บทที่ 2

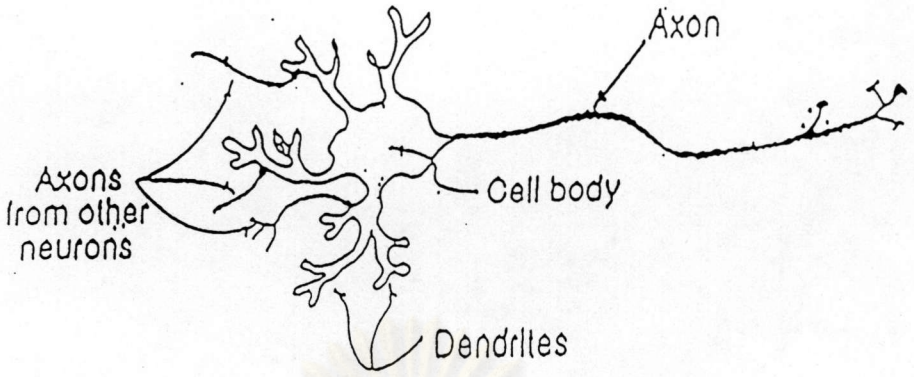
หลักการและทฤษฎีพื้นฐานของระบบนิวรอลเน็ตเวอร์ก

ในบทนี้จะขอกล่าวถึงประวัติความเป็นมาของระบบนิวรอลเน็ตเวอร์ก หลักการทำงานและการแบ่งประเภทของระบบนิวรอลเน็ตเวอร์กชนิดที่ใช้ในวิทยานิพนธ์ เพื่อเป็นพื้นฐานให้เข้าใจก่อนที่จะศึกษางานวิจัยในบทต่อๆไป

ความเป็นมาของระบบนิวรอลเน็ตเวอร์ก

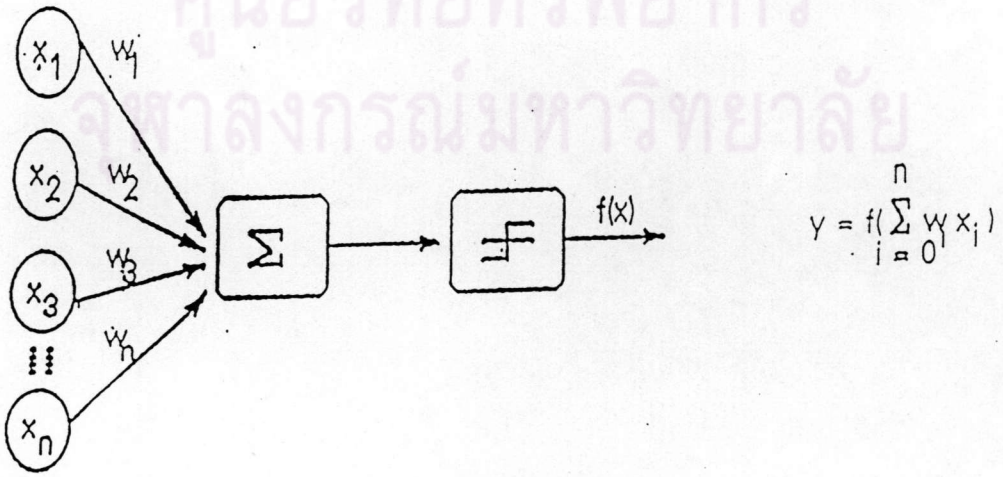
ระบบนิวรอลเน็ตเวอร์กเป็นโมเดลคอมพิวเตอร์ที่มีการเลียนแบบโครงสร้างภายในให้คล้ายคลึงกับลักษณะโครงสร้างของระบบประสาทมนุษย์ โดยการประยุกต์ใช้ความรู้และเทคโนโลยี ส่วนหนึ่งที่ได้มาจากวิชาชีววิทยาและชีวฟิสิกส์ที่เกี่ยวข้องกับเซลล์สมองมนุษย์ (Neurophysiology) รูปแบบคอมพิวเตอร์ที่กล่าวถึงจึงมีชื่อเรียกกันได้มากมายเช่น ระบบประมวลผลขนานแบบกระจายอำนาจ (Parallel Distributed Processing), โมเดลการเชื่อมโยง (Connectionist Model), ระบบประสาทประดิษฐ์ (Artificial Neural System) เป็นต้น

การจำลองแบบการทำงานของระบบนิวรอลเน็ตเวอร์กในรูปของฟังก์ชันทางคณิตศาสตร์ โดยการจำลองการทำงานจากเซลล์ประสาทในสมองของมนุษย์ จากรูปที่ 2.1 เซลล์นิวรอนรับข้อมูลอินพุตจากเซลล์นิวรอนเซลล์อื่น(Artificial Neural Systems,Jacek M.Zurada,1992) โดยผ่านทางจุดเชื่อมโยงระหว่างเซลล์ที่เรียกว่า ไซแนพส์ (Synapse) สัญญาณข้อมูลจากไซแนพส์จะถูกส่งผ่านเดนไดรต์ (Dendrite) ซึ่งจะอยู่รอบข้างของตัวเซลล์ (Cell body) สัญญาณข้อมูลอินพุตจะได้รับการประมวลภายใน จากนั้นสัญญาณข้อมูลเอาท์พุตจากเซลล์นิวรอนจะถูกส่งออกมาทางส่วนของเอ็กซอน (Axon) ซึ่งจะส่งสู่เดนไดรต์ต่อไป ส่วนวิธีการ การประมวลผลภายในโดยเซลล์นิวรอนแต่ละเซลล์มีจุดเชื่อมโยงระหว่างอินพุตและเอาท์พุตแล้วยังมีหน้าที่ขยายหรือลดขนาดของสัญญาณ ซึ่งมีฟังก์ชันการทำงาน 2 ลักษณะ คือ การกระตุ้น (Excitatory) เป็นการทำให้สัญญาณที่ผ่านมามีความถี่สูงขึ้น และการยับยั้ง (Inhibitory) เป็นการทำให้สัญญาณที่ผ่านมามีความถี่ลดลง ซึ่งแบบจำลองระบบนิวรอลเน็ตเวอร์กอัตราขยายหรือลดถูกกำหนดด้วยค่าน้ำหนัก (Weight)



รูปที่ 2.1 โครงสร้างพื้นฐานเซลล์นิวรอนของสมองมุนษย์

การพัฒนาของระบบนิวรอลเน็ตเวอร์ก(Artificial Neural Systems,Jacek M.Zurada,1992) ได้เริ่มต้นขึ้นใน ค.ศ. 1943 โดย McCulloch และ Pitts ได้แสดงถึงการทำงานระบบนิวรอลเน็ตเวอร์กในรูปของการคำนวณตรรกศาสตร์ต่อมาใน ค.ศ. 1949 Donald Hebb ได้เสนอ Hebbian learning rule เป็นส่วนในการพัฒนาทฤษฎีของนิวรอลเน็ตเวอร์กใน ค.ศ. 1958 Frank Rosenblatt ได้ประดิษฐ์เครื่องมือที่เรียกว่า Perceptron ซึ่ง เป็นเครื่องมือที่สามารถเรียนรู้ในการแยกแยะรูปแบบต่างๆ ซึ่งโครงสร้างไดอะแกรมของ Perceptron จะทำการประมวลผลโดยหาผลบวกทั้งหมดของผลของคูณระหว่างอินพุตและค่าน้ำหนัก ถ้ามีค่ามากกว่าศูนย์แล้วผลลัพธ์เอาท์พุต(y)จะเป็น 1 และถ้ามีค่าน้อยกว่าหรือเท่ากับ 0 ผลลัพธ์เอาท์พุต(y)จะเป็น 0 แสดงในรูปที่ 2.2



รูปที่ 2.2 โครงสร้าง Perceptron

ต่อมาใน ค.ศ.1960 Bernard Widraw และ Matcian Hoff ได้ประดิษฐ์ ADALINE (ADAptive LINEar comtiner) และ Widrow-Hoff learning rule ซึ่งพัฒนาอัลกอริทึม ระบบนิวรอลเน็ตเวอร์ก เพื่อใช้ในการแยกแยะรูปแบบต่างๆ ใน ค.ศ. 1972 S.Amari ได้ศึกษาและพัฒนาทฤษฎีคณิตศาสตร์ สำหรับระบบนิวรอลเน็ตเวอร์กใน ค.ศ. 1980 Kunihiko Fukushima และ Miyaka ได้พัฒนาระบบนิวรอลเน็ตเวอร์กขึ้นเป็น neocognitrons เป็นการจำลองการทำงานการรับรู้ภาพของเรตินาในดวงตาของมนุษย์ และใช้ในการรับรู้ภาพ 2 มิติ ใน ค.ศ. 1982 T.Kohonen และ James A.Anderson ได้ทำการวิจัยเกี่ยวกับการใช้งานเป็นหน่วยความจำแบบ Associative ในปีเดียวกันนี้ T.Kohonen ได้พัฒนา Self organizing map ซึ่งเป็นเน็ตเวอร์กที่เรียนรู้แบบไม่มีผู้ช่วย (Unsupervised learning) และ John Hopfield ได้แนะนำ Recurrent neural network สำหรับการใช้งานเป็นหน่วยความจำแบบ Associative ใน ค.ศ. 1985 Rumelhart, Hinton และ Williams ได้พัฒนาอัลกอริทึม back propagation ซึ่งเป็นนิวรอลเน็ตเวอร์กชนิดหลายชั้นและในการเรียนรู้แบบมีผู้ช่วย (Supervised learning) นอกจากนี้แล้วยังมีการพัฒนาระบบนิวรอลเน็ตเวอร์กแต่ละชนิดให้มีประสิทธิภาพดีขึ้นอีกมากมาย

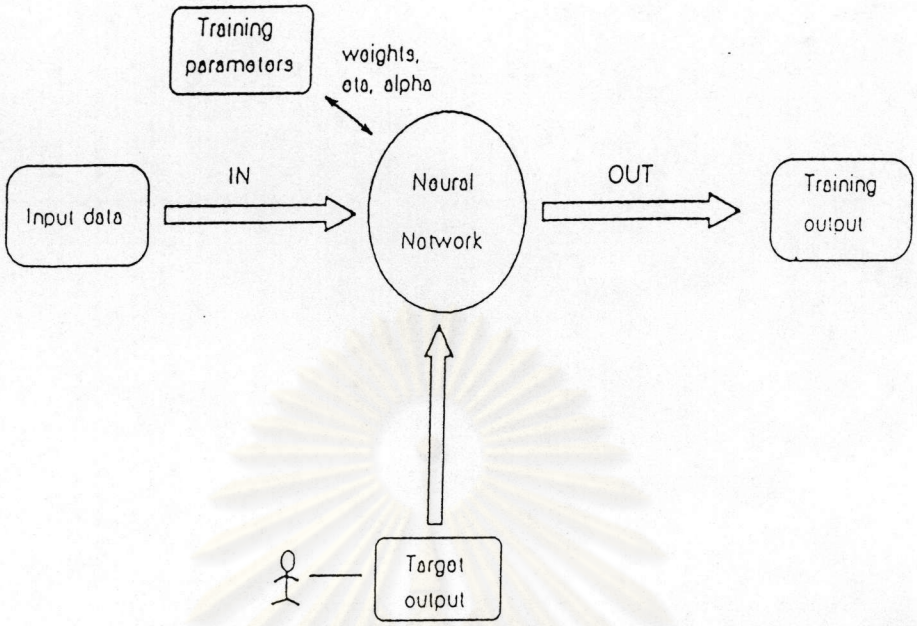
ประเภทของระบบนิวรอลเน็ตเวอร์ก

ระบบนิวรอลเน็ตเวอร์กได้ถูกพัฒนาขึ้นมากมายหลายชนิด แต่ละชนิดจะมีคุณสมบัติที่จะนำไปประยุกต์ใช้งานที่เหมาะสมแตกต่างกัน ในการจัดแบ่งประเภทระบบนิวรอลเน็ตเวอร์กแต่ละชนิดได้จัดแบ่งตามคุณสมบัติที่สำคัญ ดังนี้

1. <u>การเรียนรู้ (Learning) ของระบบนิวรอลเน็ตเวอร์กมี 2 แบบ คือ</u>

1.1 <u>การเรียนรู้แบบมีผู้ช่วย (Supervised Learning)</u>

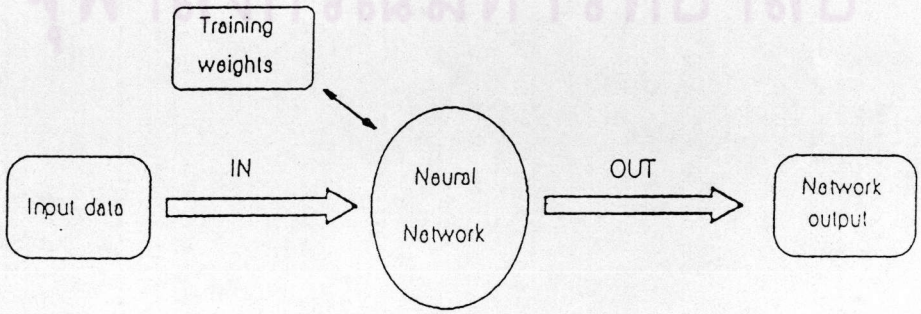
ขั้นตอนการเรียนรู้นี้ เน็ตเวอร์กจะได้รับชุดข้อมูลตัวอย่าง(pattern) ซึ่งประกอบด้วยข้อมูลอินพุต และเอาท์พุตเป้าหมาย (Target output) ข้อมูลอินพุตที่ป้อนเข้าสู่เน็ตเวอร์ก เพื่อประมวลผลแล้วนำเอาท์พุตที่ได้จากการฝึก (Training output) มาเปรียบเทียบกับเอาท์พุตที่กำหนด ความแตกต่างระหว่างค่าทั้งสองนี้จะถูกนำไปปรับค่าพารามิเตอร์ต่างๆเพื่อให้ระบบนิวรอลเน็ตเวอร์กสามารถเรียนรู้ข้อมูลอินพุตและเอาท์พุตเป้าหมายได้ตามที่กำหนด แสดงให้จากรูปที่ 2.3



รูปที่ 2.3 ไดอะแกรมของ Supervised Learning

1.2 <u>การเรียนรู้แบบไม่มีผู้ช่วย (Unsupervised learning)</u>

ขั้นตอนการเรียนรู้ จะเป็นการเรียนรู้ด้วยข้อมูลอินพุตเท่านั้น ดังนั้นการเรียนรู้จึงได้จากการส่งข้อมูลอินพุตเข้าสู่ระบบนิวรอลเน็ตเวอร์ก และการปรับค่าพารามิเตอร์จึงใช้เฉพาะเอาท์พุตจากเน็ตเวอร์กที่ได้จากการประมวลผลของเน็ตเวอร์ก แสดงได้จากรูปที่2.4

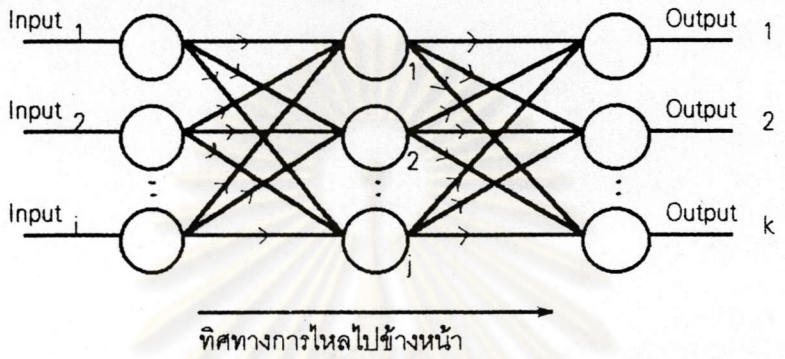


รูปที่ 2.4 ไดอะแกรมของ Unsupervised Learning

2. ทิศทางการไหลภายในของระบบนิวรอลเน็ตเวอร์กมี 2 แบบ คือ

2.1 ทิศทางการไหลแบบป้อนไปข้างหน้า (feedforward network)

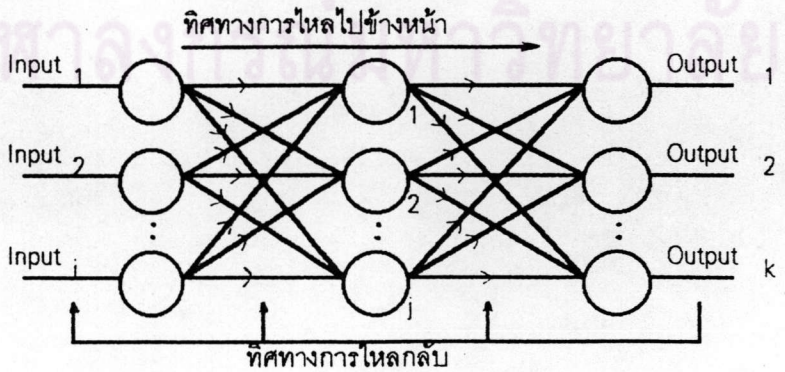
ทิศทางการไหลของข้อมูลจะเป็นลักษณะเคลื่อนไปข้างหน้าจากอินพุตเลเยอร์ผ่านเลเยอร์ภายในไปสู่เอาท์พุตเลเยอร์ แสดงได้ในรูปที่ 2.5



รูปที่ 2.5 ทิศทางการไหลแบบป้อนไปข้างหน้า

2.2 ทิศทางการไหลแบบป้อนไปข้างหน้าและป้อนกลับ (recurrent network)

ทิศทางไหลของข้อมูลจะแบ่งเป็น 2 ทิศทาง คือ ทิศทางการไหลของข้อมูลลักษณะเคลื่อนไปข้างหน้าจากอินพุตเลเยอร์ผ่านเลเยอร์ภายในไปสู่เอาท์พุตเลเยอร์ (feedforward) และทิศทางการไหลของข้อมูลลักษณะป้อนกลับ (feedback) จากเอาท์พุตเลเยอร์ไปยังอินพุตเลเยอร์หรือจากเอาท์พุตเลเยอร์ผ่านเลเยอร์ภายในไปสู่อินพุตเลเยอร์ แสดงได้ในรูปที่ 2.6



รูปที่ 2.6 ทิศทางการไหลแบบป้อนไปข้างหน้าและป้อนกลับ

ประเภทของระบบนิวรอลเน็ตเวอร์ก แต่ละชนิดได้แสดงไว้ในตารางที่ 2.1

ตารางที่ 2.1 การแบ่งประเภทของระบบนิวรอลเน็ตเวอร์ก(Artificial Neural Systems,Jacek M.Zurada,1992)

ชนิดของระบบนิวรอลเน็ตเวอร์ก	การเรียนรู้	ทิศทางการไหลของข้อมูล
Single-layer Network of Discrete and Continuous Perceptrons	S	FF
Multilayer Network of Discrete and Continuous Perceptrons	S	FF
Gradient-type Network	S	REC
Linear Associative Memory	S	FF
Autoassociative Memory	S	REC
Bidirectional Associative Memory	S	REC
Temporal Associative Memory	S	REC
Hamming Network	S	FF
MAXNET	S	REC
Clustering Network	U	FF
Counterpropagation Network	U+S	FF
Self-Organizing Neural Array	U	FF
Adaptive Resonance Theory 1 Network	U	REC

เมื่อ S - แทนการเรียนรู้แบบมีผู้ช่วย (Supervised learning)

U - แทนการเรียนรู้แบบไม่มีผู้ช่วย (Unsupervised learning)

FF - แทนทิศทางการไหลแบบป้อนไปข้างหน้า (feedforward network)

REC - แทนทิศทางการไหลแบบป้อนไปข้างหน้าและป้อนกลับ (recurrent network)

จะเห็นได้ว่าได้มีการพัฒนาหลักการและทฤษฎีในการทำงานของระบบนิวรอลเน็ตเวอร์กต่างๆมากมาย โดยในแต่ละชนิดจะขึ้นอยู่กับความเหมาะสมการนำไปประยุกต์ใช้งาน สำหรับในวิทยานิพนธ์นี้จะใช้ระบบนิวรอลเน็ตเวอร์กโดยวิธี back propagation ประยุกต์ใช้สำหรับตรวจวัดก๊าซด้วยเหตุผลที่สำคัญ ดังนี้

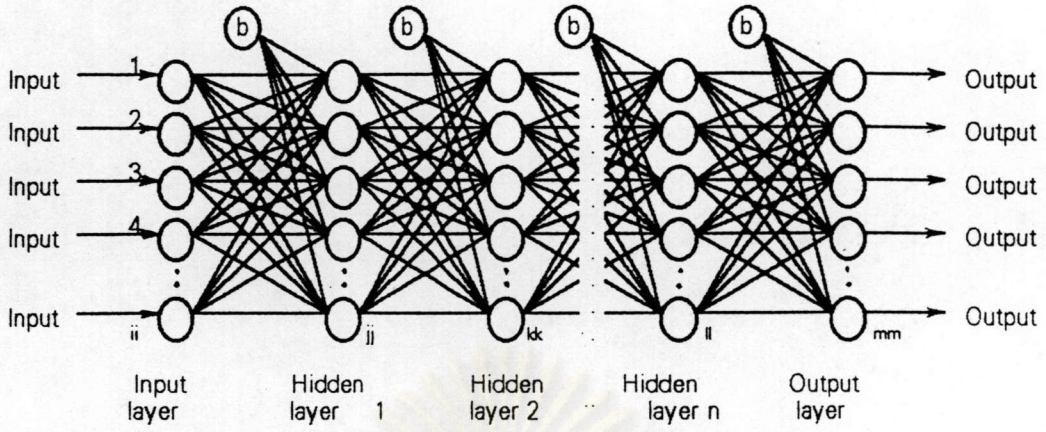
- เป็นระบบที่สามารถสร้างความสัมพันธ์ในรูปฟังก์ชัน ได้จากข้อมูลในการเรียนรู้

- เป็นระบบนิวรอลเน็ตเวอร์กแบบมีผู้ช่วย (Supervised learning) ดังนั้น การตรวจวัดก๊าซจึงสามารถตรวจวัดก๊าซได้ไม่จำกัดชนิด ซึ่งระบบนิวรอลเน็ตเวอร์กเรียนรู้จากชุดข้อมูลตัวอย่างจากการวัดก๊าซ

- เป็นระบบเน็ตเวอร์กหลายชั้นเคลื่อนหน้า (Multilayer feedforward network) ดังนั้น จึงสามารถจำรูปแบบข้อมูลตัวอย่างที่ซับซ้อนได้มากขึ้น

หลักการเรียนรู้ของระบบนิวรอลเน็ตเวอร์กโดยวิธี Back Propagation Network

ระบบนิวรอลเน็ตเวอร์กโดยวิธี back propagation เน็ตเวอร์กชนิดนี้ประกอบด้วยโหนดที่เรียงกันอยู่อย่างน้อย 3 เลเยอร์ประกอบด้วย อินพุตเลเยอร์ ,เลเยอร์ภายใน และเอาท์พุตเลเยอร์ ทิศทางการไหลของข้อมูลจะเป็นลักษณะเคลื่อนไปข้างหน้า (feedforward network) จากอินพุตเลเยอร์ผ่านเลเยอร์ภายในไปยังเอาท์พุตเลเยอร์ เน็ตเวอร์กชนิดนี้มีความสามารถในการเรียนรู้ในโหมดที่ต้องมีผู้ช่วยภายนอกในการสอน (Supervised learning) ดังแสดงในรูปที่ 2.7 ประกอบ ด้วย b เป็นโหนดไบแอส ซึ่งมีค่าเท่ากับ 1 และ n เป็นจำนวนเลเยอร์ภายใน (hidden layer)



รูปที่ 2.7 โครงสร้างไดอะแกรม Back Propagation Network

การคำนวณสัญญาณเอาท์พุต เป็นการคำนวณไปข้างหน้า (forward propagation) กล่าวคือ ชุดข้อมูลอินพุตที่ป้อนเข้าสู่เน็ตเวอร์กในชั้นของอินพุตเลเยอร์ จะถูกประมวลผลและส่งผลลัพธ์ต่อไปยังขั้นต่อไปจนถึงเอาท์พุตเลเยอร์

สมการค่าอินพุตในเลเยอร์ภายใน j ได้คือ

$$net_j = \sum w_{ji} o_i + w_{b_j} \qquad (2.1)$$

และค่าเอาท์พุตของโหนด j คือ

$$o_j = f(net_j) \qquad (2.2)$$

โดยที่ i คือ จำนวนโหนดของอินพุตเลเยอร์

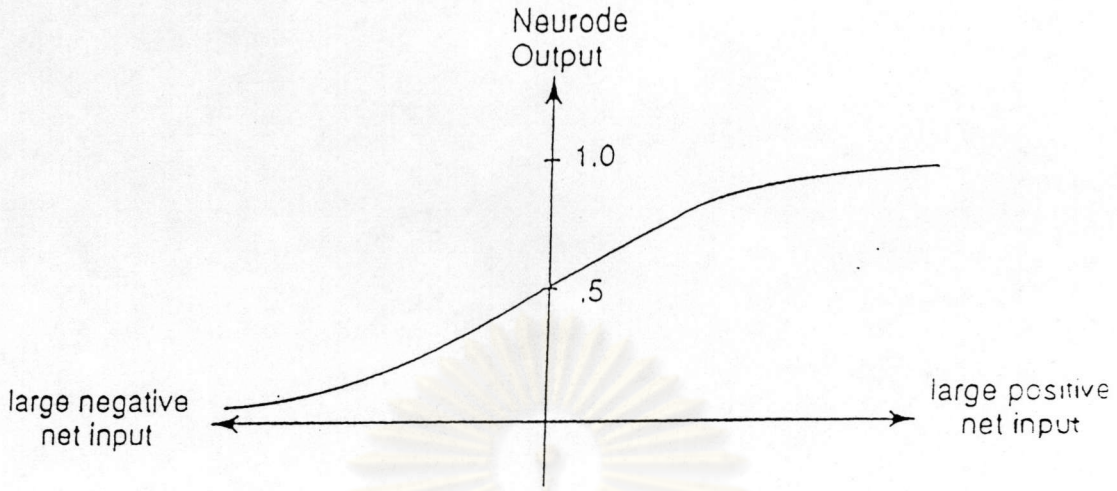
j คือ จำนวนโหนดของเลเยอร์ภายใน

w_{ji} คือ ค่าน้ำหนักระหว่างโหนดในอินพุตเลเยอร์ i กับเลเยอร์ภายใน j

w_{b_j} คือ ค่าน้ำหนักไบแอสของเลเยอร์ภายใน j

o_j คือ ค่าเอาท์พุตของโหนด j ในเลเยอร์ภายใน j

$f(.)$ คือ ฟังก์ชันซิกมอยด์ ซึ่งมีค่าเพิ่มขึ้นจาก 0 ไปถึง 1 เมื่อค่า x เปลี่ยนจากค่า $-\infty$ ไปยังค่า $+\infty$ ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 กราฟของฟังก์ชันซิกมอยด์

สำหรับฟังก์ชันซิกมอยด์เขียนอยู่ในรูปสมการ

$$O_j = \frac{1}{1 + e^{-net_j}} \qquad (2.3)$$

ความสัมพันธ์ในโหนดของเลเยอร์ภายในและเอาท์พุตมีสมการเช่นเดียวกับสมการ 2.1 และ 2.2 สามารถคำนวณได้ดังนี้คือ

ค่าอินพุตของเอาท์พุตเลเยอร์ k คือ

$$net_k = \sum w_{kj} O_j + w_{b_k} \qquad (2.4)$$

และค่าเอาท์พุตของโหนด k คือ

$$O_k = f(net_k) \qquad (2.5)$$

โดยที่ k คือจำนวนโหนดของเอาท์พุตเลเยอร์

w_{kj} คือค่าน้ำหนักระหว่างโหนดในเลเยอร์ภายใน j และเอาท์พุตเลเยอร์ k

w_{b_k} คือค่าน้ำหนักไบแอสของเอาท์พุตเลเยอร์ k

ในการเรียนรู้ของระบบซึ่งเป็นการคำนวณแบบเคลื่อนย้อนกลับ (back propagation) กล่าวคือ เอาท์พุตที่ได้จากการฝึกของเน็ตเวอร์กจะถูกนำไปเปรียบเทียบกับเอาท์พุตเป้าหมาย (Target output) ซึ่งจะได้ค่าสัญญาณความผิดพลาดที่ถูกคำนวณขึ้นและส่งผลถอยหลังกลับจากเลเยอร์เอาท์พุตไปยังโหนดต่างๆ ของเลเยอร์ภายใน และแต่ละโหนดต่างๆของเลเยอร์ถัดลงมาจนกระทั่งทุกโหนดในเน็ตเวอร์กได้รับการปรับค่าน้ำหนักใหม่ ทั้งสองขั้นตอนจะดำเนินไปเรื่อยๆ จนกระทั่งค่าสัญญาณความผิดพลาดต่ำกว่าค่าที่กำหนดไว้ค่าหนึ่งจึงจะหยุด สูตรในการคำนวณค่าสัญญาณความผิดพลาดและการปรับค่าสามารถแสดงเป็นสมการคณิตศาสตร์ได้ดังนี้

ค่าความผิดพลาด (Sum squared error) ระหว่างเอาท์พุตที่กำหนดมากับเอาท์พุตของเน็ตเวอร์ก สำหรับแต่ละชุดข้อมูลเป็น

$$E = \frac{1}{2}\sum_{k}(t_k - o_k)^2 \qquad (2.6)$$

โดยที่ t_k คือค่าข้อมูลเอาท์พุตเป้าหมายในเอาท์พุตเลเยอร์ k

o_k คือค่าข้อมูลเอาท์พุตที่ได้จากการฝึกของเน็ตเวอร์กในเอาท์พุตเลเยอร์ k

การคำนวณหาสมการการปรับค่าน้ำหนักของเน็ตเวอร์ก คำนวณจากค่าอนุพันธ์ของค่าความผิดพลาด (E) เทียบกับน้ำหนัก (w) ได้ดังนี้ คือ

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} \qquad (2.7)$$

จากสมการที่ 2-4 จะได้ว่า

$$\frac{\partial net_k}{\partial w_{kj}} = \frac{\partial\left(\Sigma w_{kj} o_j + w_{b_k}\right)}{\partial w_{kj}} = o_j \qquad (2.8)$$

และกำหนดให้

$$\delta_k = -\frac{\partial E}{\partial net_k} \qquad (2.9)$$

โดยที่ δ_k คือค่าสัญญาณความผิดพลาดของแต่ละโหนดในเอาท์พุตเลเยอร์ k

จากกฎลูกโซ่ ดังนั้นสมการ 2.9 เขียนใหม่จะได้เป็นสมการ

$$\delta_k = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k} \qquad (2.10)$$

ค่าอนุพันธ์ของค่าความผิดพลาดเทียบกับค่าเอาท์พุตของโหนด k คำนวณได้เป็นสมการ

$$\frac{\partial E}{\partial o_k} = -(t_k - o_k) \qquad (2.11)$$

และ

$$\frac{\partial o_k}{\partial net_k} = f'_k(net_k) \qquad (2.12)$$

จากสมการ 2.10, 2.11 และ 2.12 จะได้ค่าสัญญาณความผิดพลาดของแต่ละโหนดในเอาท์พุตเลเยอร์คำนวณได้ดังนี้

$$\delta_k = (t_k - o_k) f'_k(net_k) \qquad (2.13)$$

สมการอัตราการปรับค่าน้ำหนัก จะเท่ากับผลลบของอัตราการเรียนรู้ของน้ำหนัก (η_l) คูณด้วยค่าอนุพันธ์ของค่าความผิดพลาด (E) เทียบกับน้ำหนัก (w) ได้ดังนี้คือ

$$\Delta w_{kj} = -\eta_l \frac{\partial E}{\partial w_{kj}} \qquad (2.14)$$

ดังนั้น จากสมการ 2.7, 2.8, 2.13 และ 2.14 สมการอัตราการปรับค่าน้ำหนักเขียนใหม่ได้ดังนี้คือ

$$\Delta w_{kj} = \eta_l (t_k - o_k) f'_k(net_k) o_j = \eta_l \delta_k o_j \qquad (2.15)$$

และอัตราการปรับค่าน้ำหนักระหว่างเลเยอร์ภายในและอินพุตสามารถคำนวณหาได้ในลักษณะเดียวกันคือ

$$\Delta w_{ji} = -\eta_l \frac{\partial E}{\partial w_{ji}}$$

$$= -\eta_l \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$= -\eta_l \frac{\partial E}{net_j} o_i$$

$$= \eta_l \left(-\frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \right) o_i$$

$$= \eta_l \left(-\frac{\partial E}{\partial o_j} \right) f_j'(net_j) o_i$$

$$= \eta_l \delta_j o_i \qquad (2.16)$$

อย่างไรก็ตาม เฟกเตอร์ $\frac{\partial E}{\partial o_j}$ ไม่สามารถหาค่าได้โดยตรง แต่สามารถคำนวณจากสมการ 2.6 ได้ในเทอมดังนี้

$$-\frac{\partial E}{\partial o_j} = -\sum_k \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial o_j}$$

$$= \sum_k \left(-\frac{\partial E}{\partial net_k} \right) \frac{\partial \left(\sum_j w_{kj} o_j + w_{b_k} \right)}{\partial o_j}$$

$$= \sum_k \left(-\frac{\partial E}{\partial net_k} \right) w_{kj} = \sum_k \delta_k w_{kj} \qquad (2.17)$$

และค่าสัญญาณความผิดพลาดของแต่ละโหมดในเลเยอร์ภายในคำนวณได้ดังนี้

(Adaptive Pattern Recognition and Neural Network, Yoh-Han Pao, 1989)

$$\delta_j = f_j'(net_j) \sum_k \delta_k w_{kj} \qquad (2.18)$$

ดังนั้นถ้าใช้ฟังก์ชันในรูปของซิกมอยด์ คือ

$$o_j = \frac{1}{1 + e^{-(net_j)}} \tag{2.19}$$

จะสามารถเขียนได้ว่า

$$\frac{\partial o_j}{\partial net_j} = o_j(1 - o_j) \tag{2.20}$$

ดังนั้น จากสมการ 2-13 และ 2-18 สามารถเขียนได้ใหม่เป็นสมการ

$$\delta_k = (t_k - o_k) o_k (1 - o_k) \tag{2.21}$$

และ

$$\delta_j = o_j(1 - o_j) \sum_k \delta_k w_{kj} \tag{2.22}$$

การปรับค่าน้ำหนัก เพื่อช่วยในการปรับค่า เร่งให้ค่าความผิดพลาด (E) ลู่เข้าสู่ศูนย์ได้เร็วขึ้นจึงมีการใช้วิธีโมเมนตัมรวมในการปรับค่า ดังนั้นอัตราการปรับค่าน้ำหนักคำนวณได้เป็นสมการ

$$\Delta w_{kj}(n+1) = \eta_l \delta_k o_j + \alpha \Delta w_{kj}(n) \tag{2.23}$$

และ

$$\Delta w_{ji}(n+1) = \eta_l \delta_j o_i + \alpha \Delta w_{ji}(n) \tag{2.24}$$

โดยที่ η_l คืออัตราการเรียนรู้ (learning rate)

α คืออัตราโมเมนตัมของน้ำหนัก (momentum rate)

n คือจำนวนครั้งของการปรับค่าน้ำหนัก

การปรับค่าน้ำหนักไบแอสสามารถคำนวณได้โดยใช้ค่าอนุพันธ์ค่าความผิดพลาดเทียบกับค่าน้ำหนักไบแอส ซึ่งอัตราการปรับค่าน้ำหนักไบแอสคำนวณได้ดังนี้คือ

$$\Delta w_{b_k}(n+1) = \eta_b \frac{\partial E}{\partial w_{b_k}} + \alpha \Delta w_{b_k}(n) \qquad (2.25)$$

และ

$$\Delta w_{b_j}(n+1) = \eta_b \frac{\partial E}{\partial w_{b_j}} + \alpha \Delta w_{b_j}(n) \qquad (2.26)$$

โดยที่ η_b คืออัตราการไบแอส (bias rate)

สูตรในการคำนวณ ค่าสัญญาณความผิดพลาดและการปรับค่าของ Back propagation network สามารถสรุปเป็นอัลกอริทึมได้ดังนี้

<u>ขั้นตอนที่ 1</u> ค่าน้ำหนักเริ่มแรกของแต่ละโหนดถูกกำหนดโดยการสุ่มให้ค่าในช่วง ที่ต้องการ กำหนดให้ ค่าน้ำหนักที่เชื่อมโยงระหว่างอินพุตเลเยอร์กับเลเยอร์ภายใน w_{ji} เป็นเมตริกซ์ (i x j) ,ค่าน้ำหนักที่เชื่อมโยงระหว่างเลเยอร์ภายในกับเอาท์พุตเลเยอร์ w_{kj} เป็นเมตริกซ์ (j x k) ,ค่าน้ำหนักไบแอสเลเยอร์ภายใน w_{b_j} และค่าน้ำหนักไบแอสเอาท์พุตเลเยอร์ w_{b_k} เมื่อ i,j,k เป็นจำนวนโหนด ของอินพุตเลเยอร์,เลเยอร์ภายในและเอาท์พุตเลเยอร์ ตามลำดับ

<u>ขั้นตอนที่ 2</u> รับชุดข้อมูล(Pattern) ซึ่งประกอบด้วยค่าอินพุตและค่าเอาท์พุตเป้าหมาย โดยกำหนดให้ค่าอินพุตเป็น $O_0, O_1, O_2, \cdots, O_i$ และ ค่าเอาท์พุตเป้าหมายเป็น $t_0, t_1, t_2, \cdots, t_k$

<u>ขั้นตอนที่ 3</u> ทำการคำนวณหาค่าเอาท์พุต

$$net_j = \sum w_{ji} O_i + w_{b_j}$$

และค่าเอาท์พุตของโหนด j คือ

$$o_j = f(net_j)$$

โดยที่ i คือ จำนวนโหนดของอินพุตเลเยอร์

j คือ จำนวนโหนดของเลเยอร์ภายใน

w_{ji} คือ ค่าน้ำหนักระหว่างโหนดในอินพุตเลเยอร์ i กับเลเยอร์ภายใน j

w_{b_j} คือ ค่าน้ำหนักไบแอสเลเยอร์ภายใน j

o_j คือ ค่าเอาท์พุตของโหนด jในเลเยอร์ภายใน j

$f(.)$ คือ ฟังก์ชันซิกมอยด์ ซึ่งมีค่าเท่ากับ $\frac{1}{1+e^{-net_j}}$

$$net_k = \sum w_{kj} o_j + w_{b_k}$$

และค่าเอาท์พุตของโหนด k คือ

$$o_k = f(net_k)$$

โดยที่ k คือจำนวนโหนดของเอาท์พุตเลเยอร์

w_{kj} คือค่าน้ำหนักระหว่างโหนดในเลเยอร์ภายใน j และเอาท์พุตเลเยอร์ k

w_{b_k} คือค่าน้ำหนักไบแอสเอาท์พุตเลเยอร์ k

$f(.)$ คือ ฟังก์ชันซิกมอยด์ ซึ่งมีค่าเท่ากับ $\frac{1}{1+e^{-net_k}}$

<u>ขั้นตอนที่ 4</u> คำนวณหาค่าความผิดพลาด (sum squared error) ระหว่างเอาท์พุตที่กำหนดมากับเอาท์พุตของเน็ตเวอร์ก สำหรับแต่ละชุดข้อมูล

$$E = \frac{1}{2}\sum_k (t_k - o_k)^2$$

โดยที่ t_k คือค่าข้อมูลเอาท์พุตเป้าหมาย

o_k คือค่าข้อมูลเอาท์พุตที่ได้จากการฝึกของเน็ตเวอร์ก

ขั้นตอนที่ 5 คำนวณหาค่าสัญญาณความผิดพลาดของแต่ละโหนด

$$\delta_k = (t_k - o_k) o_k (1 - o_k)$$

โดยที่ δ_k คือค่าสัญญาณความผิดพลาดของแต่ละโหนดในเอาท์พุตเลเยอร์ k

$$\delta_j = o_j (1 - o_j) \sum_k \delta_k w_{kj}$$

โดยที่ δ_j คือค่าสัญญาณความผิดพลาดของแต่ละโหนดในเลเยอร์ภายใน j

ขั้นตอนที่ 6 ปรับค่าน้ำหนักระหว่างเอาท์พุตเลเยอร์กับเลเยอร์ภายใน

$$\Delta w_{kj}(n+1) = \eta_l \delta_k o_j + \alpha \Delta w_{kj}(n)$$

โดยที่ η_l คืออัตราการเรียนรู้ (learning rate)

α คืออัตราโมเมนตัมของน้ำหนัก (momentum rate)

n คือจำนวนครั้งของการปรับค่าน้ำหนัก

และปรับค่าน้ำหนักระหว่างเลเยอร์ภายในกับอินพุตเลเยอร์

$$\Delta w_{ji}(n+1) = \eta_l \delta_j o_i + \alpha \Delta w_{ji}(n)$$

ขั้นตอนที่ 7 ปรับค่าน้ำหนักไบแอสเอาท์พุตเลเยอร์ k

$$\Delta w_{b_k}(n+1) = \eta_b \frac{\partial E}{\partial w_{b_k}} + \alpha \Delta w_{b_k}(n)$$

และค่าน้ำหนักไบแอสเลเยอร์ภายใน j

$$\Delta w_{b_i}(n+1) = \eta_b \frac{\partial E}{\partial w_{b_i}} + \alpha \Delta w_{b_i}(n)$$

โดยที่ η_b คืออัตราการไบแอส (bias rate)

<u>ขั้นตอนที่ 8</u> กลับไปที่ขั้นตอนที่ 2 จนกว่าค่า E น้อยกว่าที่กำหนดจึงจะหยุด