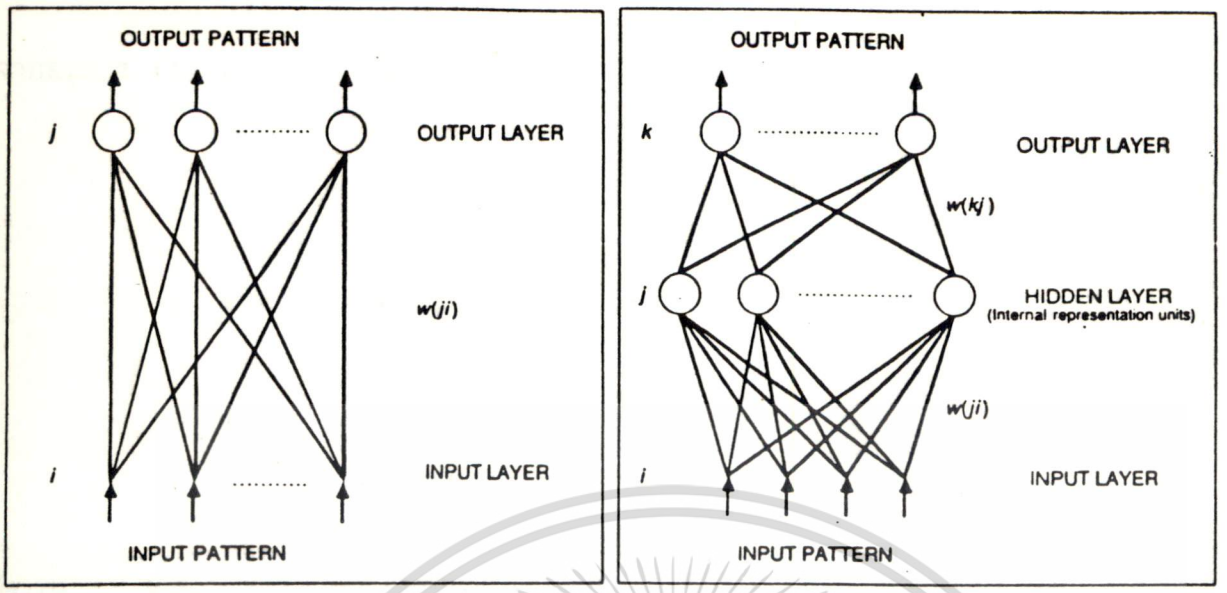
ข้อสังเกต จากสมการหาค่าผลรวมอินพุท (net) มีค่าเป็นบวกมากๆ เอาท์พุทจะมีค่าเข้าใกล้ 1 ในทางตรงกันข้าม ถ้าผลรวมอินพุท (net) น้อยมากติดลบ ค่าเอาท์พุท (out) จะมีค่าเข้าใกล้ 0 ซึ่งสอดคล้องกับสถานะเปิด (On) และ ปิด (Off)

เน็ตเวิร์คหลายนิวรอลและหลายเลเยอร์

จากที่กล่าวไปในตอนต้นนั้นเป็นเพียงขั้นตอน หรือวิธีการทำงานของนิวรอลเพียงโหนดเดียวซึ่งเทียบได้กับเซลเพียงเซลเดียว หากประกอบกันเป็นหลายเซล จะต้องให้นิวรอลแต่ละตัวเชื่อมต่อถึงกันทำให้เกิดเป็นเน็ตเวิร์คที่มีลักษณะเป็นชั้นๆ เรียกว่า เลเยอร์ (Layer) ในหนึ่งเลเยอร์สามารถมีจำนวนนิวรอลได้มากกว่า 1 นิวรอล จากรูปภายในเลเยอร์เดียวกันจะไม่มีการเชื่อมต่อถึงกัน และนิวรอลที่มีเลเยอร์ต่ำกว่า จะรับค่ามาจากเอาท์พุทของเลเยอร์ที่อยู่เหนือกว่าเพื่อเป็นอินพุทของตัวเอง จากลักษณะดังกล่าวจะเห็นว่า คุณสมบัติของนิวรอลทุกตัวที่อยู่ในเลเยอร์ที่ต่ำกว่า จะถูกถ่ายทอดไปยังนิวรอลในชั้นถัดไป

รูปที่ 5-a เป็นรูปที่แสดงลักษณะของนิวรอลที่ประกอบกันเป็นเน็ตเวิร์ค แบบเลเยอร์เดียว (Single-layer) โดยในเลเยอร์แรก หรืออินพุทเลเยอร์โดยแท้จริงแล้วไม่ใช้นิวรอลเพราะไม่มีการคำนวณอยู่ภายใน แต่จะแสดงลักษณะการกระจายของค่าอินพุทให้กับนิวรอลในชั้นถัดไปเท่านั้น

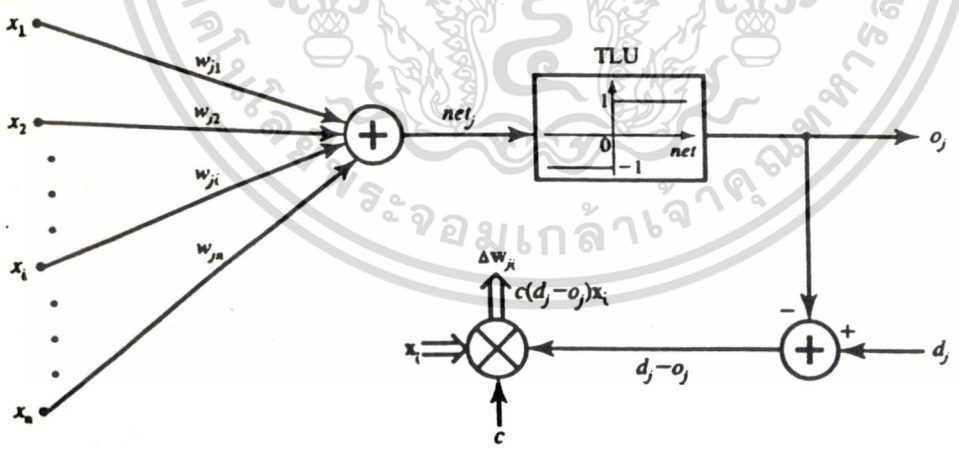
รูปที่ 5-b เป็นรูปแสดงลักษณะของเน็ตเวิร์คที่ใหญ่กว่า ซับซ้อนมากกว่า เพื่อให้ใกล้เคียงกับความเป็นจริงของนิวรอลในสมองมนุษย์ โดยอาจมองว่ามัลติเลเยอร์เน็ตเวิร์ค (Multi-layer Neural Networks) เกิดจากการรวมกันของเน็ตเวิร์คแบบเลเยอร์เดียวก็ได้ ซึ่งเอาท์พุทของเลเยอร์หนึ่งจะถูกส่งไปเป็นอินพุทในเลเยอร์ย่อยในชั้นถัดไป



a.) นิวรอลเน็ตเวิร์คแบบชั้นเดียว b. นิวรอลเน็ตเวิร์คแบบหลายชั้น

รูปที่ 5 แสดงเน็ตเวิร์คในแต่ละเลเยอร์

เน็ตเวิร์คแบบเลเยอร์เดียวจะมีโครงสร้างไม่ซับซ้อน สามารถเรียนรู้ที่จะจดจำและแยกแยะแพทเทิร์นอย่างง่ายได้ ทฤษฎีที่จะนำมาใช้กับเน็ตเวิร์คแบบเลเยอร์เดียวคือ ทฤษฎีของเปอเซพตรอน (Rosenblatt 1958) [3] ดังในรูปที่ 6 โดยใช้แอคติเวชั่นฟังก์ชั่นแบบ Hard Limiter (Binary function) ดังในรูป 3a ซึ่งใช้การเรียนรู้แบบมีครูสอน (Supervised)



รูปที่ 6 แสดงโครงสร้างเน็ตเวิร์คแบบเปอเซพตรอน

จากรูป ผลรวมของค่าอินพุทแพทเทิร์น ($x_1,..,x_n$) ในเลเยอร์ i คูณกับตัวเลขน้ำหนัก (w_{ji}) จะถูกส่งผ่านไปยังแอคติเวชั่นฟังก์ชั่นแบบ Hard Limiter ได้ค่าเอาท์พุท (o_j) ในเลเยอร์ j ซึ่งค่าเอาท์พุทจะมีค่าเป็น 1 หรือ -1 ค่าใดค่าหนึ่งเท่านั้น จากนั้นค่าเอาท์พุทจะถูกนำไปคำนวณเพื่อปรับค่าตัวเลขน้ำหนักอีกครั้งหนึ่ง ดังสมการ (3)

$$E = \frac{1}{2} \sum_{j} (d_j - o_j)^2 \qquad \ldots\ldots (5)$$

ซึ่งเท่ากับ

$$= \frac{1}{2} \sum_{j} [d_j - f(\sum w_{ji} x_i)]^2 \qquad \ldots\ldots (6)$$

เพื่อให้ได้ค่าผิดพลาดต่ำสุดควรปรับเปลี่ยนตัวเลขน้ำหนักในทิศทางลบของ gradient ได้ว่า

$$\Delta w_{ji} = - c (\nabla E) \qquad \ldots\ldots (7)$$

$$= c (d_j - o_j) f'(net_j) x_i \qquad \ldots\ldots (8)$$

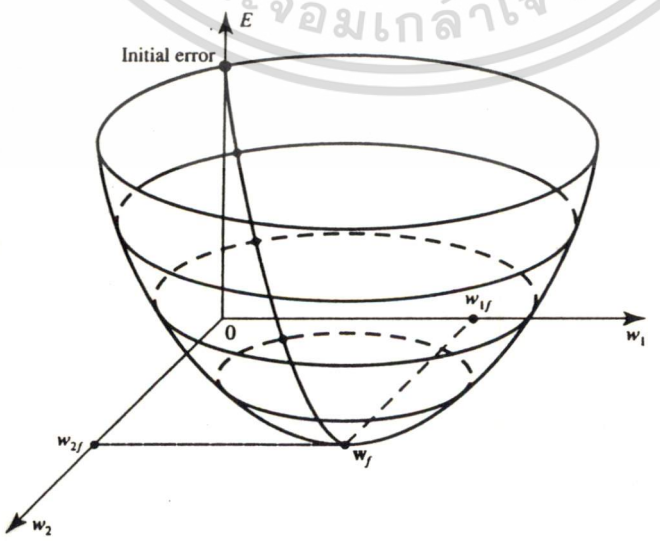
Δw_{ji} ค่าตัวเลขน้ำหนักที่เปลี่ยนไป

c ค่าคงที่การเรียนรู้

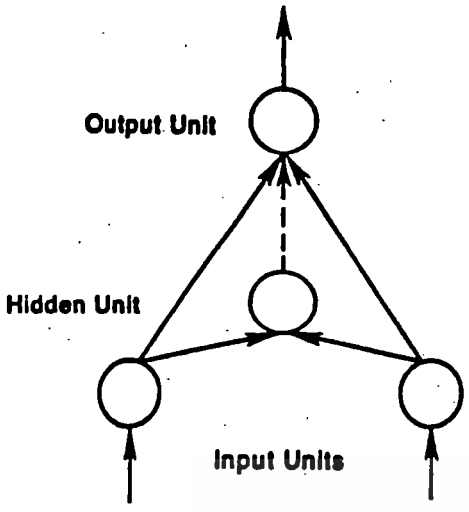
f เป็นแอคติเวชั่นฟังก์ชั่นแบบ Sigmoid

f' ดิฟเฟอเรนเชียลของแอคติเวชั่นฟังก์ชั่นแบบ Sigmoid

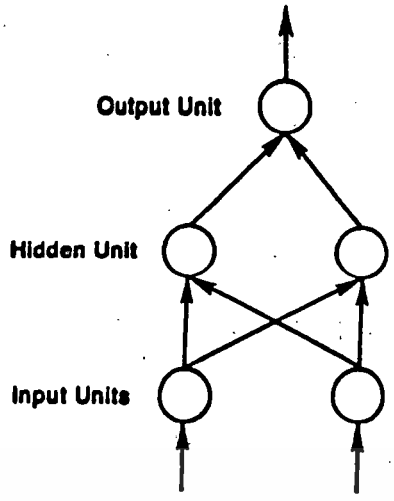
แสดงกราฟที่เกิดขึ้นตามทฤษฎี negative-gradient descent ดังในรูปที่ 8 โดยกำหนดค่าตัวเลขน้ำหนักเริ่มต้น w_1 และ w_2 เป็น 0 พิจารณากราฟที่เกิดขึ้นระหว่างค่าผิดพลาด (E) และค่าตัวเลขน้ำหนัก w_1 และ w_2 จะเห็นว่าเมื่อเวลาผ่านไป ส่วนค่าตัวเลขน้ำหนักนั้นจะถูกปรับเปลี่ยนไปจนกระทั่งได้ค่าผิดพลาด $E(w_f)=0$



ก) กราฟแสดงค่าผิดพลาดต่ำสุดขณะสอนเน็ตเวิร์ค



a) แบบที่ 1



b) แบบที่ 2

รูปที่ 9 แสดงโครงสร้างเน็ตเวิร์คสำหรับเอ็กซ์คลูซีฟออร์

รูป 9a) ในฮิดเดนเลเยอร์มีนิวรอลเพียงโหนดเดียว ลักษณะโครงสร้างเน็ตเวิร์ค
วรอลในอินพุทเลเยอร์จะเชื่อมต่อกับนิวรอลในฮิดเดนเลเยอร์และเอาท์พุทเลเยอร์โดยตรง

รูป 9b) แสดงลักษณะของเน็ตเวิร์คแบบมัลติเลเยอร์เน็ตเวิร์ค หากเปรียบเทียบกับ
ป 9a) จะเห็นว่าโหนดในอินพุทเลเยอร์จะไม่มีการเชื่อมต่อกับนิวรอลในฮิดเดนเลเยอร์โดยตรง
ละโครงสร้างในลักษณะนี้จะต้องกำหนดจำนวนนิวรอลในฮิดเดนเลเยอร์ให้มากพอ จึงจะสามารถ
ก้ปัญหาของเอ็กซ์คลูซีฟออร์ได้

จะเห็นว่า โครงสร้างเน็ตเวิร์คดังในรูปที่ 9 เป็นเน็ตเวิร์คแบบหลายเลเยอร์ซึ่งไม่
ามารถใช้ทฤษฎีของเปอเซพตรอนหรือ Delta rule อธิบายได้ ดังนั้นจะขอกล่าวถึงทฤษฎีต่อไป

ลติเลเยอร์แบ็คพร็อบพาเกชั่น (Multi-layer Back Propagation)

ทฤษฎีแบ็คพร็อบพาเกชั่น (Back Propagation) ใช้กับเน็ตเวิร์คแบบหลายเลเยอร์
วยรูปแบบการเรียนรู้แบบมีครูสอน (Supervised Learning) และมีพื้นฐานทางคณิตศาสตร์
นับสนุนเป็นอย่างดี หลักการสำคัญของแบ็คพร็อบพาเกชั่นคือการปรับและเปลี่ยนตัวเลขน้ำหนักจน
ว่าค่าผิดพลาดเฉลี่ยของระบบ (E) จะมีค่าผิดพลาดเฉลี่ยต่ำสุดที่สามารถยอมรับได้ (E_p)

$$E_p = \frac{1}{2} \sum_k (d_{pk} - o_{pk})^2 \qquad \ldots\ldots (9)$$

$$E = \frac{1}{p} \sum_{p} E_p \qquad \ldots\ (10)$$

เพื่อหาค่าต่ำสุดของผลต่างกำลังสองระหว่าง ค่าเอาท์พุทที่ได้จากการคำนวณ (o_{pk}) และค่าเอาท์พุทจริงที่ต้องการ (d_{pk}) โดยค่าผิดพลาดทั้งหมดของแพทเทิร์น p (E_p) เกิดจากผลต่างของเอาท์พุทจริงที่ต้องการ และค่าเอาท์พุทที่ได้จากการคำนวณในเลเยอร์ k จากสูตรจะเห็นว่า หากต้องการให้ค่าผิดพลาดของระบบลดลง จะต้องให้ผลต่างกำลังสองรวมเข้าใกล้ศูนย์ เนื่องจากค่า t_{pk} ไม่มีการเปลี่ยนแปลง เพราะฉะนั้นสิ่งที่จะทำได้คือ พยายามทำให้ค่า o_{pk} มีค่าเข้าใกล้ t_{pk} มากที่สุด เนื่องจาก o_{pk} มีความสัมพันธ์กับค่าตัวเลขน้ำหนักในนิวรอลเน็ตเวิร์ค ดังนั้นการปรับค่าตัวเลขน้ำหนักจนได้ค่าเหมาะสม จะมีผลทำให้ค่าผิดพลาดของระบบลดลง โดยหาค่าดิฟเฟอเรนเชียลของค่าผิดพลาดที่คำนวณได้ กับค่าตัวเลขน้ำหนักที่เปลี่ยนไปในลักษณะเชิงเส้น

$$\Delta_p w_{kj} \propto -\frac{\partial E_p}{\partial w_{kj}} \qquad \ldots\ (11)$$

เมื่อ

$$w_{kj}(t+1) = w_{kj}(t) + \Delta_p w_{kj} \qquad \ldots\ (12)$$

โดย $w_{kj}(t+1)$ ค่าตัวเลขน้ำหนักระหว่างเลเยอร์ k ไปเลเยอร์ j หลังจากปรับค่าแล้ว

$w_{kj}(t)$ ค่าตัวเลขน้ำหนักระหว่างเลเยอร์ k ไปเลเยอร์ j ก่อนการปรับค่า

$\Delta_p w_{kj}$ ค่าตัวเลขน้ำหนักที่เปลี่ยนไปได้จากการคำนวณ

t เป็นเวลาขณะใดๆ , $t+1$ เป็นเวลาถัดไป

จากสมการ เมื่อความชันของค่าความผิดพลาด ($\partial E_p / \partial w_{kj}$) ลดลง จะมีผลทำให้ค่าตัวเลขน้ำหนักที่เปลี่ยนไป ($\Delta_p w_{kj}$) ลดลงด้วย วิธีการนี้จะกระทำซ้ำไปซ้ำมา จึงจะได้ค่าที่ต้องการซึ่งเราเรียกว่า การสอน (Training) เน็ตเวิร์คจะทำงาน จนกระทั่ง ค่าความชันของค่าผิดพลาดเข้าใกล้ศูนย์ด้วย นั่นคือค่าเอาท์พุทที่ได้จากการคำนวณ จะมีค่าใกล้เคียงกับค่าเอาท์พุทที่เราต้องการมากที่สุด จากสมการ (11) สามารถทำเป็นสมการเชิงเส้นได้ดังนี้

$$\Delta_p w_{kj} = -\eta \frac{\partial E_p}{\partial w_{kj}} \qquad \ldots\ (13)$$

	INPUT			OUTPUT
	O_0	O_1	O_2	O_5
PATTERN NO. 1	1	0	0	0
PATTERN NO. 2	1	0	1	1
PATTERN NO. 3	1	1	0	1
PATTERN NO. 4	1	1	1	0

ตารางที่ 3

การเรียนรู้จะเริ่มหลังจากการกำหนดค่าเริ่มต้นแบบสุ่มให้กับ W ซึ่งเป็นค่าตัวเลขน้ำหนักแบบเวกเตอร์จากอินพุทเลเยอร์ไปยังฮิดเดนเลเยอร์ และ V ซึ่งเป็นค่าตัวเลขน้ำหนักแบบเวกเตอร์จากฮิดเดนเลเยอร์ไปยังเอาท์พุทเลเยอร์ เพื่อใช้เป็นค่าเริ่มต้นในการคำนวณต่อไป

$$W = \begin{bmatrix} w_{50} & w_{53} & w_{54} \end{bmatrix} = \begin{bmatrix} 0.082 & -0.106 & 0.086 \end{bmatrix}$$

$$V = \begin{bmatrix} w_{30} & w_{31} & w_{32} \\ w_{40} & w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.096 & -0.032 & 0.080 \\ -0.137 & 0.051 & -0.068 \end{bmatrix}$$

อินพุท o_1, o_2 เริ่มต้นกำหนดให้เป็นแพทเทิร์นแรก $o_1=0$, $o_2=0$ และ $o_0=1$ จะถูกคำนวณไปพร้อมกันแบบ Feed-forword เพื่อให้ได้ค่า o_3, o_4 และ o_5 โดยกำหนด แอคติเวชั่นฟังก์ชั่นดังสมการ (21) และ (23)

$$o_5 = f(net_5) = \frac{1}{1 + e^{-(w_{50}o_0 + w_{53}o_3 + w_{54}o_4)}} \quad \ldots (25)$$

จากนั้นจะคำนวณหาสัญญาณผิดพลาด δ_5 จากสมการ (16) เมื่อ $f'(net_5) = o_5 (1 - o_5)$

$$\delta_5 = [o_5 (1 - o_5)] (d_5 - o_5) \quad \ldots . (26)$$

และจะคำนวณหาสัญญาณผิดพลาด δ_3 และ δ_4 จากสมการ (19) สำหรับฮิดเดนเลเยอร์

$$\delta_3 = f'(net_3) \sum_{k=5} \delta_k w_{k3} \quad \text{.... (27)}$$

$$\delta_4 = f'(net_4) \sum_{k=5} \delta_k w_{k4} \quad \text{.... (28)}$$

ใช้สมการ (24) เพื่อแทนค่าหา $f'(net_3)$ และ $f'(net_4)$

$$\delta_3 = [o_3 (1 - o_3)] \delta_5 w_{53} \quad \text{.... (29)}$$

$$\delta_4 = [o_4 (1 - o_4)] \delta_4 w_{54} \quad \text{.... (30)}$$

ตอนนี้ค่าตัวเลขน้ำหนักที่ถูกปรับ $\Delta_p w_{kj}$ และ $\Delta_p w_{ji}$ จะสามารถคำนวณได้เมื่อกำหนดค่าคงที่อัตราเรียนรู้ (η) โดยจะเริ่มคำนวณค่าตัวเลขน้ำหนักในเลเยอร์เอาท์พุทก่อน จากสมการ (17)

$$\Delta w_{50} = \eta \delta_5 o_0 = \eta \delta_5 \quad \text{.... (31)}$$

$$\Delta w_{53} = \eta \delta_5 o_3 \quad \text{.... (32)}$$

$$\Delta w_{54} = \eta \delta_5 o_4 \quad \text{.... (33)}$$

การปรับค่าตัวเลขน้ำหนักในฮิดเดนเลเยอร์จะสัมพันธ์กับนิวรอลในโหนด #3 และ #4 โดยคำนวณจากสมการ (20)

$$\Delta w_{30} = \eta \delta_3 o_0 = \eta \delta_3 \quad \text{.... (34)}$$

$$\Delta w_{31} = \eta \delta_3 o_1 \quad \text{.... (35)}$$

$$\Delta w_{32} = \eta \delta_3 o_2 \quad \text{.... (36)}$$

และ

$$\Delta w_{40} = \eta \delta_4 o_0 \quad \text{.... (37)}$$

$$\Delta w_{41} = \eta \delta_4 o_1 \quad \text{.... (38)}$$

$$\Delta w_{42} = \eta \delta_4 o_2 \quad \text{.... (39)}$$

ที่ผ่านมาเป็นการคำนวณเพื่อปรับเปลี่ยนค่าตัวเลขน้ำหนัก $\Delta_p w_{kj}$ และ $\Delta_p w_{ji}$ เฉพาะแพทเทิร์นแรกเท่านั้น จะต้องมีการคำนวณซ้ำไปซ้ำมา โดยกำหนดอินพุท o_i และ o_2 เป็นแพทเทิร์นสองสามและสี่ ตามลำดับ ทั้งหมดถือเป็นหนึ่งรอบ การคำนวณจะสิ้นสุดต่อเมื่อค่าผิดพลาดเฉลี่ยของระบบ (E) จะมีค่าน้อยกว่าหรือเท่ากับค่าผิดพลาดเฉลี่ยต่ำสุดที่สามารถยอมรับได้ (E_p) หากกำหนดให้ E_p เท่ากับ 0.001 จะได้จำนวนรอบการเรียนรู้ที่เกิดขึ้นทั้งหมด 3,567 รอบและได้ผลลัพธ์สุดท้ายของค่าตัวเลขน้ำหนัก W และ V เพื่อนำไปใช้งานต่อไป

$$W = \begin{bmatrix} w_{50} & w_{53} & w_{54} \end{bmatrix} = \begin{bmatrix} -3.714 & -8.288 & 8.048 \end{bmatrix}$$

$$V = \begin{bmatrix} w_{30} & w_{31} & w_{32} \\ w_{40} & w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 2.314 & -6.007 & -6.058 \\ 5.993 & -4.061 & -4.066 \end{bmatrix}$$

บทวิเคราะห์

1. dummy นิวรอลที่เพิ่มขึ้นมาแท้จริงแล้วคือ ไบอัสของระบบ θ_k และ θ_j เมื่อกำหนดอินพุท θ_o เป็นค่าคงที่เท่ากับ 1 ดูได้จากสมการ (21) และ (23) ตามลำดับ จากการทดสอบหากไม่มี dummy นิวรอลโหนดจะไม่สามารถเรียนรู้ได้สำเร็จ ยกเว้น จะเพิ่มจำนวนนิวรอลโหนดในฮิดเดนเลเยอร์จาก 2 เป็น 4,5,...

2. หากเพิ่มจำนวนนิวรอลโหนดในฮิดเดนเลเยอร์มากขึ้น จะทำให้การเรียนรู้ดีขึ้น

3. การทดสอบปัญหาของเอ็กซ์คลูซีฟออร์ โดยใช้โครงสร้างเน็ตเวิร์คดังในรูป 9a) และ 9b) จะให้ผลลัพธ์เหมือนกัน แต่ในรายงานฉบับนี้จะขอเสนอเฉพาะการทดสอบกับโครงสร้างเน็ตเวิร์คดังในรูป 9b) เท่านั้น

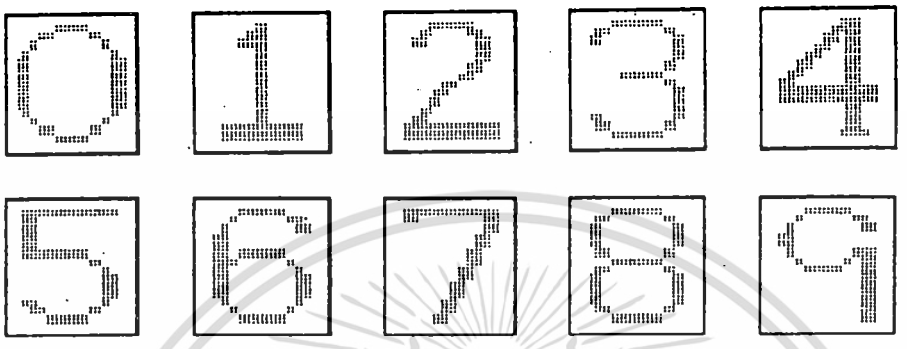
ทดสอบการทำงานตอนที่ 2

จากความรู้เรื่องเอ็กซ์คลูซีฟออร์ ทำให้สามารถเข้าใจโครงสร้างพื้นฐานอย่างง่ายของมัลติเลเยอร์เปอเซพตรอนเน็ตเวิร์คได้เป็นอย่างดี เนื่องจากโครงสร้างเน็ตเวิร์คมีขนาดเล็กจำนวนอินพุทโหนดและเอาท์พุทโหนดน้อย จึงสามารถศึกษาขั้นตอนการทำงานได้ง่ายขึ้น

ต่อไปจะนำเสนอหลักการและวิธีการทำงานของทฤษฎีแบ็คพร็อบพาเกชั่น เพื่อนำไปประยุกต์ใช้จดจำและแยกแยะแพทเทิร์นแพทเทิร์นตัวอักษรตัวเลขแบบตัวพิมพ์ เพื่อใช้เป็นแนวทางในการศึกษาและพัฒนากับงานทางด้าน OCR ในอนาคต

ก่อนที่จะแสดงลำดับและขั้นตอนการทำงานของโปรแกรม ควรที่จะกำหนดโครงสร้างเน็ตเวิร์คและกำหนดรูปแบบของแพทเทิร์นที่จะใช้ทดสอบก่อน

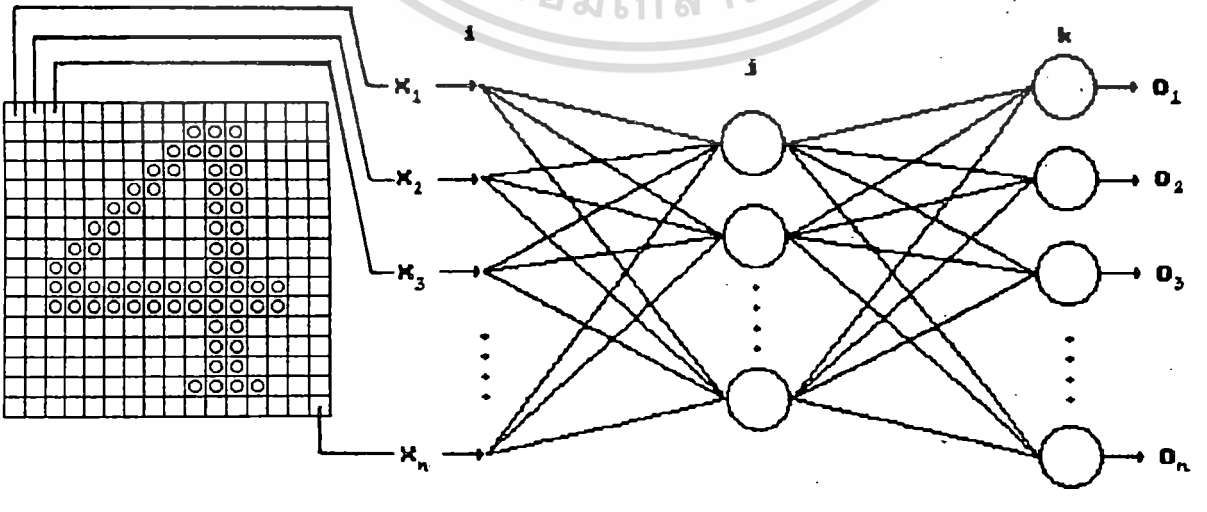
1. ข้อกำหนดเบื้องต้นเพื่อให้คอมพิวเตอร์สามารถเรียนรู้และจดจำแพทเทิร์นตัวอักษรตัวเลขแบบตัวพิมพ์ที่นำมาทดสอบ ได้มีการพัฒนาเขียนซอฟแวร์ขึ้น เพื่อทดสอบการจดจำตัวอักษรตัวเลขแบบตัวพิมพ์ตั้งแต่ "0" ถึง "9" (ทั้งหมด 10 แพทเทิร์นตัวเลข, p=1 ถึง 10) โดยแพทเทิร์นตัวเลขแต่ละตัวมีขนาด 16x16 บิทแพทเทิร์น (256 บิทแพทเทิร์น)



รูปที่ 11 แสดงแพทเทิร์นตัวอักษรตัวเลข "0".."9"

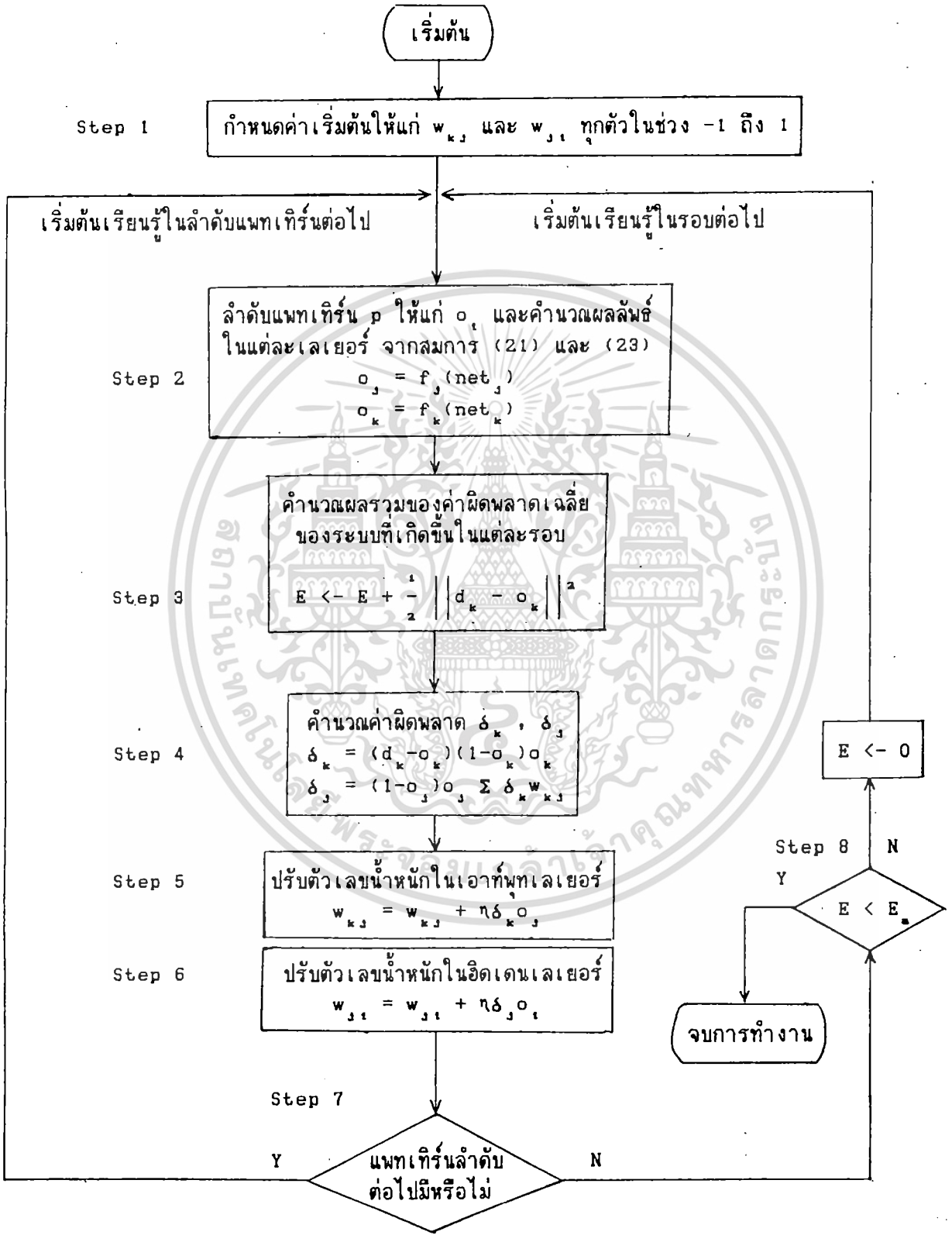
แพทเทิร์นที่ปรากฏดังในรูปที่ 11 สามารถมองเห็นได้จากจอมอนิเตอร์ แสดงผลในลักษณะเป็นบิท (ค่า 0 ลักษณะปิดหรือ 1 ลักษณะเปิด) ซึ่งจุดแต่ละจุดที่มองเห็นจะถูกนำมาเป็นอินพุท (o_{pi}) ของนิวรอลเน็ตเวิร์ค โดย i=1..256 และ ค่าเอาท์พุทที่ต้องการ (t_{pk}) ของนิวรอลเน็ตเวิร์ค ถูกกำหนดให้เป็น 1 เมื่อ p=k โดย k=1..10

2. กำหนดจำนวนเลเยอร์ทั้งหมดในการทดสอบเพียง 3 เลเยอร์คือ อินพุทเลเยอร์ ฮิดเดนเลเยอร์ และเอาท์พุทเลเยอร์ โดยฮิดเดนเลเยอร์จะมีจำนวนนิวรอล 120 นิวรอลโหนด j=1..120 ดังในรูป



รูปที่ 12 แสดงโครงสร้างการทำงานของนิวรอลเน็ตเวิร์ค

ขั้นตอนการคำนวณ



รูปที่ 13 ขั้นตอนการทำงานของโปรแกรมในรูปแบบของโฟลชาร์ต

1. เริ่มแรกจะต้องให้ค่าเริ่มต้นแก่ค่าตัวเลขน้ำหนัก (w) ทุกตัว ในลักษณะสุ่มเป็นค่าน้อยๆ โดยจำกัดในช่วง 0.0-1.0 ทุกตัว

2. จัดลำดับของอินพุท o_{pi} (o_{p1},..,o_{p256}) และค่าเอาท์พุท t_{pk} (t_{p1},..., t_{p10}) ในแต่ละแพทเทิร์นให้สัมพันธ์กัน สามารถแสดงผลออกทางจอภาพได้ในลักษณะเป็นบิท โดย

o_{1i} คือแพทเทิร์นตัวเลข 0 , t_{1k} = [1 0 0 0 0 0 0 0 0 0]

o_{2i} คือแพทเทิร์นตัวเลข 1 , t_{2k} = [0 1 0 0 0 0 0 0 0 0]

o_{3i} คือแพทเทิร์นตัวเลข 2 , t_{3k} = [0 0 1 0 0 0 0 0 0 0]

. .

. .

. .

o_{10i} คือแพทเทิร์นตัวเลข 9 , t_{10k} = [0 0 0 0 0 0 0 0 0 1]

สอนเน็ตเวิร์คจนได้ค่าตัวเลขน้ำหนักที่ไม่มีการเปลี่ยนแปลง หรือค่าค่าผิดพลาดเฉลี่ยของระบบ(E) มีค่าน้อยกว่าค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้(E_a) กำหนดให้เท่ากับ 0.001

3. คำนวณหาค่าเอาท์พุท (o_{pj}) และ (o_{pk}) ในลักษณะไปข้างหน้า (Feed-forward) โดยใช้ทฤษฎีของแบ็คพร็อบพาเกชั่น เมื่อ θ_j และ θ_k เป็นไบอัสของระบบ

$$net_{pj} = \sum_i w_{ji} o_{pi} \qquad \text{.... (40)}$$

$$o_{pj} = f_j(net_{pj}) = \frac{1}{1 + e^{-net_{pj} + \theta_j}} \qquad \text{.... (41)}$$

$$net_{pk} = \sum_j w_{kj} o_{pj} \qquad \text{.... (42)}$$

$$o_{pk} = f_k(net_{pk}) = \frac{1}{1 + e^{-net_{pk} + \theta_k}} \qquad \text{.... (43)}$$

4. ปรับค่าตัวเลขน้ำหนักในแต่ละคู่เลเยอร์ ในลักษณะป้อนกลับ (Feedback) โดยเริ่มจากเอาท์พุทเลเยอร์ก่อน

$$w_{kj}(t+1) = w_{kj}(t) + \eta\, \delta_{pk}\, o_{pj} \quad \ldots\ (44)$$

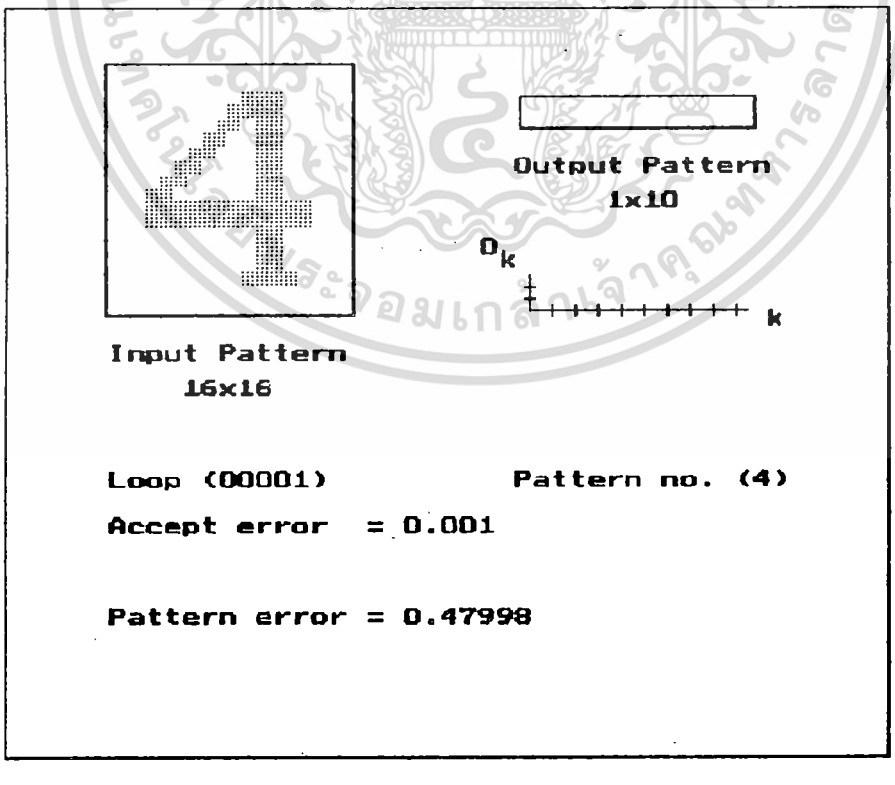
$$= w_{kj}(t) + \eta\, [o_{pk}(1-o_{pk})(t_{pk}-o_{pk})]\, o_{pj} \quad \ldots\ (45)$$

$w_{kj}(t)$ เป็นค่าตัวเลขน้ำหนักจากฮิดเดนเลเยอร์ (j) ไปยังเอาท์พุทเลเยอร์ (k) ที่เวลา t ใดๆและ t+1 เป็นเวลาถัดไป กำหนดค่าอัตราการเรียนรู้ (η) ในการทดสอบมีค่าเท่ากับ 0.5 จากนั้นปรับค่าตัวเลขน้ำหนักคู่เลเยอร์ถัดไปในลักษณะเดียวกัน

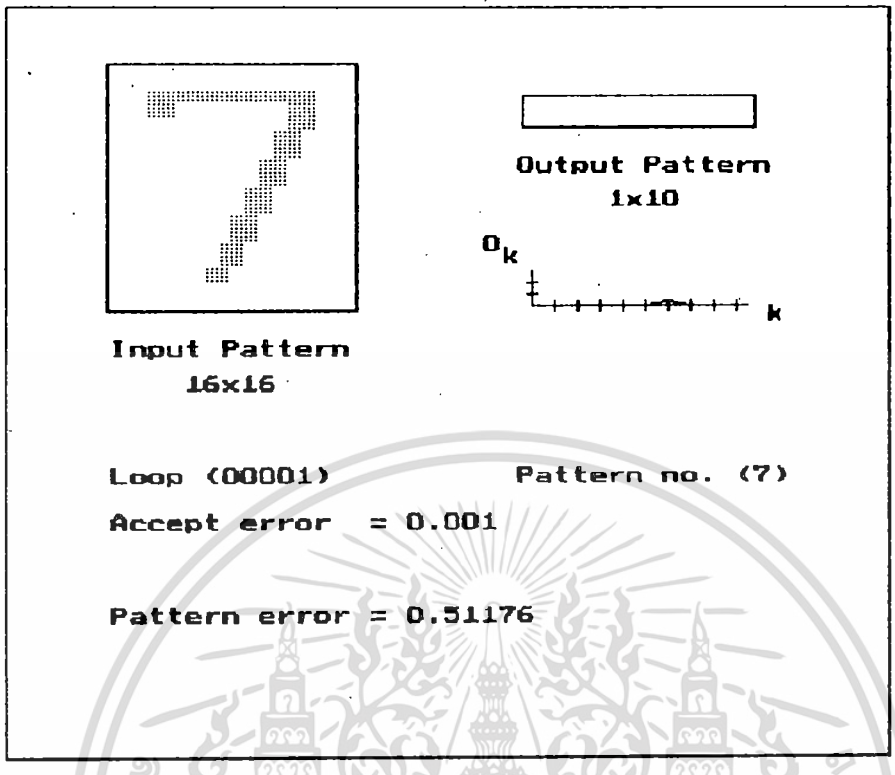
$$w_{ji}(t+1) = w_{ji}(t) + \eta\, \delta_{pj}\, o_{pi} \quad \ldots\ (46)$$

$$= w_{ji}(t) + \eta\, [o_{pj}(1-o_{pj}) \sum_{k}(\delta_{pk}\, w_{kj})]\, o_{pi} \quad \ldots\ (47)$$

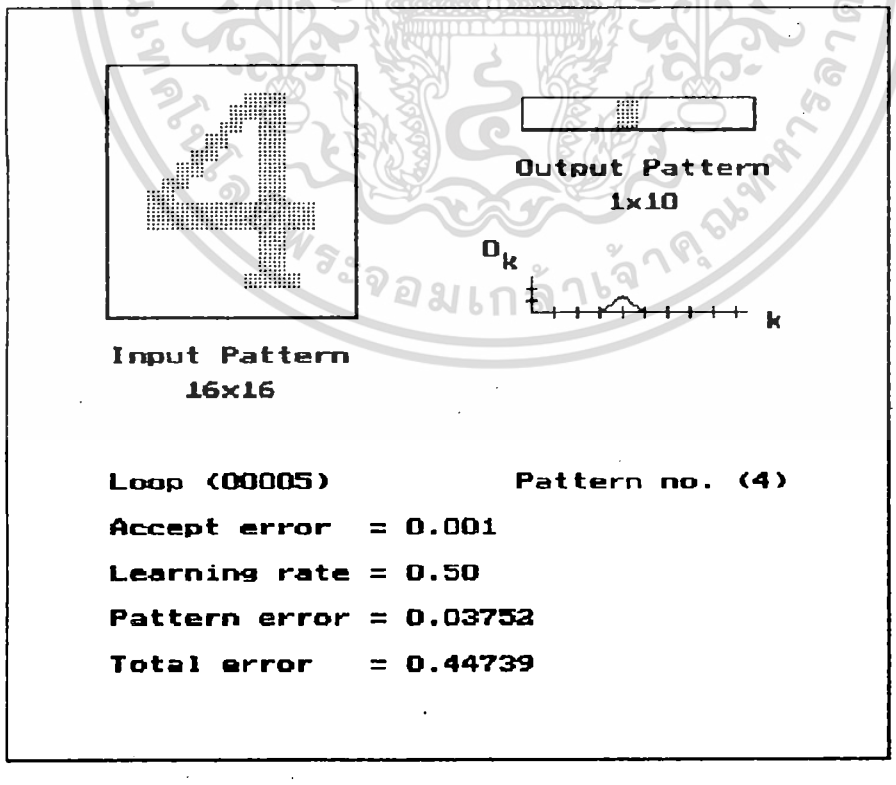
5. เปลี่ยนเป็นแพทเทิร์นลำดับต่อไป แล้วกลับไปคำนวณในข้อ 2



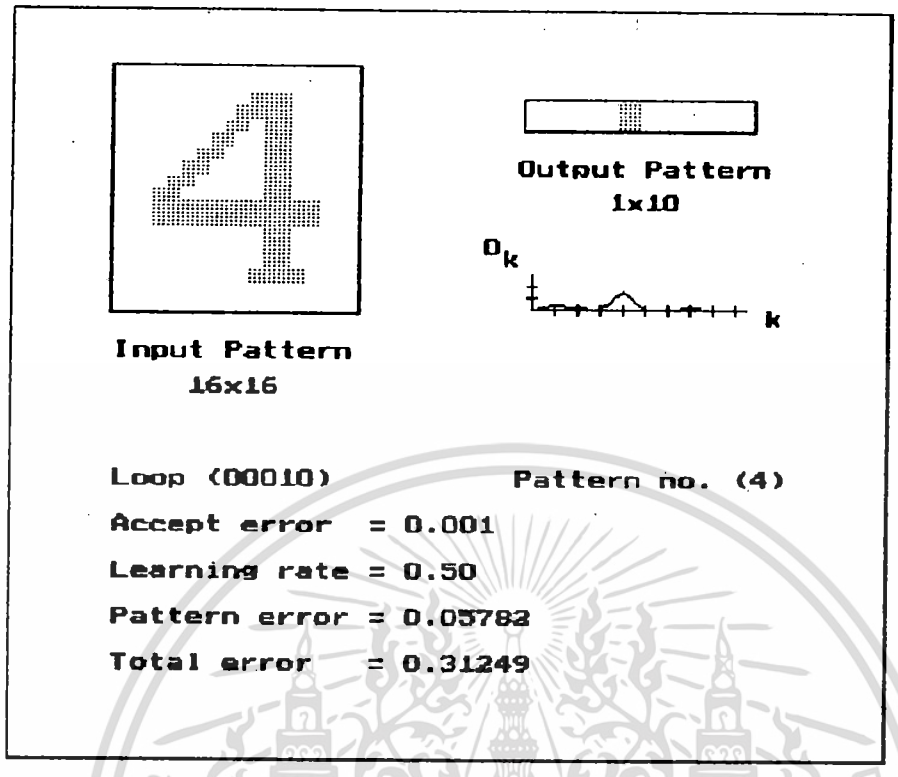
a) แสดงการสอนแพทเทิร์นตัวอักษรตัวเลข "4" ในรอบที่ 1



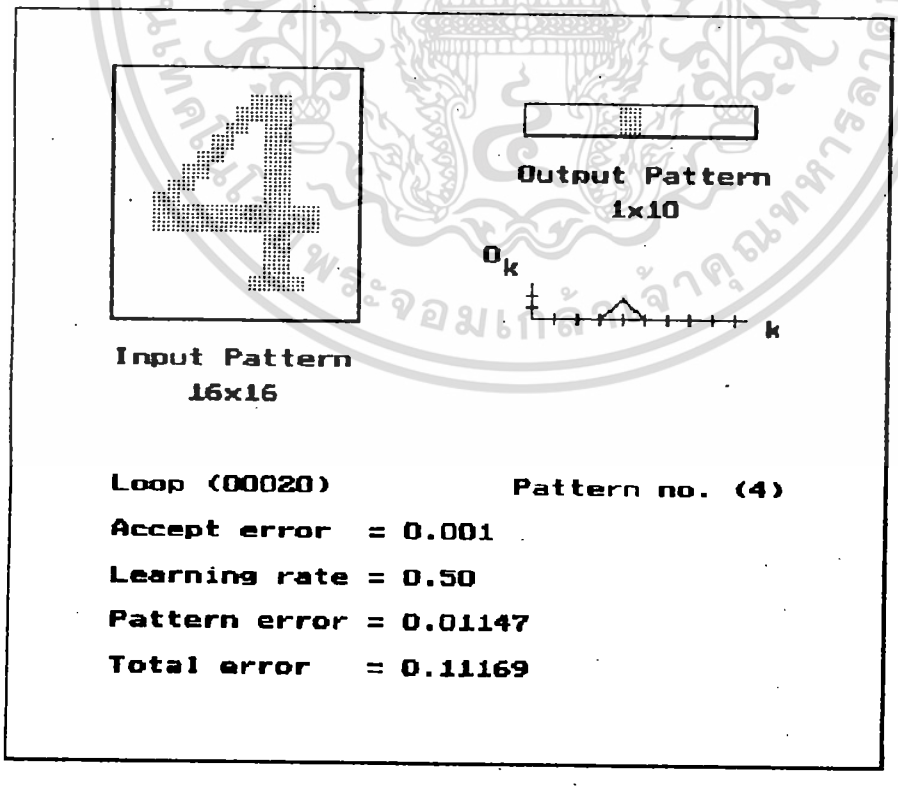
b) แสดงการสอนแพทเทิร์นตัวอักษรตัวเลข "7" ในรอบที่ 1



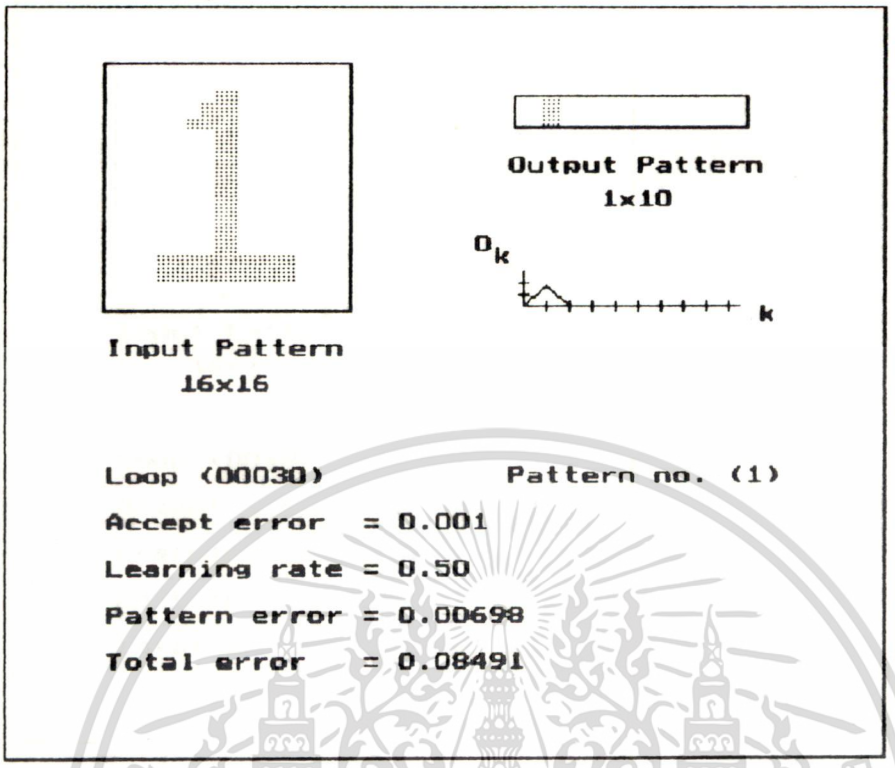
c) แสดงการสอนแพทเทิร์นตัวอักษรตัวเลข "4" ในรอบที่ 5



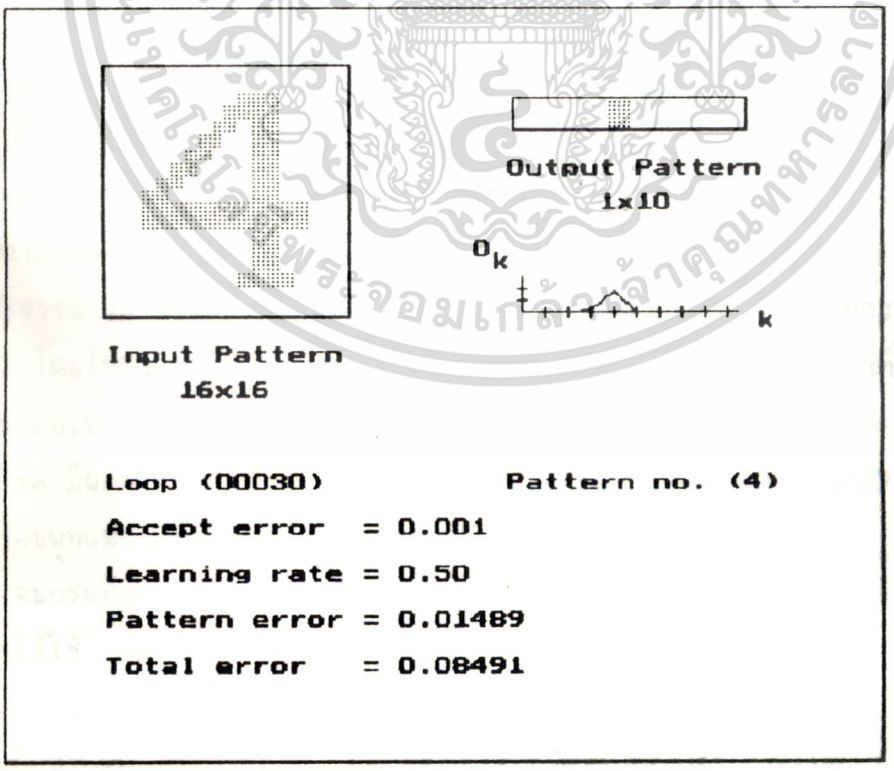
d) แสดงการสอนแพทเทิร์นตัวอักษรตัวเลข "4" ในรอบที่ 10



e) แสดงการสอนแพทเทิร์นตัวอักษรตัวเลข "4" ในรอบที่ 20



f) แสดงการสอนแพทเทิร์นตัวอักษรตัวเลข "1" ในรอบที่ 30



g) แสดงการสอนแพทเทิร์นตัวอักษรตัวเลข "4" ในรอบที่ 30

ผลการทดสอบ

ได้ทดสอบความสามารถในการแยกแยะแพทเทิร์นที่มีความผิดเพี้ยน (Distortion) หลังจากสอนแพทเทิร์นตัวอักษรตัวเลขสิ้นสุดลง โดยลองเพิ่มสัญญาณรบกวน (Noise) เข้าไปในแพทเทิร์นที่สมบูรณ์ดังในรูปที่ 15 กำหนดค่าความน่าจะเป็นของสัญญาณรบกวน P(Probability) เท่ากับ 0.10 (10 เปอร์เซ็นต์) แล้วคำนวณจำนวนบิทของสัญญาณรบกวนสำหรับแพทเทิร์นขนาด 16x16 (256 บิทแพทเทิร์น) ตามสมการ (48)

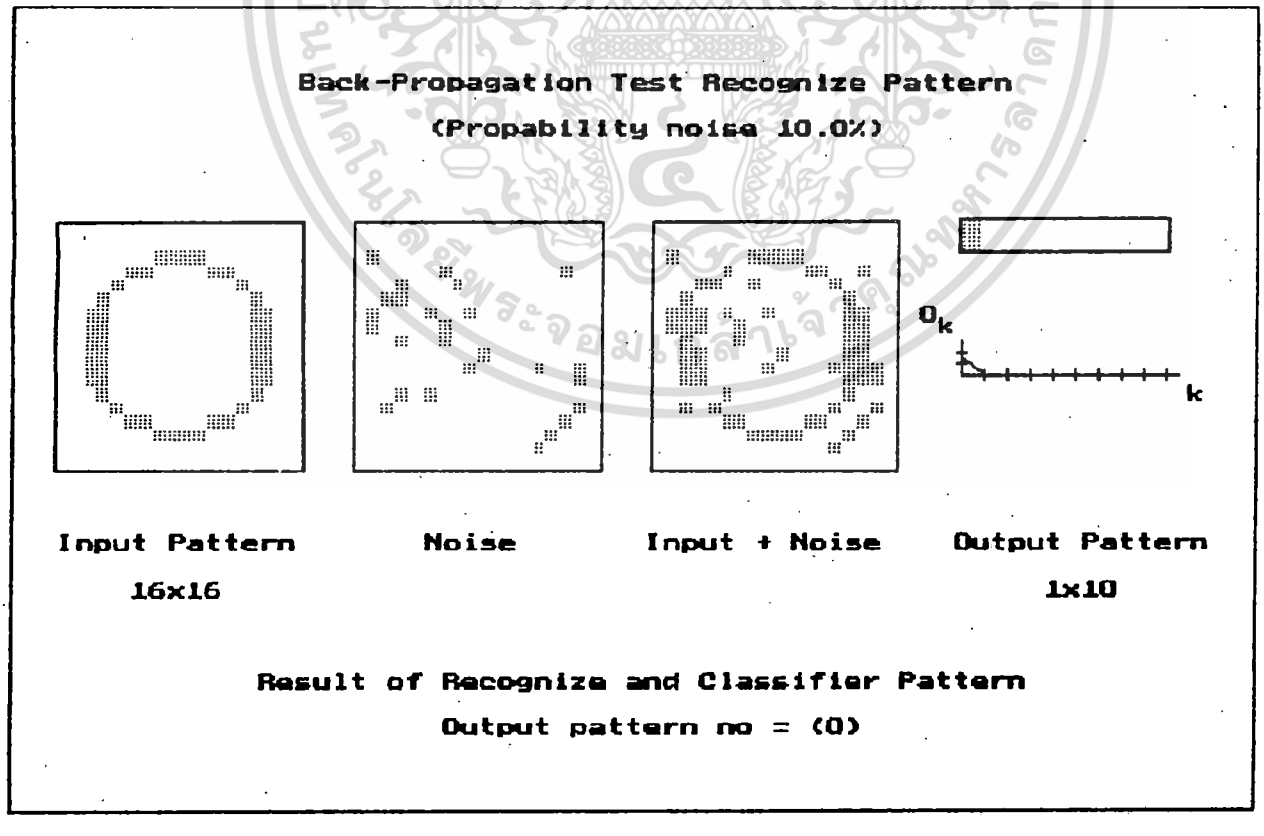
$$P = \frac{\text{Noise}}{\text{ขนาดแพทเทิร์น}} \quad \ldots\ldots (48)$$

แทนค่า $\text{Noise} = 0.1 \times 256$

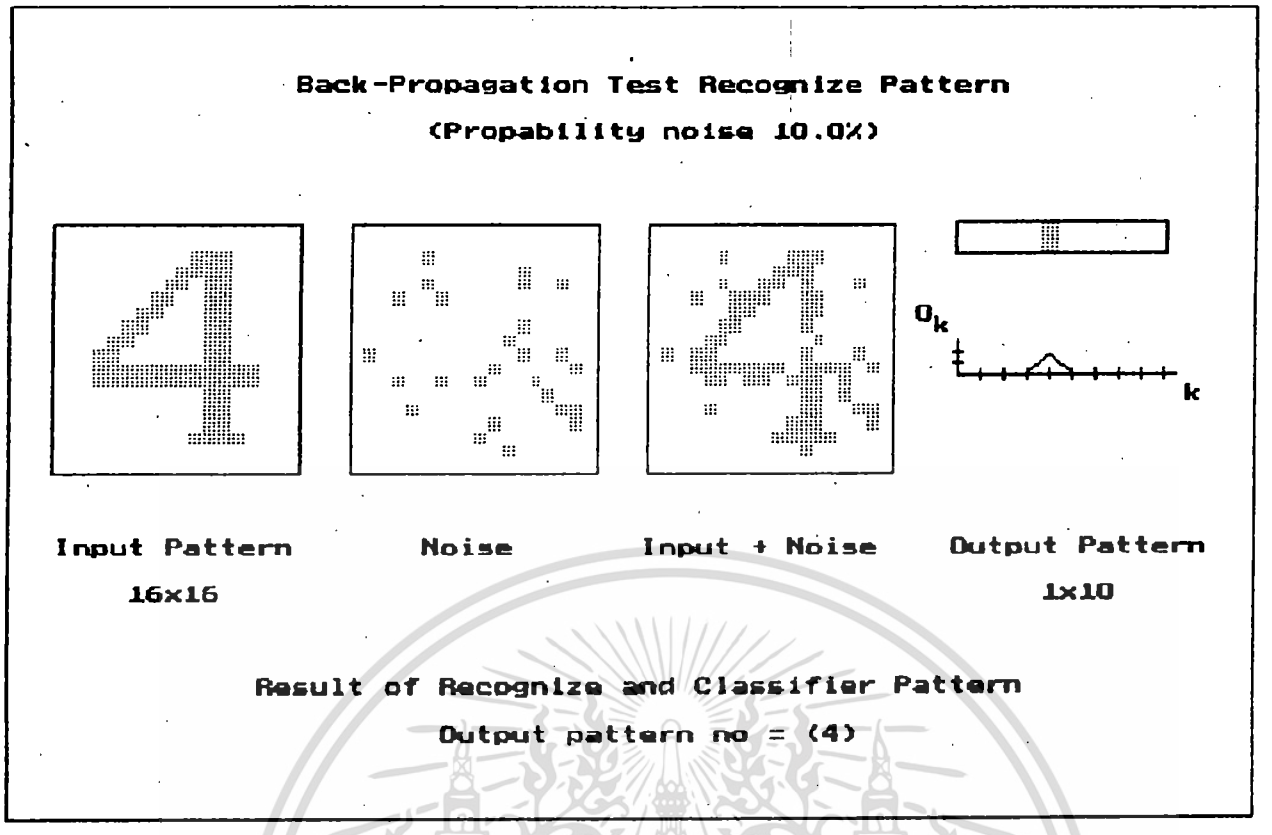
$= 25.6$

$\simeq 26$ บิทแพทเทิร์น

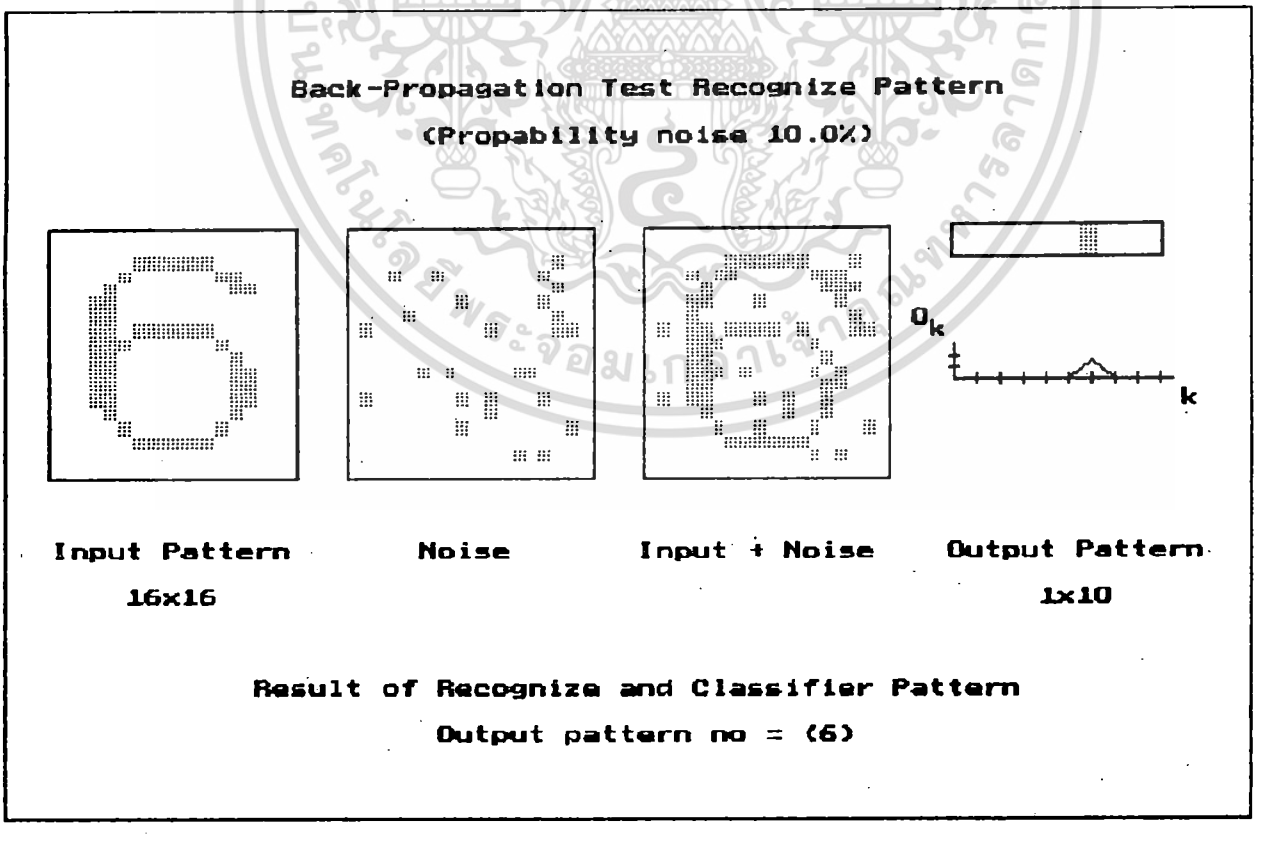
ดังนั้นจะต้องใส่ Noise จำนวน 26 บิทแพทเทิร์นลงใน 256 บิทของอินพุทแพทเทิร์น



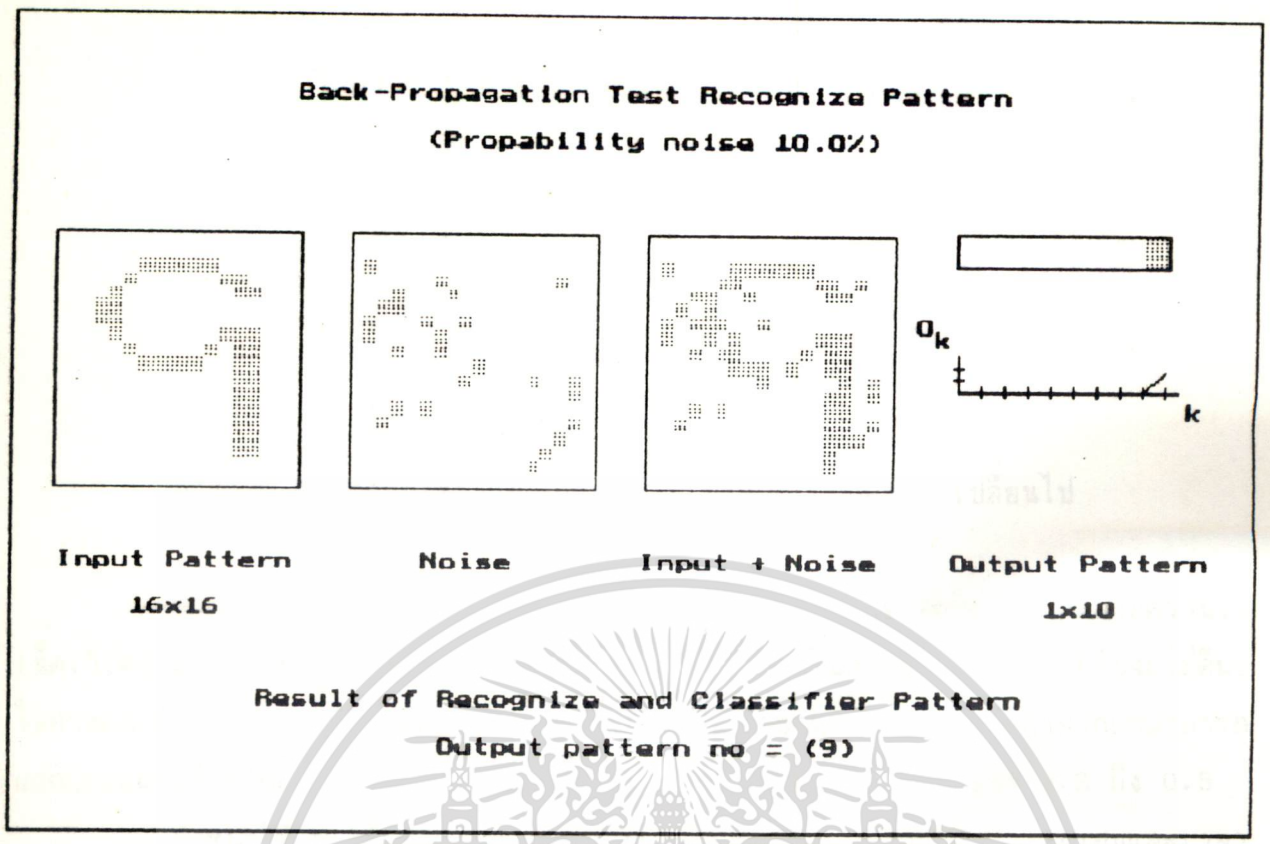
a) แสดงตัวอย่างการทดสอบการแยกแยะแพทเทิร์นตัวอักษรตัวเลข "0"



b) แสดงตัวอย่างการทดสอบการแยกแยะแพทเทิร์นตัวอักษรตัวเลข "4"



c) แสดงตัวอย่างการทดสอบการแยกแยะแพทเทิร์นตัวอักษรตัวเลข "6"



d) แสดงตัวอย่างการทดสอบการแยกแยะแพทเทิร์นตัวอักษรตัวเลข "9"

รูปที่ 15 แสดงตัวอย่างการทดสอบการแยกแยะแพทเทิร์น เมื่อข้อมูลอินพุทถูกบิดเบือน

จากรูปที่ 15 แสดงผลการทดสอบโดยดูจากจอมอนิเตอร์ ตัวอักษรตัวเลขทางซ้ายมือเป็นแพทเทิร์นอินพุทปกติ ซึ่งนำมาเป็นตัวอย่างการทดสอบ ตัวอักษรตัวเลขตรงกลางหมายถึงอินพุทแพทเทิร์นที่มีสัญญาณรบกวน และตัวอักษรตัวเลขขวามือ เป็นตัวอักษรที่ผ่านการแยกแยะแพทเทิร์นด้วยโปรแกรมคอมพิวเตอร์ โดยวิธีการของแบ็คพร็อบพาเกชั่นนิวรอลเน็ตเวิร์ค ผลจากการทดสอบแสดงให้เห็นว่า แม้อินพุทแพทเทิร์นตัวอักษรตัวเลขที่เข้ามาจะมีสัญญาณรบกวนบ้าง หากไม่มากนัก โปรแกรมสามารถแยกแยะแพทเทิร์นและแสดงเอาท์พุทที่ถูกต้องได้ โดยมีความผิดพลาดน้อยมาก

เทคนิคการ Training เพิ่มเติม

จากที่กล่าวมาแล้วข้างต้น เรากำหนดค่าอัตราการเรียนรู้ (Learning rate η) เป็นค่าคงที่ตลอดการ Training ซึ่งให้ผลดีพอสมควร แต่จะขอแนะนำเพิ่มเติมเกี่ยวกับเทคนิคการ Training ดังนี้

1) กำหนดค่าอัตราการเรียนรู้ (η) เป็นค่าคงที่ ซึ่งการเลือกค่า η ที่เหมาะสมที่สุดจำเป็นต้องทดลองแทนค่าด้วยตนเอง จากการทดสอบกำหนดให้ค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ E_p เป็นค่าคงที่เท่ากับ 0.001

2) สูตรแปรผันตามจำนวนรอบ (t) ใช้แนวความคิดมาจากสูตรการกระจายค่าของเอาท์พุท Self organization map [7] ของ Kohonen โดยเปลี่ยนมาใช้กับค่าอัตราการเรียนรู้ (η) กำหนดให้ค่า η เริ่มแรกมีค่าเท่ากับ 1.0 และค่อยๆลดลงตามจำนวนรอบที่เพิ่มขึ้นจนกระทั่งค่าผิดพลาดเฉลี่ยของระบบ E มีค่าน้อยกว่าค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ E_a ซึ่งกำหนดค่าสุดท้ายของ η มีค่าประมาณ 0.2

$$\eta = \eta_o \left(1 - \frac{t}{T}\right) \quad \ldots\ (49)$$

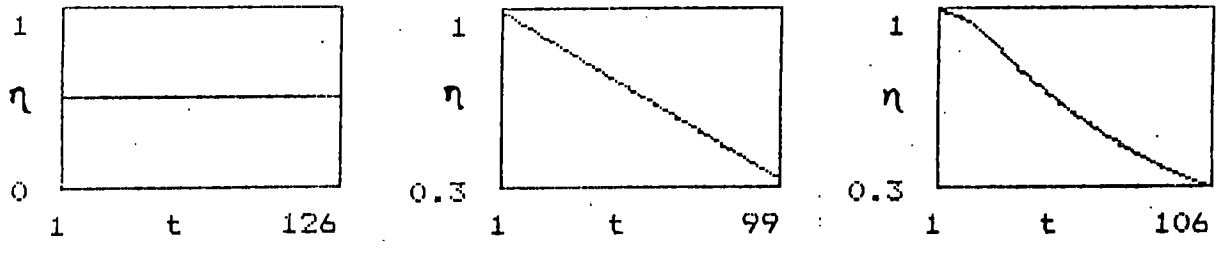
โดย η_o ค่าเริ่มต้นของอัตราการเรียนรู้ กำหนดให้เท่ากับ 1.0

t เป็นหมายเลขของรอบขณะทำการทดสอบ

T เป็นจำนวนรอบทั้งหมดที่ใช้ในการทดสอบโดยประมาณ ต้องกำหนดเอง โดยตั้งเงื่อนไขว่า ค่าเริ่มต้น η มีค่าเท่ากับ 1 และค่าสุดท้ายควรมีค่าเป็น 0.2 ดังนั้น หากดูจำนวนรอบจากตาราง 4 และ 5 จะเห็นว่าค่า T ที่เหมาะสมควรมีค่าประมาณ 180 เมื่อกำหนด E_a เท่ากับ 0.001

3) สูตรแปรผันตามค่าผิดพลาดเฉลี่ยของระบบ (E) ซึ่งเป็นวิธีที่เรานำเสนอ กำหนดให้ค่า η เริ่มแรกมีค่าเท่ากับ 1.0 และค่อยๆ ลดลงตามค่าค่าผิดพลาดเฉลี่ยของระบบ E ที่ลดลงจนกระทั่ง E มีค่าน้อยกว่าค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ E_a กำหนดค่าสุดท้ายของค่า η มีค่าประมาณ 0.2

$$\eta = 1 - \frac{0.8\ E_a}{E} \quad \ldots\ (50)$$



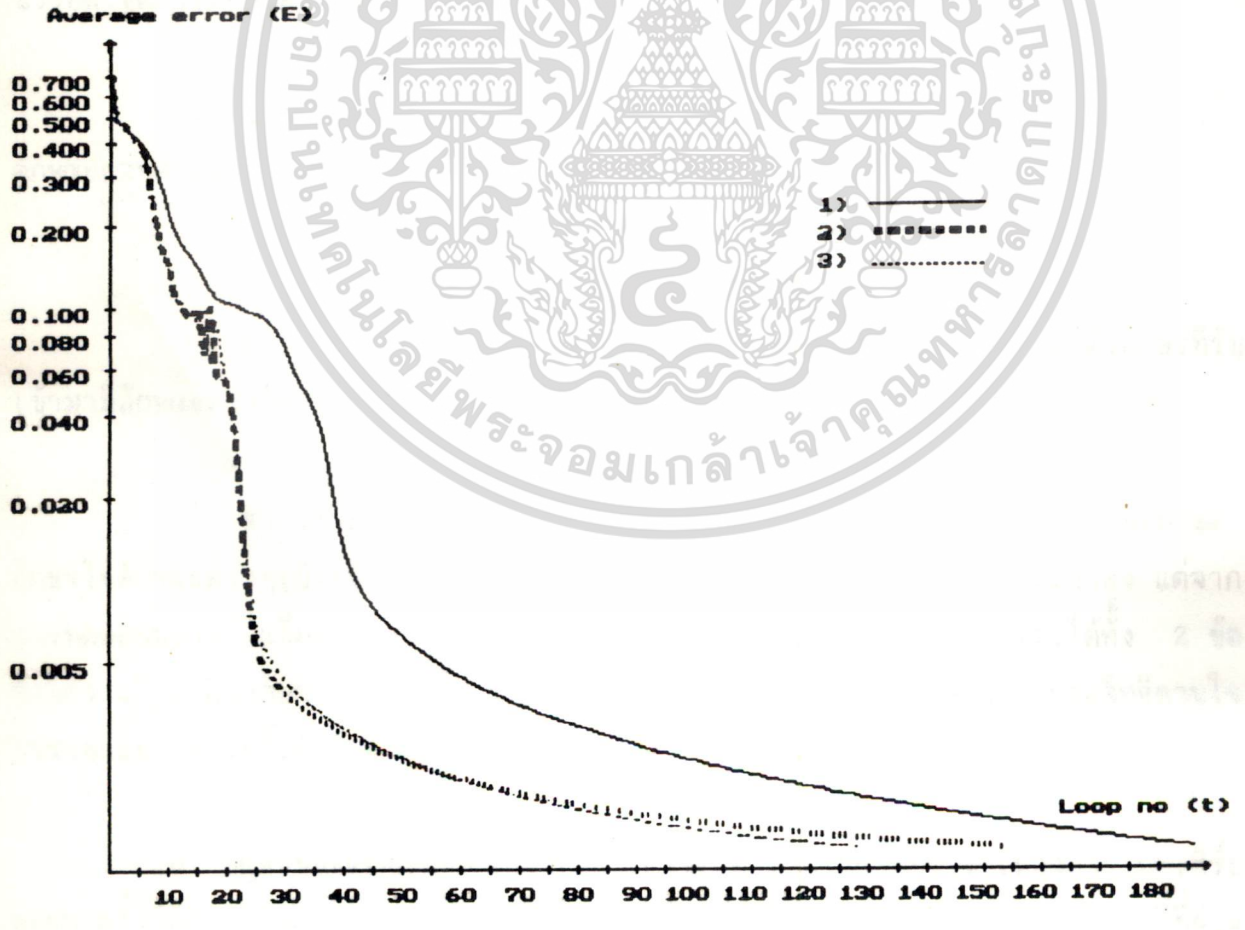
รูปที่ 16 แสดงการกำหนดค่า η แตกต่างกันตามสูตร

รูปที่ 16 เป็นกราฟแสดงความสัมพันธ์ระหว่างค่าอัตราการเรียนรู้ η กับค่าหมายเลขของรอบขณะทำการทดสอบ 16-a) กำหนดให้ η เป็นค่าคงที่ 16-b) กำหนดให้ η แปรผันตามจำนวนรอบ 16-c) กำหนดให้ η แปรผันตามค่าผิดพลาดเฉลี่ยของระบบ

เพื่อเปรียบเทียบการทดสอบอัตราการเรียนรู้ทั้ง 3 แบบ จึงนำข้อมูลของความสัมพันธ์ระหว่างค่าผิดพลาดเฉลี่ยของระบบ (E) กับจำนวนรอบที่เพิ่มขึ้น (t) ขณะ Training เขียนเป็นกราฟขึ้น ดังในรูปที่ 17 โดยแสดงค่า E ในรูปแบบของลอการิทึม

แบบที่ 1) กำหนดค่า η เป็นค่าคงที่เท่ากับ 0.5
แบบที่ 2) กำหนดค่า η แปรผันตามจำนวนรอบ (t)
แบบที่ 3) กำหนดค่า η แปรผันตามค่าผิดพลาดเฉลี่ยของระบบ (E)

จากรูป แบบที่ 2) และ แบบที่ 3) ให้ผลใกล้เคียงกัน โดยเส้นกราฟจะทับกันพอดี ซึ่งค่าผิดพลาดเฉลี่ยของระบบ E จะลดลงเร็วมากในช่วงแรก และจะช้าลงเมื่อจำนวนรอบเพิ่มขึ้น จากการทดสอบความสามารถในการแยกแยะแพทเทิร์นเปรียบเทียบกันทั้ง 3 แบบพบว่า แบบที่ 2) และแบบที่ 3) จะสามารถแยกแยะแพทเทิร์นได้ดีกว่าแบบที่ 1) เล็กน้อย



รูปที่ 17 แสดงผลการ Training โดยกำหนดค่า η แตกต่างกันตามสูตร

สรุป

การนำนิวรอลเน็ตเวิร์คซึ่งเป็นศาสตร์ที่มีความใกล้เคียงกับสมองมนุษย์มาช่วยงาน ทำให้เกิดประโยชน์อย่างมากทั้งในปัจจุบันและในอนาคต โดยคอมพิวเตอร์สามารถเรียนรู้ และจดจำแพทเทิร์นในลักษณะต่างๆได้ ในอนาคต สามารถนำนิวรอลเน็ตเวิร์คไปประยุกต์เพื่อจดจำลายมือเขียน ลายนิ้วมือ ลายเซ็นต์ และเสียงพูดของมนุษย์ได้

รายงานนี้นำเสนอวิธีการของแบ็คพร็อบพาเกชั่น (Back-Propagation) ซึ่งเป็นนิวรอลเน็ตเวิร์คในรูปหนึ่งที่มีความเหมาะสมมากที่สุด การจดจำและแยกแยะแพทเทิร์นตัวอักษรตัวพิมพ์ เพื่อประยุกต์ใช้กับโปรแกรมทางด้าน OCR จากผลการทดสอบ โดยเขียนซอฟแวร์ให้ผลเป็นที่น่าพอใจระดับหนึ่ง แต่ยังต้องมีการพัฒนาต่อไปอีก เพื่อให้ซอฟแวร์สามารถจดจำและแยกแยะแพทเทิร์นตัวอักษรซึ่งมีขนาดต่างๆกัน หรือหมุนเอียงไปในทิศใดทิศหนึ่ง

เอกสารอ้างอิง

1. David E. Rumelhart, James L. McClelland and the PDP Research Group, **Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Volume 1: Foundations**, by The Massachusetts Institute of Technology, 1986.
2. Philip D. Wasserman, **Neural Computing Theory and Practice**, ANZA Research Inc, 1989.
3. Jacek M. Zurada, **Introduction to Artificial Neural Systems**, West Info Access, 1992
4. R. P. Lippmann, "*An Introduction to Computing with Neural Nets*", **IEEE Acoustics, Speech, and Signal Processing Society**, Volume 4, Number 2, pp. 4-22, April 1987.
5. Philip Treleaven, Marco Pacheco and Marley Vellasco, **VLSI Architectures for Neural Networks**, IEEE Micro, pp. 8-27, December 1989.
6. Yoh-Han Pao, **Adaptive Pattern Neural Networks**, Addison-Wesley Inc., pp. 113-140, 1989.
7. T. Kohonen, **Self-Organization and Associative Memory**, Spriner-Verlag, 1989