บทที่ 3
ทฤษฎีเกี่ยวกับข่ายงานนิวรัล

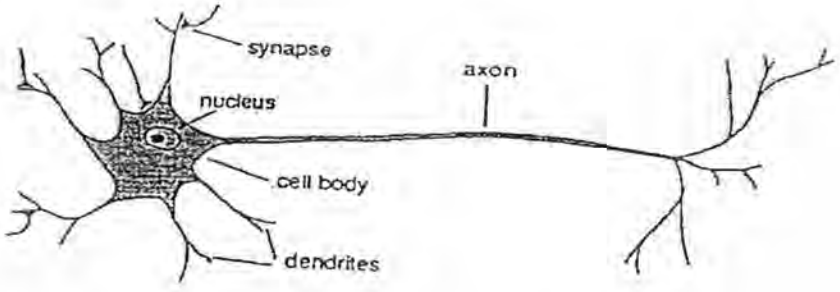
บทนี้จะกล่าวถึงทฤษฎีเกี่ยวกับข่ายงานนิวรัลโดยแบ่งหัวข้อเป็นดังนี้คือ หัวข้อ 3.1 จะกล่าวถึงชีววิทยาของนิวรัลและแบบจำลองของข่ายงานนิวรัลในยุคแรกๆ ว่ามีแนวคิดของข่ายงานนิวรัลมาจากสิ่งใด และหัวข้อ 3.2 จะกล่าวถึงแบบจำลองนิวรัลที่ใช้ในปัจจุบัน ต่อมาในหัวข้อ 3.3 กล่าวถึงการออกแบบข่ายงานนิวรัล ซึ่งต้องมีองค์ประกอบพื้นฐานคือ ข้อมูลที่ใช้ในการฝึกข่ายงาน และความรู้เกี่ยวกับการสร้างแบบจำลอง ซึ่งข้อมูลที่ใช้ในการฝึกข่ายงานนิวรัลจะใช้ข้อมูลจำนวนมากและต้องมีความแตกต่างกันของข้อมูลโดยจะต้องมีการจัดเตรียมข้อมูลที่ดี ส่วนความรู้กระบวนการที่จะสร้างแบบจำลองจะเป็นเครื่องมือผลักดันในแต่ละขั้นตอนของการสร้างแบบจำลองให้สำเร็จ โดยจะอธิบายถึงวิธีการสร้างแบบจำลองเริ่มตั้งแต่โครงสร้างของแบบจำลอง , การจำแนกการเรียนรู้ , การจำแนกข่ายงานนิวรัล จะแบ่งเป็น 2 ประเภทคือข่ายงานนิวรัลแบบมีการชี้นำและข่ายงานนิวรัลแบบไม่มีการชี้นำ , วิธีการฝึกข่ายงาน และปัญหาที่พบในการฝึกข่ายงาน ในบทนี้จะเน้นการเรียนรู้ไปที่อัลกอริธึมแบบการกระจายความผิดพลาดย้อนกลับชนิด Levenberg-Marquardt เนื่องจากเป็นอัลกอริธึมที่ใช้สำหรับงานวิจัยในครั้งนี้

3.1 ชีววิทยาของนิวรัลและแบบจำลองของข่ายงานนิวรัลในยุคแรกๆ

ระบบประสาทของมนุษย์ได้รับสัญญาณอินพุทจากหลายแหล่งและกระบวนการทำงานของระบบประสาทจะทำหน้าที่ในการแปลงสัญญาณอินพุทให้เป็นสัญญาณการตอบสนองที่เหมาะสม โดยจะมีนิวรัลเป็นล้านๆนิวรัลที่เชื่อมต่อกันที่เรียกว่าข่ายงานนิวรัล (neural network) เพื่อสร้างสัญญาณตอบสนองหรือพฤติกรรมต่างๆในการดำเนินชีวิตประจำวันของมนุษย์ ดังนั้นมีผู้วิจัยและพัฒนาแบบจำลองได้แนวคิดสำหรับข่ายงานนิวรัลโดยมีพื้นฐานมาจากการเลียนแบบโครงสร้างระบบประสาทของมนุษย์ (human neurons system) นั้นคือระบบประสาทของมนุษย์ประกอบด้วยกลุ่มของนิวรัลประมาณ 10^{11} เซล โดยสามารถติดต่อกับนิวรัลอื่นๆด้วยแอกซอน (axon) และซินแนพส์ (synapes) ซึ่งความหนาแน่นของซินแนพส์ประมาณ 10^{4} หน่วยต่อนิวรัล ข้อสมมติฐานเกี่ยวกับแบบจำลองของระบบประสาทคือนิวรัลจะทำการติดต่อกันโดยอาศัยการกระตุ้นทางไฟฟ้า (electrical impulse) แต่การทำงานของนิวรัลนั้นเกิดจากกระบวนการทางชีวเคมี ซึ่งช่วงเวลาในการประมวลผลของนิวรัล (neuron switch time) จะมีค่าน้อยกว่าหนึ่งในพันวินาทีซึ่งน้อยกว่าช่วงเวลาที่กระแสไฟฟ้าวิ่งผ่านคอมพิวเตอร์ถึงหนึ่งล้านเท่าและนิวรัลยังคงสามารถเชื่อมโยงกันได้ไวกว่าซูเปอร์คอมพิวเตอร์ในปัจจุบันถึงหนึ่งพันเท่า

ในทางชีววิทยานิวรัลมีองค์ประกอบ 3 ส่วนดังแสดงในรูปที่ 3.1

1) เดนไดรท์ (dendrite) ที่แตกแขนงออกจากตัวเซลเพื่อรับสัญญาณจากแอกซอนของนิวรัลตัวอื่นเข้ามาและส่งต่อมายังเซลนิวรัล
2) เซลนิวรัล (cell body) ทำหน้าที่รวบรวมสัญญาณจากเดนไดรท์และส่งไปให้แก่แอกซอน
3) แอกซอน (axon) ทำหน้าที่นำสัญญาณที่ออกจากเซลนิวรัลส่งไปยังเดนไดรท์ของนิวรัลอื่นๆ

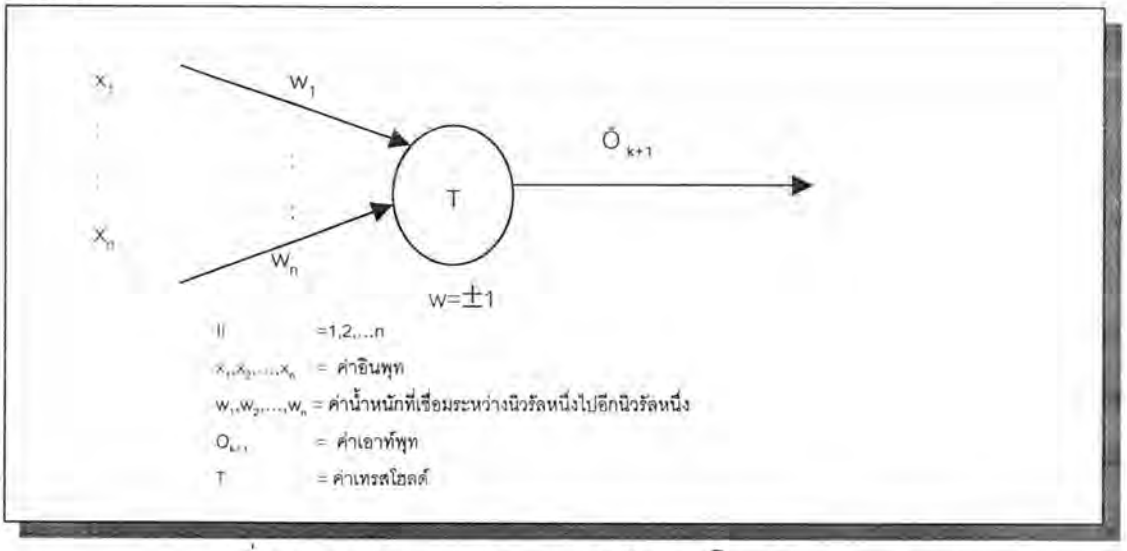


รูปที่ 3.1 โครงสร้างของเซลประสาท (neuron)

สำหรับซินแนพส์นั้นเป็นรอยต่อระหว่างแอกซอนของนิวรัลปัจจุบันกับเดนไดรท์ของนิวรัลอื่นๆนั้น คือเดนไดรท์จะรับสัญญาณจากแอกซอนของนิวรัลอื่นเข้ามาและส่งต่อให้เซลนิวรัล เซลนิวรัลจะรวมสัญญาณที่ได้ส่งต่อไปให้แอกซอนจนถึงรอยต่อซินแนพส์ โดยแอกซอนจะนำสัญญาณผ่านไปตามลำดับของคลื่นกระแสไฟฟ้า ซึ่งขึ้นกับค่าแรงดันต่างศักย์ของนิวรัลและเนื้อเยื่อของนิวรัล (membrane) ที่สร้างแรงดันต่างศักย์กระจายผ่านแอกซอนไปจนถึงรอยต่อซินแนพส์ จากแนวคิดพื้นฐานดังกล่าวนี้มีนักวิจัยได้สนใจที่จะศึกษากลไกและโครงสร้างของระบบประสาทของมนุษย์ซึ่งนำไปสู่การพัฒนาแบบจำลองสำหรับการแก้ปัญหาที่ซับซ้อนได้แก่ ปัญหาแบบไม่เชิงเส้น โดยเริ่มต้นจากในปี ค.ศ. 1943 McCulloch และ Pitts ได้เป็นผู้ริเริ่มในการจำลองลักษณะการทำงานของเซลประสาทสมองโดยใช้สมการคณิตศาสตร์อย่างง่าย ซึ่งมีข้อจำกัดคือ สามารถจัดการเฉพาะกับปัญหาที่เป็นลักษณะทางตรรกศาสตร์อย่างง่าย เช่น ปัญหาทางตรรกศาสตร์ Nor OR และ AND เป็นต้น และจากจุดเริ่มต้นนี้ได้มีการพัฒนาแบบจำลองดังกล่าวจนเป็นที่ยอมรับและนิยมใช้กันอย่างแพร่หลายในเวลาต่อมา ดังแสดงในรูปที่ 3.2 เป็นการเสนอโครงสร้างข่ายงานนิวรัลของ McCulloch และ Pitts โดยอยู่ในรูปแบบจำลองการคำนวณโดยการสร้างตรรกเทร์สโฮลด์ (threshold logic) อย่างง่ายๆ แต่แบบจำลองมีข้อจำกัดคือค่าอินพุทและเอาท์พุทเป็น 0 กับ 1 เท่านั้น และค่าน้ำหนักกำหนดให้มีค่าคงที่ รวมทั้งนิวรัลมีเพียงหน่วยเดียวเท่านั้นทำให้ไม่มีการติดต่อกันระหว่างนิวรัลหน่วยอื่น สัญญาณที่ออกจากนิวรัลนั้นได้ถูกกำหนดเงื่อนไขดังสมการ (3.1)

$$O_{k+1} = \left\{ \begin{array}{ll} 1 & \text{if} \quad \sum_{i=1}^{n} w_i x_{ik} \geq T \\ 0 & \text{if} \quad \sum_{i=1}^{n} w_i x_{ik} < T \end{array} \right\} \qquad \ldots(3.1)$$

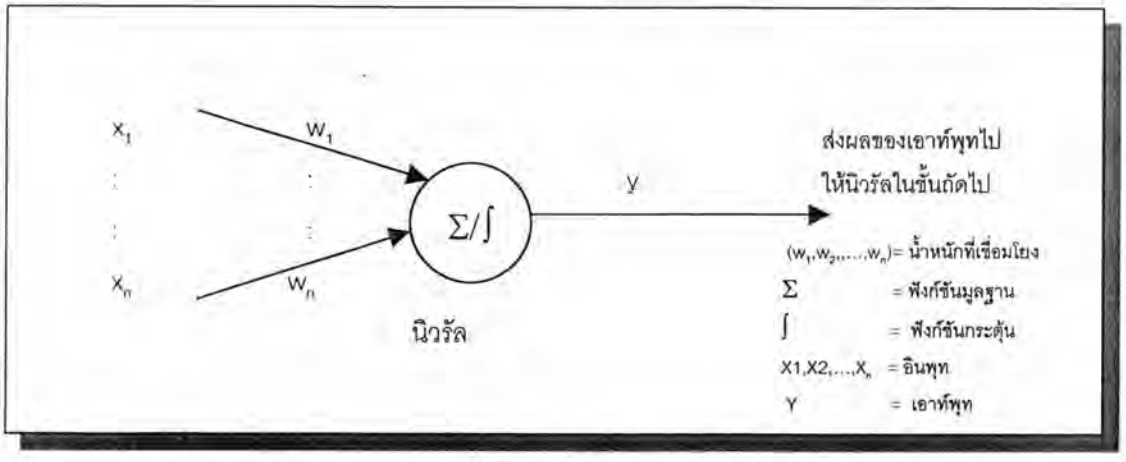
และในปี ค.ศ 1958 Rosenblatt ได้เสนอแบบจำลองเปอร์เซปตรอนซึ่งเป็นแบบจำลองข่ายงานแบบตัดสินใจเพื่อแยกข้อมูล 2 ระดับออกจากกัน ต่อมาใน ค.ศ. 1969 Minsky และ Papert ทำการพิสูจน์คุณสมบัติของเปอร์เซปตรอนและชี้ให้เห็นถึงข้อจำกัดของแบบจำลองนี้ซึ่งสามารถแก้ปัญหาเฉพาะระบบที่เป็นเชิงเส้นเท่านั้น



รูปที่ 3.2 แบบจำลองการทำงานของนิวรัลเสนอโดย McCulloch และ Pitts

3.2 แบบจำลองของข่ายงานนิวรัลในปัจจุบัน (Artificial Neural Network)

แบบจำลองของข่ายงานนิวรัลในปัจจุบันประกอบด้วยชั้นของนิวรัลและส่วนเชื่อมโยงระหว่างนิวรัลหนึ่งไปอีกนิวรัลหนึ่งเรียกว่าน้ำหนัก (weight) อินพุทของนิวรัลจะถูกนำมารวมกันโดยฟังก์ชันมูลฐาน (basis function) และส่งเอาท์พุทที่ออกจากฟังก์ชันมูลฐานไปแปลงด้วยฟังก์ชันกระตุ้น (activation function) ซึ่งจะได้เอาท์พุทของนิวรัลและส่งเอาท์พุทของนิวรัลนี้ไปเป็นอินพุทของนิวรัลในชั้นถัดไป ดังนั้นองค์ประกอบและหน้าที่ของนิวรัลที่เลียนแบบการทำงานของระบบประสาทมนุษย์จะมีดังต่อไปนี้



รูปที่ 3.3 หน้าที่และการทำงานของนิวรัล

3.2.1 องค์ประกอบและหน้าที่ของนิวรัล

องค์ประกอบของนิวรัลประกอบด้วย 3 ส่วนที่สำคัญคือ

1. หน่วยประมวลผล (processing elements) เรียกว่า node
2. การเชื่อมต่อแบบซินแนพส์ (synaptic connection) เป็นส่วนเชื่อมต่อระหว่างนิวรัลในแต่ละชั้นสำหรับการส่งข้อมูลที่ประมวลผลได้จากนิวรัลหนึ่งไปยังอีกนิวรัลหนึ่ง
3. ค่าน้ำหนัก (weight) และค่าไบแอส (bias) ทำหน้าที่ในการขยายหรือลดขนาดของสัญญาณที่เข้าสู่นิวรัลโดยผ่านการเชื่อมต่อแบบซินแนพส์

ข่ายงานนิวรัลแต่ละนิวรัลจะมีการเชื่อมโยงระหว่างชั้นจากนิวรัลหนึ่งไปสู่อีกนิวรัลหนึ่งโดยใช้ค่าน้ำหนัก และในนิวรัลหนึ่งๆ จะมีสภาวะภายในในการแปลงค่าอินพุทที่ได้รับเข้ามาให้เป็นเอาท์พุทโดยการใช้ฟังก์ชันมูลฐานและฟังก์ชันกระตุ้น ซึ่งค่าเอาท์พุทของนิวรัลนี้จะส่งเป็นอินพุทให้กับนิวรัลในชั้นถัดไปดังรายละเอียดต่อไปนี้

แต่ละนิวรัล y จะได้รับอินพุทที่ส่งมาจากชั้นก่อนหน้านี้ ซึ่งอินพุทคือ $x_1, x_2, ..., x_n$ และค่าน้ำหนักที่เชื่อมระหว่าง x_i กับ y คือ w_i

วิธีการหาเอาท์พุทของนิวรัล y คือ ผลรวมของอินพุทคูณกับน้ำหนัก ถ้ามีเทอมไบแอส (b) ก็สามารถนำเทอมไบแอสรวมด้วย

$$y_{in} = b+w_1x_1+w_2x_2+w_nx_n \qquad ...(3.2)$$

และนำค่า y_{in} คำนวณผ่านฟังก์ชันกระตุ้นเช่นฟังก์ชันกระตุ้นแบบซิกมอยด์คือ logistic sigmoid function

$$\text{เอาท์พุท} = y = f(y_{in}) = 1/(1+\exp(-y_{in})) \qquad ...(3.3)$$

ถ้าเป็นฟังก์ชันกระตุ้นแบบเชิงเส้น

$$\text{เอาท์พุท}= y = f(y_{in})=y_{in} \qquad ...(3.4)$$

เอาท์พุท y นี้ก็จะส่งไปเป็นเป็นอินพุทให้กับนิวรัลในชั้นถัดไป

สรุปหน้าที่ของนิวรัลแต่ละหน่วยซึ่งแสดงในรูป 3.3 มีดังนี้

1. รับสัญญาณจากนิวรัลหน่วยอื่นๆ (ทำงานคล้ายเดนไดรท์)
2. รวมสัญญาณจากนิวรัลหน่วยอื่นเข้าด้วยกันโดยใช้ฟังก์ชันมูลฐาน (ทำงานคล้ายเซลนิวรัล)
3. แปลงสัญญาณที่รวมได้โดยใช้ฟังก์ชันกระตุ้น (ทำงานคล้ายเซลนิวรัล)
4. ส่งผลลัพธ์ที่ได้จากฟังก์ชันกระตุ้นไปยังนิวรัลถัดไป (ทำงานคล้ายแอกซอน)

3.2.2 พารามิเตอร์ที่มีความสำคัญสำหรับการทำงานของนิวรัล

3.2.2.1 น้ำหนัก (weight)

น้ำหนักหรือที่เรียกว่าสัญญาณเชื่อมโยงระหว่างนิวรัลจะถูกปรับเปลี่ยนค่าอยู่ตลอดเวลา ในระหว่างการเรียนรู้ข่ายงานสร้างแบบจำลองภายในขึ้นมาเพื่อให้มีค่าใกล้เคียงกับระบบที่สนใจ แบบจำลองภายในที่สร้างขึ้นมาจะสามารถใช้แทนแบบจำลองของกระบวนการได้ก็ต่อเมื่อผลลัพธ์ที่ออกจากระบบมีค่าใกล้เคียงกับผลลัพธ์ที่ออกจากข่ายงาน นั้นคือ ความแตกต่างของผลลัพธ์ที่ออกจากระบบกับผลลัพธ์ที่ออกจากข่ายงานต้องมีค่าน้อยที่สุดอยู่ในช่วงที่ยอมรับได้

3.2.2.2 ฟังก์ชันมูลฐาน (basis function)

ฟังก์ชันมูลฐานคือการแทนการแมพพิงก์ (mapping) ทางคณิตศาสตร์ด้วยฟังก์ชัน $u(w,x)$ เมื่อ w คือ แมทริกซ์ของน้ำหนักและ x คืออินพุทเวคเตอร์ ฟังก์ชันมูลฐานทำหน้าที่ในการรวมสัญญาณที่ได้รับมาจากนิวรัลหน่วยอื่นๆเข้าด้วยกัน และฟังก์ชันกระตุ้นจึงแปลงสัญญาณนี้เป็นเอาท์พุทของนิวรัล ฟังก์ชันมูลฐานสามารถแบ่งเป็น 2 ประเภท ดังแสดงในรูป 3.4 คือ ฟังก์ชันมูลฐานเชิงเส้นและฟังก์ชันมูลฐานรัศมี

ก. ฟังก์ชันมูลฐานเชิงเส้น (linear basis function)

ฟังก์ชันมูลฐานเชิงเส้นเป็นฟังก์ชันชนิดไฮเปอร์แพลน (hyperplane) โดยเป็นฟังก์ชันมูลฐานอันดับที่ 1 ค่า net ที่ได้เป็นผลรวมเชิงเส้นของค่าอินพุทซึ่งแสดงในสมการ (3.5)

$$u_i(w,x) = \sum_{j=1}^{n} w_{ij} x_j \qquad ...(3.5)$$

ข. ฟังก์ชันมูลฐานรัศมี (radial basis function RBF)

ฟังก์ชันมูลฐานรัศมีเป็นฟังก์ชันชนิดไฮเปอร์สเฟียร์ (hypersphere) โดยเป็นฟังก์ชันมูลฐานอันดับที่ 2 (ไม่เชิงเส้น) ค่า net ที่ได้จะแทนระยะทางไปยังชุดข้อมูลอ้างอิง (reference pattern)

$$u_i(w,x) = \sqrt{\sum_{i=1}^{n}(x_i - w_{ij})^2} \qquad ...(3.6)$$

นอกจากนี้ฟังก์ชันอันดับที่ 2 ยังสามารถขยายไปเป็นฟังก์ชันมูลฐานอิลิปติค (eliptic basis funciton) ได้อีกด้วย

(ก) ฟังก์ชันมูลฐานเชิงเส้น linear(hyperplane) (ข) ฟังก์ชันมูลฐานรัศมี radial (hypersphere)

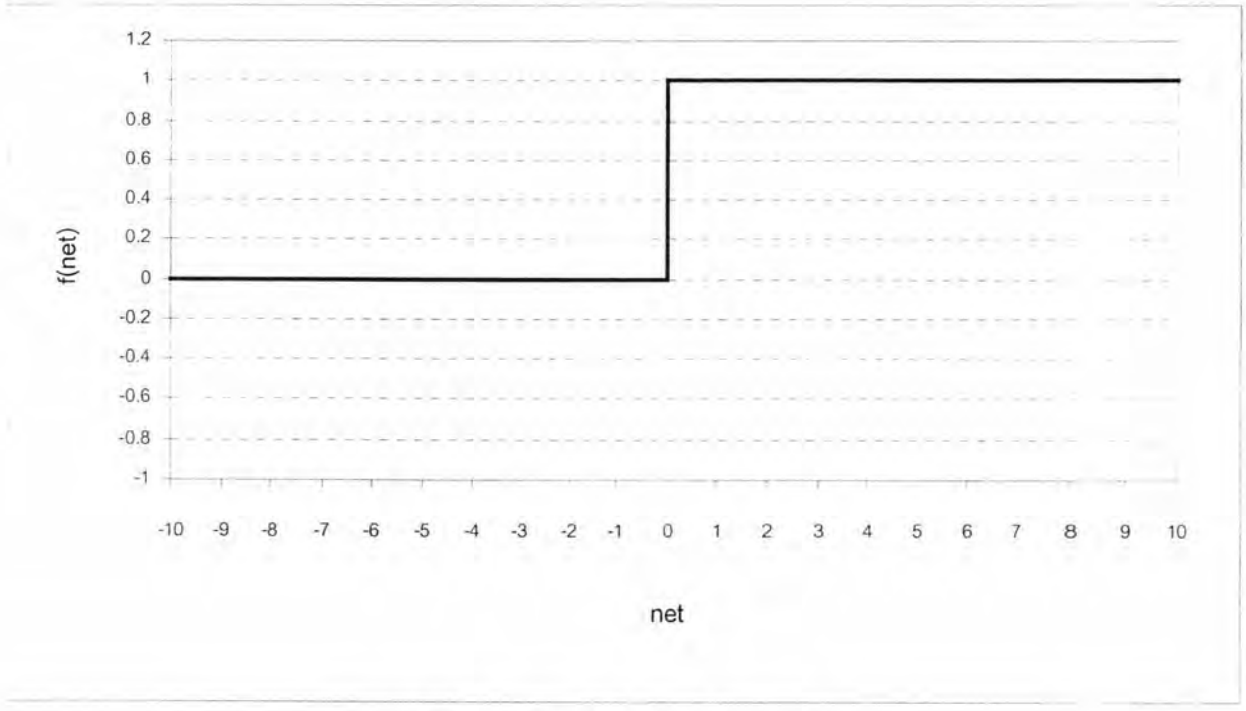
รูปที่ 3.4 ฟังก์ชันมูลฐาน

3.2.2.3. ฟังก์ชันกระตุ้น (activation function)

ค่า net ที่อยู่ในรูปของฟังก์ชันมูลฐาน u(w,x) จะถูกแปลงค่าโดยใช้ฟังก์ชันกระตุ้นไม่เชิงเส้น (nonlinear activation function) ฟังก์ชันที่ส่วนใหญ่นิยมใช้ทั่วไป คือฟังก์ชันขั้นบันได(Step Function) , ฟังก์ชันแรมป์ (Ramp Function) , ซิกมอยด์ (Sigmoid Function) , ไบโพลาร์ (Bipolar Function), ฟังก์ชันเกาส์เสียน (Gaussian Function)

ก. ฟังก์ชันสเต็พ (Step function)

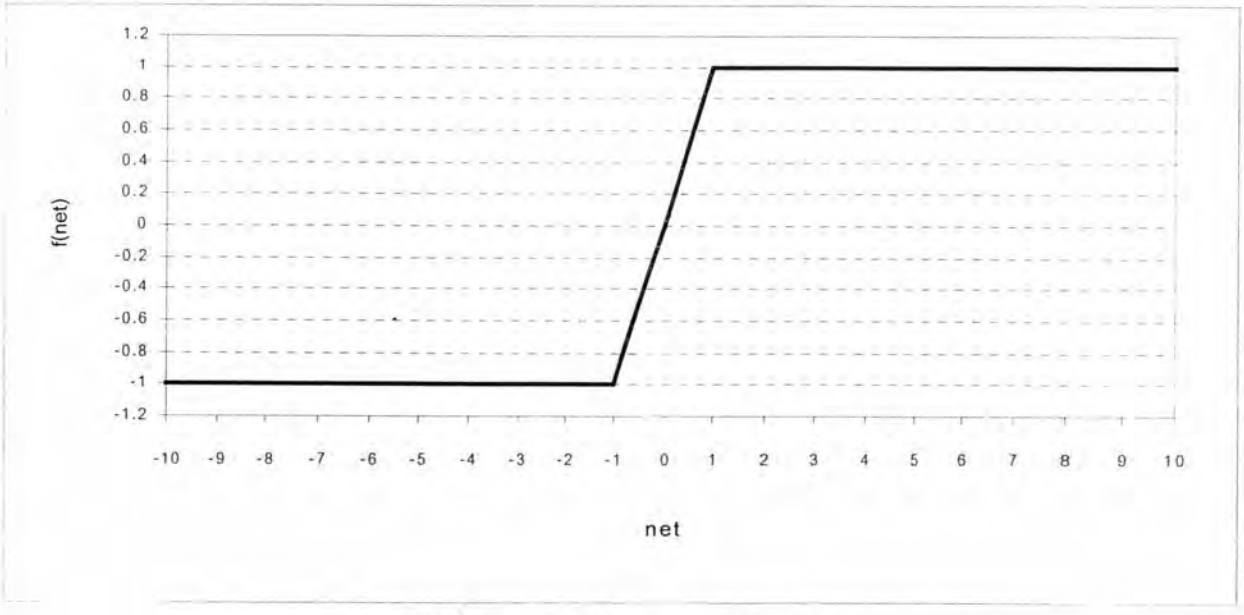
$$f(net) = \left\{ \begin{array}{l} 1 \text{ if net} > 0 \\ 0 \text{ otherwise} \end{array} \right\} \qquad ...(3.7)$$



(ก) ฟังก์ชันแบบสเต็พ (Step Function)

ข. ฟังก์ชันแบบแรมพ์ (Ramp Function)

$$f(net) = \begin{cases} 1 & \text{if } net \geq 1 \\ net & \text{if } |net| < 1 \\ -1 & \text{if } net \leq -1 \end{cases} \qquad \ldots(3.8)$$



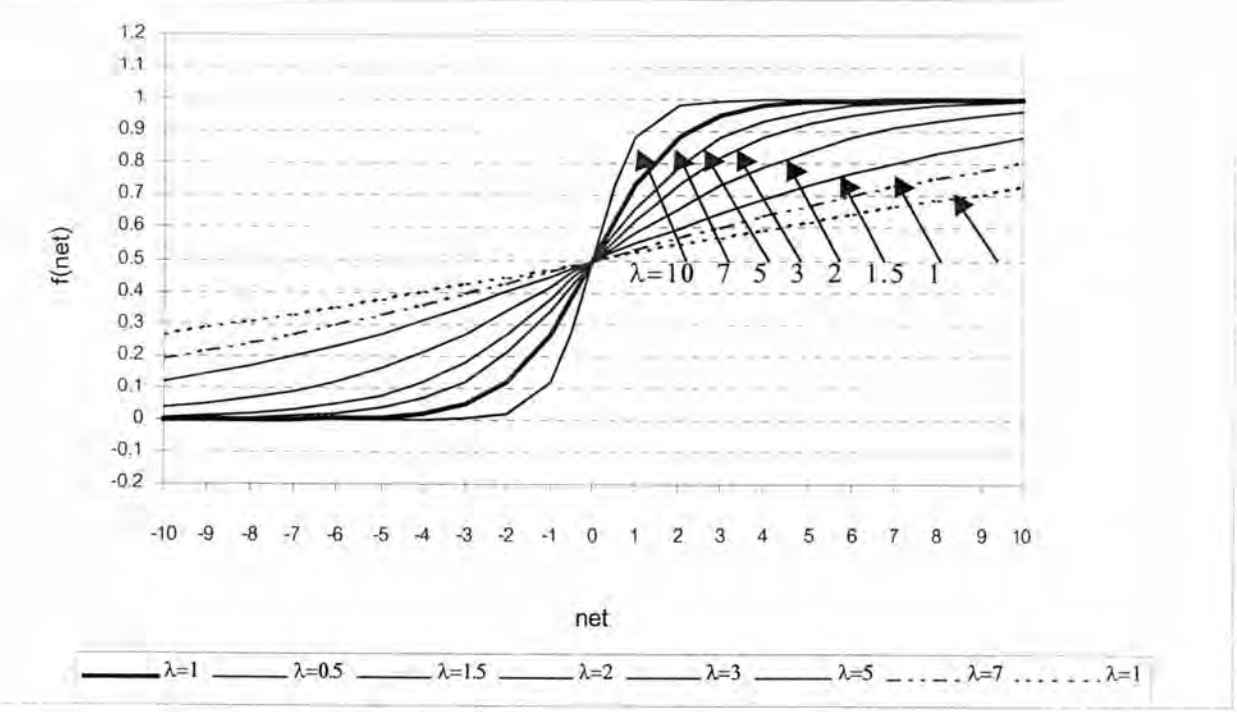
(ข) ฟังก์ชันแบบแรมพ์ (Ramp Function)

รูปที่ 3.5 ฟังก์ชันกระตุ้น (Activation Funciton)

ค. ฟังก์ชันซิกมอยด์ (Sigmoid Function)

$$f(net) = \frac{1}{1 + e^{-\frac{net}{\lambda}}} \quad ...(3.9)$$

ส่วนใหญ่แล้วฟังก์ชันกระตุ้นที่นิยมคือฟังก์ชันซิกมอยด์ซึ่งจะมีการแปลงค่าอินพุทที่มีค่าบวกมากๆ ให้เข้าใกล้ 1 หรือค่าลบมากๆ ให้ใกล้ 0 ดังสมการ 3.9 ดังนั้นช่วงของฟังก์ชันซิกมอยด์จะมีค่าระหว่าง [0,1]



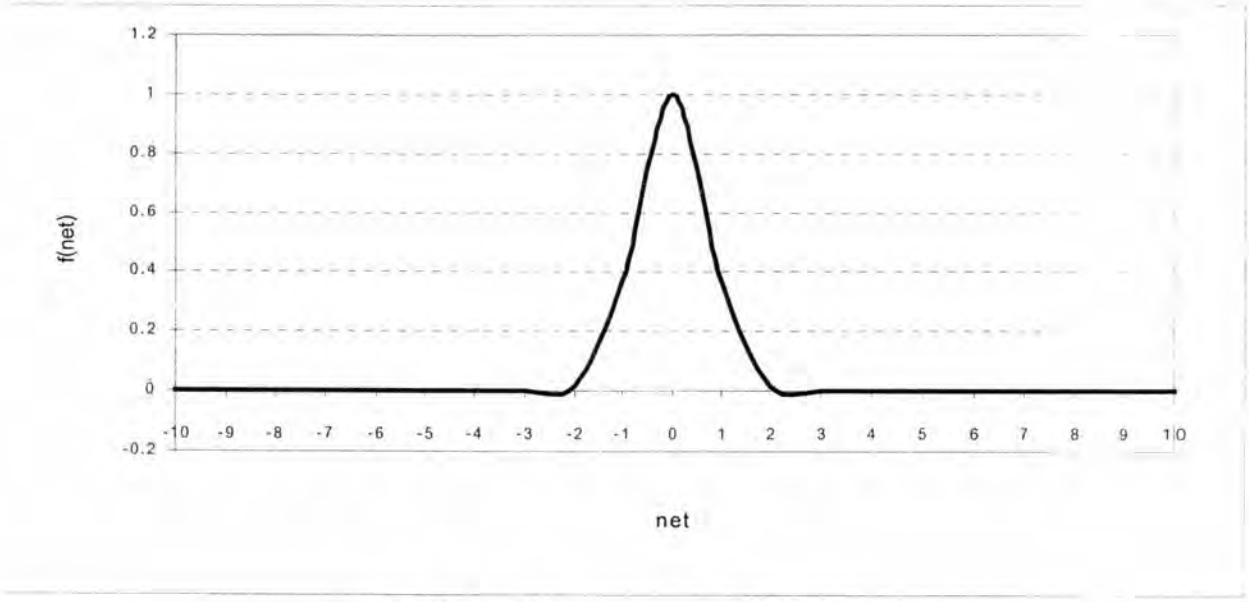
(ค) ฟังก์ชันซิกมอยด์ (Sigmoid Function)

รูปที่ 3.5 (ต่อ) ฟังก์ชันกระตุ้น (Activation Funciton)

ง. ฟังก์ชันเกาส์เสียน (Gaussian Function)

$$f(net) = ce^{-\frac{net^2}{\lambda^2}} \quad ...(3.10)$$

ฟังก์ชันเกาส์เสียนนี้มีค่ามากสุดอยู่ที่ 1 ส่วนใหญ่จะใช้กับอัลกอริธึมแบบข่ายงานรัศมี (radial basis function network) สำหรับฟังก์ชันอื่นที่ใช้กันแต่ไม่นิยมมากนักได้แก่ ฟังก์ชันไบโพลาร์, ฟังก์ชันอาร์คแทนเจนท์ , ฟังก์ชันไบโพลาร์อาร์คแทนแจนท์



(ง) ฟังก์ชันเกาส์เสียน (Gaussian Function)

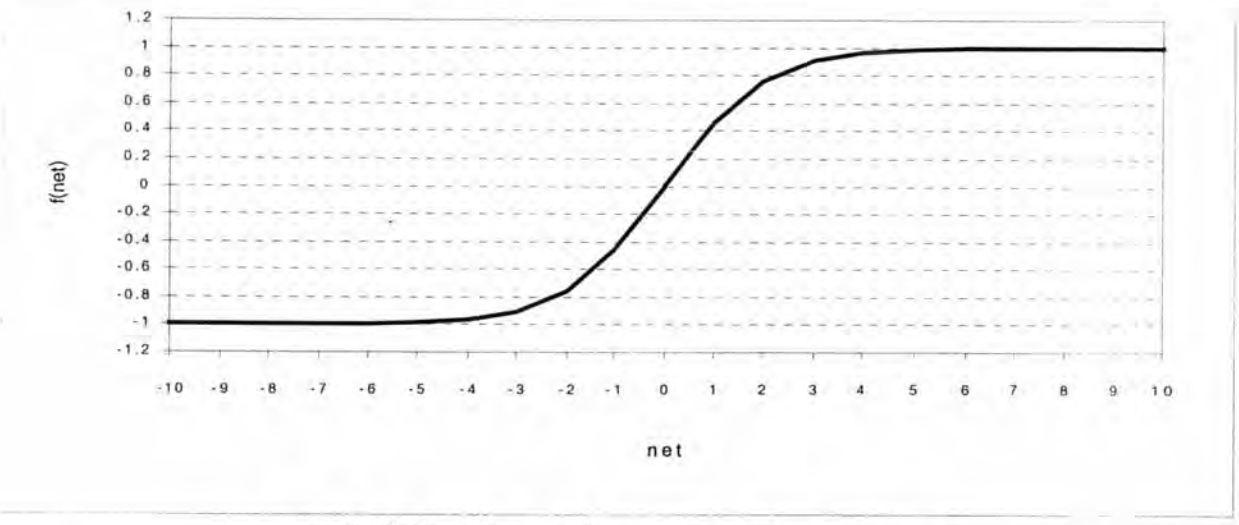
จ. *ฟังก์ชันไบโพลาร์ซิกมอยด์ (Bipolar Sigmoid Function)*

ฟังก์ชันไบโพลาร์มีสมการดังนี้

$$f(net) = \frac{1}{1 + \exp^{(-\lambda net)}} - 1 \qquad ...(3.11)$$

ฟังก์ชันไบโพลาร์ซิกมอยด์มีสมการดังนี้

$$f(net) = -1+2/(1+\exp(-net)) \qquad ...(3.12)$$



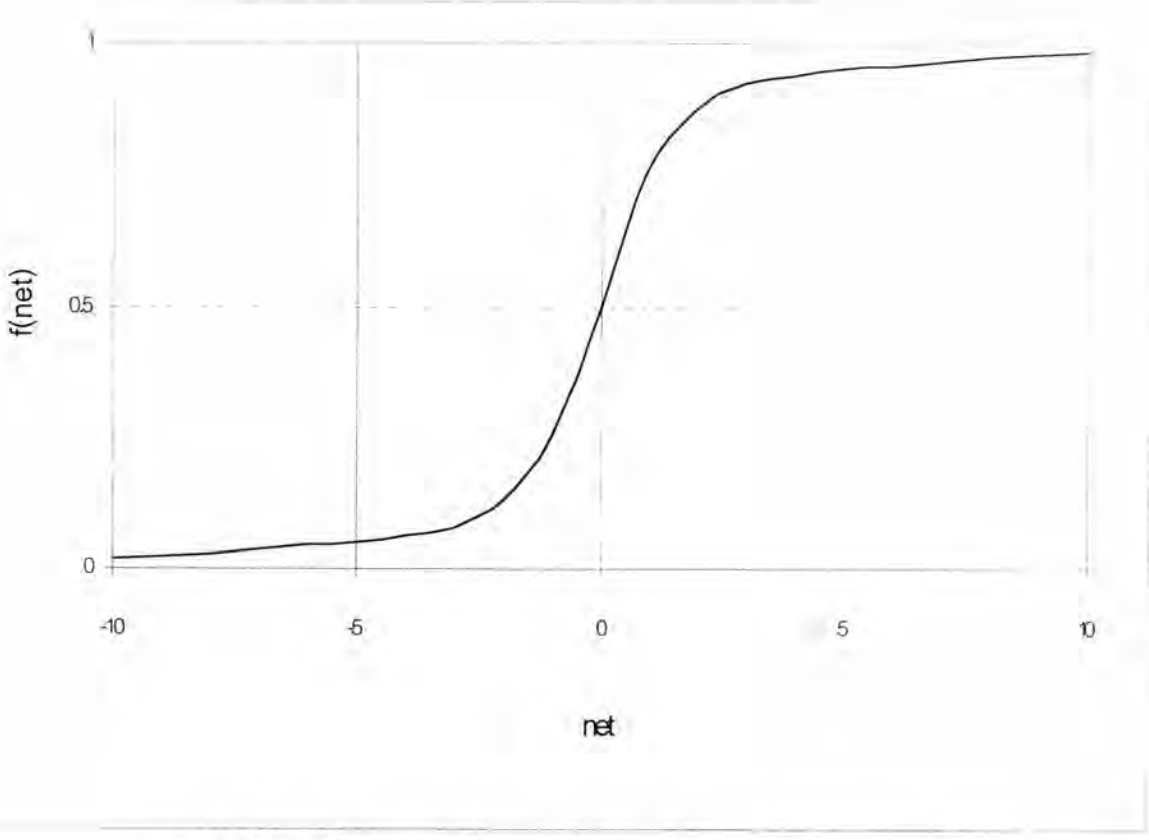
(จ) ฟังก์ชันไบโพลาร์ซิกมอยด์ (Bipolar Sigmoid Function)

รูปที่ 3.5 (ต่อ) ฟังก์ชันกระตุ้น (Activation Funciton)

ฉ. *ฟังก์ชันกระตุ้นแบบอาร์คแทนเจนท์ (Arc Tangent Transfer Function)*

$$f(net) = \arctan(net/3.14159)+0.5 \quad ...(3.13)$$

ฟังก์ชันกระตุ้นนี้มีค่าอยู่ระหว่าง [0,1] ข้อดีของฟังก์ชันนี้ คือเป็นทางเลือกหนึ่ง สำหรับฟังก์ชันซิกมอยด์ เมื่อต้องการปรับค่าน้ำหนักที่มีค่ามาก



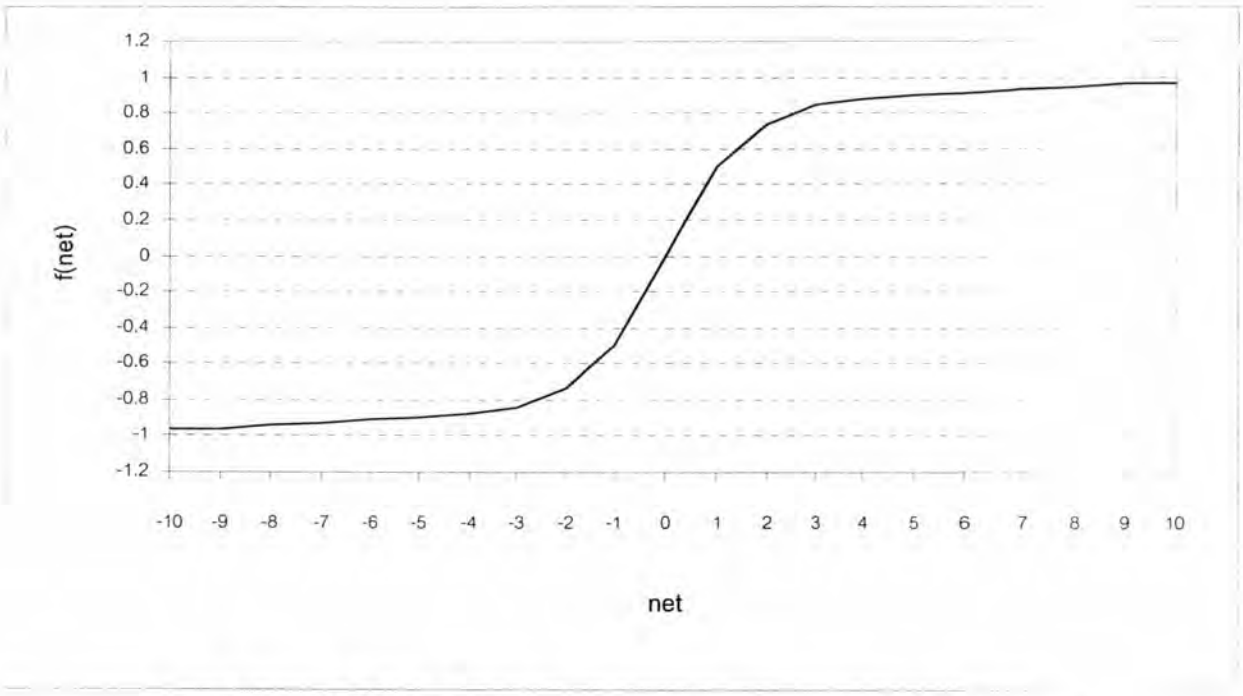
(ฉ) ฟังก์ชันกระตุ้นแบบอาร์คแทนเจนท์ (Arc Tangent Transfer Function)

รูปที่ 3.5 (ต่อ) ฟังก์ชันกระตุ้น (Activation Funciton)

ช. *ฟังก์ชันกระตุ้นแบบไบโพลาร์อาร์คแทนเจนท์ (Bipolar Arc Tangent Transfer Function)*

$$f(net) = \arctan((2.0*net)/3.14159) \quad ...(3.14)$$

ฟังก์ชันนี้ให้ค่าเอาท์พุทอยู่ในช่วง [-1,1] ข้อดีจะเหมือนกับแบบฟังก์ชันอาร์คแทน

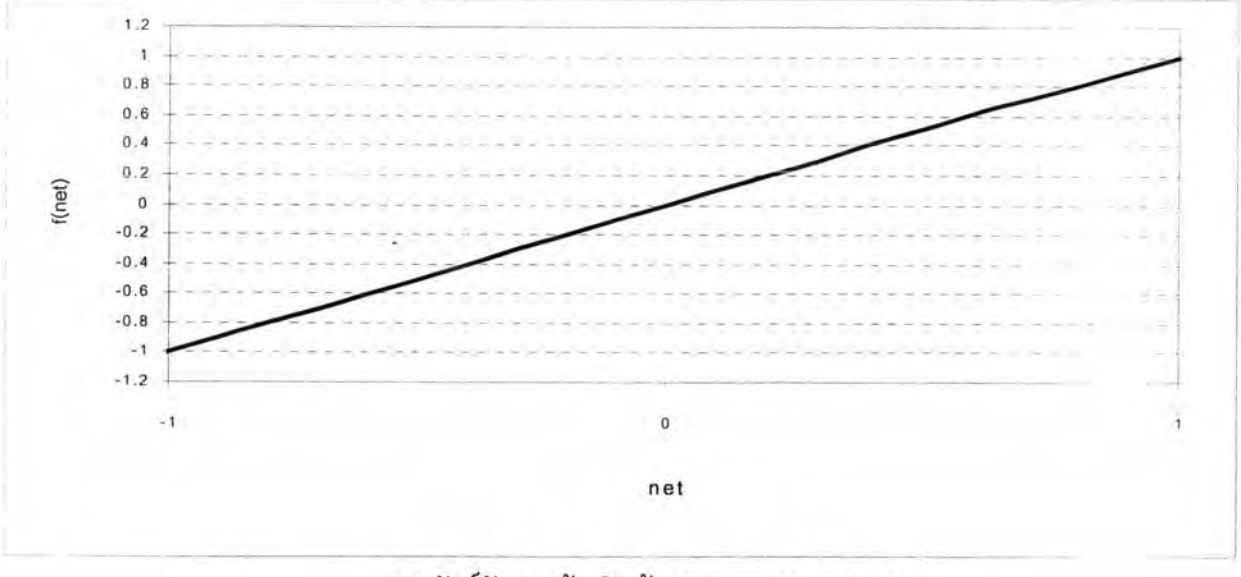


(ช) ฟังก์ชันกระตุ้นไบโพลาร์อาร์คแทน (Bipolar Arc Tangent Transfer Function)

ซ. ฟังก์ชันกระตุ้นแบบเชิงเส้น (Linear Transfer Function)

$$f(net) = net \qquad ...(3.15)$$

ฟังก์ชันนี้จะให้ค่าผลลัพธ์เท่ากับค่าอินพุท ฟังก์ชันนี้ปกติจะถูกใช้ในชั้นเอาท์พุท



(ซ) ฟังก์ชันกระตุ้นเชิงเส้น (Linear Function)

รูปที่ 3.5 (ต่อ) ฟังก์ชันกระตุ้น (Activation Funciton)

เหตุผลที่ต้องใช้ฟังก์ชันกระตุ้น คือต้องการทำให้ข่ายงานเป็นแบบไม่เชิงเส้น ซึ่งถ้าเป็นแบบจำลองเชิงเส้นก็สามารถใช้ข่ายงานแบบเปอร์เซบตรอน (perceptron ที่ไม่มีชั้นซ่อน)และฟังก์ชันกระตุ้นแบบเชิงเส้นได้ แต่ถ้าต้องการแบบไม่เชิงเส้นต้องมีฟังก์ชันกระตุ้นในการแทนความไม่เป็นเชิงเส้น นั้นคือการทำข่ายงานแบบหลายชั้น และมีฟังก์ชันกระตุ้นที่ไม่เป็นเชิงเส้นเช่น ซิกมอยด์ ได้แก่ logistic, tanh และ เกาส์เสียน โดยปกติจะใช้บ่อย ซึ่งถ้าเป็นฟังก์ชัน tanh จะให้ค่าเอาท์พุท +/- จะช่วยให้สามารถฝึกข่ายงานได้เร็วกว่าค่าบวกอย่างเดียว ส่วนฟังก์ชัน logistic และ tanh ต้องทำการสเกลช่วงของเป้าหมายให้อยู่ในช่วงเอาท์พุทหรือช่วงเอาท์พุทให้อยู่ในช่วงเป้าหมาย(การสเกลช่วงคือการคูณด้วยค่าคงที่ที่เหมาะสม) แต่ถ้าเป้าหมายไม่มีขอบเขต ฟังก์ชันกระตุ้นควรจะเป็นฟังก์ชันกระตุ้นที่ไม่มีขอบเขต (unbounded activation function) ถ้าค่าเป้าหมายเป็นค่าบวกแต่ไม่ทราบค่าสูงสุดควรใช้ฟังก์ชันกระตุ้นแบบเอ็กโปเนนเชียล (exponential activation function) แต่ควรระวังการเกิดเอาท์พุทมีค่ามากเกินไป (overflow)

3.2.2.4 ฟังก์ชันความผิดพลาด (error function หรือ loss function)

เป้าหมายของการฝึกข่ายงานคือการลดค่าความผิดพลาดรวมของข่ายงานให้น้อยที่สุด ดังนั้นการเลือกชนิดความผิดพลาดจะมีอิทธิพลต่อความสำเร็จในการฝึกข่ายงาน ฟังก์ชันความผิดพลาดนี้เป็นการวัดค่าความแตกต่างของค่าเป้าหมายกับค่าเอาท์พุทที่ได้จากข่ายงาน ซึ่งจะมีหลายประเภทดังตัวอย่างต่อไปนี้

ก. ความผิดพลาดรวมกำลังสอง (sum square error)

ส่วนใหญ่แล้วจะใช้ค่าผลรวมของความผิดพลาดกำลังสอง (sum square error)

$$E(y,p)=(y-p)^2 \qquad ..(3.16)$$

ซึ่ง E คือความผิดพลาด , y คือค่าเอาท์พุท และ p คือค่าเป้าหมาย

ถ้าเป็นความผิดพลาดของทั้งชุดข้อมูลก็นำมารวมกันจะได้เป็นผลรวมความผิดพลาดยกกำลังสอง หรือ ความผิดพลาดรวม (sum square error, SSE หรือ total error)

ข. ค่าความผิดพลาดเฉลี่ยกำลังสอง (Mean Square Error)

ในบางครั้งอาจใช้ค่าเฉลี่ยของค่าความผิดพลาดยกกำลังสอง (mean square error (MSE), average square error (ASE)) โดยค่าเฉลี่ยของความผิดพลาดกำลังสองใช้กับการแบ่งประเภทความผิด (ใช้กับข้อมูลไบนารี) การใช้ค่าเฉลี่ยของความผิดพลาดยกกำลังสองแทนความผิดพลาดรวมกับอัลกอริธึมการ

กระจายความผิดพลาดย้อนกลับ ซึ่งจะให้อัตราการเรียนรู้ไม่ค่อยรวดเร็วกับชุดข้อมูลในการฝึกได้ แต่ถ้าเป็นค่าผิดพลาดรวมต้องใช้ค่าอัตราการเรียนรู้น้อยมากสำหรับชุดข้อมูลการฝึกที่มีขนาดใหญ่

ให้ $z_m = (x_i, y)$, $i=1, ..., m$ ซึ่ง m คือลำดับของชุดข้อมูลฝึกข่ายงาน

W = ค่าน้ำหนักในการเชื่อมต่อข่ายงาน

g(x,W) = เอาท์พุทของฟังก์ชันข่ายงาน

ค่าความผิดพลาดเฉลี่ยกำลังสอง (Mean Square Error) สามารถคำนวณได้ดังสมการต่อไปนี้

$$m\text{-}1((y_1-g(x_1,W))^2+(y_1-g(x_1,W))^2+...+(y_m-g(x_m,W))^2) \quad ...(3.17)$$

ค. ค่าความผิดพลาดเฉลี่ยสัมบูรณ์ (Mean Absolute Error)

สมการแสดงค่าความผิดพลาดเฉลี่ยสัมบูรณ์แสดงดังสมการต่อไปนี้

$$m\text{-}1(|y_1-g(x_1,W)|+|y_1-g(x_1,W)|+..+|y_m-g(x_m,W)|) \quad ...(3.18)$$

เป็นการลดค่าความผิดพลาดของข้อมูลที่มีสัญญาณรบกวน ค่าที่ได้รับการฝึกจะมีค่าเบี่ยงเบนจากปกติ ซึ่งค่าความผิดพลาดชนิดนี้ออกแบบให้ละเลยหรือลดผลกระทบนี้ให้น้อยที่สุดซึ่งค่าความผิดพลาดเฉลี่ยกำลังสองจะมีผลกระทบมากเนื่องจากเกิดการยกกำลังสอง ค่าที่ผิดพลาดจำนวนเล็กน้อยก็จะทำให้การเรียนรู้ผิดพลาดเนื่องจะผลรวมจะไปเพิ่มมากขึ้นในค่าความผิดพลาดรวม

ง. ค่าความผิดพลาดเฉลี่ยกำลังสี่ (Mean Fourth Power Error)

สมการแสดงค่าความผิดพลาดกำลังสี่แสดงดังในสมการต่อไปนี้

$$m\text{-}1((y_1-g(x_1,W))^4+(y_1-g(x_1,W))^4+...+(y_m-g(x_m,W))^4) \quad ...(3.19)$$

ส่วนใหญ่จะใช้กำลังสี่เมื่อต้องการเน้นความสำคัญของค่าความผิดพลาดให้มีขนาดใหญ่ ซึ่งวิธีนี้เป็นการคำนวณให้อัลกอริธึมเรียนรู้ในกรณีที่ค่าความผิดพลาดมีค่ามากที่สุด

จ. ค่าความผิดพลาดไฮเปอร์โบลิกกำลังสอง (Hyperbolic Square Error)

ค่าความผิดพลาดแบบไฮเปอร์โบลิกกำลังสองคือฟังก์ชันค่าความผิดพลาดที่พัฒนาโดย Scott Fahlman เป็นวิธีที่ปรับปรุงประสิทธิภาพของการฝึกข่ายงานแบบการกระจายความผิดพลาดย้อนกลับ ค่าความผิดพลาดในรูปแบบเดิมคือ E จะถูกคำนวณเป็นค่าผลต่างระหว่างค่าเอาท์พุทกับค่าเป้าหมาย แต่วิธีนี้ E จะมีสมการดังต่อไปนี้ โดยค่าความผิดพลาดแบบไฮเปอร์โบลิกจะใช้คู่กับฟังก์ชันกระตุ้นชนิดที่ไม่เป็นไบโพลาร์ (non-bipolar transfer function)

$$E= \log((1+E)/(1-E)) \qquad ...(3.20)$$

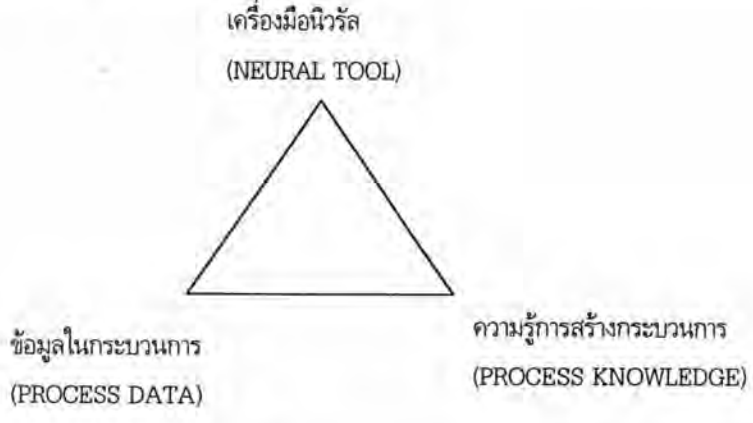
ฉ . ค่าความผิดพลาดไบโพลาร์ไฮเปอร์โบลิกกำลังสอง (Bipolar Hyperbolic Square Error)

ค่าความผิดพลาดไบโพลาร์ไฮเปอร์โบลิกจะคล้ายกับค่าความผิดพลาดไฮเปอร์โบลิกกำลังสอง แต่ชนิดนี้จะใช้กับฟังก์ชันกระตุ้นชนิดไบโพลาร์ (bipolar transfer function) เท่านั้น ดังสมการต่อไปนี้

$$E= \log((2+E)/(2-E)) \qquad ...(3.21)$$

3.3 การออกแบบข่ายงานนิวรัล

การออกแบบข่ายงานนิวรัลที่ดีต้องมีองค์ประกอบพื้นฐาน 2 อย่างคือ ข้อมูลที่ใช้ในการฝึกข่ายงานนิวรัล (process data) และ ความรู้เกี่ยวกับกระบวนการสร้างแบบจำลอง (process knowledge) ดังแสดงในรูป 3.6 ถ้าต้องการแบบจำลองข่ายงานนิวรัลที่ดีควรมีข้อมูลที่ใช้ในการฝึกข่ายงานนิวรัลที่สร้างจากฐานข้อมูล (data-based model) ส่วนความรู้กระบวนการที่จะสร้างแบบจำลองจะเป็นเครื่องมือผลักดันให้แต่ละขั้นตอนของการสร้างแบบจำลองประสบความสำเร็จ



รูปที่ 3.6 พื้นฐานการออกแบบข่ายงานนิวรัล

3.3.1 ข้อมูลที่ใช้ในการฝึกข่ายงาน (Training Data)

ก. การเลือกชุดข้อมูลอินพุทและเอาท์พุท

หลักในการเลือกชุดข้อมูลอินพุทเข้าสู่ข่ายงานนิวรัลคือในกรณีที่แบบจำลองในสภาวะคงที่ให้เลือกอินพุทที่มีผลกระทบต่อค่าเป้าหมาย แต่ถ้าเป็นแบบจำลองที่เป็นพลศาสตร์ (dynamics) นอกจากจะพิจารณาตัวแปรที่มีผลกระทบต่อค่าเป้าหมายแล้วยังต้องพิจารณาอนุกรมของช่วงเวลาของชุดข้อมูลอินพุทและเอาท์พุท ในงานวิจัยนี้จะเลือกค่าอินพุทที่ขึ้นกับเวลา ซึ่งมีอิทธิพลต่อค่าเอาท์พุท ความรู้ของระบบจะเป็นตัวสร้างแบบจำลองเพื่อใช้เป็นแนวทางเริ่มต้นในการตัดสินใจเลือกชุดข้อมูลอินพุทที่ขึ้นกับเวลา ส่วนการเลือกชุดข้อมูลเอาท์พุทคือชุดข้อมูลที่ต้องการในการทำนาย ตัวแปรทั้งหมดถูกวัดในหน่วยที่ต่างกัน และในช่วงใช้งานที่ต่างกัน ถ้าตัวแปรใดมีค่ามากว่าก็จะมีความสำคัญไม่เท่ากัน เนื่องมาจากวิธีการปรับค่าน้ำหนัก หรือกระบวนการออฟติไมซ์ ดังนั้นจึงต้องมีการลดค่าของตัวแปรอินพุทให้อยู่ในขอบเขตของค่าสูงต่ำเดียวกันทุกตัวแปร เพื่อจะให้ได้ค่าน้ำหนักที่เหมาะสมสำหรับตัวแปรอินพุท และค่าเอาท์พุทก็จะมีค่าอยู่ในขอบเขตหนึ่งซึ่งมาจากลักษณะของฟังก์ชันกระตุ้น

รูปแบบของอินพุทไฟล์ (input file format) ที่ใช้เป็น ASCII ในรูปของข้อมูลตัวอักษร (text) ข้อมูลแต่ละคอลัมน์แทนด้วยนิวรัลอินพุทหรือค่าเป้าหมายของชั้นเอาท์พุท ข้อมูลแต่ละแถวแทนด้วยจำนวนชุดข้อมูลที่ใช้ฝึก (pattern)

Pattern 1 <input node1> <input node 2> <input node 2>... <output node n>
Pattern 2 <input node1> <input node 2> <input node 2>... <output node n>

.

Pattern m <input node1> <input node 2> <input node 2>... <output node n>

ชุดข้อมูลที่ใช้ในการสร้างข่ายงานจะแบ่งชุดข้อมูลเป็น 3 ชุด ได้แก่ชุดข้อมูลชุดแรกจะใช้ในการฝึกข่ายงานเพื่อสร้างแบบจำลองภายในของข่ายงาน โดยใช้การปรับค่าน้ำหนักของข่ายงานจนมีค่าความผิดพลาดระหว่างค่าเป้าหมายกับค่าเอาท์พุทที่ได้จากข่ายงานน้อยจนกระทั่งอยู่ในขอบข่ายที่ยอมรับได้ ส่วนข้อมูลอีกชุดหนึ่งเป็นชุดในการทดสอบแบบจำลอง (cross validation data set) ซึ่งใช้เป็นตัวทดสอบความสามารถของข่ายงานว่าสามารถทำนายค่าเอาท์พุทได้ถูกต้องเพียงใด โดยชุดข้อมูลนี้จะคล้ายกับชุดฝึกข่ายงาน ชุดที่ 3 จะเป็นชุดทดสอบในการใช้งานจริงโดยใช้แบบจำลองที่เป็นตัวแทนของกระบวนการที่ผ่านจากการทดสอบด้วยชุดข้อมูลทดสอบชุดที่ 2 (validation set)

ข. การเตรียมข้อมูล (data preparation)

การเตรียมข้อมูลเป็นสิ่งสำคัญที่จะทำให้แบบจำลองสำเร็จหรือไม่สำเร็จเพราะจะทำให้ค่าของชุดข้อมูลที่ใช้ในการฝึกต่างกัน แบบจำลองก็จะต่างกันด้วย

ค. การลดขนาดของช่วงในข้อมูล (data normalization)

การลดขนาดของช่วงข้อมูลจะทำอยู่ในช่วงระหว่างค่า 0 ถึง 1 เพราะว่าค่าสัญญาณเอาท์พุทจะมีค่าอยู่ในช่วง 0 ถึง 1 และเป็นการลดขนาดของอินพุทที่มีขนาดใหญ่มากเกินไปกว่าอินพุทอื่น เพื่อทำให้มีความสำคัญเท่าเทียมกัน การลดขนาดช่วงของข้อมูลมี 4 วิธี

(1) Mean/std Deviation

วิธีนี้เป็นวิธีที่นิยมมาก แต่ละอินพุทจะเป็นอิสระต่อกัน

$$x_i' = (x_i - \text{mean}) / \text{std deviation}$$

xi' = อินพุทใหม่หลังจากการลดช่วงข้อมูลแล้ว

mean = ค่าเฉลี่ยของแต่ละอินพุท

std deviation = ค่าการเบี่ยงเบนมาตฐานของแต่ละอินพุท

(2) Max Min Preprocessing

วิธีนี้จะคำนวณจากค่าสูงสุดและต่ำสุดของแต่ละอินพุทในชุดข้อมูลที่ใช้ในการฝึกและทดสอบข่ายงาน

$$x_i' = (x_i - (\max_i - \min_i)/2) / (\max_i - \min_i)$$

โดย $\max_i$ = ค่ามากที่สุดของอินพุท

$\min_i$ = ค่าน้อยที่สุดของอินพุท

(3) Sum to 1 Normalization

$$x_i' = x_i / (\Sigma x_i)$$

วิธีนี้จะไม่ประสบผลสำเร็จถ้ามีค่าอินพุทที่เป็นค่าบวกและค่าลบมากๆ

(4) Sum of Squares to 1 Preprocessing

วิธีนี้จะคล้ายวิธี Sum to 1 Normalization

$$x_i' = x_i / (\Sigma(x_i)^{1/2})$$

วิธีนี้จะสูญเสียค่าของข้อมูลที่มีขนาดใหญ่ซึ่งจะสัมพันธ์กับเวกเตอร์ของอินพุทขณะที่ยังรักษาทิศทางหรือความสัมพันธ์ระหว่างอินพุทเหมือนเดิม

วิธีที่ 1 และ 2 วิธีนี้จะไม่สูญเสียข้อมูลใดๆ และสามารถขยายช่วงกลับได้โดยใช้สมการทางคณิตศาสตร์ แต่ วิธีที่ 3 และ 4 ไม่สามารถขยายช่วงกลับได้วิธีนี้จะเป็นวิธีที่สูญเสียค่าของข้อมูล

ง . จำนวนชุดข้อมูลในการฝึก (training pattern)

จำนวนชุดข้อมูลที่ใช้ในการฝึกข่ายงานก็เป็นสิ่งสำคัญ จำนวนชุดข้อมูลในการฝึกยิ่งมาก แบบจำลองก็ยิ่งดี แต่หมายความรวมถึงปริมาณข้อมูลและค่าความแตกต่างของชุดข้อมูลแต่ละชุดด้วย ถ้าข่ายงานมีชุดข้อมูลการฝึกที่ดีก็จะมีความสัมพันธ์ระหว่างอินพุทและเอาท์พุทดีขึ้น การเพิ่มจำนวนชุดข้อมูลนั้นคือการเพิ่มความแตกต่างของชุดข้อมูลจะเป็นการเพิ่มความซับซ้อนของการสร้างและออกแบบชั้นซ่อน ซึ่งจะแบ่งชุดข้อมูลนี้ประมาณ 10-20% สำหรับใช้เป็นชุดข้อมูลในการทดสอบ (test set)

3.3.2 ความรู้เกี่ยวกับการสร้างแบบจำลอง

ความรู้ในการสร้างแบบจำลองจะต้องประกอบไปด้วยองค์ประกอบดังต่อไปนี้

3.3.2.1. โครงสร้างของแบบจำลองและขนาดของโครงสร้าง

จากหน้าที่ของนิวรัลแต่ละหน่วยจะสามารถนำมาเขียนเป็นโครงสร้างของข่ายงานได้คือ โครงสร้างของข่ายงานประกอบด้วยชั้น (layer) อย่างน้อย 3 ชั้นเพื่อใช้ในการรับและส่งสัญญาณเอาท์พุท ดังต่อไปนี้

(1) ชั้นอินพุท (input layer): ทำหน้าที่รับข้อมูลที่ใช้ในการฝึกข่ายงานนิวรัลและกระจายไปให้กับนิวรัลในชั้นซ่อนต่อไป

(2) ชั้นภายใน (hidden layer) : จะทำการประมวลผลโดยการรวมสัญญาณ (โดยใช้ฟังก์ชันมูลฐาน) จากชั้นรับข้อมูลไปแปลงค่า (transform) เป็นสัญญาณเอาท์พุท โดยใช้ฟังก์ชันกระตุ้นและสัญญาณเอาท์

พุทจะถูกส่งต่อไปให้นิวรัลในชั้นถัดไป โดยข้อมูลความรู้ต่างๆ ที่ประมวลได้ จะถูกเก็บอยู่ในรูปค่าน้ำหนักและไบแอส ลักษณะการทำงานดังกล่าวจะกระทำซ้ำกันไปจนถึงชั้นเอาท์พุท ซึ่งชั้นซ่อนจะใช้เป็นตัวสร้างแบบจำลองภายในของระบบ ชั้นซ่อนสามารถมีได้มากกว่า 1 ชั้น

(3) ชั้นเอาท์พุท (output layer): รวมสัญญาณจากชั้นซ่อนไปแปลงค่าเป็นผลลัพธ์ออกจากข่ายงานนิวรัลซึ่งจะเป็นคำตอบของปัญหาที่ต้องการต่อไป

การกำหนดโครงสร้างและจำนวนของนิวรัลที่ต้องการในแต่ละชั้นในแต่ละงานจะไม่มีกฎเกณฑ์ที่แน่นอน เช่นในงานที่เป็นโครงสร้างการเชื่อมต่อแบบป้อนไปข้างหน้าที่มีชั้นซ่อน 1 ชั้น จะเหมาะสมในการใช้งานส่วนใหญ่ โดยปกติสำหรับการเลือกชั้นซ่อนจะเริ่มต้นที่ชั้นซ่อน 1 ชั้นก่อน และค่อยทำการแก้ไขขนาดของโครงสร้างจากตอนเริ่มต้นโดยใช้วิธีลองผิดลองถูก เพื่อให้แบบจำลองเป็นที่พอใจโดยมีค่าความผิดพลาดอยู่ในช่วงที่ยอมรับได้ แต่ถ้าค่าความผิดพลาดอยู่ในช่วงที่ยอมรับไม่ได้ ขนาดของโครงสร้างต้องถูกปรับใหม่และทำการหาค่าความผิดพลาดใหม่ซ้ำไปเรื่อยๆ

โครงสร้างข่ายงานที่ต่างกันต้องการอัลกอริธึมในการฝึกข่ายงานที่ต่างกันและเวลาในการฝึกข่ายงานก็ไม่เท่ากันซึ่งสามารถทำให้เวลาลดลงได้โดยเลือกอัลกอริธึมที่เหมาะสม อย่างไรก็ตามในงานวิจัยนี้จะใช้อัลกอริธึมแบบการกระจายความผิดพลาดย้อนกลับแบบ Levenberg-Marquardt ในการลดค่าผิดพลาดระหว่างค่าเป้าหมายกับค่าที่เอาท์พุทที่ทำนายได้จากข่ายงาน ดังนั้นในการออกแบบโครงสร้างของแบบจำลองต้องกำหนดดังต่อไปนี้

- ชั้นอินพุทและชั้นเอาท์พุท

ชั้นอินพุทของข่ายงานนิวรัลมีจุดประสงค์ในการกระจายค่าอินพุทไปที่ชั้นซ่อนชั้นแรก จำนวนนิวรัลในชั้นอินพุทจะเท่ากับจำนวนค่าตัวแปรที่ใช้เป็นอินพุทในแบบจำลอง

จำนวนนิวรัลในชั้นเอาท์พุทของข่ายงานนิวรัลเท่ากับจำนวนของตัวแปรที่ต้องการทำนาย

- ชั้นซ่อน

การเลือกจำนวนของชั้นซ่อน และจำนวนของนิวรัลในแต่ละชั้นซ่อน จะไม่สามารถกำหนดได้เพราะมีหลายปัจจัยที่มีอิทธิพลกับการหาโครงสร้างข่ายงานที่เหมาะสม ปัจจัยนี้ได้แก่ ปริมาณของชุดข้อมูลที่ฝึก (training pattern) จำนวนของนิวรัลในชั้นอินพุทและชั้นเอาท์พุท และความสัมพันธ์ระหว่างอินพุทและเอาท์พุท จะก่อให้เกิดโครงสร้างที่มีจำนวนชั้นซ่อนขึ้นมากมาย ซึ่งเรียกว่าสมองยิ่งใหญ่แบบจำลองยิ่งดี เมื่อแบบจำลองใหญ่มากและซับซ้อนจะมีลักษณะการจดจำชุดข้อมูลอินพุทและเอาท์พุทมากกว่าการเรียนรู้ความสัมพันธ์ของชุดข้อมูล ข่ายงานที่มีการฝึกที่ดีแต่เวลาทดสอบด้วยชุดทดสอบ (validation set) อาจจะไม่ดีก็ได้เพราะ

- ค่าของข้อมูลอาจอยู่นอกช่วงของชุดข้อมูลฝึกก็ได้
- ในบางครั้งเวลาที่ใช้ในการฝึกจะเพิ่มขึ้นเมื่อข่ายงานใหญ่และซับซ้อนโดยไม่จำเป็น

- หลักการการเรียนรู้จะเป็นการเรียนรู้แบบจดจำ

ดังนั้นการเริ่มต้นที่ดีด้วยการออกแบบข่ายงานที่มีจำนวนของชั้นซ่อนและจำนวนนิวรัลในชั้นซ่อนน้อยๆ ถ้าการเรียนรู้ไม่เพียงพอหรือมีความสัมพันธ์กันไม่ดีพอ ควรสร้างความซับซ้อนของข่ายงานเพิ่มขึ้นโดยการปรับการเรียนรู้หรือการเพิ่มโครงสร้างของข่ายงาน

3.3.2.2 ชนิดของการเชื่อมโยงน้ำหนักในโครงสร้างของข่ายงานนิวรัล (network connection)

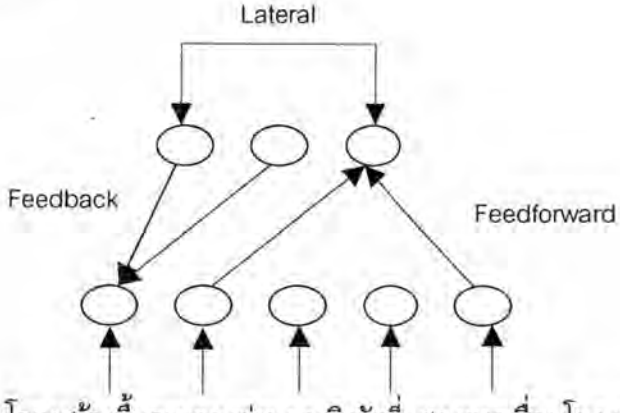
ข่ายงานนิวรัลประกอบด้วยนิวรัลและแมทริกซ์ของน้ำหนัก พฤติกรรมของข่ายงานขึ้นกับการเชื่อมโยงระหว่างนิวรัลสองชั้นติดต่อกันโดยผ่านข่ายงานเชื่อมต่อที่เรียกว่าน้ำหนัก ซึ่งการติดต่อสื่อสารกันระหว่างชั้นแบ่งเป็น 4 ประเภท แสดงในรูปที่ 3.7 คือ

(1) การเชื่อมโยงแบบป้อนไปข้างหน้า (feedforward connection) เป็นการนำข้อมูลของนิวรัลจากชั้นที่ต่ำกว่าและนำไปใช้ในชั้นที่อยู่เหนือกว่า

(2) การเชื่อมโยงแบบป้อนกลับ (feedback connection) เป็นการนำข้อมูลจากชั้นที่อยู่เหนือกว่า และส่งกลับลงมายังชั้นที่อยู่ต่ำกว่า

(3) การเชื่อมโยงภายในชั้น (lateral connection) เป็นการนำข้อมูลภายในชั้นเดียวกันและกระจายกลับมาใช้ในชั้นเดิม

(4) การเชื่อมโยงแบบหน่วงเวลา (time-delayed connection) เป็นการนำข้อมูลมาหน่วงเวลาแล้วนำเข้าไปรวมกับการเชื่อมโยงของข้อมูลเพื่อให้ได้แบบจำลองเป็นไดนามิก (temporal dynamics) ซึ่งนำมาประยุกต์ใช้กับการจดจำรูปแบบข้อมูล (temporal pattern recognition)



รูปที่ 3.7 โครงสร้างพื้นฐานของข่ายงานนิวรัลที่แสดงการเชื่อมโยงแบบต่างๆ

การออกแบบข่ายงานที่เกี่ยวข้องกับการควบคุมการเชื่อมต่อของชั้นซ่อนภายในข่ายงานควรพิจารณาว่ามีการเชื่อมต่ออย่างไร ซึ่งปกติแล้วหลักที่สำคัญคือจะใช้การเชื่อมต่อข่ายงานแบบกระจายทั้งหมดมากกว่ากระจายบางส่วน

3.3.2.3 ฟังก์ชันกระตุ้น (transfer function)

ฟังก์ชันกระตุ้นใช้เพื่อจุดประสงค์สำหรับควบคุมความเข้มของสัญญาณขาออก (output signal strength) สำหรับนิวรัลในแต่ละชั้น (ยกเว้นชั้นอินพุทจะใช้ค่าในชั้นอินพุทเหล่านั้น) ฟังก์ชันกระตุ้นจะถูกกำหนดตามชนิดของฟังก์ชันกระตุ้น เช่นกำหนดความเข้มของสัญญาณขาออกอยู่ระหว่าง 0 ถึง 1 อินพุทที่จะมาผ่านฟังก์ชันกระตุ้น โดยส่วนใหญ่ได้มาจากการทำผลคูณแบบเวกเตอร์ (dot product) ระหว่างสัญญาณอินพุทของนิวรัลกับค่าน้ำหนัก (weight vector) ซึ่งฟังก์ชันกระตุ้นที่นิยมใช้ส่วนใหญ่มี ฟังก์ชันซิกมอยด์ (sigmoid), เกาส์เสียน (gaussian) , ไฮเปอร์โบลาร์ แทนเจนท์ (hyperbolar tangent) ซึ่งฟังก์ชันเหล่านี้สามารถกำหนดแบบใดก็ได้ในแต่ละชั้น

3.3.2.4 การกำหนดค่าน้ำหนักเริ่มต้น (weight initialization)

การกำหนดค่าน้ำหนักเริ่มต้นจะมีผลต่อความเร็วและคุณภาพของการฝึกข่ายงานนิวรัล การกำหนดค่าน้ำหนักเริ่มต้นจะกำหนดเป็นค่าน้อยๆ คือเป็นตัวเลขสุ่ม เช่น ในช่วง -0.5 ถึง 0.5 (Bhat and McAvoy, 1954) ดังนั้นน้ำหนักที่ทำการเชื่อมต่อแต่ละชั้นจะมีความแตกต่างกันเล็กน้อยระหว่างการฝึกข่ายงานและมีผลกระทบต่อการหยุดการฝึกที่มีค่าผิดพลาดต่ำที่สุด ถ้าค่าความผิดพลาดไม่อยู่ในช่วงที่พอใจ จะทำการปรับค่าน้ำหนักใหม่ และทำการเรียนรู้ซ้ำอีกครั้ง และถ้าไม่ได้ผลอีกให้ทำการกำหนดโครงสร้างของข่ายงานใหม่ ซึ่งวิธีเหล่านี้เป็นการปรับปรุงประสิทธิภาพของการฝึกข่ายงาน

3.3.2.5การจำแนกการเรียนรู้ของข่ายงานนิวรัล

ในการออกแบบข่ายงานต้องทำการออกแบบอัลกอริธึมในการเรียนรู้เพื่อให้ข่ายงานสามารถจำลองข่ายงานได้ใกล้เคียงกับสิ่งที่ต้องการเรียนรู้ การเรียนรู้ของระบบประสาทของมนุษย์ได้จากการเรียนรู้จากประสบการณ์ การจำลองการเรียนรู้โดยใช้ข่ายงานนิวรัลนี้จึงเรียกว่า machine learning algorithms เพราะว่าเป็นการปรับค่าน้ำหนัก ระหว่างนิวรัล เพื่อทำให้เกิดการเรียนรู้คำตอบของปัญหา

ในการฝึกข่ายงานนิวรัลให้เรียนรู้เพื่อหาคำตอบของปัญหาหนึ่ง ๆ นั้นอาศัยชุดของปัญหาและคำตอบของปัญหาที่ถูกต้อง เพื่อนำมาใช้ในการฝึกข่ายงานนิวรัลให้สามารถสร้างคำตอบของปัญหาให้คล้ายคลึงกับชุดของข้อมูลที่ได้รับการฝึกให้เรียนรู้ โดยทั่วไปแล้วข่ายงานนิวรัลสามารถจำแนกการเรียนรู้ได้ 2 ประเภท ดังนี้

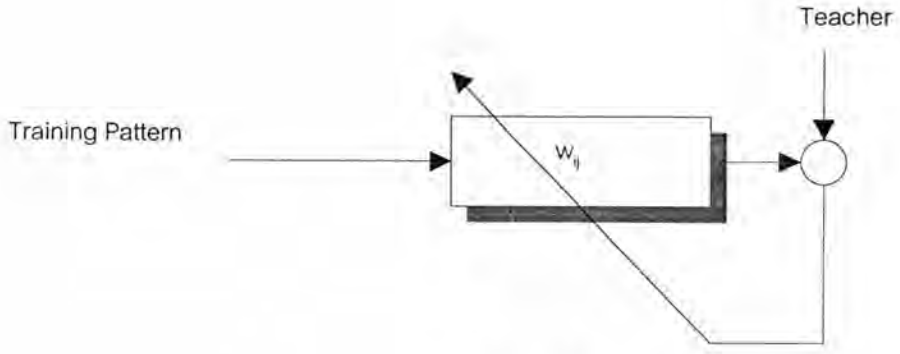
ก. ข่ายงานนิวรัลแบบมีการชี้นำ (supervised neural network)

เป็นการฝึกเครือข่ายนิวรัลให้เลียนแบบชุดข้อมูลตามที่กำหนดและสามารถส่งสัญญาณเอาท์พุทได้คล้ายคลึงกับชุดข้อมูลที่ป้อนให้เรียนรู้

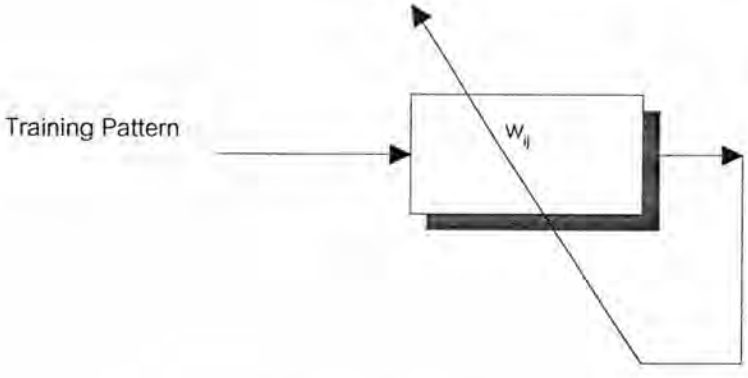
ข่ายงานนิวรัลที่มีการชี้นำถูกฝึกโดยใช้คู่ลำดับของค่าอินพุท และค่าเป้าหมาย ค่าอินพุทใช้เป็นอินพุทให้แก่ข่ายงานและค่าเป้าหมายใช้เป็นผู้สอน ดังแสดงในรูปที่ 3.8 (ก) ซึ่งถือว่าเป็นข้อได้เปรียบของการเรียนรู้วิธีนี้เพราะมีเป้าหมายที่ต้องการเป็นผู้สอนโดยค่าที่แตกต่างกันระหว่างค่าเอาท์พุทที่ออกจากข่ายงานกับค่าเป้าหมายที่ต้องการจะเป็นข้อมูลที่ใช้ในการเรียนรู้เพื่อทำให้ค่าแตกต่างระหว่างค่าเป้าหมายและเอาท์พุทของข่ายงานมีค่าน้อยที่สุด วิธีการปรับค่าน้ำหนักของข่ายงานเพื่อให้เครือข่ายนิวรัลเรียนรู้และจดจำข้อมูลสัญญาณอินพุทและสร้างสัญญาณเอาท์พุทได้สอดคล้องกับข้อมูลสัญญาณเป้าหมายตามที่กำหนดในชุดของข้อมูลที่ใช้ฝึก ซึ่งกระบวนการนี้อาจทำซ้ำจนค่าผิดพลาดอยู่ในช่วงที่ยอมรับได้ ในขณะที่การเรียนรู้แบบไม่มีการชี้นำ (unsupervised neural network) จะมีเพียงชุดข้อมูลอินพุทของการฝึกเท่านั้น โดยการเรียนรู้จะเป็นการปรับตัวบนพื้นฐานของประสบการณ์เพื่อจัดกลุ่ม (cluster) ให้เข้ากับชุดข้อมูลที่ใช้เรียนรู้ที่ผ่านมา ในทางปฏิบัตินั้นการเรียนรู้แบบชี้นำจะสอดคล้องกับวิธีการออพติไมเซชันแบบสโตชาสติคเกรเดียนท์เดสเซ็นท์ (stochastic gradient optimization) ตัวอย่างของข่ายงานประเภทนี้ได้แก่เปอร์เซปตรอน (perceptron) ,อะดาไลน์ (adaline) , และการกระจายย้อนกลับ (backpropagation) ดังแสดงในรูปที่ 3.9

ข. ข่ายงานนิวรัลแบบไม่มีการชี้นำ (unsupervised neural network)

ข่ายงานนิวรัลที่มีการเรียนรู้แบบไม่มีการชี้นำต้องการเพียงค่าอินพุทในการฝึกข่ายงานโดยไม่มีผู้สอนหรือค่าเป้าหมาย การปรับค่าพารามิเตอร์ของข่ายงานนิวรัลใช้สัญญาณเอาท์พุทอย่างเดียว ซึ่งแสดงในรูป 3.8 (ข) ในระหว่างการฝึกค่าน้ำหนักของข่ายงานจะถูกปรับโดยอินพุทที่คล้ายๆกันสร้างเอาท์พุทที่คล้ายกัน ซึ่งวิธีการปรับมีหลายวิธี ในแต่ละวิธีมีวัตถุประสงค์เพื่อให้การเรียนรู้ของข่ายงานประเภทนี้คล้ายกับสมองของมนุษย์ที่เปลี่ยนแปลงโครงสร้างของมันภายใต้อิทธิพลของประสบการณ์โดยไม่มีผู้สอน ตัวอย่างการเรียนประเภทนี้ได้แก่ AG(Additive Grossberg), Self Organizing Map (SOM) และ ART (Adaptive Resonance Theory) ดังแสดงในรูปที่ 3.9

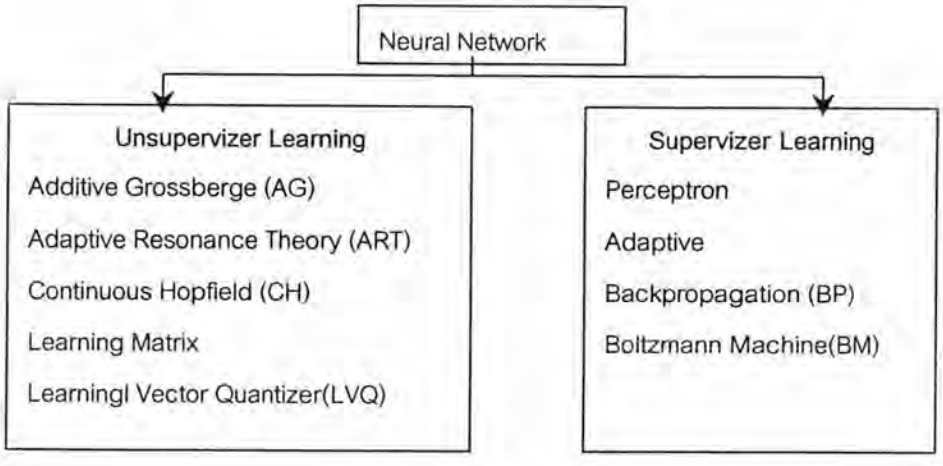


(ก) ข่ายงานที่มีการชี้นำ (supervised learning)



(ข) ข่ายงานที่ไม่มีการชี้นำ (unsupervised learning)

รูปที่ 3.8 การเรียนรู้ของข่ายงานนิวรัลในการปรับค่าน้ำหนัก



รูปที่ 3.9 ข่ายงานนิวรัลสามารถจำแนกเป็นแบบมีการชี้นำและไม่มีการชี้นำ

3.3.2.5 ตัวอย่างข่ายงานนิวรัลแบบมีการชี้นำ (supervised neural network) และไม่มีการชี้นำ (unsupervised neural network)

หัวข้อนี้จะกล่าวถึงข่ายงานนิวรัลแบบมีการชี้นำและข่ายงานที่ไม่มีการชี้นำ

3.3.2.5.1 ข่ายงานนิวรัลแบบมีการชี้นำ

ข่ายงานนิวรัลแบบมีการชี้นำได้แก่ ข่ายงานเปอร์เซปตรอน (perceptron) ข่ายงานแบบอะดาไลน์ และข่ายงานแบบการกระจายความผิดพลาดย้อนกลับ ข่ายงานเปอร์เซปตรอนและอะดาไลน์เป็นข่ายงานที่มีชั้นของน้ำหนักเพียงชั้นเดียว ดังนั้นข่ายงานทั้งสองจึงสามารถแก้ปัญหาได้เฉพาะปัญหาแบบเชิงเส้น สำหรับข่ายงานแบบกระจายย้อนกลับเป็นข่ายงานของน้ำหนักตั้งแต่ 2 ชั้นขึ้นไป ดังนั้นข่ายงานจึงสามารถแก้ปัญหาแบบไม่เชิงเส้นได้เป็นอย่างดี ข่ายงานการเรียนรู้แบบการกระจายย้อนกลับเป็นอัลกอริธึมที่นำมาใช้ในฝึกข่ายงานนิวรัลเพื่อใช้ในการทำนายค่าอัตราการไหลและความหนาแน่นของพอลิเมอร์ของงานวิจัยนี้โดยข่ายงานนิวรัลจะถูกฝึกด้วยการใช้ชุดข้อมูลซึ่งมีการปรับค่าน้ำหนักเพื่อให้ค่าเป้าหมายกับเอาท์พุทของข่ายงานมีค่าใกล้เคียงกัน ฟังก์ชันมูลฐานที่ใช้เป็นฟังก์ชันเชิงเส้น ฟังก์ชันกระตุ้นเป็นฟังก์ชันซิกมอยด์ และโครงสร้างการเชื่อมโยงเป็นแบบป้อนไปข้างหน้า โดยจำนวนชั้นในข่ายงานนิวรัลหมายถึงชั้นของน้ำหนัก ชั้นของน้ำหนักเป็นตัวแทนของการเปลี่ยนแปลงแบบเชิงเส้นแต่ชั้นของนิวรัลแต่ละชั้นจะเกิดการเปลี่ยนแปลงแบบไม่เชิงเส้น ในทางปฏิบัติพบว่าชั้นซ่อนเพียงสองชั้นเพียงพอต่อการแก้ปัญหาระบบต่างๆ ได้ทุกระบบ (สุรพล,1995) ข่ายงานนิวรัลนี้จะใช้อัลกอริธึมการกระจายย้อนกลับในการฝึกข่ายงานนิวรัล และข่ายงานนิวรัลแบบมีการชี้นำจะใช้ชุดข้อมูลที่ประกอบไปด้วยค่าอินพุทและเป้าหมายเป็นคู่ที่สอดคล้องกัน

$$[u,O] = \{ [u_1,O_1], [u_2,O_2], [u_3,O_3],..., [u_4,O_4] \} \qquad ....(3.22)$$

เมื่อ n คือจำนวนชุดข้อมูลที่ใช้ในการฝึก วัตถุประสงค์ในการฝึกข่ายงานคือหาค่าน้ำหนักที่เหมาะสม และทำให้ความแตกต่างของเป้าหมายกับค่าผลลัพธ์ที่ออกจากข่ายงานมีค่าน้อยที่สุด ข้อกำหนดของค่าความแตกต่างคือ ผลรวมของความแตกต่างระหว่างค่าเป้าหมายกับเอาท์พุทของข่ายงานกำลังสอง (sum square error,SSE) ฟังก์ชันของผลต่างกำลังสองคือ

$$E = [t - f(u,w)]^2 \qquad ...(3.23)$$

E คือฟังก์ชันพลังงานหรือฟังก์ชันต้นทุน (energy function or cost function) ของระบบแบบจำลองของข่ายงานเป็นฟังก์ชันของอินพุท และแมทริกซ์ของน้ำหนัก, y = f(u,w) แมทริกซ์ของน้ำหนักสามารถปรับค่าเพื่อให้ฟังก์ชันพลังงานมีค่าต่ำลงให้เหลือน้อยที่สุดไปในทิศทางลดลงของเกรเดียนท์ (gradient)

กฎการเรียนรู้โดยทั่วไป (General Learning Rule)

กำหนดให้

W	ค่าน้ำหนักและค่าไบแอส
X	สัญญาณอินพุท
r	สัญญาณที่ใช้ในการเรียนรู้ (Learning signal)
d	สัญญาณที่ใช้ในการสอน (Teachering Signal)
O	สัญญาณเอาท์พุทจากนิวรัล
α	ค่าคงที่ของการเรียนรู้ (Learning Constant)
k	ลำดับเวลา (Time Step)

สามารถเขียนสมการรูปทั่วไปของสัญญาณที่ใช้ในการเรียนรู้ดังสมการ

$$r = r(W,X,d) \qquad ...(3.24)$$

ค่าน้ำหนักของเครือข่ายนิวรัลที่เปลี่ยนไป หลังจากได้รับสัญญาณที่ใช้ในการเรียนรู้สามารถเขียนได้ดังสมการที่ 3.25

$$\Delta W(k) = \alpha.r\ (\ W(k),X(k),d(k)\ )\ X(k) \qquad ...(3.25)$$

ดังนั้นในทุกๆรอบของกระบวนการเรียนรู้ ค่าน้ำหนักของเครือข่ายนิวรัลจะถูกปรับดังสมการต่อไปนี้

$$\Delta W(k+1) = W(k) + \Delta W(k) \qquad ...(3.26)$$

ชนิดของสัญญาณที่ใช้ในการเรียนรู้ อาศัยกฎที่มีลักษณะแตกต่างกันออกไป อาทิเช่น Hebbian Learning Rule, Perceptron Learning Rule, Delta Learning Rule เป็นต้น ในการเลือกใช้กฎใด ขึ้นกับลักษณะของงานที่นำไปประยุกต์ใช้

ในที่นี้จะพิจารณาข่ายงานนิวรัลแบบการกระจายย้อนกลับ ซึ่งเป็นกฎการเรียนรู้ปรับปรุงดัดแปลงจาก Delta Learning Rule ซึ่งจะกล่าวถึงต่อไป

ก. ข่ายงานแบบเฮิรบเน็ท (Hebb net)

ข่ายงานนี้เป็นข่ายงานที่ง่ายที่สุด ซึ่งเรียกว่า Hebb rule น้ำหนักเริ่มต้น =0 การปรับค่าน้ำหนักจะเกิดขึ้นเมื่อ

$$w1(new) = w1(old) + X_i t \qquad ...(3.27)$$

wi = น้ำหนักที่เชื่อมระหว่างชั้น

t = ค่าเป้าหมาย

ข่ายงานนี้ไม่สามารถเรียนรู้ถ้าค่าเป้าหมายเท่ากับ 0 และมีข้อจำกัดคือในบางครั้งจะให้ค่าเอาท์พุทไม่ถูกต้อง นั้นคือค่าเอาท์พุทไม่ใกล้เคียงเป้าหมายหลังจากที่ทำการฝึกข่ายงาน

ข. ข่ายงานแบบเปอร์เซปตรอน (Perceptron)

ข่ายงานเปอร์เซปตรอนเป็นแบบจำลองข่ายงานนิวรัลแบบตัดสิน เสนอโดย F.Rosenblatt ในปี ค.ศ. 1958 ซึ่งออกแบบเพื่อใช้ในการแยกข้อมูล 2 คลาสออกจากกัน โดยใช้ขอบเขตการตัดสินเชิงเส้น (linear decision) ดังแสดงในรูปที่ 3.10 และ 3.11 เป็นการเรียนรู้ที่ดีกว่า Hebb net การเรียนรู้วิธีนี้จะวนซ้ำจนกระทั่งได้ค่าน้ำหนักที่ถูกต้องนั้นคือในการเรียนรู้อาศัยสัญญาณการเรียนรู้จากค่าความแตกต่างระหว่างค่าเป้าหมายกับค่าเอาท์พุทที่ออกจากข่ายงานนิวรัล ดังนั้นวิธีการเรียนรู้แบบนี้จึงเป็นการเรียนรู้แบบชี้นำ ค่าความผิดพลาดแสดงดังสมการที่ 3.28

$$r = d_i - O_i \qquad ...(3.28)$$

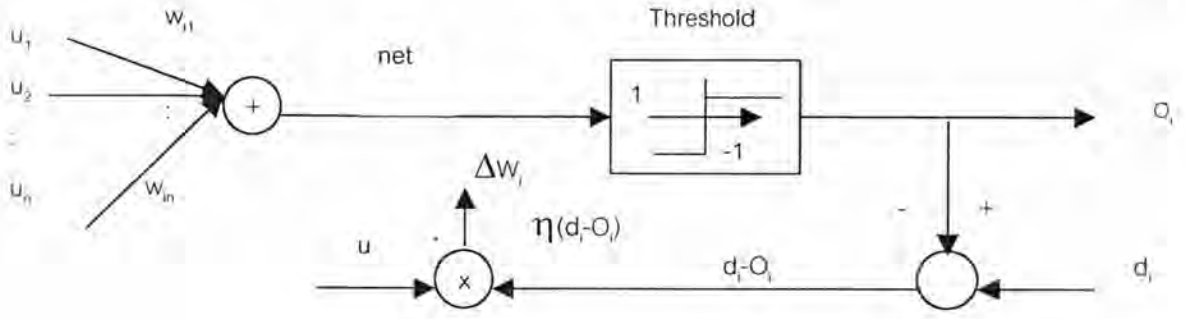
เมื่อ $O_i = f(u,w)$, u = อินพุท และ d_i คือค่าเป้าหมายโดยการปรับค่าน้ำหนักแสดงดังสมการ 3.29

$$\Delta w_{ij} = \eta[d_i - f(w_{ij}^T u)x_j] \qquad \text{เมื่อ } j = 1,2,...,n \qquad ...(3.29)$$

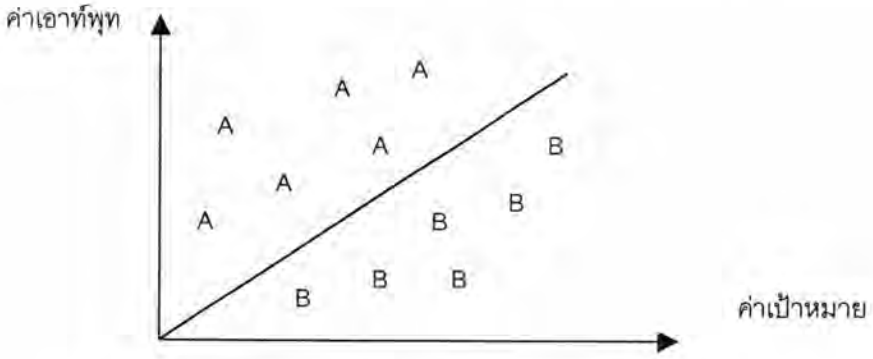
การเรียนรู้แบบเปอร์เซปตรอนนำไปประยุกต์ใช้ได้เฉพาะระบบตัวเลขไบนารีเท่านั้น ดังนั้นแบบจำลองนี้จึงสามารถนำไปใช้ในการแยกข้อมูล 2 ประเภทออกจากกัน ในการปรับค่าน้ำหนักจะถูกกระทำเมื่อค่า

เอาท์พุทจากข่ายงานนิวรัลไม่ตรงกับค่าเป้าหมายเท่านั้น ในกรณีที่เลือกใช้ฟังก์ชันแบบไบโพลาร์เป็นฟังก์ชันกระตุ้น ดังนั้นค่าเป้าหมายคือ -1 กับ 1 การปรับค่าน้ำหนักจะลดลงเหลือ

$$\Delta w_i = \pm 2\eta u \qquad ...(3.30)$$



รูปที่ 3.10 ข่ายงานเรียนรู้แบบเปอร์เซปตรอน



รูปที่ 3.11 การปรับหาขอบเขตแยกข้อมูล 2 ประเภท โดยข่ายงานเปอร์เซปตรอนเชิงเส้น

ค. การเรียนรู้แบบเดลตา (delta learning rule :ADALINE)

McCelland และ Rumelhart (1986) เสนอแบบจำลองข่ายงาน ADALINE (Adaptive Linear Element) ที่ใช้กฎการเรียนรู้แบบเดลตาในการหาค่าน้ำหนัก

กฎการเรียนรู้แบบเดลตา (delta learning rule) เป็นการเรียนรู้แบบชี้นำเพียงชนิดเดียวที่ใช้กับฟังก์ชันกระตุ้นแบบต่อเนื่องซึ่งได้แก่ฟังก์ชันซิกมอยด์ และฟังก์ชันเกาส์เสียน ฟังก์ชันมูลฐานเป็นฟังก์ชันเชิงเส้น โครงสร้างการเรียนรู้ของข่ายงานแบบอะดาไลน์มีชั้นของน้ำหนักเพียง 1 ชั้น ดังแสดงในรูปที่ 3.12 โดยมีนิยามของการเรียนรู้ (learning signal,r) ดังนี้

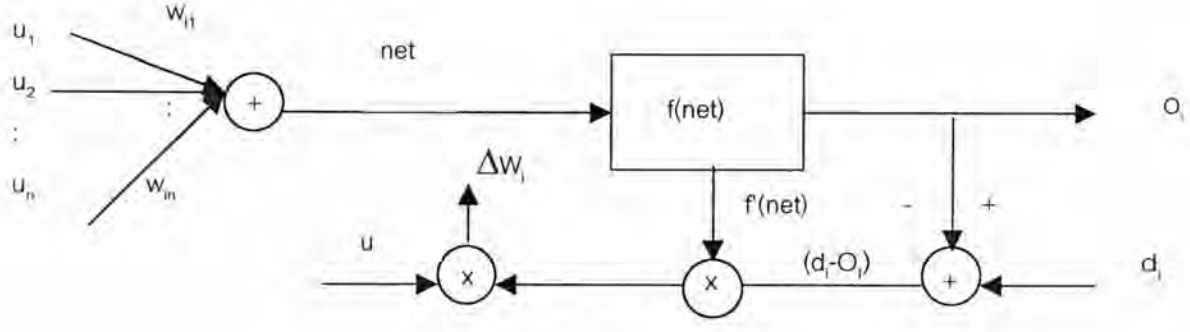
$$r = [d_i - f(net)]\, f'(net) \qquad ...(3.31)$$

เมื่อ f'(net) เป็นอนุพันธ์ของฟังก์ชันกระตุ้น ,f(net) ซึ่งคำนวณจาก net = $w_{ij}u$

กฎการเรียนรู้สามารถหาได้จากค่าผิดพลาดระหว่าง O_i และ d_i ยกกำลังสอง การคำนวณเกรเดียนท์เวคเตอร์ (gradient vector)เพื่อลดค่าผิดพลาด สามารถหาได้จากการหาค่าอนุพันธ์ของค่าผิดพลาดเทียบกับ w_i เพื่อปรับค่าน้ำหนักให้ค่าความผิดพลาดมีค่าต่ำที่สุดซึ่งการเรียนรู้อยู่บนพื้นฐานการลดค่ากำลังสองของค่าความแตกต่างให้มีค่าน้อยที่สุดโดยมีนิยามดังนี้

$$\text{minimum} \quad E = \tfrac{1}{2} (\Sigma(d_i - o_i)^2) \qquad ...(3.32)$$

เมื่อเปรียบเทียบการเรียนรู้ข่ายงานแบบเดลตากับเปอร์เซปตรอนพบว่ากฎการเรียนรู้แบบเดลตาใช้ค่าความแตกต่างของค่าเป้าหมาย, d_i เอาท์พุทของข่ายงาน ,o_i ป้อนกลับโดยใช้การออพติไมซ์ดังสมการ 3.31 แต่การเรียนรู้แบบเปอร์เซปตรอนใช้ค่าความแตกต่างระหว่างค่าเป้าหมายกับค่าเอาท์พุทของข่ายงานนิวรัลโดยตรงเพื่อใช้ในการปรับค่าน้ำหนัก ดังสมการ 3.28



รูปที่ 3.12 การเรียนรู้แบบเดลตา (delta learning rule)

3.3.2.5.2 ข่ายงานที่ไม่มีการชี้นำ

ข่ายงานที่ไม่มีการชี้นำจะ ใช้สำหรับชุดข้อมูลที่มีค่าอินพุทอย่างเดียวไม่มีค่าเป้าหมาย โดยจะยกตัวอย่าง 2 ประเภทได้แก่ Kohonen self-Organizing network และ Adaptive Resonance Theory (ART)

ก. Kohonen Self-Organizing พัฒนาโดย Teuvo Kohonen ในช่วงต้นปี 1980 เป็นข่ายงานที่ไม่ใช้ค่าเป้าหมายในการฝึกข่ายงาน Fausett ได้กล่าวถึง Self-Organization process เป็นการแบ่งกลุ่ม

โดยนิวรัลที่มีค่าเอาท์พุทมากที่สุดถือว่าเป็นเอาท์พุทของข่ายงาน และสามารถยับยั้งหรือกระตุ้นนิวรัลที่อยู่ข้างเคียงได้ และสามารถปรับน้ำหนักได้ โดยชุดทดสอบให้ค่าความผิดพลาดต่ำสุดคือ

$$D(g)= \sum_{i}(w_{ij}-w_{i})^{2} \qquad ...(3.33)$$

ใช้น้ำหนักเป็นค่าในการแบ่งกลุ่มแทนการสเกลใช้อินพุทดังนั้นการปรับค่าน้ำหนักคือ

$$w_{ij}(new) = w_{ij}(old)+ \alpha[w_{i}-w_{ij}(old)] \qquad ..(3.34)$$

การปรับน้ำหนักนิวรัลทุก i ของเอาท์พุท j ค่า α จะลดลงถ้าการเรียนรู้เพิ่มขึ้น ข่ายงานแบบ Kohonen จะคล้ายกับแบบข่ายงาน Competitive, ข่ายงาน Max และแบบ Learning Vector Quantization

ข. Adaptive Learning Resonant Theory (ART)

ART พัฒนาโดย Stephen Grossberg ในกลางปี 1970 ซึ่งจะวิเคราะห์ตามความสำคัญของอินพุทและการแบ่งประเภทอินพุทเวกเตอร์ใช้ในการเรียนรู้แบบไม่มีการชี้นำ นิวรัลมี 3 ชั้นคือ นิวรัลจะมีชั้นอินพุท เอาท์พุท และชั้นเชื่อมระหว่างอินพุทและเอาท์พุท ระหว่างการเชื่อมของชั้นอินพุทกับชั้นเชื่อม (interface) หรือชั้นเชื่อมไปชั้นเอาท์พุท จะมีการเชื่อมของน้ำหนักกลับไปมา คือ การเชื่อมของน้ำหนักแบบไปข้างหน้าระหว่างชั้นเอาท์พุทกับชั้นเชื่อม จะทำการแบ่งกลุ่มค่าอินพุท ซึ่งจะถูกกำหนดเป็นข้อมูลอินพุท และ การเชื่อมของน้ำหนักย้อนกลับมาจากชั้นเชื่อมไปสู่ชั้นอินพุทเพื่อหาว่าข้อมูลอินพุทคล้ายกับเวกเตอร์ตัวอย่างของการแบ่งกลุ่มหรือไม่ ถ้าคล้ายถึงจะมีการปรับน้ำหนัก ถ้าไม่คล้ายกลุ่มนั้นจะถูกเปลี่ยนชุดอินพุทใหม่ ซึ่งจะทำการวนรอบซ้ำโดยการเรียนรู้ตามอัลกอริธึม

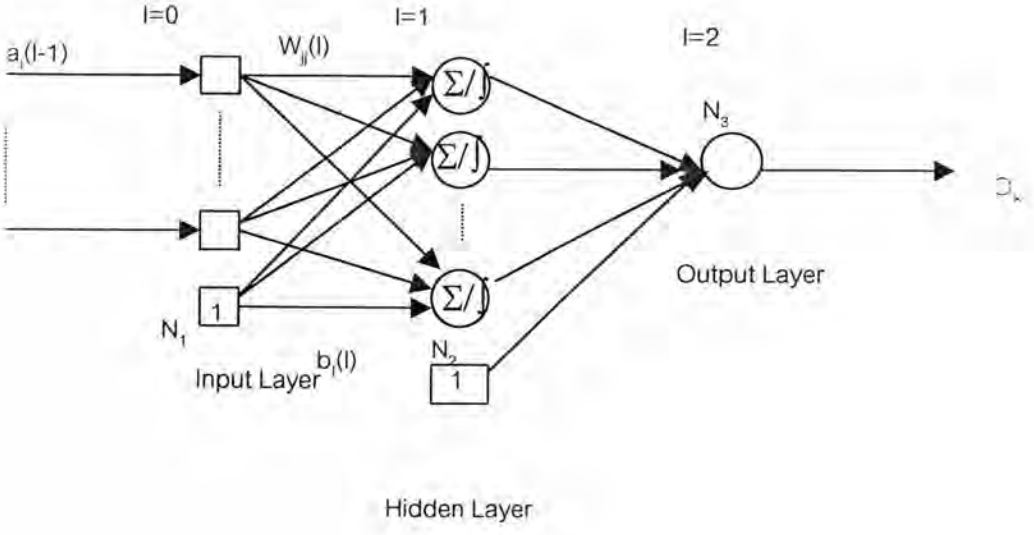
3.3.2.5.3 ข่ายงานนิวรัลที่ใช้แบบกระจายความผิดพลาดย้อนกลับ (backpropagation neural network)

Werbos(1974) พัฒนาอัลกอริธึมการกระจายความผิดพลาดย้อนกลับ (backproagation algorithm) Rumelhart (1986) เสนออัลกอริธึมนี้อีกครั้งทำให้อัลกอริธึมนี้เป็นที่รู้จักกันอย่างกว้างขวางและเป็นอัลกอริธึมมาตรฐานที่ใช้ในการฝึกข่ายงานแบบมีการชี้นำ อัลกอริธึมการกระจายย้อนกลับเป็นวิธีที่คำนวณเกรเดียนท์อย่างได้ผล (Kung,1993)

เครือข่ายนิวรัลแบบการกระจายความผิดพลาดย้อนกลับ เป็นเครือข่ายนิวรัลที่ใช้วิธีการประมวลผลเป็นชั้นๆ โดยนำข้อมูลจากชุดข้อมูลที่ต้องการใช้ในการฝึกเครือข่าย (Training set) ป้อนข้อมูลเข้าที่ชั้นอิน

พุท (Input layer) ผ่านชั้นซ่อน (Hidden layer) จนถึงชั้นเอาท์พุท (output layer) สัญญาณออกจากชั้นเอาท์พุทถูกนำไปเปรียบเทียบกับสัญญาณตามที่ต้องการ (Desired Signal) หรือสัญญาณเป้าหมาย (Target Signal) ซึ่งได้จากชุดของข้อมูลที่นำมาใช้ฝึกเครือข่ายนิวรัล ค่าแตกต่างระหว่างสัญญาณทั้งสองถูกแผ่ย้อนกลับเข้าสู่เครือข่ายนิวรัลอีกครั้ง โดยผ่านทางชั้นเอาท์พุทผ่านชั้นซ่อนจนถึงชั้นอินพุท เพื่อปรับค่าพารามิเตอร์ของเครือข่ายนิวรัลในแต่ละชั้น อันได้แก่ ค่าน้ำหนัก ค่าไบแอส และค่าพารามิเตอร์ต่างๆ ของกฎที่ใช้ในการเรียนรู้ ทำให้เครือข่ายนิวรัลสามารถเรียนรู้ชุดข้อมูลที่แฝงในรูปของค่าพารามิเตอร์ของเครือข่าย และสามารถสร้างสัญญาณออกซึ่งสอดคล้องกับข้อมูลที่ใช้ในการฝึกเครือข่าย

ข่ายงานนิวรัลแบบกระจายย้อนกลับมีฟังก์ชันมูลฐานเป็นฟังก์ชันมูลฐานเชิงเส้น และมีฟังก์ชันกระตุ้นเป็นฟังก์ชันซิกมอยด์มีสมการไดนามิกดังสมการ (3.35) และ (3.36) ตามลำดับ ดังแสดงในรูปที่ 3.13



รูปที่ 3.13 แสดงแบบจำลองนิวรัลในชั้นที่ l

a_i (l-1) = ค่าอินพุทที่ i ของชั้นอินพุท

O_i = ค่าเอาท์ที่ต้องการในชั้นเอาท์พุทตัวที่ i

$a_i(l)$ = $f(a_i)$ = ค่าเอาท์พุทจริงของนิวรัล i

$u_i(l)$ = ค่าผลรวมของค่าน้ำหนักกับอินพุทโดยใช้การคำนวณแบบฟังก์ชันมูลฐาน

w_{ij} = ค่าน้ำหนักที่เชื่อมต่อระหว่างนิวรัล i ในชั้นที่ l-1 กับนิวรัล j ในชั้นที่ l

E = ค่าความผิดพลาด (global error)

η = ค่าอัตราการเรียนรู้

$b_i(l)$ = ค่าไบอัสของชั้นที่ i

$$u_i(l) = \sum_{j=1}^{N} w_{ij}(l) a_j(l-1) + b_i(l) \quad ...(3.35)$$

$$a_i(l) = f(u_i(l)) \quad 1 \le i \le N_l, \quad 1 \le l \le L \quad ...(3.36)$$

ข่ายงานนิวรัลแบบกระจายย้อนกลับ ประกอบไปด้วยหลายๆ นิวรัลที่มีการเชื่อมโยงแบบป้อนไปข้างหน้า โครงสร้างของข่ายงานนิวรัลจะประกอบไปด้วยชั้นอินพุท,ชั้นซ่อนและชั้นเอาท์พุท ชั้นอินพุทประกอบด้วยนิวรัลที่ไม่มีการคำนวณใดๆ แต่จะกระจายค่าอินพุทให้แก่ทุกๆ นิวรัลในชั้นถัดไป ชั้นซ่อนที่อยู่ระหว่างชั้นอินพุทและชั้นเอาท์พุทซึ่งข่ายงานนิวรัลสามารถมีชั้นซ่อนหลายชั้นแต่จะพิจารณาในที่นี้เพียงชั้นเดียวเท่านั้น เพื่อความง่ายต่อการเข้าใจ และชั้นสุดท้ายคือชั้นเอาท์พุทโดยจะทำหน้าที่ส่งค่าเอาท์พุทสุดท้ายของข่ายงานนิวรัล นิวรัลในชั้นอินพุทรับค่าอินพุทเป็น $i_1,..,i_{n1}$ และมีน้ำหนักเป็น $w_{1j},...,w_{nj}$ และในแต่ละชั้นแทนด้วย l=0 เป็นชั้นอินพุท , l=1 เป็นชั้นซ่อนที่ 1 , l=2 เป็นชั้นเอาท์พุท

ผลรวมน้ำหนักของอินพุทของนิวรัลลำดับที่ j ในชั้นซ่อนคือ

$$u_j(1) = \sum_{i=1}^{N_1} w_{ij}(1) a_i(0) + b_j(1) \quad 1 \le j \le N_2 \quad ..(3.37)$$

เอาท์พุทของนิวรัลลำดับที่ j ในชั้นซ่อนคือ

$$a_j(1) = f(u_j(1)) 1 \le j \le N_2 \quad ..(3.38)$$

เอาท์พุทของนิวรัลในชั้นซ่อนใช้เป็นอินพุทให้แก่นิวรัลในชั้นเอาท์พุท ดังสมการ

$$u_j(2) = \sum_{i=1}^{N_2} w_{ij}(2) a_i(1) + b_j(2) \quad 1 \le j \le N_3 \quad ...(3.39)$$

นิวรัลในชั้นเอาท์พุทสร้างเอาท์พุทของกระบวนการที่ถูกทำนายในช่วง 1 รอบที่วนในการคำนวณจากชั้นอินพุทจนถึงชั้นเอาท์พุท ดังแสดงในสมการ (3.40)

$$a_j(2) = f(u_j(2)) \quad ...(3.40)$$

เมื่อ b_l (l) เป็นเทอมไบอัส (bias term) ของชั้นที่ l ซึ่งเป็นน้ำหนัก โดยใช้ค่าอินพุทมีค่าเป็น 1 และ a_2 หมายถึงค่าเอาท์พุทจากข่ายงานนิวรัล

หมายเหตุ f แทนด้วยฟังก์ชันกระตุ้น ฟังก์ชันกระตุ้นมักจะเป็นฟังก์ชันซิกมอยด์ (sigmoid function) โดยมีสมการดังนี้คือ

$$f(x) = \frac{1}{1 + e^{(-x)}} \qquad ...(3.41)$$

และอนุพันธ์ของฟังก์ชันซิกมอยด์คือ

$$f'(x) = x(1-x) \qquad ...(3.42)$$

ค่าอนุพันธ์ของฟังก์ชันซิกมอยด์จะนำไปใช้ในอัลกอริธึมการกระจายย้อนกลับที่จะกล่าวถึงในหัวข้อถัดไป

วิธีการออพติไมซ์ (optimize method)

กระบวนการที่ไม่เป็นเชิงเส้นหรือที่ซับซ้อนจะมีการเชื่อมโยงอินพุทไปสู่เอาท์พุทโดยใช้ชุดของน้ำหนักในข่ายงาน วัตถุประสงค์ของอัลกอริธึมก็คือต้องการหาค่าน้ำหนักให้ถูกต้องเพื่อที่สามารถจำลองแบบกระบวนการนั้นให้สามารถทำนายค่าเอาท์พุทจากอินพุทที่รับเข้าไปได้อย่างถูกต้อง ดังนั้นอัลกอริธึมการกระจายย้อนกลับเป็นอัลกอริธึมหนึ่งที่ต้องการลดค่าผิดพลาดกำลังสองระหว่างค่าเป้าหมายกับผลลัพธ์จากข่ายงาน โดยใช้วิธีการออพติไมซ์แบบเกรเดียนท์เดสเซนท์ซึ่งเป็นวิธีการลดความผิดพลาด (optimization method) ซึ่งแสดงในสมการ 3.43

สำหรับข้อมูลแต่ละชุด

minimize:

$$E = \frac{1}{2}\sum_{i=1}^{N_L} \{t_i - a_i(L)\}^2 \qquad ...(3.43)$$

subject to : $t_{max} > t_i > t_{min}$ ...(3.44)

$$a_{max} > a_i(L) > a_{min} \qquad ...(3.45)$$

เมื่อ E คือค่าความผิดพลาด(global error) ที่ชั้นเอาท์พุท ได้มาจากผลรวมกำลังสองของความแตกต่างของเอาท์พุทที่ได้จากการคำนวณกับค่าเป้าหมายในชั้น

เอาท์พุท

t_i คือค่าเป้าหมาย

t_{max} คือค่าสูงสุดของเป้าหมาย

t_{min} คือค่าต่ำสุดของเป้าหมาย

$a_i(L)$ คือค่าเอาท์พุทของข่ายงาน

a_{max} คือค่าสูงสุดของเอาท์พุทของข่ายงาน

a_{min} คือค่าต่ำสุดของเอาท์พุทของข่ายงาน

เกรเดียนท์เดสเซ็นท์คือการลดความผิดพลาดให้ต่ำที่สุดโดยการปรับค่าน้ำหนัก ดังนั้นถ้าทราบค่าอนุพันธ์ย่อย (partial derivative) ของค่าผิดพลาดเทียบกับค่าน้ำหนักแต่ละค่าจะทำให้รู้ทิศทางที่น้ำหนักไปในทิศทางที่ลดค่าผิดพลาด นั้นคือ $\partial E/\partial w_{ij}$ เป็นค่าเกรเดียนท์ในทิศทางลบของพื้นผิวความผิดพลาด เครื่องหมายลบแทนด้วยทิศทางเกรเดียนลดความผิดพลาดดังนั้นวิธีการปรับค่าน้ำหนักใหม่ในทางลดความผิดพลาดให้ต่ำสุด ซึ่งควรจะเป็นความผิดพลาดจริง (global error) มากกว่าที่จะเป็นความผิดพลาดเฉพาะที่ (local error) สำหรับอัลกอริธึมการกระจายย้อนกลับจะทำการปรับค่าน้ำหนัก (w_{ij}) เพื่อให้ค่า E มีค่าต่ำสุด สมการการเรียนรู้แบบเกรเดียนท์แสดงดังในสมการ 3.46

$$W_{ij}^{m+1}(l) = W_{ij}^{m}(l) + \Delta W_{ij}(l) \qquad ...(3.46)$$

อนุพันธ์ของค่าผิดพลาดจะใช้วิธีการตามเทคนิคของกฎลูกโซ่ (chain rule) ของชุดของข้อมูลการฝึก (iteration)

$$\Delta w_{ij}^{m}(l)\alpha - \frac{\partial E}{\partial w_{ij}^{m}(l)} \qquad ...(3.47)$$

$$\Delta w_{ij}^{m}(l) = -\eta \frac{\partial E}{\partial w_{ij}^{m}(l)}$$

$$= -\eta \frac{\partial E}{\partial a_i^{m}(l)} \frac{\partial a_i^{m}(l)}{\partial w_{ij}^{m}(l)} \qquad ...(3.48)$$

จากสมการ 3.46 นำมาแยกพิจารณาทีละเทอมและจากสมการ 3.47 และ 3.48 จะได้ว่า

$$\frac{\partial a_i^{m}(l)}{\partial w_{ij}^{m}(l)} = \frac{\partial f(u_i^{m}(l))}{\partial w_{ij}^{m}(l)}$$

$$= \frac{\partial f(\sum_{i=1}^{N} w_{ij}^{m}(l) a_i^{m}(l-1) + b_i(l))}{\partial w_{ij}^{m}(l)}$$

$$= f'(u_i^{m}(l) a_j^{m}(l-1)) \qquad ...(3.49)$$

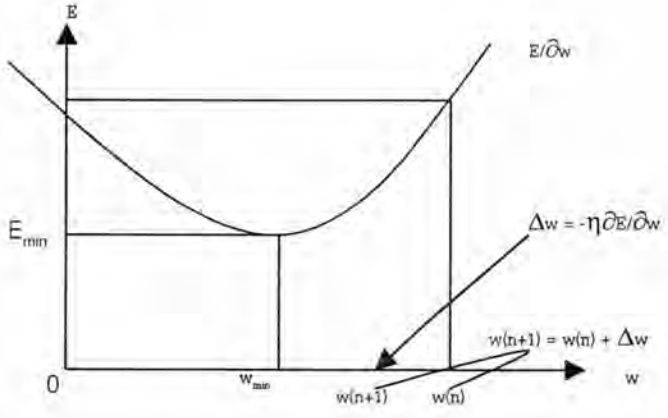
กำหนดให้สัญญาณความผิดพลาด, $\delta_i(l)$ เป็น

$$\delta_i^m(l) = -\frac{\partial E}{\partial a_i^m(l)} \qquad ..(3.50)$$

สมการที่ 3.49 และ 3.50 แทนในสมการ 3.48 จะได้สมการ 3.51 ได้ (การพิสูจน์สมการ 3.51 แสดงดังในภาคผนวก ฉ.)

$$\Delta w_{ij}^m(l) = \eta \delta_i^m(l) f'(u_i^m(l) a_j^m(l-1)) \qquad ...(3.51)$$

η คืออัตราการเรียนรู้ (learning rate) ถ้าเลือกอัตราการเรียนรู้ที่มีค่ามาก แต่ละขั้นตอนที่มีการปรับน้ำหนักค่าความผิดพลาดจะสูงเกินจุดต่ำสุด และนำไปสู่การแกว่งตัวรอบจุดต่ำสุด ในทางตรงกันข้ามถ้าอัตราการเรียนรู้มีค่าน้อยมากจะเป็นสาเหตุให้เข้าสู่จุดต่ำสุดช้าซึ่งต้องใช้อัลกอริธึมที่ช่วยเร่งในการเรียนรู้เช่น Rprop หรือ Levenberg-Marquardt



รูปที่ 3.14 แสดงการปรับน้ำหนักสำหรับอัลกอริธึมการกระจายความผิดพลาดย้อนกลับ (ถ้าเกรเดียนท์ $\partial E/\partial w$ เป็น บวก ค่าน้ำหนักจะถูกลบด้วย Δw เพื่อที่จะทำให้เกรเดียนท์มีค่าน้อยที่สุด ถ้าเกรเดียนท์มีค่าเป็นลบค่าน้ำหนักจะเพิ่มไปทางบวก)

ดังนั้นค่าน้ำหนักจะถูกปรับจนกระทั่งค่าความแตกต่างของค่าเป้าหมายกับผลลัพธ์จากข่ายงานมีค่าน้อยที่สุดในช่วงที่ยอมรับได้

ขั้นตอนของการเรียนรู้อัลกอริธึมการกระจายย้อนกลับ

ในการเรียนรู้ของอัลกอริธึมการกระจายย้อนกลับสำหรับข่ายงานนิวรัลแสดงในรูปที่ 3.13 ต้องกำหนดรูปแบบ (pattern) ของข้อมูลการเรียนรู้ ซึ่งประกอบด้วยกลุ่มข้อมูลของอินพุท u_i และค่าเป้าหมาย t_i จำนวน p คู่ เพื่อให้เครือข่ายนิวรัลสร้างสัญญาณออกลักษณะเดียวกัน ชุดของข้อมูลที่ใช้ในการฝึกเครือข่ายสามารถเขียนแทนได้ดังสมการต่อไปนี้

$$P=\{ (u_1,t_1),(u_2,t_2),...,(u_p,t_p)\} \qquad ..(3.52)$$

โดยที่ P ชุดของข้อมูลที่ใช้ในการฝึกเครือข่าย (Training Set)

p จำนวนชุดข้อมูลที่ใช้ในการฝึกเครือข่าย (Training pair)

ขั้นตอนที่ 1 การกำหนดค่าเริ่มต้นสำหรับพารามิเตอร์ของเครือข่ายนิวรัล

การฝึกเครือข่ายนิวรัลจำเป็นต้องกำหนดค่าอัตราการเรียนรู้ (learning rate, η) ให้มีค่ามากกว่าศูนย์ กำหนดค่าผิดพลาดต่ำสุด (E_{min}) และกำหนดค่าเริ่มต้นให้กับน้ำหนัก (weight) ที่เชื่อมต่อระหว่างนิวรัลแต่ละชั้น $w_{ij}(l)$ โดยใช้ค่าตัวเลขสุ่ม กำหนดข้อมูลคู่แรกเข้ามา p = 1 ค่าผิดพลาดเริ่มต้น E=0 และเริ่มการเรียนรู้ด้วยการวนรอบครั้งที่1 (q=iteration) q=1 เพื่อใช้เป็นจุดเริ่มต้นในกระบวนการประมวลผลของเครือข่าย

ขั้นตอนที่ 2 กระบวนการประมวลผล

กระบวนการประมวลผลของเครือข่ายนิวรัลแบ่งเป็น 2 ขั้นตอน

ขั้นตอนที่ 2.1 การคำนวณไปข้างหน้า (Feedforward Calculation)

การคำนวณเริ่มจากชั้นอินพุท ผ่านชั้นซ่อนภายในจนถึงชั้นเอาท์พุทแสดงได้ดังสมการที่ 3.53 ส่งอินพุทและค่าเป้าหมายที่สอดคล้องกันให้นิวรัลของชั้นอินพุทเพื่อรับข้อมูลซึ่ง $u = u_p$, $t=t_p$ และคำนวณค่าผลลัพธ์ให้นิวรัลในชั้นซ่อนและชั้นเอาท์พุทโดยใช้ฟังก์ชันซิกมอยด์เป็นฟังก์ชันกระตุ้น

$$\begin{aligned} u_i(l) &= \sum_{j=1}^{N} w_{ij}(l)a_j(l-1) + b_i(l) \\ a_i(l) &= f(u_i(l)) \quad 1 \le i \le N_l, \quad 1 \le l \le L \end{aligned} \qquad ..(3.53)$$

เมื่ออินพุทคือ $x_i = a_i(0)$ และเอาท์พุทแทนด้วย $y_i = a_i(l)$ เมื่อ l คือจำนวนชั้นทั้งหมดของข่ายงานนิวรัล และ i คือนิวรัลที่ i ในชั้นที่ j

ขั้นตอนที่ 2.2 การแผ่ความคลาดเคลื่อนย้อนกลับ (Backpropagation of Error)

การปรับค่าน้ำหนักของเครือข่ายนิวรัลเพื่อลดค่าความคลาดเคลื่อนหรือความแตกต่างระหว่างข้อมูลเป้าหมาย t_i กับสัญญาณออกจากเครือข่ายนิวรัลในชั้นเอาท์พุท a_i วิธีการออพติไมซ์แบบเกรเดียนท์ โดยกำหนดฟังก์ชันความคลาดเคลื่อน (Error Function :E) ดังแสดงในสมการที่ 3.54

คำนวณค่าผิดพลาด (E = ค่าผิดพลาดระหว่างค่าเป้าหมาย (t_i) และผลลัพธ์จากข่ายงาน ($a_i(L)$))

$$E^{new}=E^{old}+\frac{1}{2}\sum_{i=1}^{N}\{t_i-a_i(L)\}^2 \qquad ...(3.54)$$

การคำนวณค่าเวกเตอร์สัญญาณผิดพลาด

สัญญาณความผิดพลาดสำหรับชั้นเอาท์พุท

$$\delta(L)=(t_i-a_i(L)) \quad ,i=1,2,...,N_L \qquad ..(3.55)$$

สัญญาณความผิดพลาดสำหรับชั้นซ่อนแต่ละชั้น

$$\delta_i(l)=\sum_{j=1}^{N_{l+1}}(\delta_j(l+1)W_{ij}(l+1)a_i(l)(1-a_i(l))) \quad ,i=1,2,...,N_l \qquad ...(3.56)$$

ขั้นตอนที่ 3 การปรับค่าน้ำหนักที่เชื่อมระหว่างชั้นเอาท์พุทกับชั้นซ่อน หรือชั้นอินพุทกับชั้นซ่อน

หลังจากการประมวลผลในขั้นที่ 2 เครือข่ายนิวรัลจะปรับฐานข้อมูลความรู้ภายในใหม่ ซึ่งอยู่ในรูปของการปรับค่าน้ำหนัก และค่าน้ำหนักไบแอสในแต่ละชั้นให้ทันสมัยและสอดคล้องกับชุดข้อมูลความรู้ใหม่ที่ได้เรียนรู้ เป็นผลให้ฟังก์ชันความคลาดเคลื่อนดังสมการ 3.57 มีค่าลดลง

เมื่อ $i = 1,2,...,n_l$ และ $j=1,2,...,n_{l-1}$

w_{ij} = weight ของ nodeที่ i ใน layer ชั้นที่ j

$$W_{ij}^{new}(l) = W_{ij}^{old}(l) + \eta\delta_i(l)\, a_i(l-1) \qquad ...(3.57)$$

ขั้นตอนที่ 4 ตรวจสอบจำนวนชุดข้อมูลการเรียนรู้

ข้อมูลที่ใช้ในการฝึกเครือข่าย (Training Set) จะถูกป้อนเข้ามาครั้งละ 1 คู่ลำดับ (Training Pair :(u_i,t_i)) ซึ่งประกอบไปด้วย ข้อมูลอินพุทและข้อมูลเป้าหมาย แต่เนื่องจากชุดข้อมูลที่ใช้ในการฝึกเครือข่ายมีจำนวนทั้งสิ้น p ชุด ดังนั้นจำเป็นต้องตรวจสอบจำนวนข้อมูลที่ใช้ในการฝึกเครือข่ายได้รับมาครบหรือไม่ ถ้าชุดข้อมูลยังไม่ถูกทำงานจนครบรอบ ($P<p$) ให้ส่งข้อมูลของชุดถัดไปไปยังนิวรัลในชั้นอินพุท โดยที่

p=p+1 และ q=q+1 แล้วจึงกลับไปทำขั้นตอนที่ 2 แต่ถ้าข้อมูลเรียนรู้จนครบแล้ว (p=p) ให้ไปดำเนินการขั้นตอนที่ 5

ขั้นตอนที่ 5 ตรวจสอบความผิดพลาดกับค่าต่ำสุดที่ตั้งไว้

วัตถุประสงค์หลักของการฝึกเครือข่ายนิวรัลคือ การลดฟังก์ชันความคลาดเคลื่อนระหว่างข้อมูลที่ออกจากเครือข่ายนิวรัลและข้อมูลเป้าหมาย ดังนั้นในขั้นตอนการเรียนรู้จะเสร็จสมบูรณ์เมื่อค่าผิดพลาดในการเรียนรู้มีค่าน้อยกว่าค่าผิดพลาดต่ำสุดที่ตั้งไว้ ($E<E_{min}$) ถ้า $E>E_{min}$ แล้วให้ E=0 และ p=1 เริ่มวงจรการเรียนรู้ใหม่ตามขั้นตอนที่ 2

การเรียนรู้โดยวิธีการเพิ่มโมเมนตัม

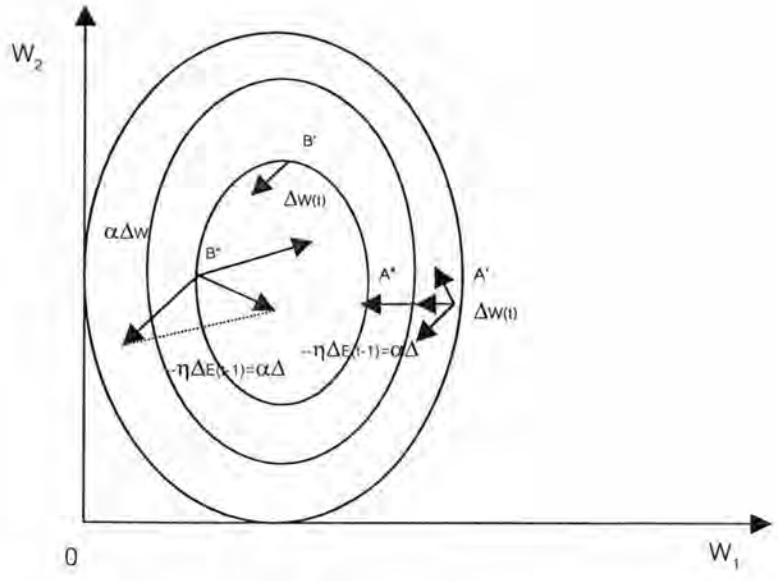
การเรียนรู้โดยการใช้โมเมนตัมช่วยเร่งให้การเรียนรู้ของข่ายงานเข้าสู่ค่าเป้าหมายได้เร็วขึ้น โดยการนำค่าน้ำหนักล่าสุดมาใช้ในการปรับหาค่าน้ำหนักในปัจจุบันตามขั้นตอนที่ 5 ด้วยสัดส่วนของค่าโมเมนตัมดังสมการ 3.58

$$\Delta w(t) = -\eta \nabla e(t) + \alpha \Delta w(t-1) \qquad ..(3.58)$$

เมื่อ t คือ จำนวนรอบของข้อมูลที่ฝึกข่ายงาน

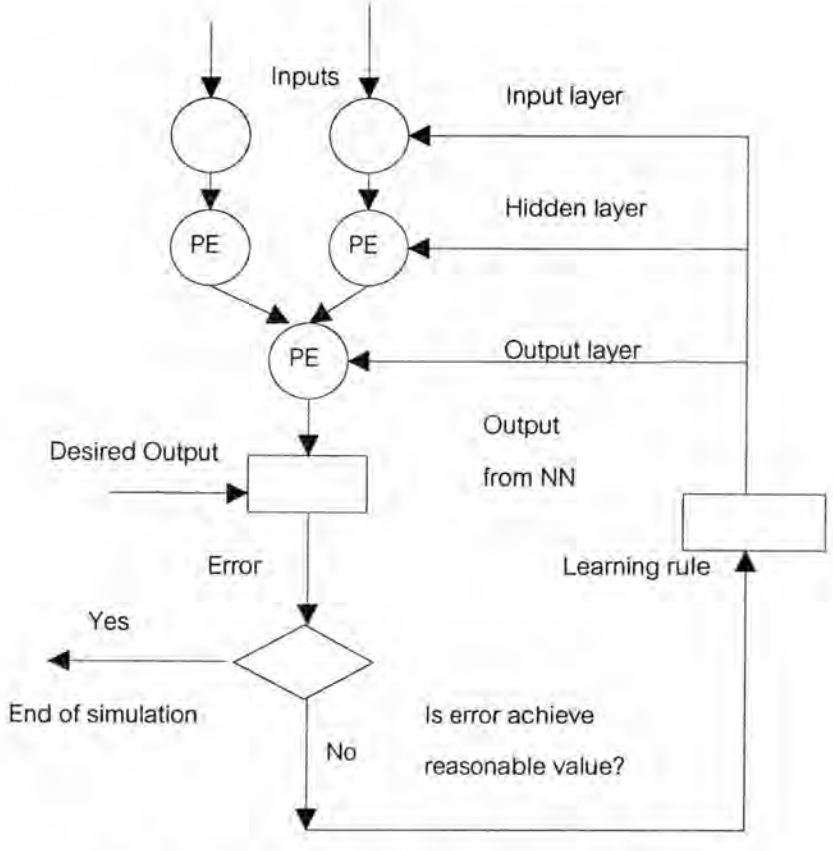
α คือแฟกเตอร์โมเมนตัมที่มักเลือกค่าอยู่ระหว่าง 0 –1

รูปที่ 3.15 แสดงวิธีการเร่งการเรียนรู้ของข่ายงานโดยการใช้โมเมนตัม และการเรียนรู้เริ่มต้นที่จุด a' ค่าเกรเดียนท์ของค่าโมเมนตัมมาใช้ในการปรับค่าน้ำหนัก ความผิดพลาดเมื่อเทียบกับค่าน้ำหนัก $\partial e/\partial w_1$ และ $\partial e/\partial w_2$ ที่จุด a' และ a" มีเครื่องหมายตรงกัน ดังนั้นองค์ประกอบเกรเดียนท์จะถูกนำมารวมกันซึ่งทำให้การเข้าสู่เป้าหมายรวดเร็วขึ้นโดยการปรับน้ำหนักที่จุด a" ซึ่งอาศัยค่าน้ำหนักที่จุด a' เข้ามาร่วมด้วยตามสัดส่วนของค่าโมเมนตัม



รูปที่ 3.15 การเรียนรู้แบบการกระจายความผิดพลาดย้อนกลับที่มีการรวมค่าแฟกเตอร์

ดังนั้นการเรียนรู้แบบการกระจายย้อนกลับสามารถอธิบายการทำงานได้ตามรูปที่ 3.16



รูปที่ 3.16 แสดงขั้นตอนการเรียนรู้ของข่ายงานนิวรัล

ชนิดของอัลกอริธึมการกระจายความผิดพลาดย้อนกลับ

1) Standard Backpropagation (Vanilla Backpropagation)

สมการอัลกอริธึม เสนอขึ้นโดย P.Werbos แสดงดังต่อไปนี้

$$\Delta w_{ij} = \eta \delta_j o_i \qquad ...(3.59)$$

$$\delta_j = f_j'(net_j)(t_j - o_j) \qquad \text{ถ้า } j \text{ คือ เอาท์พุทนิวรัล} \qquad ...(3.60)$$

หรือ

$$\delta_j = f_j'(net_j)\Sigma_k \delta_k w_{jk} \qquad \text{ถ้า } j \text{ คือ นิวรัลชั้นซ่อน} \qquad ...(3.61)$$

อัลกอริธึมนี้อาจเรียกว่า online backpropagation ได้เพราะว่าจะมีการปรับค่าน้ำหนักหลังจากมีการฝึกข่ายงานครบทุกชุดข้อมูล

2) Enhanced Backpropagation

Enhanced Backpropagation ใช้เทอมของโมเมนตัมเข้ามาช่วยและกำจัดจุดที่ไม่มีการเปลี่ยนแปลง (flat spot) ซึ่งเทอมของโมเมนตัมจะถูกนำเพิ่มในสมการของการปรับค่าน้ำหนักใหม่ โดยการนำไปรวมกับค่าน้ำหนักเก่า ซึ่งจะหลีกเลี่ยงปัญหาการแกว่งตัวเมื่อค่าพื้นผิวความผิดพลาดมีค่าแคบมากของอัลกอริธึมการกระจายความผิดพลาดย้อนกลับแบบมาตรฐาน การปรับน้ำหนักใหม่ของอัลกอริธึมนี้สามารถแสดงดังต่อไปนี้

$$\Delta w_{ij}(t+1) = \eta * \delta_j * o_i + \alpha \Delta w_{ij}(t) \qquad ...(3.62)$$

โดย α คือค่าคงที่ของโมเมนตัม

3) Backpropagation with Weight Decay

Weight Decay เสนอขึ้นโดย P.Werbos ซึ่งเป็นการลดน้ำหนักของการเชื่อมต่อโดยค่า d ของน้ำหนักเดิมดังสมการต่อไปนี้

$$\Delta w_{ij}(t+1) = \eta * \delta_j * o_i - d w_{ij}(t) \qquad ...(3.63)$$

ค่าน้ำหนักจะถูกลดเข้าใกล้ค่า 0 จนกว่าจะได้ข่ายงานที่ต้องการ

4) Batch Backpropagation

Batch Backpropagation จะใช้สูตรเหมือนกับ vanilla backpropagation ความแตกต่างคือการปรับค่าน้ำหนัก นั้นคือ vanilla backpropagation จะมีการปรับน้ำหนักในแต่ละชุดข้อมูลที่เรียนรู้ ในขณะที่ batch backpropagation จะทำการปรับน้ำหนักเมื่อครบทุกชุดข้อมูล (ทุก 1 รอบของชุดข้อมูล) การปรับค่าน้ำหนักวิธีนี้เหมาะสำหรับการฝึกข่ายงานแบบขนาน

5) Quickprop

วิธีนี้เร่งการเรียนรู้โดยการใช้เส้นโค้งของพื้นผิวความผิดพลาด ซึ่งใช้การคำนวณอนุพันธ์อันดับสองของฟังก์ชันความผิดพลาด Quickprop มีสมมติฐานว่าค่าพื้นผิวความผิดพลาดจะเป็นแบบควอดราติกและพยายามที่จะข้ามไปอีกขั้นตอนหนึ่งจากตำแหน่งปัจจุบันไปสู่ค่าน้อยที่สุดของรูปพาราโบลา

Quickprop จะคำนวณอนุพันธ์ในทิศทางการปรับน้ำหนัก หลังจากที่ได้คำนวณเกรเดียนท์แรกด้วยการกระจายความผิดพลาดย้อนกลับแบบมาตรฐาน ขั้นตอนที่จะลดค่าความผิดพลาดคือ

$$\Delta(t+1)\, w_{ij} = S(t+1)/(S(t)-S(t+1))\, \Delta\,(t)\, w_{ij} \qquad ..(3.64)$$

โดย w_{ij} คือค่าน้ำหนักการเชื่อมต่อระหว่างนิวรัล i และ j

$\Delta(t+1)$ คือการเปลี่ยนแปลงค่าน้ำหนักจริง

S(t+1) คืออนุพันธ์บางส่วนของฟังก์ชันความผิดพลาด$1\text{-}v'\, w_{ij}$

S(t) คืออนุพันธ์บางส่วนที่เป็นค่าสุดท้ายทของฟังก์ชันความผิดพลาด

6) Rprop

Rprop มีการเปลี่ยนแปลง 2 แบบ แบบแรกคือการปรับปรุงขั้นตอนการกระจายย้อนกลับที่ทำอย่างรวดเร็วคือถ้ามีการข้ามจุดต่ำสุดได้ แบบที่ 2 คือในเทอมของการทำให้น้ำหนักลดลง (weight-decay) ถูกนำเข้ามารวมด้วย ซึ่ง พารามิเตอร์ของการทำให้น้ำหนักลดลง คือ α (พารามิเตอร์การเรียนรู้ตัวที่ 3) ซึ่งมี 2 เป้าหมายคือ เพื่อลดค่าความผิดพลาดของเอาท์พุท (เป้าหมายทั่วไป) และ ลดขนาดของน้ำหนัก (เป้าหมายที่เพิ่มขึ้น) ฟังก์ชันความผิดพลาดแสดงดังต่อไปนี้

$$E = \sum(t_i-o_i)^2+10^{-\alpha}\sum w_{ij}^{\,2} \qquad ...(3.65)$$

เทอมของการทำให้น้ำหนักลดลง (weight decay, α) เป็นเทอมเอ็กซ์โปเนนเชียล ซึ่งจะทำให้ค่าน้ำหนักที่ลดลงเมื่อเปรียบเทียบกับอินพุทมีค่าเล็กมาก เช่น พารามิเตอร์การเรียนรู้ตัวที่ 3 (α) = 4 ค่าน้ำหนักที่ลดลง (weight decay) ต่อความผิดพลาดของเอาท์พุทคือ 1:10000

Rprop ย่อมาจาก Resilient backpropagation และ เป็นการเรียนรู้แบบ adaptive learning การเรียนรู้จะเป็นแบบชี้นำ ซึ่ง Rprop จะช่วยแก้ไขปัญหาเรื่องขนาดอนุพันธ์บางส่วนของค่าน้ำหนัก อนุพันธ์ที่จะแสดงนี้จะแสดงในทิศทางการปรับค่าน้ำหนัก ขนาดของน้ำหนักที่เปลี่ยนแปลงหามาจากค่าน้ำหนักเฉพาะ ที่เรียกว่า update-value $\Delta_{ij}^{(t)}$

ขั้นตอนที่ 1 การหาขนาดของน้ำหนักที่จะเปลี่ยนแปลง

$$\Delta w_{ij}^{(t)} = -\Delta_{ij}^{(t)} \text{ , if } \partial E/\partial w_{ij}^{(t)} > 0$$
$$= +\Delta_{ij}^{(t)} \text{ , if } \partial E/\partial w_{ij}^{(t)} < 0$$
$$= 0 \text{ , if else}$$

โดย $\partial E/\partial w_{ij}^{(t)}$ คือผลรวมของเกรเดียนทุกชุดข้อมูล

ถ้าแทน $\Delta_{ij}^{(t)}$ โดยค่าคงที่ update-value Δ สมการนี้จะเรียกว่า "Manhattan-update rule"

ขั้นตอนที่ 2 ของการเรียนรู้ Rprop คือการหา update-value ใหม่ $\Delta_{ij}^{(t)}$ ดังสมการต่อไปนี้

$$\Delta_{ij}^{(t)} = \eta^{+} * \Delta_{ij}^{(t-1)} \text{ ,if } \partial E/\partial w_{ij}^{(t-1)} * \partial E/\partial w_{ij}^{(t)} > 0$$
$$= \eta^{-} * \Delta_{ij}^{(t-1)} \text{ ,if } \partial E/\partial w_{ij}^{(t-1)} * \partial E/\partial w_{ij}^{(t)} < 0 \quad \text{...(3.66)}$$
$$= \Delta_{ij}^{(t-1)} \text{ ,if else}$$

ซึ่ง $0<\eta^{-}<1<\eta^{+}$

วิธีการปรับมีดังต่อไปนี้ ทุกอนุพันธ์บางส่วนของการเปลี่ยนแปลงค่าน้ำหนัก w_{ij} จะเปลี่ยนตามเครื่องหมายซึ่งแสดงว่าการปรับน้ำหนักครั้งสุดท้ายมีค่ามากเกินไป และอัลกอริธึมก็ข้ามจุดต่ำสุดเฉพาะที่ (local minimum) update-value $\Delta_{ij}^{(t)}$ จะลดลงโดยค่า η^{-} ถ้าอนุพันธ์มีเครื่องหมายตรงกันข้าม update-value จะมีค่าเพิ่มขึ้นเพื่อที่จะเร่งเข้าสู่ค่าต่ำสุด ในกรณีที่ $\partial E/\partial w_{ij}^{(t-1)} = 0$ จะไม่มีการปรับตัวเกิดขึ้นในขั้นตอนการเรียนรู้ และเพื่อจะลดจำนวนของพารามิเตอร์จึงทำการกำหนดค่า η ให้อยู่ในช่วง $\eta^{-} = 0.5$, $\eta^{+} = 1.2$ Rprop พยายามที่จะทำการปรับกระบวนการเรียนรู้โดยใช้ฟังก์ชันความผิดพลาด ซึ่งเรียกว่า batch learning

หรือ learning by epoch ซึ่งหมายถึงการปรับค่าน้ำหนักจะถูกทำหลังจากได้เกรเดียนท์ของชุดข้อมูลทั้งหมดแล้ว

อัลกอริธึมการกระจายความผิดพลาดย้อนกลับแบบ Levenberg- Marquardt

วิธี Levenberg-Marquardt เป็นวิธีการประมาณแบบ Newton's method อัลกอริธึมนี้ใช้อนุพันธ์อันดับสองเป็นฟังก์ชันความผิดพลาด (cost function) ดังนั้นการเข้าสู่ค่าที่ต้องการจึงดีกว่าวิธีเกรเดียนเดสเซนท์ โดยเฉพาะดีกว่าอนุพันธ์อันดับหนึ่ง ซึ่งอัลกอริธึมนี้สามารถแสดงดังสมการต่อไปนี้

$$e = d - F(\phi,u) \qquad ...(3.67)$$

$$J = e^2/2 \qquad ...(3.68)$$

$$\Delta\phi = -(\nabla^2 J(\phi))^{-1}\nabla J(\phi) \qquad .(3.69)$$

โดย $\nabla^2 J(\phi)$ คือ Hessian matrix

$\nabla J(\phi)$ คือ เกรเดียนความผิดพลาดของ j

$$J_s = \begin{bmatrix} \frac{\partial e_1}{\partial \phi_1} & \cdots & \frac{\partial e_1}{\partial \phi_B} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_L}{\partial \phi_1} & \cdots & \frac{\partial e_L}{\partial \phi_B} \end{bmatrix} \qquad ...(3.70)$$

ค่าความผิดพลาดในสมการที่ 3.67 จะทำการลดลงให้น้อยที่สุดดังสมการที่ 3.68 โดยการใช้กฎในสมการที่ 3.69 เป้าหมายในการลดลงให้น้อยที่สุดดังสมการที่ 3.68 ถ้าใช้สมการ Taylor series ในการกระจาย $e(\phi)$ รอบจุดนั้นๆ สมการอนุพันธ์อันดับที่ 1 ของ Jacobian แสดงดังสมการ 3.70 และการปรับพารามิเตอร์แสดงในสมการที่ 3.71

$$\Delta\phi = N_\phi = -(J_s^T J_s + uI)^{-1} J_s^T e \qquad ..(3.71)$$

โดย B คือ จำนวนพารามิเตอร์ที่สามารถปรับได้

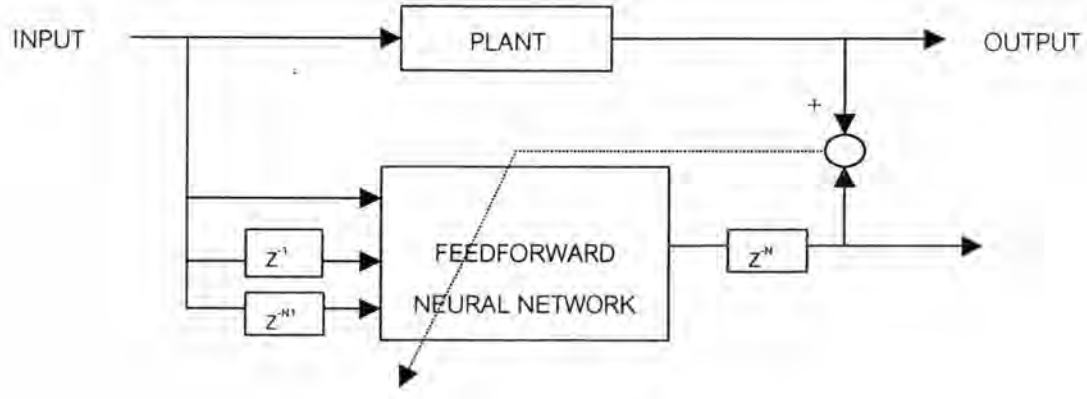
L คือ จำนวนเอาท์พุท

3.3.2.6 แบบจำลองกระบวนการของข่ายงานนิวรัล

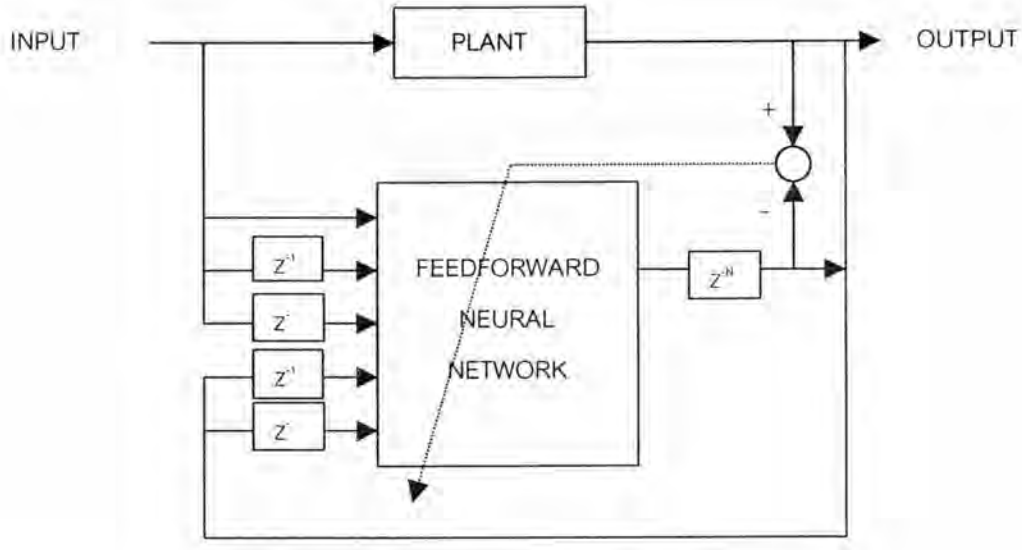
แบบจำลองที่ใช้ในการหาแบบจำลองกระบวนการของข่ายงานนิวรัลประกอบด้วยแบบจำลอง 2 แบบที่มีความแตกต่างกันในเรื่องของการจัดการข้อมูลอินพุทกับเอาท์พุทที่สอดคล้องกับการหน่วงเวลาก่อนที่จะป้อนให้ข่ายงานนิวรัลได้เรียนรู้ แบบจำลองทั้งสอง ได้แก่ แบบจำลองแบบไม่กลับกระแสและแบบจำลองแบบกลับกระแส โดยกระบวนการที่ใช้อาจเป็นแบบสถิต (static) หรือ พลศาสตร์ (dynamic) ดังต่อไปนี้

ก. แบบจำลองแบบไม่กลับกระแส (non-recurrent model) ใช้สำหรับจำลองกระบวนการแบบสถิต (static process) ซึ่งเอาท์พุทเป็นฟังก์ชันกับอินพุทที่เวลาปัจจุบัน และในอดีต และอินพุทเวกเตอร์ของข่ายงานนี้เป็นค่าอินพุทปัจจุบันและอินพุทในอดีตซึ่งแสดงในรูปที่ 3.17 การหาแบบจำลองของระบบนี้เป็นการนำข้อมูลที่ใช้ป้อนเข้าสู่กระบวนการมาหน่วงเวลาตามช่วงระยะเวลาที่ต้องการ จำนวนของการหน่วงเวลาเท่ากับจำนวนนิวรัลในชั้นอินพุท โดยที่ไม่มีค่าเอาท์พุทจากกระบวนการมาใช้เพื่อเป็นข้อมูลในการป้อนให้กับนิวรัลในชั้นอินพุท

ข. แบบจำลองแบบไม่กลับกระแส (recurrent model) ใช้สำหรับกระบวนการทางพลศาสตร์ (dynamic process) ซึ่งเอาท์พุทปัจจุบันของข่ายงานขึ้นกับค่าอินพุทในปัจจุบัน , อินพุทในอดีตและเอาท์พุทในอดีตซึ่งแสดงในรูปที่ 3.18 จำนวนของการหน่วงเวลาที่ต้องการจะเป็นอินพุทและเอาท์พุททั้งเส้นทางป้อนไปข้างหน้าและป้อนกลับ ซึ่งขึ้นอยู่กับคุณสมบัติของกระบวนการ การหาแบบจำลองของระบบนี้ โดยการนำข้อมูลที่ป้อนเข้าสู่ระบบกับเอาท์พุทของกระบวนการมาหน่วงเวลาก่อนป้อนเข้าสู่ชั้นอินพุทของข่ายงาน ซึ่งวิธีการนี้เป็นที่นิยมมากที่สุด เนื่องจากค่าเป้าหมายในอดีตของระบบจะเป็นตัวช่วยกำหนดแนวทางในการเรียนรู้โดยทำให้ทิศทางในการหาคำตอบได้รวดเร็ว ดังนั้นจึงใช้วิธีการนี้สร้างแบบจำลอง ในงานวิจัยนี้



รูปที่ 3.17 แบบจำลองแบบไม่กลับกระแส



รูปที่ 3.18 แบบจำลองแบบกลับกระแส

3.3.2.7 วงจรการทำงานของอัลกอริธึมแบบการกระจายความผิดพลาดย้อนกลับ

วงจรการทำงานของอัลกอริธึมแบบการกระจายความผิดพลาดย้อนกลับมี 3 ขั้นตอนหลักๆคือ

(1) ขั้นตอนการกำหนดโครงสร้างของข่ายงานนิวรัล (definition phase) คือจะมีการกำหนดโครงสร้างของข่ายงานนิวรัล ได้แก่ จำนวนชั้นในข่ายงาน จำนวนนิวรัลในแต่ละชั้น การเชื่อมต่อระหว่างนิวรัล การกำหนดค่าของน้ำหนักและไบอัส กำหนดอัตราการเรียนรู้ กำหนดค่าโมเมนตัม

(2) ขั้นตอนการฝึกข่ายงาน (training phase) จะนำชุดข้อมูลที่ประกอบไปด้วยอินพุทและค่าเป้าหมายมาใช้ในการฝึกข่ายงาน โดยจะมีการฝึกข่ายงานให้เรียนรู้โดยการปรับค่าน้ำหนัก จนกระทั่งค่าความผิดพลาด(ผลต่างของค่าเอาท์พุทที่ได้จากข่ายงานกับค่าเป้าหมาย) จะอยู่ในระดับที่ยอมรับได้จึงถือว่าเสร็จสิ้นสำหรับการฝึกข่ายงาน

(3) ขั้นตอนการทดสอบ (testing phase) ข่ายงานเมื่อได้รับการฝึกข่ายงานอย่างเหมาะสมในขั้นตอนที่ 2 เรียบร้อยแล้ว จะนำแบบจำลองนั้นมาทดสอบโดยใช้อินพุทอีกชุดหนึ่งในการทดสอบเพื่อสังเกตผลของค่าเอาท์พุทของแบบจำลองข่ายงานนิวรัลมาเปรียบเทียบกับค่าเป้าหมายว่าอยู่ในระดับที่ยอมรับได้หรือไม่ โดยไม่ต้องมีการปรับค่าน้ำหนักย้อนกลับไปในข่ายงาน

การเรียนรู้มากๆจะเกิดขึ้นเมื่อแบบจำลองเริ่มที่จะเรียนรู้ แต่ว่าถ้าชุดข้อมูลมีการเรียนรู้มากเกินไป ประสิทธิภาพการเรียนรู้ในชุดข้อมูลเหล่านั้นเพิ่มขึ้นแต่ประสิทธิภาพในชุดทดสอบจะลดลงไปซึ่งต้องขึ้นอยู่กับความเหมาะสมของการฝึกข่ายงานและความพึงพอใจของค่าความผิดพลาดระหว่างค่าเป้าหมายกับค่าเอาท์พุทที่

ได้จากการทำนายจากข่ายงาน ส่วนใหญ่จะพิจารณาค่าผิดพลาดในเทอมของ ผลรวมของค่าความผิดพลาดยกกำลังสอง (sum squared error) ชุดทดสอบ

3.3.2.7.1 วิธีการฝึกข่ายงาน

การฝึกข่ายงานคือกระบวนการหาค่าออฟติไมซ์ของน้ำหนักที่เชื่อมต่อระหว่างนิวรัลแต่ละชั้นและค่าไบแอส เริ่มต้นที่การกำหนดค่าน้ำหนักเริ่มต้นทั้งหมดของข่ายงานด้วยตัวเลขสุ่มที่มีค่าน้อย ๆ (ทั้งค่าบวกและลบ) กระบวนการฝึกจะทำการวนซ้ำไปหลายรอบจนกระทั่งค่าความผิดพลาดระหว่างค่าเป้าหมายและเอาท์พุทที่ทำนายได้จากข่ายงานจะเป็นที่พอใจ ซึ่งการคำนวณแต่ละรอบ(ครบทุกชุดข้อมูล)จะเรียกว่า epoch ค่าเอาท์พุทจากข่ายงานที่สัมพันธ์กับชุดข้อมูลอินพุททั้งหมดในชุดข้อมูลการฝึกจะถูกทำนายออกมาและค่าน้ำหนักจะถูกปรับในทิศทางลดค่าความผิดพลาดลง ค่าน้ำหนักจะถูกปรับสำหรับทุกชุดข้อมูลและทุกรอบเพื่อให้เข้าสู่ค่าออฟติไมซ์

ค่า nt = จำนวนของชุดข้อมูล x,y โดย x คือเมตริกซ์ของอินพุทและ y คือเมตริกซ์ของค่าเป้าหมาย แล้วในหนึ่งรอบหมายถึงการป้อนข้อมูลจำนวน nt ชุด

ค่าความผิดพลาดของชั้นเอาท์พุทที่ได้ในแต่ละรอบจะกระจายย้อนกลับลงมาที่ชั้นซ่อนที่ถัดลงมาจากชั้นเอาท์พุทโดยจะกระจายย้อนกลับลงมาถึงชั้นอินพุทเพื่อปรับค่าน้ำหนักให้ค่าความผิดพลาดของค่าเป้าหมายและค่าเอาท์พุทจากข่ายงานมีค่าลดลง

3.3.2.7.2 การทดสอบข้อมูล (Model Validation)

ในการทดสอบข้อมูลจะมีการนำชุดทดสอบเข้ามาป้อนเป็นอินพุทเพื่อทดสอบเอาท์พุทกับค่าเป้าหมายว่ามีค่าความผิดพลาดอยู่ในช่วงที่ยอมรับได้หรือไม่โดยไม่มีการปรับน้ำหนัก

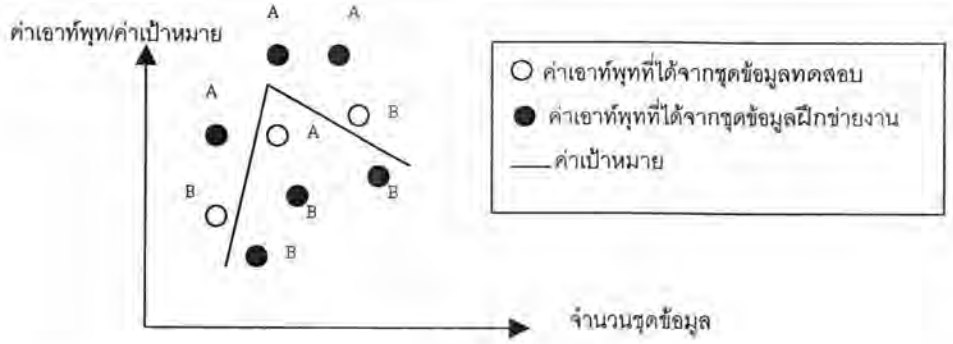
แบบจำลองที่เรียนรู้ชุดข้อมูลมากเกินไป ประสิทธิภาพการเรียนรู้ในชุดข้อมูลจะเพิ่มขึ้น แต่ประสิทธิภาพในชุดข้อมูลทดสอบจะลดลง ซึ่งต้องขึ้นอยู่กับความเหมาะสมการฝึกข่ายงานและความพึงพอใจในผลการทดสอบ

3.3.2.8 ปัญหาที่พบสำหรับการฝึกข่ายงาน

ก. การประมาณระหว่างเส้นโค้งมากเกินไป (Overfitting)

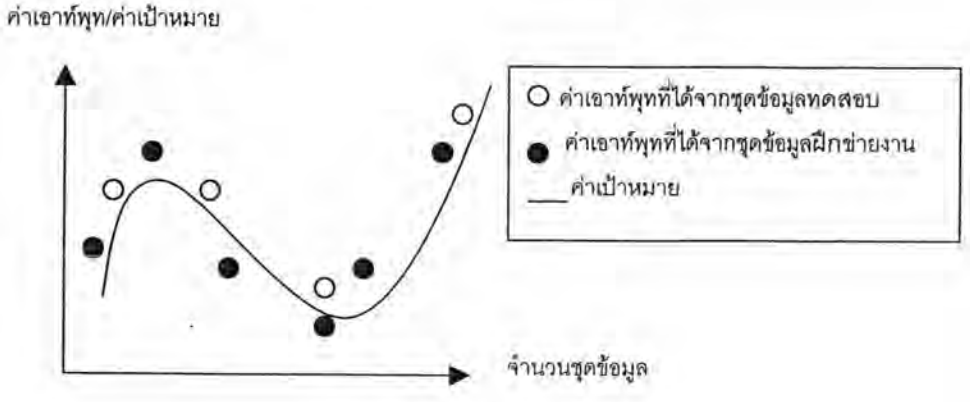
ข่ายงานมีการเรียนรู้มากเกินไปทำให้ประสิทธิภาพในการเรียนรู้ชุดข้อมูลในการทดสอบหรือข้อมูลที่ป้อนเข้ามาใหม่น้อยลงถึงแม้ว่าประสิทธิภาพในการเรียนรู้ชุดข้อมูลในการฝึกเพิ่มขึ้นก็ตาม

ในบางข้อมูลของชุดทดสอบจะถูกแบ่งผิดขอบเขตปัญหาคือ ข่ายงานมีอิสระมากเกินไป และมีการยึดติดตามข้อมูลของการฝึกข่ายงานมากเกินไป รูปที่ 3.19 แสดงการประมาณค่าระหว่างเส้นอย่างเหมาะสม

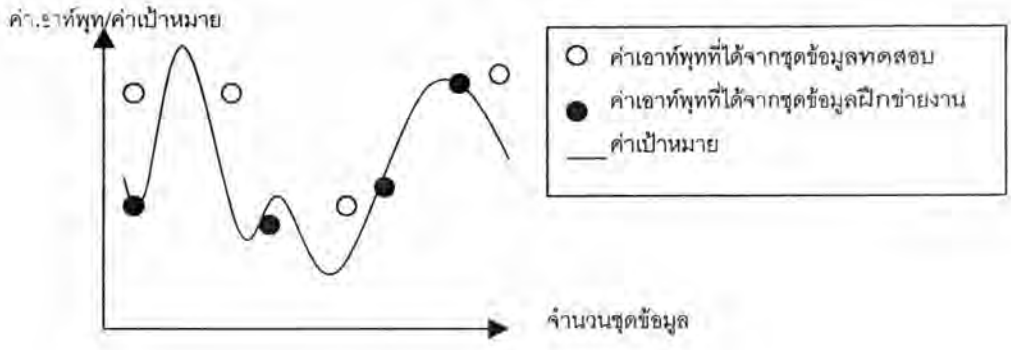


รูปที่ 3.19 ข่ายงานมีการประมาณค่าระหว่างเส้นตรงอย่างเหมาะสม

ในกราฟรูปที่ 3.20 แสดงเอาท์พุทที่ตอบสนองกับอินพุท แทนที่ด้วยจุดทึบ และจุดกลวงจะเป็นข้อมูลที่ใช้ในการทดสอบข่ายงาน และเส้นโค้งก็มีการพาดผ่านจุดของข้อมูลฝึกข่ายงาน ความผิดพลาดจะเข้าใกล้ศูนย์ แต่ถ้ามีการเปลี่ยนแปลงโครงสร้างข่ายงานเช่นจำนวนชั้นซ่อน เอาท์พุทอาจจะใกล้เคียงกับชุดเป้าหมายมากขึ้นก็ได้ ถ้ามีจำนวนนิวรัลในชั้นซ่อนมาก ข่ายงานมีอิสระมาก เอาท์พุทสามารถเปลี่ยนแปลงอย่างรวดเร็วเมื่อเกิดการเปลี่ยนแปลงอินพุท จึงจะได้กราฟรูปที่ 3.21



รูปที่ 3.20 ข่ายงานมีการประมาณค่าระหว่างเส้นโค้งอย่างเหมาะสม

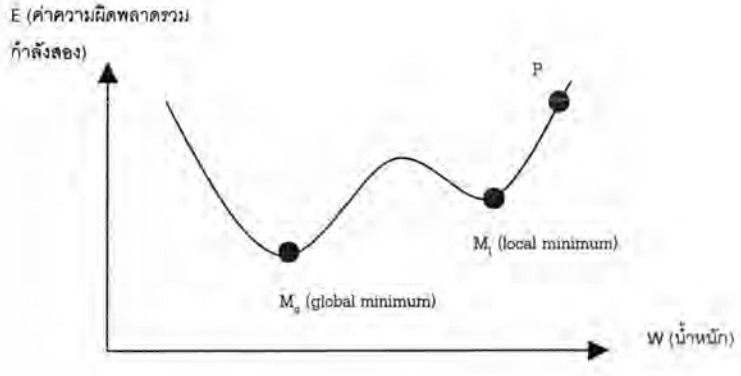


รูปที่ 3.21 ข่ายงานมีการแบ่งประเภทผิดเนื่องจากมีการประมาณค่าระหว่างเส้นโค้งมากเกินไป (overfitting)

จากกราฟจะพบว่าเส้นโค้งพาดผ่านแต่ละชุดข้อมูลฝึกข่ายงาน ซึ่งมีค่าความผิดพลาดเข้าใกล้ศูนย์ แต่สำหรับชุดข้อมูลทดสอบจะไม่อยู่ตำแหน่งใกล้เคียงกับกราฟฝึกข่ายงาน ข่ายงานนี้มีการเรียนรู้ที่ไม่ดี

ข . ค่าความผิดพลาดเฉพาะที่ (local minima)

พิจารณาดังรูป 3.22 ซึ่งเริ่มต้นจากการกำหนดค่าน้ำหนักที่จุด P ถ้าทำการปรับน้ำหนักตามเกรเดียนท์เดสเซนท์ จะพบว่าค่าต่ำสุดที่เข้าหาคือ M_l ซึ่งไม่ใช่ M_g ซึ่งเรียกว่า ความผิดพลาดเฉพาะที่ (local minimum) และสัมพันธ์กับคำตอบของข่ายงานเพียงบางส่วนสำหรับชุดข้อมูลในการฝึกข่ายงาน M_g คือ ค่าความผิดพลาดจริง (global minimum) ซึ่งจะต้องการหา จนกว่าจะหนีออกจากค่า M_l ดังนั้นไม่เคยมาถึงจุด M_g นี้



รูปที่ 3.22 ความผิดพลาดเฉพาะที่ (local minima)

3.4 บทสรุป

อัลกอริธึมการกระจายความผิดพลาดย้อนกลับจะสามารถใช้กับข่ายงานหลายชั้นแบบป้อนไปข้างหน้าทั้งข่ายงานที่เป็นแบบเชิงเส้นและไม่เป็นเชิงเส้นโดยสามารถใช้ร่วมกันฟังก์ชันกระตุ้นได้หลายชนิด รวมทั้งเหมาะกับการแก้ปัญหาที่เป็นการประมาณฟังก์ชันหรือการแบ่งประเภทของข้อมูล ซึ่งลักษณะของจำนวนอินพุทและจำนวนเอาท์พุทเป็นข้อมูลที่ต้องการสำหรับข่ายงาน ส่วนจำนวนชั้นซ่อนและจำนวนนิวรัลในแต่ละชั้นซ่อนจะขึ้นอยู่กับการออกแบบที่เหมาะสม

ในประเภทของอัลกอริธึมการกระจายความผิดพลาดย้อนกลับแล้ว อัลกอริธึมแบบมาตรฐานจะใช้เวลาฝึกข่ายงานช้ามาก เพราะต้องใช้การอัตราการเรียนรู้น้อยสำหรับการเรียนรู้ให้เสถียร และความผิดพลาดของข่ายงานจะตกอยู่ที่จุดความผิดพลาดเฉพาะที่ (local minima) มากกว่าจุดผิดพลาดต่ำสุดจริง (global minima) ซึ่งการตกอยู่ที่จุดผิดพลาดเฉพาะที่จะดีหรือไม่ขึ้นอยู่กับว่าจุดผิดพลาดเฉพาะที่อยู่ใกล้กับจุดผิดพลาดจริงมากเท่าใด และความต้องการค่าความผิดพลาดมีค่าขนาดต่ำเท่าใด ส่วนปัญหาที่ควรป้องกันคือปัญหาการประมาณระหว่างเส้นโค้งมากเกินไปนั้น คือ ถ้ามีนิวรัลในชั้นซ่อนมาก ข่ายงานจะมีความเป็นอิสระมาก (ตัวแปรที่ต้องการออฟติไมซ์ก็มีมากขึ้น) หรือเกิดการประมาณระหว่างเส้นโค้งมากเกินไป (overfitting) แต่นิวรัลเพิ่มขึ้นก็จะให้ค่าความผิดพลาดเฉพาะที่มีค่าต่ำลง อย่างไรก็ตามถ้ามีการเพิ่มเทอมโมเมนตัมเข้าในอัลกอริธึมจะช่วยลดความน่าจะเป็นที่ความผิดพลาดของข่ายงานจะตกบนค่าความผิดพลาดเฉพาะที่ที่มีค่ามากๆ และสามารถช่วยลดเวลาในการฝึกข่ายงานเนื่องจากมีการเรียนรู้แบบ adaptive learning เข้ามาช่วย เนื่องจากจะทำให้ค่าอัตราการเรียนรู้มีค่ามากขึ้น อัตราการเรียนรู้สามารถทำได้อย่างรวดเร็วและยังทำให้ข่ายงานยังสามารถเสถียรอยู่ แต่วิธีที่สามารถช่วยเร่งการเรียนรู้ที่ดีที่สุดคือ Levenberg-Marquardt optimization เป็นวิธีที่ช่วยเร่งการเรียนรู้จากวิธีอัลกอริธึมมาตรฐานสามารถลดเวลาในการฝึกข่ายงานได้ถึง 78 เท่าแต่มีข้อเสียเนื่องจากต้องใช้หน่วยความจำมาก ดังนั้นในงานวิจัยนี้จึงเลือกใช้อัลกอริธึมแบบ Levenberg-Marquardt กับโครงสร้างข่ายงานป้อนไปข้างหน้าหลายชั้นโดยใช้แบบจำลองแบบกลับกระแสจึงเป็นโครงสร้างที่คัดเลือกว่าเหมาะสมกับงานวิจัยนี้