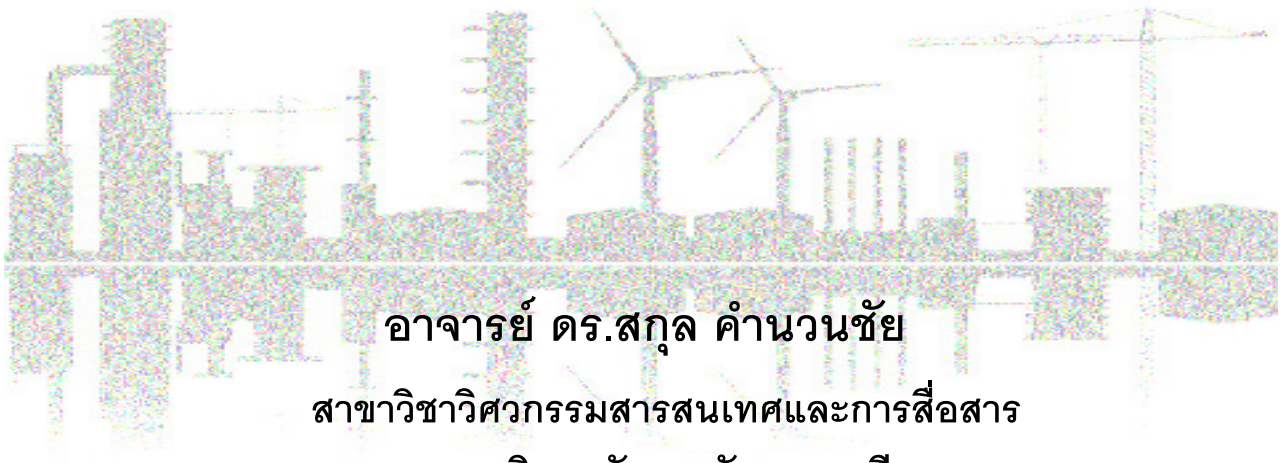
โครงข่ายประสาทเทียม

(Artificial Neural Network)



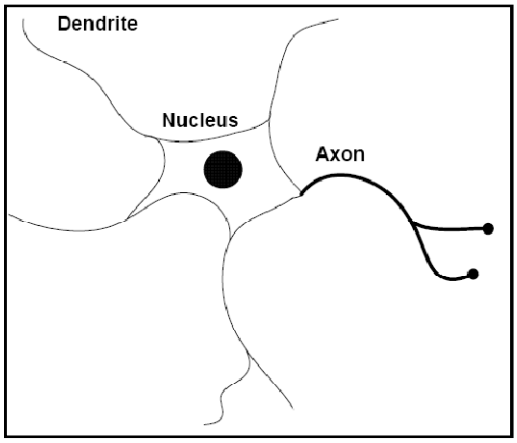
อาจารย์ ดร.สกุล คำนวนชัย

สาขาวิชาวิศวกรรมสารสนเทศและการสื่อสาร

มหาวิทยาลัยราชภัฏเทพสตรี

บทนำ

ปัจจุบันนักวิจัยให้ความเห็นตรงกันว่าโครงข่ายประสาทเทียมมีโครงสร้างเหมือนและแตกต่างจากโครงข่ายในสมองของมนุษย์ ในด้านที่ว่าโครงข่ายงานประสาทเทียม เป็นการรวมกลุ่มแบบขนานของหน่วยประมวลผลย่อยๆ และมีการเชื่อมต่อ เป็นส่วนสำคัญที่ทำให้เกิดความรู้ใหม่หรือสติปัญญาของโครงข่าย เมื่อเทียบกับขนาดของสมองมนุษย์ ที่มีขนาดใหญ่กว่าโครงข่ายประสาทเทียมอย่างมาก รวมทั้งเซลล์ประสาทยังมีความซับซ้อนกว่าหน่วยย่อยของโครงข่าย แต่โครงข่ายประสาทเทียมมีหน้าที่สำคัญเหมือนสมอง เช่นในด้านการเรียนรู้ ยังคงสามารถถูกจำลองขึ้นอย่างง่ายเหมือนโครงข่ายประสาทของมนุษย์ ดังรูปที่ 1

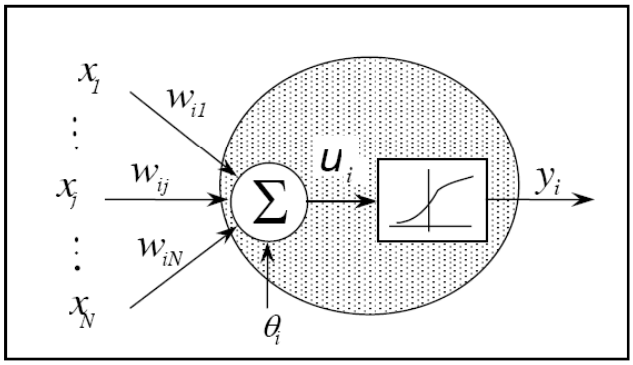


รูปที่ 1 แสดง Schematic ของ Biological Neuron

จากรูปที่ 1 แสดงเซลล์ประสาทของมนุษย์ 1 เซลล์ประสาท (neurons) ประกอบด้วย dendrite หลายๆ dendrite เพื่อใช้เป็นส่วนที่รับข้อมูลจาก เซลล์ประสาทอื่น เพื่อทำการตีความหรือประมวลผลได้ข้อมูล และมี 1 axon เพื่อใช้ส่งข้อมูลไปยัง เซลล์ประสาทอื่น

แบบจำลองเซลล์ประสาท (Neuron)

McCulloch-Pitts ได้เสนอแบบจำลองอย่างง่ายดังแสดงในรูปที่ 2



รูปที่ 2 รูปแบบอย่างง่าย McCulloch-Pitts 's Model [1]

$x = [x_1, x_2, \ldots, x_N]^T$ คือข้อมูลนำเข้าอินพุต

$w = [w, w, \ldots, w_N]^T$ คือค่าน้ำหนักความสำคัญที่ให้กับข้อมูลอินพุตแต่ละตัว

ฟังค์ชันของโครงข่าย (Network Functions) คือผลรวมทั้งหมดของผลคูณของข้อมูลนำเข้า กับค่าน้ำหนักความสำคัญที่ให้กับข้อมูลอินพุตแต่ละตัว บวกกับ biased term โดย ฟังค์ชันของโครงข่าย (Net Functions) อาจจะ map จากอินพุตไปเป็นเอาพุตต์แบบเป็นเส้นตรง หรือ ไม่ใช่เส้นตรงก็ได้ (Hyperplane)

$$u_i = \sum_{j=1}^{N} w_{ij} x_j + \theta_i = \mathbf{w}_i^T x + \theta_i = \begin{bmatrix} \theta_i & \mathbf{w}_i^T \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix}$$

ในการทำงานของโครงข่ายประสารทเทียมที่ประกอบด้วยอินพุต และเอาพุตต์ที่เหมือนกัน โดยจำลองให้อินพุตแต่ละอันมีค่าน้ำหนักเป็นตัวกำหนดน้ำหนักของอินพุต โดยนิวรอลแต่ละหน่วยจะมีค่าเทรดโฮรด (threshold) เป็นตัวกำหนดว่าให้ค่าน้ำหนักรวมของอินพุตมีขนาดเท่าใดจึงจะสามารถส่งเอาพทุไปยังนิวรอลอื่นได้ เมื่อนำนิวรอล แต่ละหน่วยมาต่อกันให้ทำงานร่วมกัน การทำงานนี้เปรียบเสมือนกับปฏิกิริยาเคมีที่เกิดในสมอง เพียงแต่ในโครงข่ายประสาทเทียมทุกอย่างเป็นตัวเลขเท่านั้น

การทำงานของโครงข่ายประสาทเทียมเป็นการนำอินพุตเข้ามายังโครงข่ายนำมาคูณกับค่าน้ำหนักความสำคัญของแต่ละขา ผลที่ได้จากอินพุตทุก ๆ ขาของโครงข่ายจะเอามารวมกันแล้วก็เอามาเทียบกับเทรดโฮรด ที่กำหนดไว้ ถ้าผลรวมมีค่ามากกว่าเทรดโฮรดแล้วนิวรอลก็จะส่งเอาพุตต์ออกไปเอาพุตต์นี้ก็จะถูกส่งไปยังอินพุตของนิวรอล อื่น ๆ ที่เชื่อมกันในโครงข่ายถ้าค่าน้อยกว่าเทรดโฮรดก็จะไม่เกิดเอาพุตต์ ซึ่งสามารถเขียนเป็นสมการได้ดังนี้

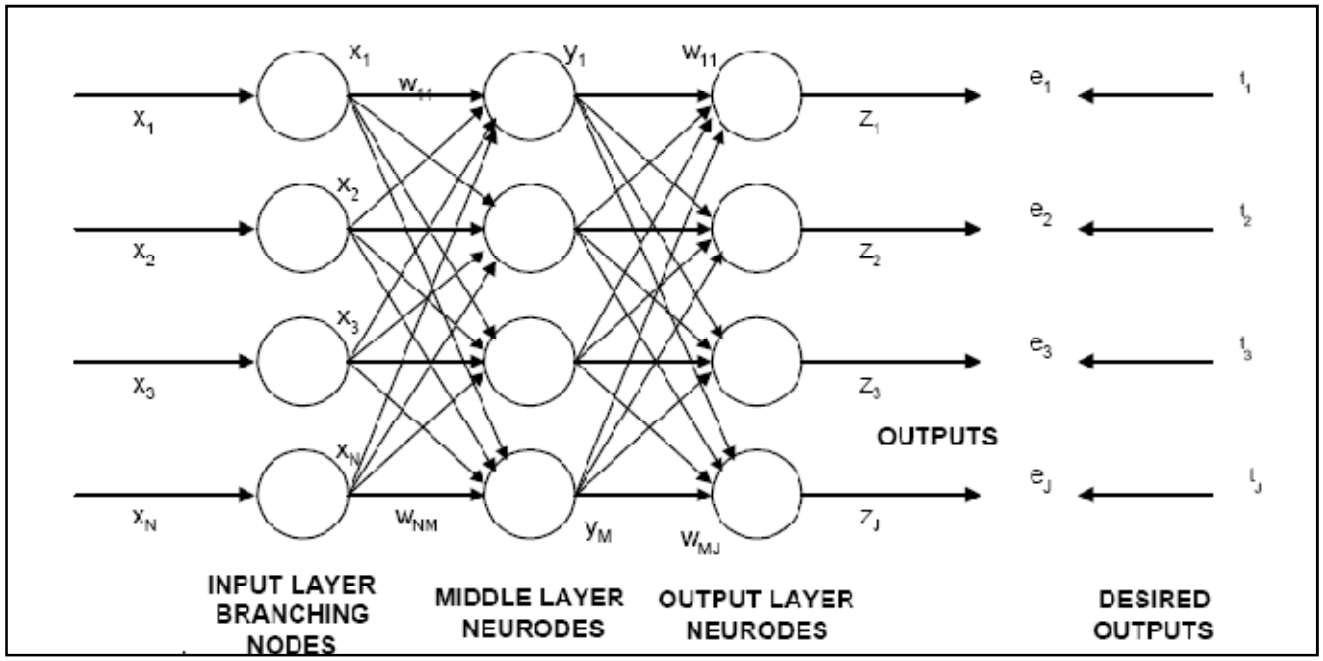
if (sum(input * weight) > threshold) then output

ค่าน้ำหนักความสำคัญ (weight) และเทรดโฮรด (threshold) ที่ต้องการเพื่อให้โครงข่ายประสาทเทียมรู้จำต้องเป็นค่าที่ไม่แน่นอน แต่สามารถกำหนดให้โครงข่ายปรับค่าเหล่านั้นได้โดยการสอนให้มันรู้จักรูปแบบ (pattern) ของสิ่งที่เราต้องการให้มันรู้จำเรียกว่า "back propagation" ซึ่งเป็นกระบวนการแพร่ย้อนกลับของการรู้จำในการฝึกแบบ feed-forward neural networks จะมีการใช้อัลกอริทึมแบบ back-propagation เพื่อใช้ในการปรับปรุงค่าน้ำหนักความสำคัญคะแนนของเครือข่าย (network weight) หลังจากใส่รูปแบบข้อมูลสำหรับการฝึก (training) ให้แก่โครงข่ายในแต่ละครั้งแล้ว ค่าที่ได้รับ (output) จากเครือข่ายจะถูกนำไปเปรียบเทียบกับผลที่คาดหวัง แล้วทำการคำนวณหาค่าความผิดพลาด ซึ่งค่าความผิดพลาดนี้จะถูกส่งกลับเข้าสู่เครือข่ายเพื่อใช้แก้ไขค่าน้ำหนักคะแนนต่อไป

การทดลอง

วิธีการเรียนรู้แบบแพร่ย้อนกลับ

โครงข่ายประสาทเทียมที่ได้ทำการทดลองเป็น แบบแพร่ย้อนกลับแบบพื้นฐานที่มีชั้นกลาง 1 ชั้น พร้อมตัวแปรต่างๆ แสดงในรูปที่ 3



รูปที่ 3 แสดงสถาปัตยกรรมของโครงข่ายประสาทเทียมแบบแพร่ย้อนกลับ

ในการทดลองได้กำหนดความหมายของตัวแปรต่าง ๆที่ใช้ แสดงตัวแปลต่าง ๆ พร้อมความหมายดังนี้

x_n = อินพุตโหนดที่ n มีทั้งหมด N โหนด

s_m = เอาต์พุตของชั้นซ่อน ก่อนทำการปรับค่า (activation) เป็น y_m

y_m = เอาต์พุตของชั้นซ่อน หลังทำการปรับค่าของโหนดที่ m มีทั้งหมด M โหนด

v_j = เอาต์พุตของชั้นเอาต์พุต ก่อนทำการปรับค่า (activation) เป็น z_j

z_j = ค่าเอาต์พุตที่ได้ทำการปรับค่าแล้วของชั้นเอาต์พุตโหนดที่ j มีทั้งหมด J โหนด

t_j = ค่าเอาต์พุตที่ต้องการที่ชั้นเอาต์พุตโหนดที่ j มีทั้งหมด J โหนด

w_{nm} = น้ำหนักของเส้นเชื่อมระหว่างชั้นอินพุต กับชั้นซ่อน

w_{mj} = น้ำหนักของเส้นเชื่อมระหว่างชั้นซ่อน กับชั้นเอาต์พุต

η = อัตราการเรียนรู้มีค่าอยู่ระหว่าง 0 ถึง 1

r = จำนวนรอบที่จะทำการเรียนรู้ มี R เป็นจำนวนรอบที่กำหนด

q = จำนวนชุดของข้อมูลตัวอย่าง มี Q เป็นตัวกำหนด

$e^{(q)}$ = ค่าผิดพลาดของข้อมูลตัวอย่าง

E = ค่าผิดพลาดรวมเฉลี่ยของข้อมูลตัวอย่าง

ขั้นตอนการเรียนรู้ของโครงข่ายประสาทเทียมแบบแพร่ย้อนกลับ แบ่งเป็นขั้นตอนดังต่อไปนี้

1. กำหนดจำนวนโหนดอินพุต (N), จำนวนโหนดเอาต์พุต (J), จำนวนโหนดของชั้นซ่อน (M), ข้อมูลอินพุต และข้อมูลเอาต์พุต ต่อจากนั้นทำการรับจำนวนรอบสูงสุดที่จะทำการเรียนรู้ (R) และค่าผิดพลาดที่ยอมรับได้
2. ตั้งค่าพารามิเตอร์ของอัตราการเรียนรู้ (η) ให้อยู่ในช่วง [0, 1]
3. สุ่มน้ำหนักเริ่มต้นให้กับทุกๆเส้นในโครงข่ายประสาทเทียมในทั้ง 2 ชั้น โดยให้มีค่าอยู่ระหว่าง [-0.5, 0.5]
4. รับค่าอินพุตของข้อมูลชุดแรก เพื่อใช้ในการคำนวณหาค่าเอาต์พุตของโครงข่ายประสาทเทียม
5. คำนวณค่าเอาต์พุตของชั้นซ่อน นำค่าอินพุตของชุดที่จะทำการคำนวณหาค่าเอาต์พุตของชั้นซ่อนออกมาแล้วจึงทำการคำนวณค่าเอาต์พุตของชั้นซ่อน ก่อนทำการปรับค่า (Activation) แล้วทำการปรับค่าเอาต์พุตของชั้นซ่อน ให้อยู่ในช่วง [0, 1] สำหรับแต่ละโหนดของชั้นซ่อน

ค่าเอาต์พุตของชั้นซ่อนก่อนทำการปรับค่า มีสมการดังนี้

$$S_m = \sum_{n=1}^{N} x_n * w_{nm}$$ สมการ (1)

ค่าเอาต์พุตของชั้นซ่อนหลังทำการปรับค่า มีสมการดังนี้

$$y_m = f(s_m)$$ สมการ (2)

ฟังก์ชั่นที่ใช้ปรับค่า f(x) มีสมการดังนี้

$$f(x) = \frac{1}{1+e^{-x}}$$ สมการ (3)

คำนวณค่าเอาต์พุตของชั้นเอาต์พุต ก่อนทำการปรับค่า แล้วทำการปรับค่าเอาต์พุตของชั้นเอาต์พุต ให้อยู่ในช่วง [0, 1] สำหรับแต่ละโหนดของชั้นเอาต์พุต ค่าเอาต์พุตของชั้นเอาต์พุตก่อนทำการปรับค่า มีสมการดังนี้

$$v_j = \sum_{m=1}^{M} y_m * w_{mj}$$ สมการ (4)

หาค่าความผิดพลาดและปรับน้ำหนัก นำเอาต์พุตที่ได้ กับ เอาต์พุตที่ได้กำหนดไว้ มาหาค่าความผิดพลาดของข้อมูล ถ้าค่าผิดพลาดของข้อมูลน้อยกว่าค่าผิดพลาดที่ยอมรับได้ ทำการรับข้อมูลชุดต่อไป ถ้าไม่ใช่ปรับน้ำหนักแล้วทำการรับข้อมูลของชุดถัดไปแล้วจึงกลับไปทำ **คำนวณค่าเอาต์พุตของชั้นซ่อน** แต่ถ้าเป็นข้อมูลชุดสุดท้ายทำหา **ค่าผิดพลาดรวมเฉลี่ย** ค่าความผิดพลาดในแต่ละชุดของข้อมูลตัวอย่าง มีสมการดังนี้

$$z_j = f(v_j)$$ สมการ (5)

การปรับน้ำหนัก มีสมการดังนี้

$$e^{(q)} = \frac{1}{2}\sum_{j=1}^{J}(t_j - z_j)^2$$ สมการ (6)

หาค่าผิดพลาดรวมเฉลี่ย นำค่าผิดพลาดของชุดข้อมูลแต่ละชุดมารวมกัน แล้วทำการหาค่าเฉลี่ย เพื่อใช้ในการตรวจสอบว่าผลลัพธ์ของทุกๆ ข้อมูลในแต่ละรอบนั้นมีค่าน้อยกว่าค่าผิดพลาดที่ยอมรับได้ในทุกๆ ข้อมูลหรือไม่ ถ้าใช่แสดงว่าโครงข่ายประสาทเทียมสามารถให้ผลลัพธ์ที่ถูกต้องของทุกๆข้อมูลแล้วจบการเรียนรู้ ถ้าไม่ใช่กลับไปทำ

รับค่าอินพุตของข้อมูลชุดแรก

$$w_{mj}^{(r+1)} = w_{mj}^{(r)} + \eta\{(t_j^{(q)} - z_j^{(q)}) * [z_j^{(q)}(1 - z_j^{(q)})] * y_m^{(q)}\}$$ สมการ (7)

$$w_{nm}^{(r+1)} = w_{nm}^{(r)} + \eta\{\sum_{j=1}^{J}(t_j^{(q)} - z_j^{(q)})[z_j^{(q)}(1 - z_j^{(q)})]w_{mj}^{(r)}\} * [y_m^{(q)}(1 - y_m^{(q)})][x_n^{(q)}]$$ สมการ (8)

ค่าผิดพลาดรวมเฉลี่ย

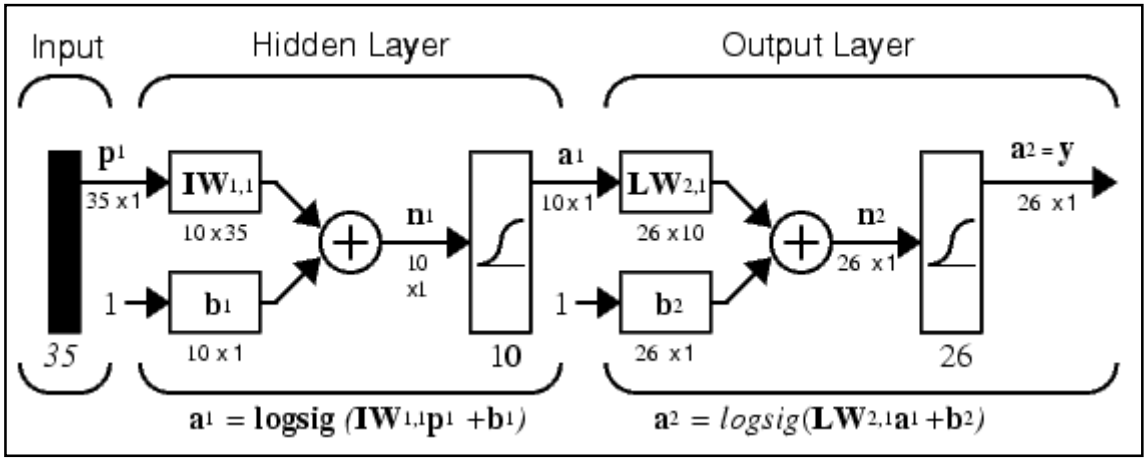
$$E = \frac{1}{Q}\sum_{q=1}^{Q} e^{(q)}$$ สมการ (9)

วิธีการทดลอง

โครงข่ายประสาทเทียม การออกแบบและได้รับการฝึกฝนให้รู้จัก 26 ตัวอักษรของตัวอักษร เป็นภาพถ่ายดิจิตอล ตัวอักษรแต่ละตัวเป็นตัวแทนในฐานะ 5x7 พิกเซล โดยมีองค์ประกอบต่างๆ ดังนี้

การทดลองใช้รูปภาพตัวอักษรที่มีขนาด 5x7 พิกเซล ถ้าทำการนำทุกพิกเซลมาเป็นอินพุทจะทำให้ขนาดของนิวรอล มีจำนวน เท่ากับ 35 เวคเตอร์อิลิเมนต์ ใช้ในการชี้กับเวคเตอร์เอาต์พุตของตัวอักษร จำนวน 26 อิลิเมนต์ โดยถ้ามีผลการทำงานที่ถูกต้องจะตอบสนองค่าเป็น 1 ของลำดับของตัวอักษร จากจำนวน 26 ตัวอักษร ส่วนตำแหน่ง เอาต์พุทให้การเป็น 0

โดยมีโครงสร้างของโครงข่ายประสาทเทียมกำหนดให้มีชั้นอินพุตจำนวน 35 นิวรอล และ 26 นิวรอล เป็นส่วนของชั้นเอาต์พุต ใช้ในการระบุตัวอักษรที่ระบบการรู้จำตัวอักษร โดยระบบที่ใช้ในการชี้ตัวอักษร เป็นแบบ log-sigmoid และ log-sigmoid โดยระบบ sigmoid ฟังก์ชัน เป็นฟังก์ชันที่ทำค่าระหว่างช่วงเอาท์พุท (0 ถึง 1) คือการเรียนรู้ที่สมบูรณ์แบบสำหรับการส่งออกค่าในรูปของบูลีน โดยมีสถาปัตยกรรมของโครงข่ายประสาทเทียมดังรูปที่ 4



รูปที่ 4 แสดงสถาปัตยกรรมในการทดลอง

จากสถาปัตยกรรมในการทดลอง ชั้นซ่อนจำนวน 10 นิวรอล และทำเริ่มเตรียมระบบด้วยฟังก์ชัน

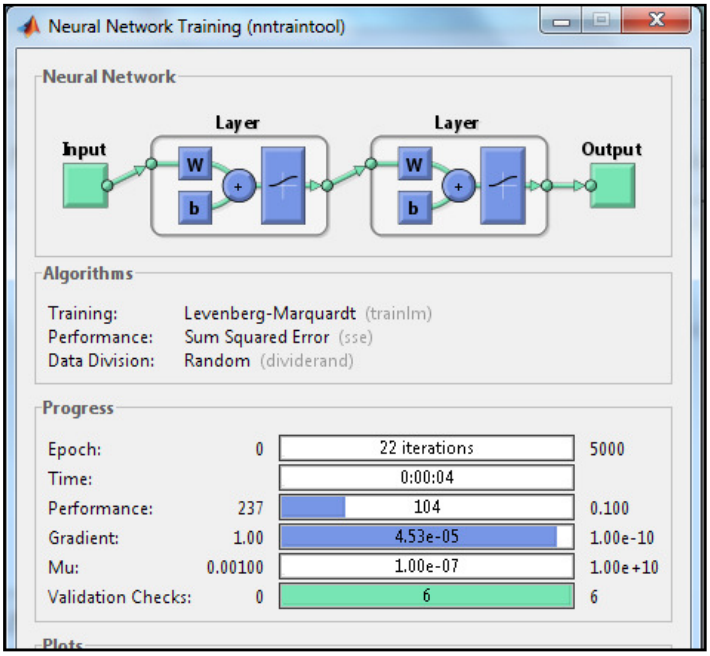
net = newff(alphabet,targets,10,{'logsig','logsig'});

ในการฝึกสอน (train) ระบบให้โครงข่ายในการรู้จำทั้งสองรูปแบบ แบบที่ไม่มีการรบกวน (without noise) กับแบบมีการใส่ส่วนของการรบกวน (with noise) โดยการฝึกสอนด้วยแบบการแพร่ย้อนกลับ (backpropagation) และใช้ส่วนเพิ่มแบบ (adaptive learning) ด้วยฟังก์ชั่น trainbpx.

โดยการ Training Without Noise กำหนดค่าการ train สูงสุด 5,000 epochs หรือจนกระทั่ง ค่า sum squared error อยู่ที่ 0.1 โดยค่าที่ทำการกำหนดให้ระบบเป็นดังนี้

```
P = alphabet;
T = targets;
net.performFcn = 'sse';
net.trainParam.goal = 0.1;
net.trainParam.show = 20;
net.trainParam.epochs = 5000;
net.trainParam.mc = 0.95;
[net,tr] = train(net,P,T);
```

ได้ผลการทดลองดังรูปที่ 5

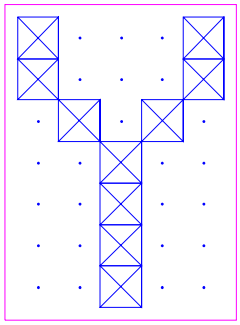


ดังรูปที่ 5 แสดงการ train ด้วยฟังก์ชัน nntraintool

โดยการสร้างสัญญาณรบกวน ตัวอย่างตัวอักษร J ได้ด้วยฟังก์ชันด้านล่าง

```
noisyJ = alphabet(:,10)+randn(35,1) * 0.2;
plotchar(noisyJ);
A2 = sim(net,noisyJ);
A2 = compet(A2);
answer = find(compet(A2) == 1);
plotchar(alphabet(:,answer));
```

แสดงตัวอย่างผลการสร้างสัญญาณรบกวนของตัวอักษร J ดังรูปที่ 6

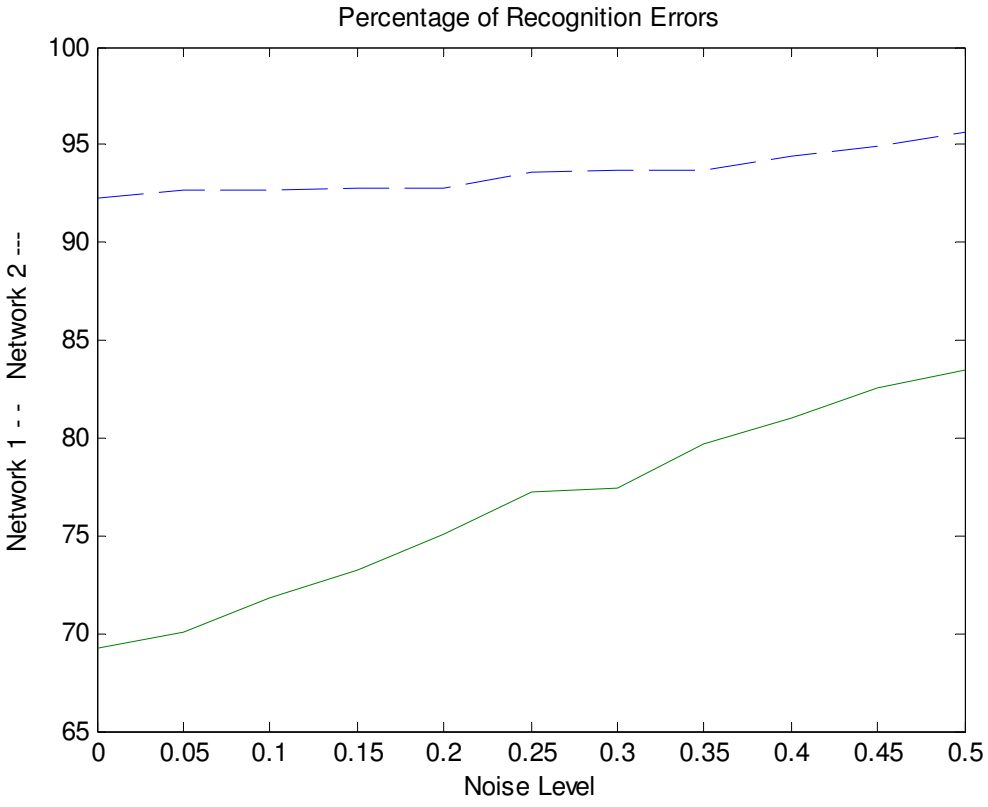


รูปที่ 6 แสดงผลการรบกวนของตัวอักษร J

โดยการ Training with Noise กำหนดค่าการ train สูงสุด 300 epochs หรือจนกระทั่ง ค่า sum squared error อยู่ที่ 0.6 โดยค่าที่ทำการกำหนดให้ระบบเป็นดังนี้

```
netn = net;
netn.trainParam.goal = 0.6;
netn.trainParam.epochs = 300;
T = [targets targets targets targets];
P1 = P;
[R,Q] = size(P);
for pass = 1:10
  P = [P1,P1,(P1 + randn(R,Q)*0.1),(P1 + randn(R,Q)*0.2)];
  [netn,tr] = train(netn,P,T);
end
```

ซึ่งจากการทดลองได้ผลการทดลองดังรูปที่ 7



รูปที่ 7 แสดงกราฟเปรียบเทียบผลการรู้จำตัวอักษร

จากกราฟรูปที่ 7 แสดงถึงการการทำงานของระบบ โครงข่ายประสาทเทียม Network 1, trained without noise จะเกิดผลการผิดพลาดมากกว่า การฝึกสอนที่มี noise ดังกราฟ Network 2, ซึ่งเป็น trained with noise

บรรณานุกรม

[1] Patterson, Dan W., 1930. Artificial Neural Network : Theory and Application. Includes Bibliographical Reference and Index ISBN 0-13-295353-6 (pbk.)

[2] Patterson, Dan W., 1930. Artificial Neural Network : Theory and Application. Includes Bibliographical Reference and Index ISBN 0-13-295353-6 (pbk.)

[3] Patterson, Dan W., 1930. Artificial Neural Network : Theory and Application. Includes Bibliographical Reference and Index ISBN 0-13-295353-6 (pbk.)

[4] MATLAB(R), MATLAB(R) Compiler(TM), and other MATLAB family products U.S. Patent Nos. 6,857,118; 6,973,644; 6,993,772; 7,010,364; 7,051,333; 7,051,338; 7,096,154; 7,139,686; 7,165,253; 7,181,745; 7,228,239; 7,231,631; 7,237,237

[5] Simulink(R), Real-Time Workshop(R), and other Simulink family products U.S. Patent Nos. 4,796,179; 5,133,045; 5,612,866; 6,868,526; 7,010,744; 7,020,850; 7,089,223; 7,139,686; 7,139,687; 7,165,253; 7,167,817; 7,170,433; 7,178,112; 7,246,056; 7,275,026; 7,292,964; 7,296,054; 7,313,449; 7,319,941