การใช้นิวรอลเนตเวิร์กเพื่อจดจำแพตเทิร์น

USING NEURAL NETWORKS FOR PATTERN RECOGNITON

เศรษฐพล ลิ้นปราชญา
Settapol Linprachya

กิตติ ไพฑูรย์วัฒนกิจ
Kitti Paithoonwattanakij

ภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
Department of Electronics, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang

บทคัดย่อ

ปัจจุบันมีผู้สนใจนำนิวรอลเนตเวิร์ก (Neural Networks) ซึ่งเป็นศาสตร์ที่จำลองแบบความสามารถของมนุษย์มาช่วยงานประมวลผลข้อมูลด้านการเรียนรู้ ด้านการจดจำและจำแนกแพตเทิร์น (Pattern) เนื่องจากมีความยืดหยุ่นในการทำงานสูง และสามารถปรับตัวเองให้รับรู้สภาพที่เปลี่ยนแปลงได้ บทความนี้ได้เสนอหลักการ และวิธีการของแบ็กพรอบพาเกชัน (Back Propagation) ซึ่งเป็นหนึ่งในอีกหลายวิธีของนิวรอลเนตเวิร์ก เพื่อใช้เรียนรู้ การจดจำและจำแนกแพตเทิร์นตัวอักษรแบบตัวพิมพ์ได้ สามารถนำไปประยุกต์ใช้กับงานด้าน OCR (Optical Character Reader) ได้เป็นอย่างดี ผลจากการทดสอบพบว่า วิธีการของแบ็กพรอบพาเกชัน นิวรอลเนตเวิร์กสามารถจดจำและจำแนกแพตเทิร์นตัวอักษรได้ดี ถึงแม้ว่าข้อมูลจะมีความผิดเพี้ยน (Distortion) เนื่องจากสัญญาณรบกวน (Noise) หรือแพตเทิร์นตัวอักษรมีคุณภาพไม่ดี

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. It can apply for OCR to achieve for more 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

การจดจำแพตเทิร์นต่าง ๆ โดยสมองมนุษย์ ก่อให้เกิดความรู้ ความเข้าใจและนำไปใช้สร้างภูมิปัญญา โดยมนุษย์เองสามารถจดจำสิ่งที่พบเห็นในลักษณะของแพตเทิร์น และบอกได้ว่าเคยพบเห็นมาแล้วหรือไม่ และสิ่งที่เห็นคืออะไร ซึ่งสิ่งที่พบเห็นได้แก่ ภาษา เสียงพูด และรูปวาดของสิ่งต่าง ๆ เพื่อทำให้เกิดการสื่อความหมายของสิ่งที่พบเห็นอยู่ภายในตัวแพตเทิร์นโดยอาศัยประสบการณ์จากการเรียนรู้และการฝึกฝนที่สะสมมาเป็นเวลานาน

ปัจจุบันเมื่อวิทยาการเจริญมากขึ้น มนุษย์เริ่มนำหลักการพื้นฐานที่มีลักษณะใกล้เคียงกับการทำงานของสมองมนุษย์ ในรูปแบบของอาร์ติฟิเชียลนิวรอลเนตเวิร์ก (Artificial Neural Networks) มาช่วยในการตัดสินใจกับงานทางด้านอุตสาหกรรม คมนาคม การสื่อสาร และอื่น ๆ มากขึ้น

นิวรอล (Neural) ในสมองมนุษย์

ระบบเส้นประสาทของมนุษย์ประกอบด้วยเซลล์เล็ก ๆ มากมายที่เรียกว่า นิวรอล หรือ เซลล์ประสาทที่รวมตัวกันอยู่ในรูปโครงสร้างที่ซับซ้อน มีอยู่ประมาณ 10^{11} เซลล์ในสมองมนุษย์[5] โดยนิวรอลแต่ละตัวประกอบด้วยส่วนสำคัญ 3 ส่วน คือ

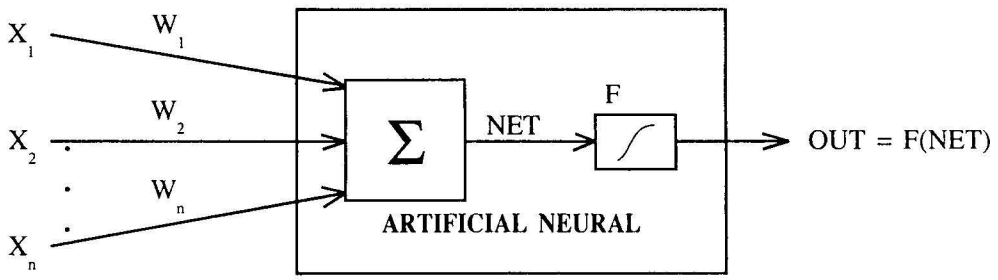
1. เดนไดรต์ (dendrite) ซึ่งเป็นตัวรับสัญญาณ อินพุต (input)
2. แอกซอน (axon) ซึ่งเป็นตัวส่งสัญญาณ เอาต์พุต (output)
3. ซินแนปส์ (synapse) ซึ่งเป็นจุดเชื่อมต่อ (connection point)

แต่ละนิวรอลจะเชื่อมต่อกับนิวรอลอื่นๆ อีกหลายนิวรอล โดยเดนไดรต์จะทำหน้าที่รับสัญญาณที่จุดเชื่อมต่อที่เรียกว่า ซินแนปส์ ซึ่งมีผลทำให้เกิดการลดหรือเพิ่มความเข้มของสัญญาณ ซึ่งบางสัญญาณสามารถกระตุ้นตัวเซลล์ บางสัญญาณสามารถยับยั้งตัวเซลล์ เมื่อผลรวมของสัญญาณทั้งหมดมากกว่าค่าเทรสโฮล (threshold) นิวรอลจะส่งสัญญาณโดยผ่านแอกซอนไปให้นิวรอลอื่นทันที กล่าวโดยสรุป นิวรอลหรือเซลล์ประสาทจะทำหน้าที่ 2 ประการคือ ทำการคำนวณและส่งผลการคำนวณผ่านปลายข้างหนึ่งของเซลล์ไปให้เซลล์อื่น ๆ ต่อไป

คอมพิวเตอร์นิวรอลเนตเวิร์ก

นิวรอลเนตเวิร์กในแง่ของคอมพิวเตอร์ประกอบด้วยหน่วยหรือส่วนประมวลผล (Processing Elements) เชื่อมโยงกันหลาย ๆ ตัวในลักษณะขนาน คล้ายกับนิวรอลในสมองมนุษย์ ซึ่งมีลักษณะเป็น

สถาปัตยกรรมแบบขนาน (Parallel Architecture) เพื่อแปลงข้อมูลจากรูปหนึ่งไปเป็นอีกรูปหนึ่ง การใช้งานนิวรอลเนตเวิร์กจะเป็นไปในรูปการฝึกสอน (training)



แผนผังแสดงอาร์ติฟิเชียลนิวรอลกับแอกติเวชันฟังก์ชัน[6] (Activation Function)

จากแผนผังจะได้ว่า $net = \Sigma w_I x_I$ .....(1)

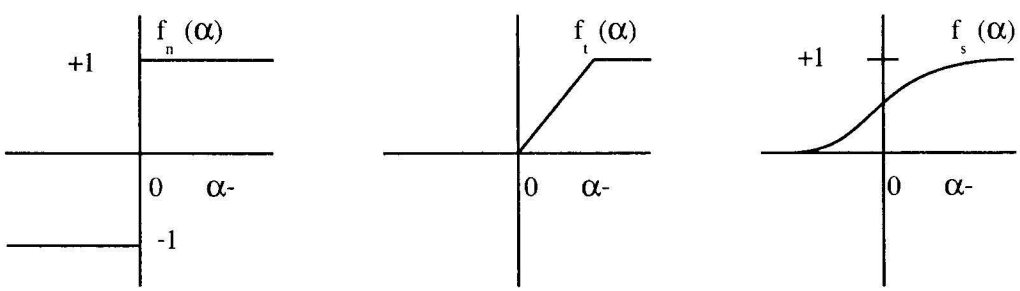
$= x_1 w_1 + x_2 w_2 + ... + x_n w_n$

โดย $x_1, x_2, ..., x_n$ เป็นกลุ่มของข้อมูลอินพุต

$w_1, w_2, ..., w_n$ เป็นกลุ่มของตัวเลขน้ำหนัก

net เป็นผลรวมสุทธิที่เกิดจากค่าอินพุต (x) คูณกับ ตัวเลขน้ำหนัก (w)

จากแผนผังแสดงว่า นิวรอล 1 โหนด (node) จะมีค่าอินพุตหลายค่า แต่จะมีค่าเอาต์พุตเพียงค่าเดียวโดยเอาต์พุตในแต่ละนิวรอล เกิดจากผลรวมของผลคูณระหว่างอินพุต และตัวเลขน้ำหนักโดยส่งผ่านให้กับแอกติเวชันฟังก์ชัน ดังกราฟ



a) HARD UMITER b) THRESHOLD LOGIC c) SIGMOID

กราฟแสดงชนิดของแอกติเวชันฟังก์ชัน[2]

โดยทั่วไปจะเลือกใช้ Sigmoid ฟังก์ชัน เป็นแอกติเวชันฟังก์ชัน[3] ซึ่งมีรูปร่างคล้ายตัวอักษร "S" ในภาษาอังกฤษ เพื่อช่วยขยายสัญญาณ net ในช่วงแคบ ๆ ได้ ทำให้ได้ค่าจำกัดในขอบเขตที่ต้องการ โดยได้ผลลัพธ์เพียงค่าเดียวคือ out ซึ่งมีค่าอยู่ในช่วง 0 ถึง 1

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

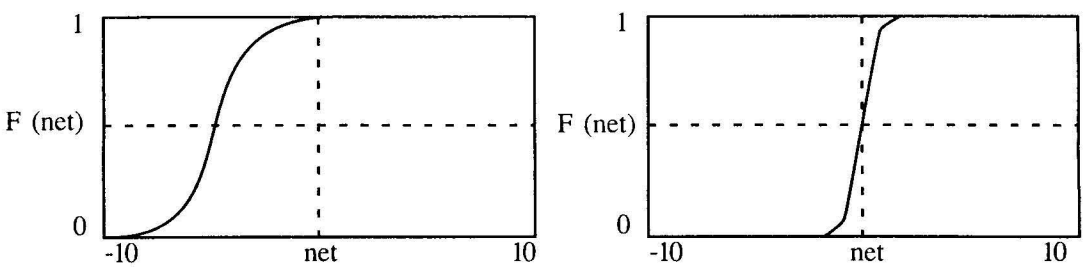
โดย out = เป็นค่าสัญญาณเอาต์พุต

f = เป็นแอกติเวชันฟังก์ชัน ดังในกราฟ

θ = เป็นค่าเทรสโฮล หรือ ค่าไบอัส (bias)

θ_0 = เป็นค่าที่ใช้ปรับรูปร่างของ Sigmoid ฟังก์ชัน

ถ้า θ มีค่าเป็นบวก รูป Sigmoid ฟังก์ชันจะเลื่อนไปทางซ้ายตามแนวแกนนอน และถ้า θ มีค่าน้อยมาก จะทำให้รูป Sigmoid ฟังก์ชันมีลักษณะคล้ายรูปฟังก์ชันขั้นบันได[3]

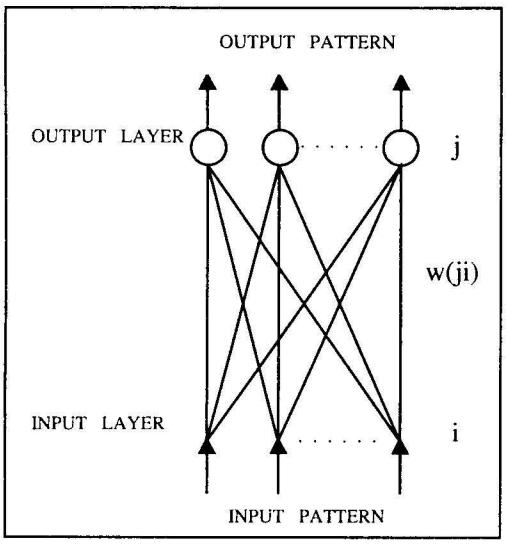


a) θ มีค่ามากกว่า 0 b) θ_0 มีค่าน้อย ๆ

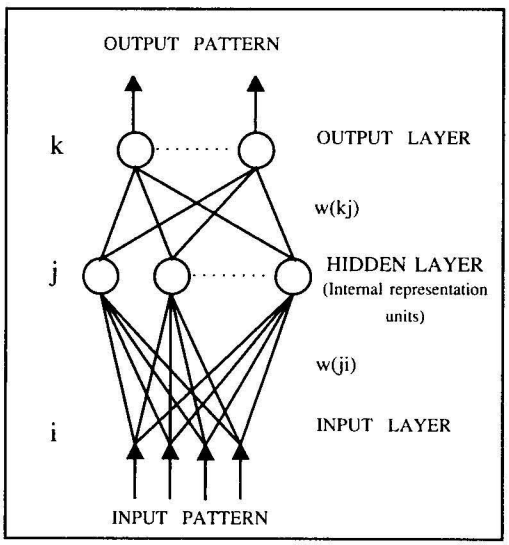
กราฟแสดงผลของไบอัส และค่าที่ใช้ปรับรูปร่างของ Sigmoid ฟังก์ชัน

ข้อสังเกต จากสมการหากค่าผลรวมอินพุต (net) มีค่าเป็นบวกมาก ๆ เอาต์พุตจะมีค่าเข้าใกล้ 1 ในทางตรงกันข้าม ถ้าผลรวมอินพุตน้อยมากจนติดลบ ค่าเอาต์พุตจะมีค่าเข้าใกล้ 0 ซึ่งสอดคล้องกับสถานะเปิด (on) และ ปิด (off)

เนตเวิร์กหลายนิวรอลและหลายเลเยอร์



a) นิวรอลเนตเวิร์กแบบชั้นเดียว



b) นิวรอลเนตเวิร์กแบบหลายชั้น

จากที่กล่าวมาแล้วเป็นเพียงขั้นตอน หรือวิธีการทำงานของนิวรอลเพียงโหนดเดียวซึ่งเทียบได้กับเซลล์เพียงเซลล์เดียว หากประกอบกันเป็นหลาย ๆ เซลล์ จะต้องให้นิวรอลแต่ละตัวเชื่อมต่อถึงกันทำให้เกิดเป็นเนตเวิร์กที่มีลักษณะเป็นชั้น ๆ เรียกว่า เลเยอร์ (layer) ในหนึ่งเลเยอร์สามารถมีจำนวนนิวรอลได้มากกว่า 1 นิวรอล จากรูปภายในเลเยอร์เดียวกันจะไม่มีการเชื่อมต่อถึงกัน และนิวรอลที่มีเลเยอร์สูงกว่าจะรับค่ามาจากเอาต์พุตของเลเยอร์ที่อยู่ต่ำกว่าเพื่อเป็นอินพุตของตัวเอง จากลักษณะดังกล่าวจะเห็นว่า คุณสมบัติของนิวรอลทุกตัวที่อยู่ในเลเยอร์ที่ต่ำกว่าจะถูกถ่ายทอดไปยังนิวรอลในชั้นถัดไป

a) เป็นรูปแสดงลักษณะของนิวรอลที่ประกอบกันเป็นเนตเวิร์ก แบบเลเยอร์เดียว (Single-layer) โดยในเลเยอร์แรกหรืออินพุตเลเยอร์โดยแท้จริงแล้วไม่ใช้นิวรอล เพราะไม่มีการคำนวณอยู่ภายใน แต่จะแสดงลักษณะการกระจายของค่าอินพุตให้กับนิวรอลในชั้นถัดไปเท่านั้น

b) เป็นรูปแสดงลักษณะของเนตเวิร์กที่ใหญ่กว่า ซับซ้อนมากกว่า เพื่อให้ใกล้เคียงกับความเป็นจริงของนิวรอลในสมองมนุษย์ โดยอาจกล่าวได้ว่ามัลติเลเยอร์นิวรอลเนตเวิร์ก (Multilayer Neural Networks) เกิดจากการรวมกันของเนตเวิร์กแบบเลเยอร์เดียวก็ได้ ซึ่งเอาท์พุตของเลเยอร์หนึ่งจะถูกส่งไปเป็นอินพุตในเลเยอร์ย่อยในชั้นถัดไป

จากหลักการดังกล่าว สิ่งที่ต้องกระทำคือ หาค่าตัวเลขน้ำหนักที่เหมาะสมเพื่อคูณกับข้อมูลทางด้านอินพุตให้ได้ค่าเอาต์พุตตามต้องการ การหาตัวเลขน้ำหนักที่เหมาะสมจะอาศัยหลักการและวิธีการของแบ็ก-พรอบพาเกชัน (Back Propagation)

แบ็กพรอบพาเกชัน (Back Propagation)

แบ็กพรอบพาเกชัน มีลักษณะการทำงานเป็นเนตเวิร์กแบบหลายเลเยอร์ ใช้การเรียนรู้แบบ Supervised Learning[2] คือต้องมีครูสอน โดยมีพื้นฐานทางคณิตศาสตร์สนับสนุนเป็นอย่างดี หลักการสำคัญของแบ็กพรอบพาเกชันคือการปรับ และเปลี่ยนตัวเลขน้ำหนักจนกว่าค่าผิดพลาดเฉลี่ยของระบบ (E) จะมีค่าต่ำสุดที่สามารถยอมรับได้ (E_a)

$$E_p = \frac{1}{2} \sum_k (t_{pk} - o_{pk})^2 \qquad .....(3)$$

$$E = \frac{1}{P} \sum_p E_p \qquad .....(4)$$

เพื่อหาค่าต่ำสุดของผลต่างกำลังสองระหว่าง ค่าเอาต์พุตที่ได้จากการคำนวณ (o_{pk}) และค่าเอาต์พุตจริงที่ต้องการ (t_{pk}) โดยค่าผิดพลาดทั้งหมดของแพตเทิร์น p (E_p) เกิดจากผลต่างของเอาต์พุตจริงและค่าเอาต์พุตที่ได้จากการคำนวณในเลเยอร์ k จากสูตรจะเห็นว่าหากต้องการให้ค่าผิดพลาดของระบบลดลงจะต้องให้ผลต่างกำลังสองรวมเข้าใกล้ศูนย์ เนื่องจากค่า t_{pk} ไม่มีการเปลี่ยนแปลง เพราะฉะนั้นสิ่งที่จะทำได้คือ พยายามทำให้ค่า o_{pk} มีค่าเข้าใกล้ t_{pk} มากที่สุด เนื่องจาก o_{pk} มีความสัมพันธ์กับค่าตัวเลขน้ำหนักในนิวรอลเนตเวิร์ก ดังนั้นการปรับค่าตัวเลขน้ำหนักจนได้ค่าที่เหมาะสมจะมีผลทำให้ค่าผิดพลาดของระบบลดลง โดยหาค่าดิฟเฟอเรนเชียลของค่าผิดพลาดที่คำนวณได้กับค่าตัวเลขน้ำหนักที่เปลี่ยนไปในลักษณะสมการเชิงเส้น

$$\Delta_p W_{ji} \quad \alpha \quad - \frac{\partial E_p}{\partial W_{ji}} \qquad .....(5)$$

$$W_{ji}(t+1) = W_{ji}(t) + \Delta_p W_{ji} \qquad .....(6)$$

โดย $W_{ji}(t+1)$ = ค่าตัวเลขน้ำหนักระหว่างเลเยอร์ j ไปเลเยอร์ i หลังจากปรับค่าแล้ว

$W_{ji}(t)$ = ค่าตัวเลขน้ำหนักระหว่างเลเยอร์ j ไปเลเยอร์ i ก่อนการปรับค่า

$\Delta_p W_{ji}$ = ค่าตัวเลขน้ำหนักที่เปลี่ยนไปจากการคำนวณ

t = เป็นเวลาขณะใด ๆ , $t+1$ = เป็นเวลาถัดไป

จากสมการ (5) เมื่อความชันของค่าความผิดพลาด ($\partial E_p / \partial W_{ji}$) ลดลง จะมีผลทำให้ค่าตัวเลขน้ำหนักที่เปลี่ยนไป ($\Delta_p W_{ji}$) ลดลงด้วย วิธีการนี้จะกระทำซ้ำไปซ้ำมา จึงจะได้ค่าที่ต้องการซึ่งเรียกว่า การฝึกสอน เนตเวิร์กจะทำงานจนกระทั่งค่าความชันของค่าผิดพลาดเข้าใกล้ศูนย์ด้วย นั่นคือค่าเอาต์พุตที่ได้จากการคำนวณ จะมีค่าใกล้เคียงกับค่าเอาต์พุตที่ต้องการมากที่สุด จากสมการ (5) สามารถทำเป็นสมการเชิงเส้นได้ดังนี้

$$\Delta_p W_{ji} = -\eta \frac{\partial E_p}{\partial W_{ji}} \qquad .....(7)$$

โดย η เป็นค่าของอัตราการเรียนรู้ (learning rate) ซึ่งมีค่าอยู่ระหว่าง 0.01 - 1.00

แบ็กพรอบพาเกชันมีพื้นฐานมาจากสมการของ Delta rule[4] ซึ่งเป็นลักษณะการทำงานของนิวรอลเนตเวิร์กแบบชั้นเดียว มีเฉพาะอินพุตเลเยอร์ (Input layer) และเอาต์พุตเลเยอร์ (Output layer) เท่านั้น สมการที่ใช้ปรับค่าตัวเลขน้ำหนักคือ

$$\Delta_p W_{ji} = \eta (t_{pi} - o_{pj}) o_{pi} = \eta \delta_{pj} o_{pi} \qquad .....(8)$$

โดย δ_{pj} เป็นค่าแสดงผลต่างระหว่างเอาต์พุตจริงที่ต้องการ และค่าเอาต์พุตที่ได้จากการคำนวณ O_{pi} เป็นค่าอินพุตในเลเยอร์ i จากสมการ Delta rule นี้เทียบกับสมการ (7) ของแบ็กพรอบพาเกชันจะได้ว่า

$$\frac{\partial E_p}{\partial W_{ji}} = \delta_{pj} o_{pi} \qquad .....(9)$$

จากการพิสูจน์เพื่อหาค่า δ โดยวิธีของแบ็กพรอบพาเกชันในแต่ละเลเยอร์ สามารถแสดงค่า δ สำหรับเอาต์พุตเลเยอร์

$$\delta_{pk} = f_k'(net_{pk}) (t_{pk} - o_{pk}) \qquad .....(10)$$

$$f_k'(net_{pk}) = \frac{\partial o_{pk}}{\partial net_{pk}} \qquad .....(11)$$

แสดงค่า δ สำหรับทุก ๆ ฮิดเดนเลเยอร์ (Hidden layer)

$$\delta_{pj} = f_j'(net_{pj}) \Sigma_k (\delta_{pk} - W_{kj}) \quad .....(12)$$

$$f_j'(net_{pk}) = \frac{\partial o_{pj}}{\partial net_{pj}} \quad .....(13)$$

สมการที่ (11) และ (13) เป็นสมการในลักษณะทั่วไป ขึ้นอยู่กับแอกติเวชันฟังก์ชันที่ใช้ หากใช้รูป Sigmoid ฟังก์ชัน ดังสมการ (4) จะได้ว่า

$$o_{pj} = f_j(net_{pj}) = \frac{1}{1 + e^{-(net_{pk} + \theta_j)}} \quad .....(14)$$

จะได้ว่า $$f_j'(net_{pj}) = o_{pj}(1 - o_{pj}) \quad .....(15)$$

$$o_{pk} = f_k(net_{pk}) = \frac{1}{1 + e^{-(net_{pk} + \theta_k)}} \quad .....(16)$$

จะได้ว่า $$f_k'(net_{pk}) = o_{pk}(1 - o_{pk}) \quad .....(17)$$

อุปกรณ์และวิธีการ

1. ข้อกำหนดเบื้องต้นเพื่อให้คอมพิวเตอร์สามารถเรียนรู้และจดจำแพตเทิร์นตัวอักษร ตัวเลข แบบตัวพิมพ์ที่นำมาทดสอบ ได้พัฒนาการเขียนซอฟแวร์ขึ้น เพื่อทดสอบการจดจำตัวอักษรแบบตัวพิมพ์ตั้งแต่ "A" ถึง "Z" โดยแต่ละตัวอักษรมีขนาด 8x8 บิตแพตเทิร์น (64 บิตแพตเทิร์น) ดังรูปที่ 1

แพตเทิร์นที่ปรากฏดังในรูปที่ 1 สามารถมองเห็นได้จากจอภาพแสดงผล ในลักษณะเป็นบิต (ค่า 0 หรือ 1) ซึ่งจุดแต่ละจุดที่มองเห็นจะถูกนำมาเป็นอินพุต (o_{pi}) และค่าเอาต์พุตที่ต้องการ (t_{pk}) ของนิวรอลเนตเวิร์ก โดย i และ k มีค่าตั้งแต่ 1 ถึง 64 (64 นิวรอลโหนด)

2. กำหนดจำนวนเลเยอร์ทั้งหมดในการทดสอบเพียง 3 เลเยอร์คือ อินพุตเลเยอร์ ฮิดเดนเลเยอร์ และเอาต์พุตเลเยอร์ โดยฮิดเดนเลเยอร์จะมีจำนวนนิวรอลเพียง 20 นิวรอล ดังรูปที่ 2

ขั้นตอนการคำนวณ

1. เริ่มแรกจะต้องให้ค่าเริ่มต้นแก่ค่าตัวเลขน้ำหนัก (w) ทุกตัว ในลักษณะสุ่มเป็นค่าน้อย ๆ โดยจำกัดในช่วง 0.0-1.0 ทุกตัว และให้ค่าเทรสโฮล $\theta_j = 0.05$ และ $\theta_k = 0.05$

2. จัดลำดับของอินพุต o_{pi} $(o_{p1}, \ldots, o_{p64})$ และค่าเอาต์พุต t_{pk} $(t_{p1}, \ldots, t_{p64})$ ในแต่ละแพตเทิร์นให้สัมพันธ์กัน สามารถแสดงผลออกทางจอภาพได้ในลักษณะเป็นบิต โดยกำหนดขั้นตอนการคำนวณตามลำดับการทำงานของโปรแกรม

ฝึกสอนเนตเวิร์กจนได้ค่าตัวเลขน้ำหนักที่ไม่มีการเปลี่ยนแปลง หรือค่าผิดพลาดเฉลี่ยของระบบ (E) มีค่าน้อยกว่าค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ (E_a) โดยกำหนดให้ = 0.05

3. คำนวณหาค่าเอาต์พุต (o_{pj}) และ (o_{pk}) ในลักษณะไปข้างหน้า (feed forward) โดยใช้ทฤษฎีของแบ็กพรอบพาเกชัน

$$net_{pj} = \Sigma_i W_{ji} o_{pi} \qquad .....(18)$$

$$o_{pj} = f_j(net_{pj}) = \frac{1}{1 + e^{-(net_{pj} + 0.05)}} \qquad .....(19)$$

$$net_{pk} = \Sigma_j W_{kj} o_{pj} \qquad .....(20)$$

$$o_{pk} = f_k(net_{pk}) = \frac{1}{1 + e^{-(net_{pk} + 0.05)}} \qquad .....(21)$$

4. ปรับค่าตัวเลขน้ำหนักในแต่ละคู่เลเยอร์ ในลักษณะป้อนกลับ (feedback) โดยเริ่มจากเอาต์พุตเลเยอร์ก่อน

$$W_{kj}(t+1) = W_{kj}(t) + \eta\, \delta_{pk}\, o_{pj} \qquad .....(22)$$

$$= W_{kj}(t) + \eta\, [o_{pk}(1-o_{pk})\,(t_{pk}-o_{pk})]\, o_{pj} \qquad .....(23)$$

$W_{kj}(t)$ เป็นค่าตัวเลขน้ำหนักฮิดเดนเลเยอร์ (j) ไปยังเอาต์พุตเลเยอร์ (k) ที่เวลา t ใด ๆ และ t+1 เป็นเวลาถัดไป กำหนดค่าอัตราการเรียนรู้ (η) ในการทดสอบมีค่า = 0.5 จากนั้นปรับค่าตัวเลขน้ำหนักคู่เลเยอร์ถัดไปในลักษณะเดียวกัน

$$W_{ji}(t+1) = W_{ji}(t) + \eta\, \delta_{pj}\, o_{pi} \qquad .....(24)$$

$$= W_{ji}(t) + \eta\, [o_{pj}(1-o_{pj})\, \Sigma_k\, (\delta_{pk}\, w_{kj})]\, o_{pi} \qquad .....(25)$$

5. กลับไปคำนวณในข้อ 2

รูปที่ 3 แสดงการฝึกสอนเนตเวิร์กรวมทั้งหมด 126 รอบ โดยแต่ละรอบจะฝึกสอนแพตเทิร์นตัวอักษร ตั้งแต่ "A" ถึง "Z" (26 แพตเทิร์นตัวอักษร) ทุกครั้ง ดังนั้น จอภาพจะแสดงรูปทั้งหมด 126 x 26 = 3,276 รูป ซึ่งไม่สามารถพิมพ์ผลได้ทั้งหมด จึงเลือกเฉพาะบางภาพมาแสดง

รูปที่ 3 a-g แสดงการเรียนรู้ของแพตเทิร์นตัวอักษร "A" โดยในรอบแรก ๆ เอาต์พุตแพตเทิร์นที่ได้จากอินพุตแพตเทิร์นเป็นเอาต์พุตที่ไม่ถูกต้อง แต่เมื่อจำนวนรอบเพิ่มขึ้นมีการปรับค่าตัวเลขน้ำหนักในทุกรอบ โดยวิธีการของแบ็กพรอบพาเกชัน-นิวรอลเนตเวิร์ก มีผลทำให้เอาต์พุตดีขึ้นตามลำดับ

รูปที่ 3-e (รอบที่ 20) แสดงเอาต์พุตแพตเทิร์นที่ได้จากอินพุตแพตเทิร์นมีลักษณะถูกต้องแล้ว แต่ค่า E ยังมีค่ามากกว่าค่า E_a โปรแกรมจึงทำงานต่อไปจนกระทั่งค่า E มีค่าน้อยกว่าค่า E_a โปรแกรมจะหยุดทำงานและมีการเก็บค่าตัวเลขน้ำหนักสุดท้ายไว้ใช้งานต่อไป

ผลการทดสอบ

ได้มีการทดสอบความสามารถของ แบ็กพรอบพาเกชัน-นิวรอลเนตเวิร์ก เรื่องการลดความผิดเพี้ยน

โดยทดลองเพิ่มสัญญาณรบกวนเข้าไปในแพตเทิร์นที่สมบูรณ์ ดังรูปที่ 4 ซึ่งกำหนดค่าความน่าจะเป็นของสัญญาณรบกวน P (probability) เท่ากับ 0.10 (10%) คำนวณจำนวนบิตของสัญญาณรบกวนสำหรับแพตเทิร์นขนาด 8x8 (64 บิตแพตเทิร์น) ตามสมการ (26)

$$P = \frac{\text{Noise}}{\text{ขนาดของแพตเทิร์น}} \qquad \text{.....26}$$

แทนค่า Noise $= 0.1 \times 64$

$= 6.4$

$\simeq$ 7 บิตแพตเทิร์น

ดังนั้น จะต้องใส่สัญญาณรบกวนจำนวน 7 บิตแพตเทิร์น ลงใน 64 บิตของอินพุตแพตเทิร์น

รูปที่ 4 แสดงผลการทดสอบโดยดูจากจอภาพ ตัวอักษรทางซ้ายมือเป็นแพตเทิร์นอินพุตปกติ ซึ่งนำมาเป็นตัวอย่างการทดสอบ ตัวอักษรตรงกลางหมายถึงอินพุตแพตเทิร์นที่มีสัญญาณรบกวน และตัวอักษรขวามือเป็นตัวอักษรที่ผ่านการจำแนกแพตเทิร์น ด้วยโปรแกรมคอมพิวเตอร์โดยวิธีการของแบ็กพรอบพาเกชัน-นิวรอลเนตเวิร์ก ผลจากการทดสอบแสดงให้เห็นว่า แม้อินพุตแพตเทิร์นที่เข้ามาจะมีสัญญาณรบกวน หากไม่มากนักโปรแกรมสามารถจำแนกแพตเทิร์นและแสดงเอาต์พุตที่ถูกต้องได้ โดยมีความผิดพลาดน้อยมาก

เทคนิคการฝึกสอนเพิ่มเติม

จากที่กล่าวมาแล้วข้างต้น การกำหนดค่าอัตราการเรียนรู้ (Learning rate η) เป็นค่าคงที่ตลอดการฝึกสอน ซึ่งให้ผลดีพอสมควร แต่หากเพิ่มเทคนิคการฝึกสอน อีก 2 วิธีรวมเป็น 3 วิธี ดังนี้

1. กำหนดค่าอัตราการเรียนรู้ (η) เป็นค่าคงที่ ซึ่งการเลือกค่า η ที่เหมาะสมที่สุดจำเป็นต้องทดลองแทนค่าด้วยตนเอง จากการทดสอบกำหนดให้ค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ E_a เป็นค่าคงที่ = 0.05 (ตารางที่ 1)

ถ้าให้ η มีค่ามาก จำนวนรอบของการฝึกสอนจะมีค่าน้อย หมายความว่า เนตเวิร์กสามารถเรียนรู้ได้เร็ว ใช้เวลาน้อย แต่ประสิทธิภาพในการจำแนกแพตเทิร์นจะไม่ดีพอ ในทางตรงกันข้าม ถ้าให้ η มีค่าน้อย ขณะฝึกสอน จำนวนรอบทั้งหมดจะมีค่ามาก แต่สามารถจำแนกแพตเทิร์นได้ดี ซึ่งผลจากการทดสอบ พบว่าค่า η ที่เหมาะสมอยู่ในช่วง 0.2 - 0.5

กำหนดให้ η เป็นค่าคงที่บ้างแล้วทดลองเปลี่ยนค่า E_a ปรากฏว่า จำนวนรอบและเวลาทั้งหมดในการฝึกสอน จะเพิ่มขึ้นมากเมื่อ E_a มีค่าลดลง แต่ประสิทธิภาพในการจำแนกแพตเทิร์นตัวอักษรจะดีขึ้น

เพียงเล็กน้อย ดังตารางที่ 2 กำหนดให้ η เป็นค่าคงที่ = 0.5

ข้อสรุปจากการทดสอบคือ เมื่อให้ η มีค่ามากขณะฝึกสอน ค่าตัวเลขน้ำหนักจะมีการเปลี่ยนแปลงอย่างรวดเร็วตลอดเวลา อาจทำให้เกิดการแกว่งของระบบขึ้น[4] (oscillate) เมื่อค่าผิดพลาดเฉลี่ยของระบบ E เข้าใกล้ค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ E_a แต่ถ้าให้ η มีค่าน้อยการเรียนรู้จะใช้เวลามาก ดังนั้นเพื่อแก้ปัญหาดังกล่าว จึงเพิ่มเทคนิคการฝึกสอนขึ้นอีก 2 วิธีตามข้อ 2 และ 3 โดยเริ่มแรกจะให้ η มีค่ามาก และค่อย ๆ ลดลงจนกระทั่ง E มีค่าน้อยกว่า E_a จึงหยุดทำงาน ดังนั้นปัญหาเรื่องการแกว่งของระบบก็จะหมดไป

2. สูตรแปรผันตามจำนวนรอบ (t) ใช้แนวความคิดมาจากสูตรการกระจายค่าของเอาต์พุต Self organization map[1] ของ Kohonen โดยเปลี่ยนมาใช้กับค่าอัตราการเรียนรู้ (η) กำหนดให้ค่า η เริ่มแรกมีค่า = 1.0 และค่อย ๆ ลดลงตามจำนวนรอบที่เพิ่มขึ้น จนกระทั่งค่าผิดพลาดเฉลี่ยของระบบ E มีค่าน้อยกว่าค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ E_a ซึ่งกำหนดค่าสุดท้ายของ η มีค่าประมาณ 0.3

$$\eta = \eta_0 \left(1 - \frac{t}{T}\right) \qquad .....(27)$$

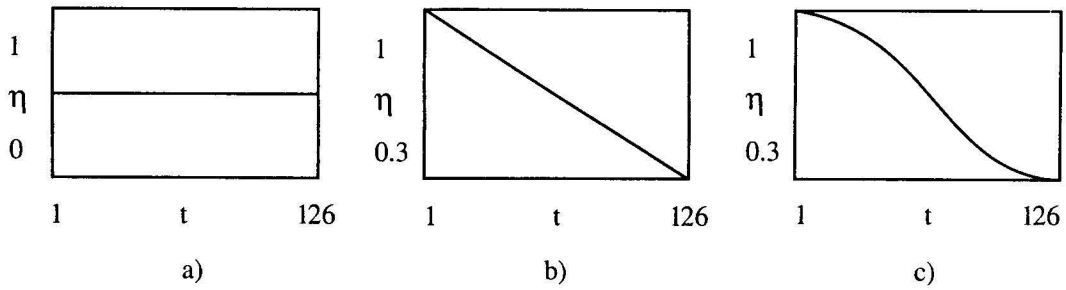
โดย η_0 = ค่าเริ่มต้นของอัตราการเรียนรู้ให้ค่า = 1.0

t = หมายเลขของรอบขณะทำการทดสอบ

T = จำนวนรอบทั้งหมดที่ใช้ในการทดสอบโดยประมาณต้องกำหนดเอง โดยพิจารณาจากจำนวนรอบทั้งหมดของการฝึกสอน เมื่อให้ค่า η เป็นค่าคงที่จากตาราง 1 และ 2 หากกำหนดค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ E_a = 0.05 แล้วค่า T ควรมีค่าประมาณ 150

3. สูตรแปรผันตามค่าผิดพลาดเฉลี่ยของระบบ (E) ซึ่งเป็นวิธีที่นำเสนอ กำหนดให้ค่า η เริ่มแรกมีค่า = 1.0 และค่อย ๆ ลดลงจนกระทั่งค่าผิดพลาดเฉลี่ยของระบบ E มีค่าน้อยกว่าค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ E_a กำหนดค่าสุดท้ายของค่า η มีค่าประมาณ 0.3

$$\eta = 1 - \frac{0.7\,E_a}{E} \qquad .....(28)$$



กราฟแสดงการกำหนดค่า η แตกต่างกันตามสูตร

กราฟนี้แสดงความสัมพันธ์ระหว่างค่าอัตราการเรียนรู้ η กับค่าหมายเลขของรอบขณะทำการทดสอบ a) กำหนดให้ η เป็นค่าคงที่ b) กำหนดให้ η แปรผันตามจำนวนรอบ c) กำหนดให้ η แปรผันตามค่าผิดพลาดเฉลี่ยของระบบ

เพื่อเปรียบเทียบการทดสอบอัตราการเรียนรู้ทั้ง 3 แบบ จึงนำข้อมูลของความสัมพันธ์ระหว่างค่าผิดพลาดเฉลี่ยของระบบ (E) กับจำนวนรอบที่เพิ่มขึ้น (t) ขณะทำการฝึกสอนเขียนเป็นกราฟขึ้น ดังรูปที่ 5 โดยแสดงค่า E ในรูปแบบของลอการิทึม

แบบ 1) กำหนดค่า η เป็นค่าคงที่ = 0.5

แบบ 2) กำหนดค่า η แปรผันตามจำนวนรอบ (t)

แบบ 3) กำหนดค่า η แปรผันตามค่าผิดพลาดเฉลี่ยของระบบ (E)

จากรูปที่ 5 แบบที่ 2) และแบบที่ 3) ให้ผลใกล้เคียงกัน โดยเส้นกราฟจะทับกันพอดี ซึ่งค่าผิดพลาดเฉลี่ยของระบบ E จะลดลงเร็วมากในช่วงแรก และจะช้าลงเมื่อจำนวนรอบเพิ่มขึ้น จากการทดสอบความสามารถในการจำแนกแพตเทิร์นเปรียบเทียบกันทั้ง 3 แบบพบว่า แบบที่ 2) และแบบที่ 3) จะสามารถจำแนกแพตเทิร์นได้ดีกว่าแบบที่ 1)

อภิปรายและวิจารณ์

1. หากกำหนดจำนวนฮิดเดนโหนด (Hidden node) น้อยเกินไป จะทำให้ความสามารถในการจำแนกแพตเทิร์นตัวอักษรให้ผลได้ไม่ดีเท่าที่ควร ซึ่งจำนวนที่เหมาะสมสามารถหาได้จากการทดลอง

2. ค่าผิดพลาดเฉลี่ยของระบบที่ยอมรับได้ E_a ที่เหมาะสม ขึ้นอยู่กับงานที่ประยุกต์ใช้หากเป็นงานที่ต้องการความละเอียดสูง จะกำหนดให้ค่า E_a มีค่าน้อย ๆ

3. หากกำหนดให้ η เป็นค่าคงที่ตลอดการฝึกสอนและมีค่ามาก อาจทำให้เกิดการแกว่งของระบบ

ในขณะทำการฝึกสอนได้ ดังนั้นค่า η ควรแปรเปลี่ยนตามจำนวนรอบของการฝึกสอนหรือค่าผิดพลาดเฉลี่ยของระบบ

4. เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบมี CPU 80386DX-25 และ MATH-CO80387 ซึ่งมีประสิทธิภาพและสมรรถนะค่อนข้างสูงแล้ว แต่ยังใช้เวลาในการฝึกสอนมาก หากเป็นนิวรอลเนตเวิร์กที่มีระบบใหญ่มากควรทดสอบโปรแกรมบนเครื่องที่มีประสิทธิภาพสูงกว่านี้

5. หากใช้ขนาดแพตเทิร์นอินพุตมากกว่า 8x8 เช่น 16x16 เป็นต้น สัญญาณรบกวนจะมีลักษณะกระจายมากขึ้น ซึ่งจะช่วยให้โปรแกรมสามารถจดจำและจำแนกตัวอักษรตัวพิมพ์ได้ดีขึ้นกว่าเดิม แม้ว่าความน่าจะเป็นสัญญาณรบกวนจะเพิ่มมากขึ้นก็ตาม

6. ในสภาพความเป็นจริงนั้น การใช้สแกนเนอร์เพื่อสแกนกราฟิกของตัวอักษรในลักษณะกลุ่มคำหลายบรรทัด เมื่อพิจารณาดูจะพบปัญหา 2 ประการคือ

6.1 มีความผิดเพี้ยนของบิตบางบิตภายในตัวอักษรของแพตเทิร์น

6.2 แพตเทิร์นของตัวอักษรที่คอมพิวเตอร์จดจำกับแพตเทิร์นของตัวอักษรที่รับเข้ามามีลักษณะต่างกัน เช่น ตัวอักษรตัวพิมพ์ “A” หรือ “*A*” เป็นต้น

หากใช้วิธีโดยทั่วไปที่ไม่ใช้นิวรอลเนตเวิร์ก การจดจำและจำแนกแพตเทิร์นตัวอักษรในลักษณะต่าง ๆ เป็นเรื่องที่ยากมากและแทบเป็นไปไม่ได้เลยที่จะมีความแม่นยำสูง แต่จากการทดสอบพบว่าแบ็กพรอบพาเกชัน-นิวรอลเนตเวิร์กสามารถแก้ปัญหาดังกล่าวได้ทั้ง 2 ข้อ ซึ่งมีความใกล้เคียงกับการใช้งานในความเป็นจริงมากกว่า นอกจากนี้ยังพบว่า ประสิทธิภาพในการจำแนกแพตเทิร์นตัวอักษรจะสูงกว่า 60% ขึ้นไป

7. ปัญหาของแบ็กพรอบพาเกชันคือ ใช้เวลามากในการเรียนรู้เพื่อจดจำแพตเทิร์นและทุกครั้งที่มีการเพิ่มแพตเทิร์นตัวอักษรตัวใหม่ จะต้องสอนเนตเวิร์กใหม่ทุกครั้ง แต่ในกรณีที่ผ่านการฝึกสอนมาแล้ว ตัวเลขน้ำหนักที่ได้จากการฝึกสอนจะถูกเก็บไว้และสามารถนำกลับมาใช้ใหม่ได้

สรุป

การนำนิวรอลเนตเวิร์กซึ่งเป็นศาสตร์ที่มีความใกล้เคียงกับสมองมนุษย์ มาช่วยในการปฏิบัติงานทำให้เกิดประโยชน์อย่างมากทั้งในปัจจุบันและในอนาคต โดยคอมพิวเตอร์สามารถเรียนรู้และจดจำแพตเทิร์นในลักษณะต่าง ๆ ได้ ในอนาคตสามารถนำนิวรอลเนตเวิร์กไปประยุกต์เพื่อจดจำลายมือเขียน ลายนิ้วมือ ลายเซ็น และเสียงพูดของมนุษย์ได้

บทความนี้ได้เสนอวิธีการของแบ็กพรอบพาเกชัน ซึ่งเป็นนิวรอลเนตเวิร์กแบบหนึ่งที่มีความเหมาะสมมากที่สุด การจดจำและจำแนกแพตเทิร์นตัวอักษรตัวพิมพ์เพื่อประยุกต์ใช้กับโปรแกรมทางด้าน OCR จากผลการทดสอบโดยเขียนซอฟแวร์ให้ผลเป็นที่น่าพอใจระดับหนึ่ง แต่ยังต้องมีการพัฒนาต่อไปอีก เพื่อให้ซอฟแวร์สามารถจดจำและจำแนกแพตเทิร์นตัวอักษรซึ่งมีขนาดต่าง ๆ กัน หรือเอนเอียงไปในทิศใดทิศหนึ่งได้

เอกสารอ้างอิง

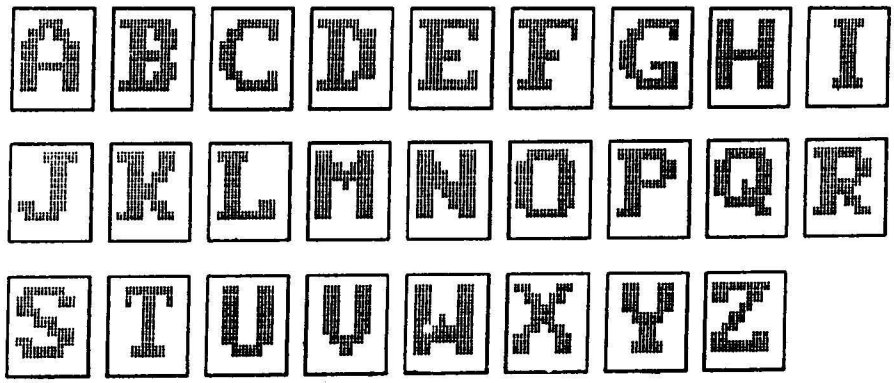
1. Kohonen, T. Self-Organization and Associative Memory, Spriner-Verlag, USA, 1989, 130-133.
2. Lippmann, R.P. An Introduction to Computing with Neural Nets. *IEEE Acoustics, Speech, and Signal Processing Society Magazine*, April, 1987, 4(2), 4-22.
3. Pao, Y. H. Adaptive Pattern Neural Networks. Addison-Wesley Inc., USA, 1989, 113-140.
4. Rumelhart, D.E., McClelland, J.L. and the PDP Research Group. Parallel Distributed Processing : Exploration in the Microstructure of Cognition, Volume 1 : Foundations. The Massachusetts Institute of Technology, 1986, 318-334.
5. Treleaven, P., Pacheco, M. and Vellasco, M. VLSI Architectures for Neural Networks. IEEE Micro, University College London, December, 1989, 8-27.
6. Wasserman, P.D. Neural Computing Theory and Practice. ANZA Research Inc, USA, 1989, 11-59.

ตารางที่ 1. แสดงผลจำนวนรอบและเวลาทั้งหมด เมื่อค่า η เปลี่ยนไป

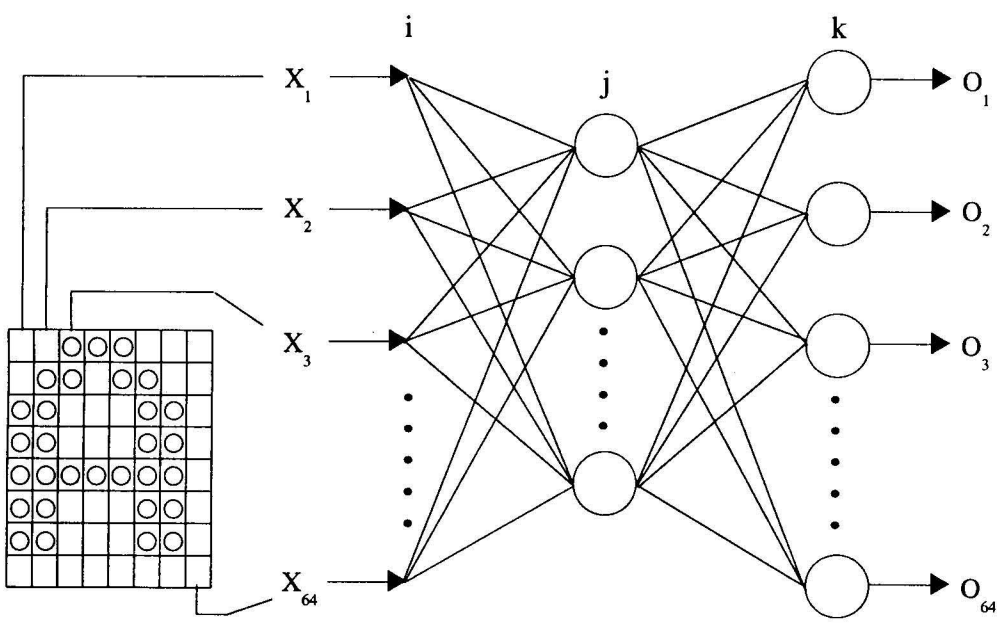
ค่า η	จำนวนรอบทั้งหมด	เวลาทั้งหมด (นาที)
1.0	65	7 : 11
0.7	95	10 : 12
0.5	126	14 : 26
0.2	305	32 : 12

ตารางที่ 2. แสดงผลจำนวนรอบและเวลาทั้งหมด เมื่อค่า E_a เปลี่ยนไป

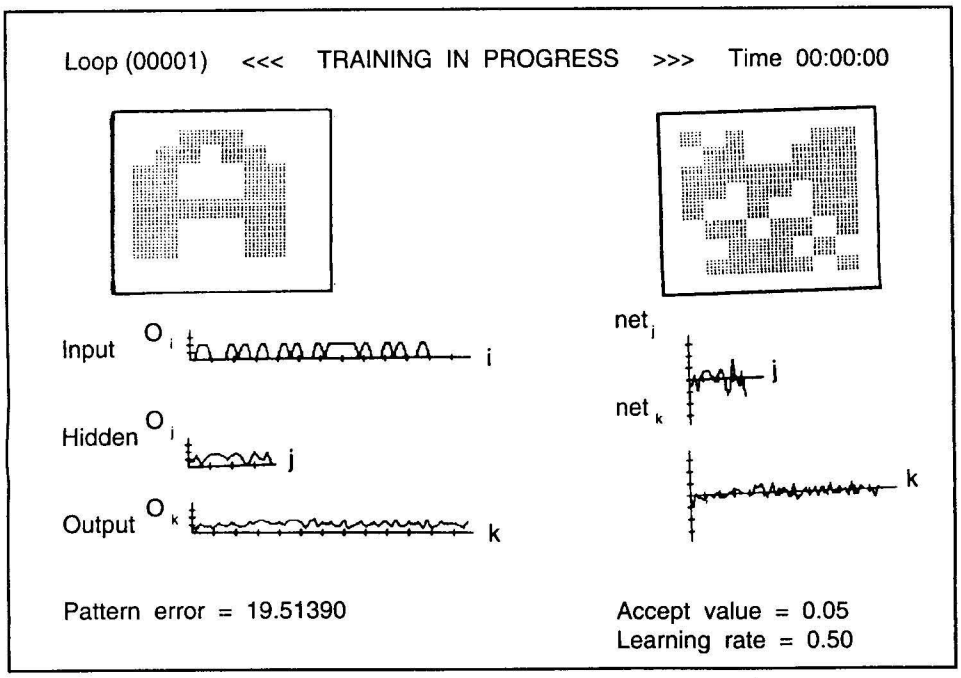
ค่า E_a	จำนวนรอบทั้งหมด	เวลาทั้งหมด (ชั่วโมง)
0.05	126	00:14:26
0.01	523	00:58:06
0.001	4538	08:16:41



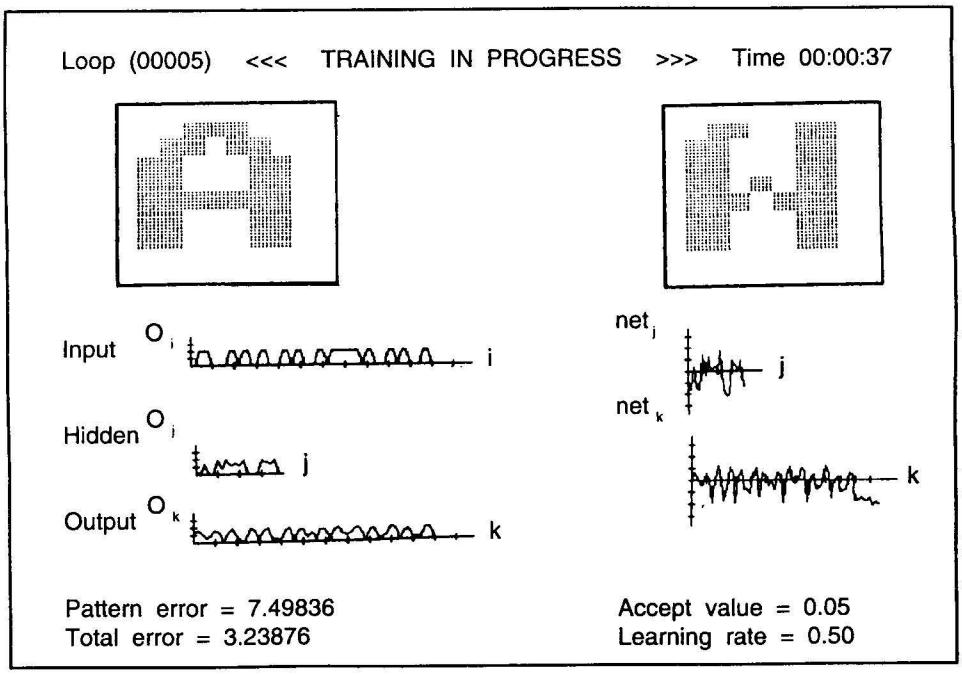
รูปที่ 1. แสดงแพตเทิร์นตัวอักษร “A” . . “Z”



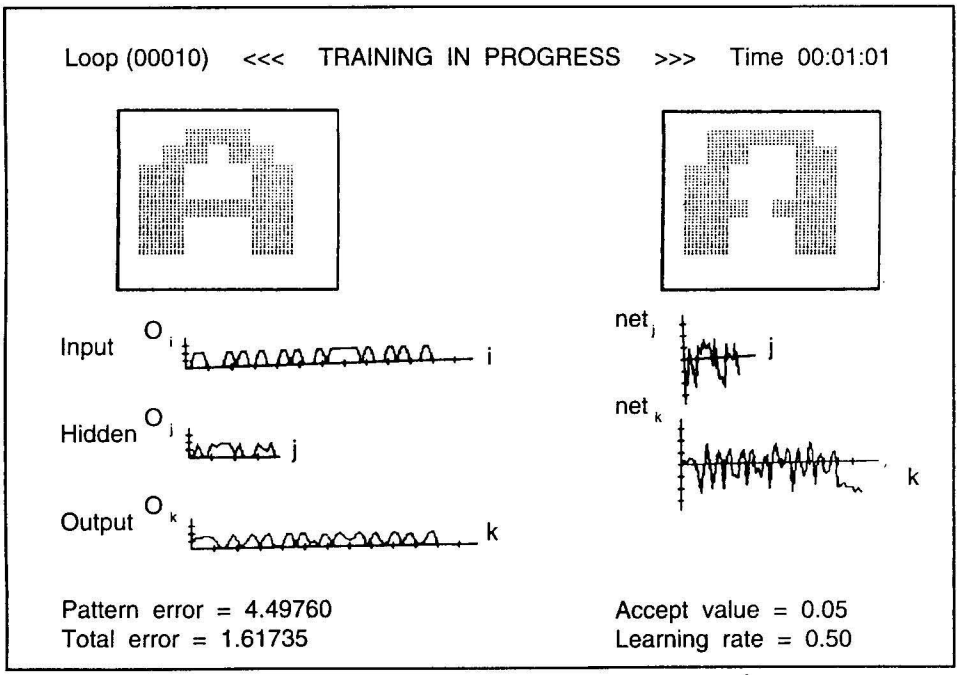
รูปที่ 2. แสดงโครงสร้างการทำงานของนิวรอลเนตเวิร์ก



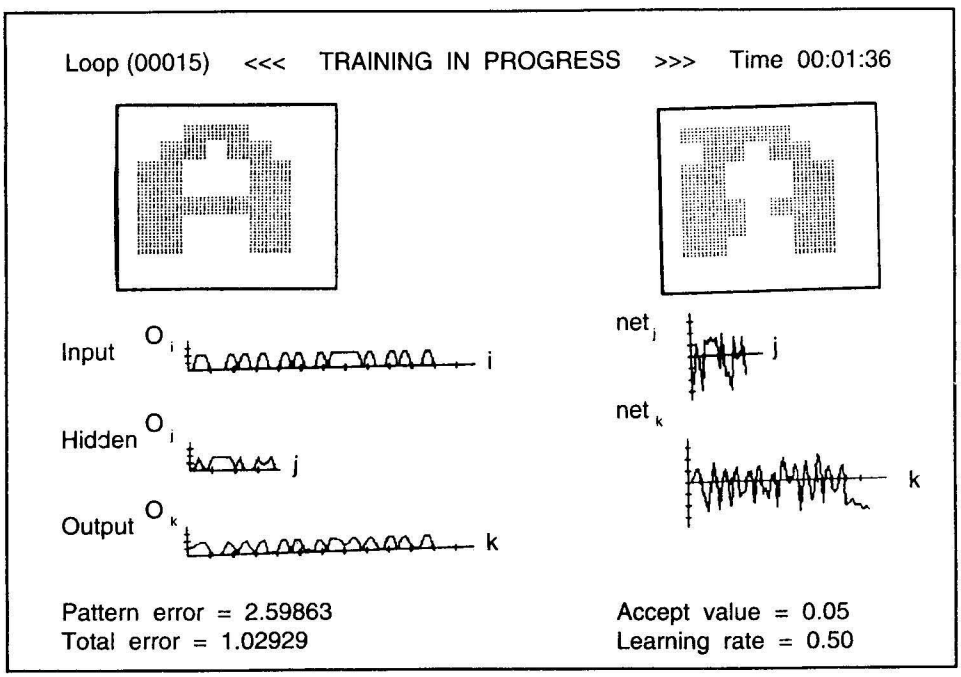
a) แสดงการฝึกสอนแพตเทิร์นตัวอักษร "A" ในรอบที่ 1



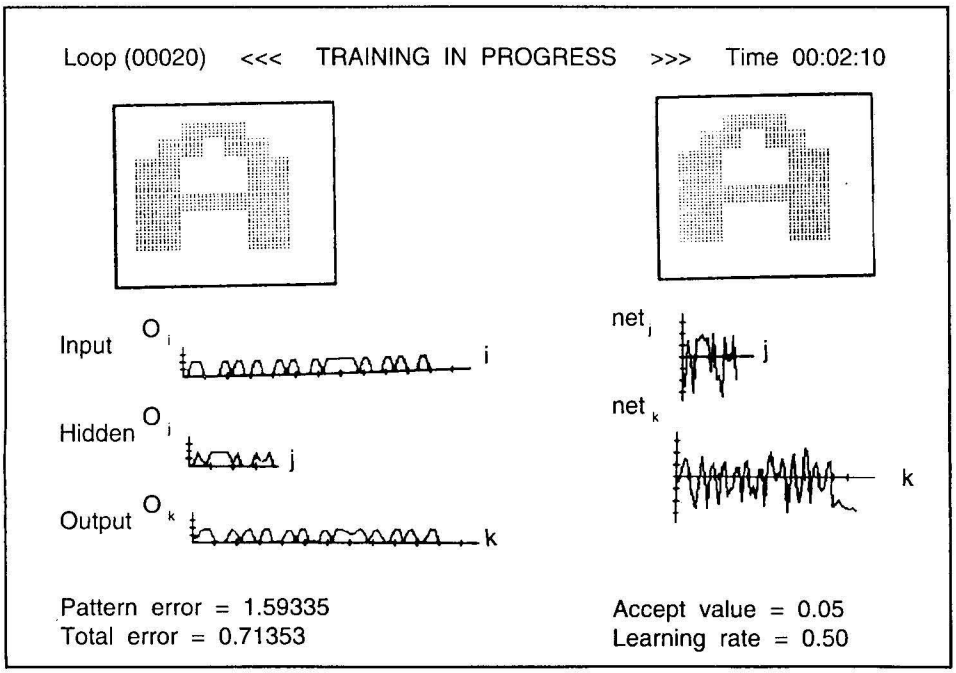
b) แสดงการฝึกสอนแพตเทิร์นตัวอักษร "A" ในรอบที่ 5



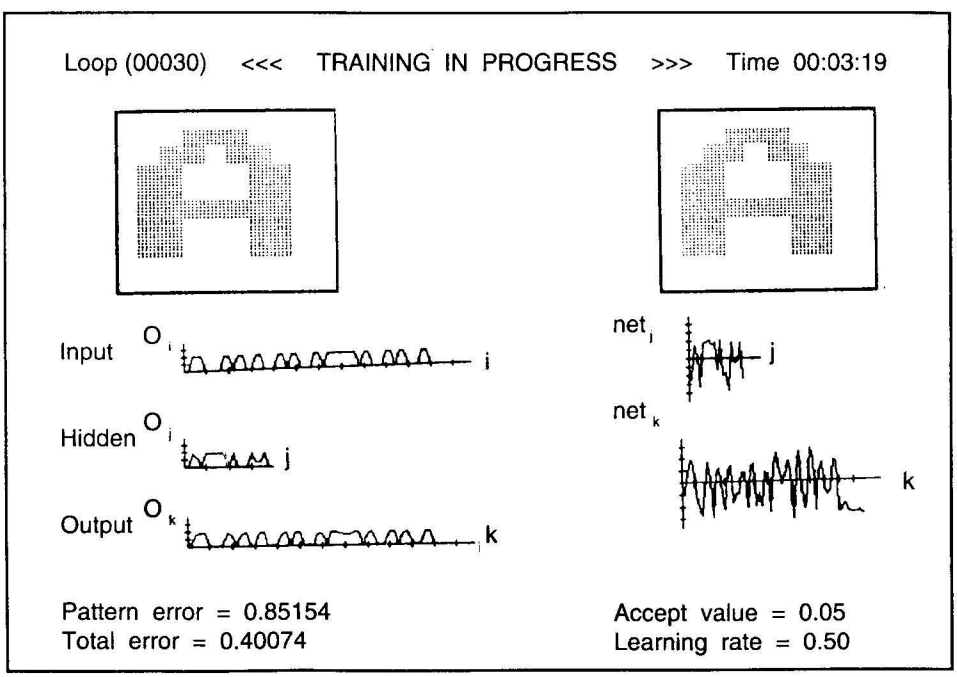
c) แสดงการฝึกสอนแพตเทิร์นตัวอักษร "A" ในรอบที่ 10



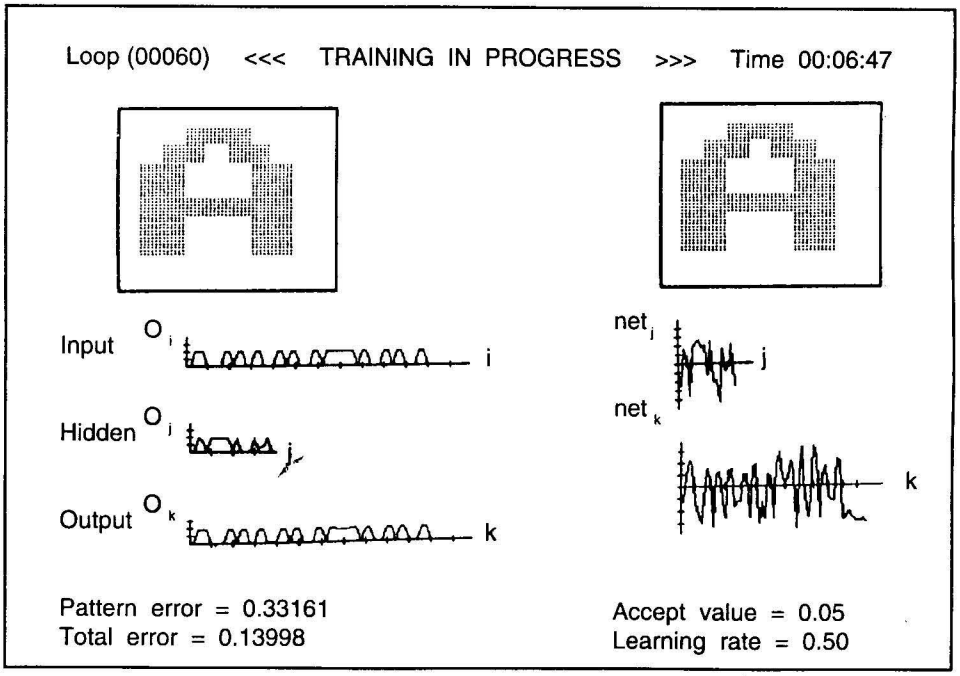
d) แสดงการฝึกสอนแพตเทิร์นตัวอักษร "A" ในรอบที่ 15



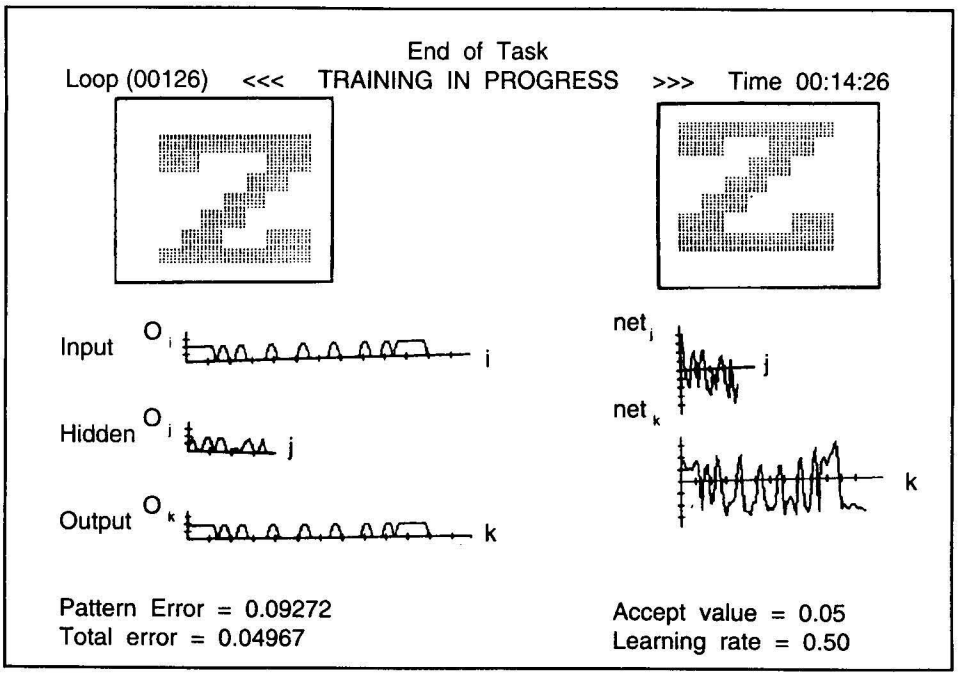
e) แสดงการฝึกสอนแพตเทิร์นตัวอักษร "A" ในรอบที่ 20



f) แสดงการฝึกสอนแพตเทิร์นตัวอักษร "A" ในรอบที่ 30

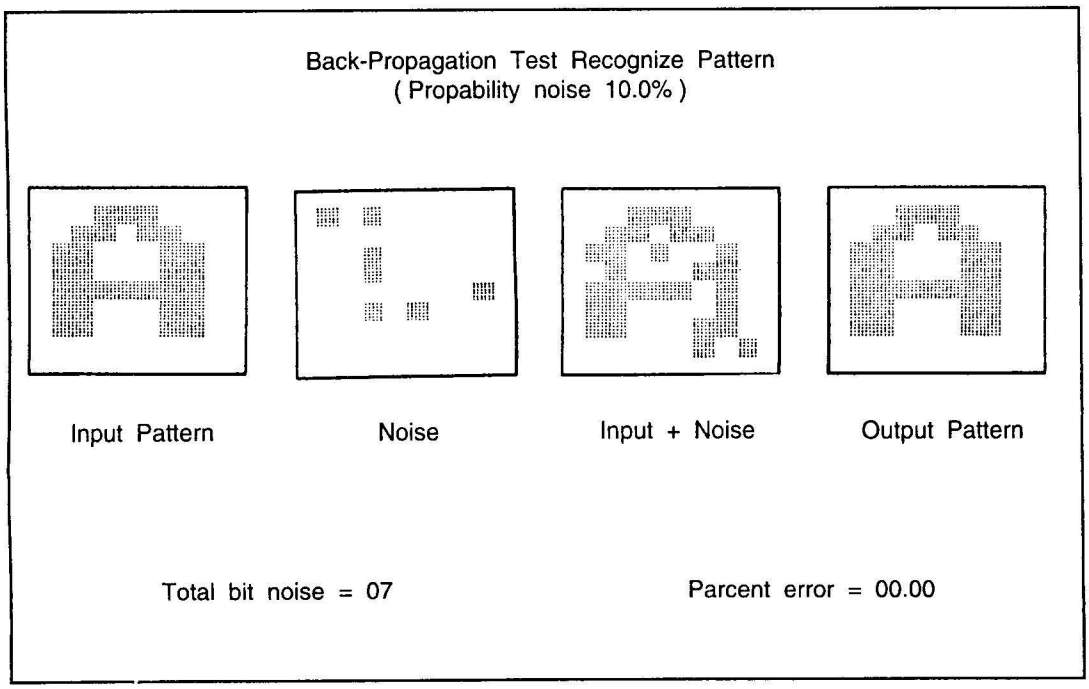


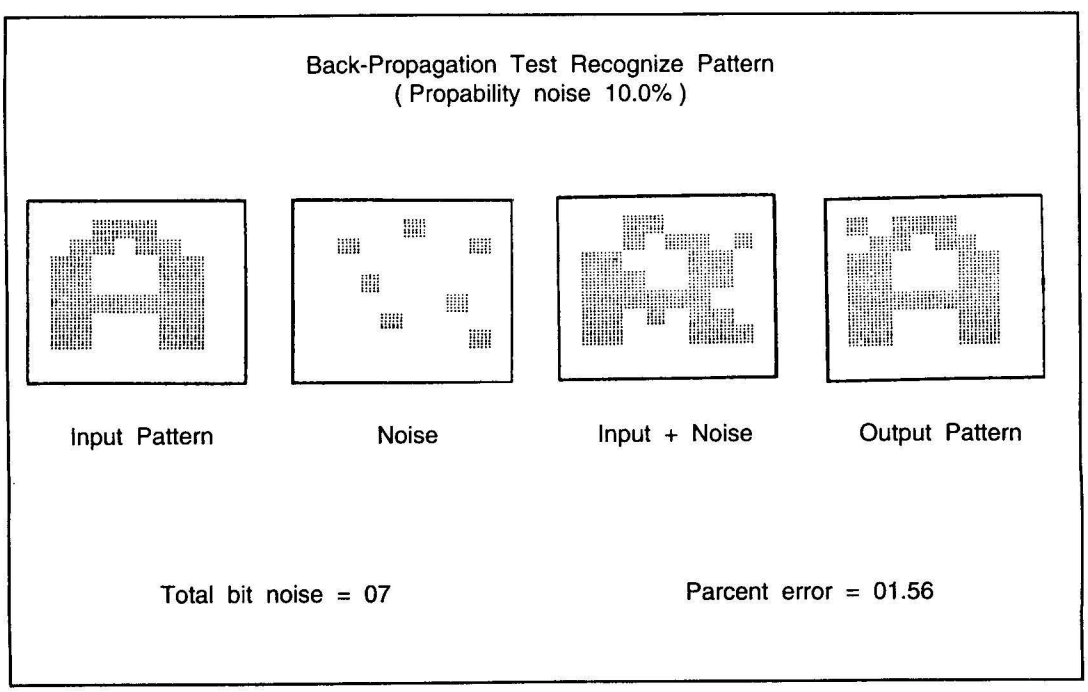
g) แสดงการฝึกสอนแพตเทิร์นตัวอักษร "A" ในรอบที่ 60



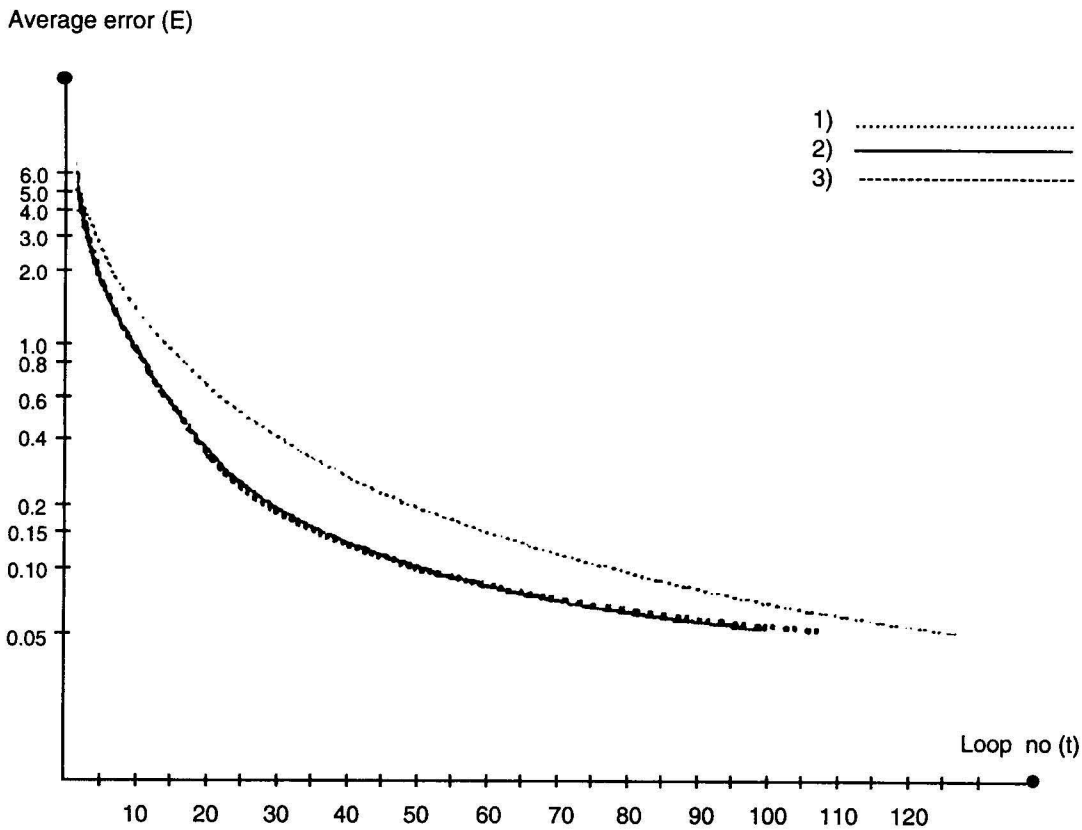
h) แสดงการฝึกสอนแพตเทิร์นตัวอักษร "Z" ในรอบที่ 126

รูปที่ 3. แสดงการฝึกสอนแพตเทิร์นตัวอักษร "A". . "Z"





รูปที่ 4. แสดงตัวอย่างการทดสอบการจำแนกแพตเทิร์นเมื่อข้อมูลอินพุตถูกบิดเบือน



รูปที่ 5. แสดงผลการฝึกสอน โดยกำหนดค่า η แตกต่างกันตามสูตร