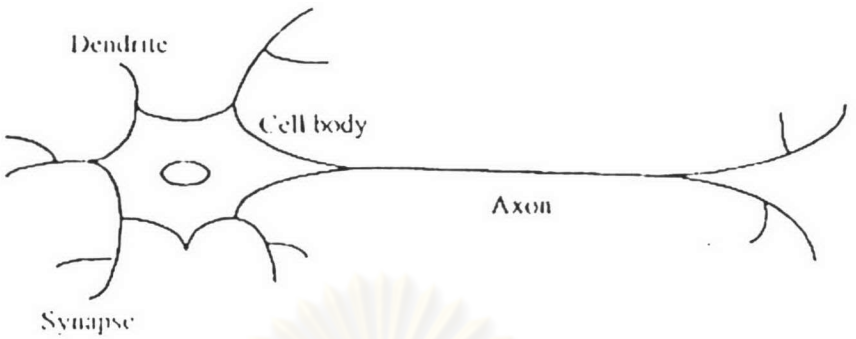
บทที่ 3
แนวคิดและทฤษฎีของโครงข่ายใยประสาทเทียม

พฤติกรรมที่เกี่ยวกับความฉลาดทางด้านวิเคราะห์และความสามารถในการจดจำบางสิ่งบางอย่างเป็นเรื่องยาก แต่สิ่งมีชีวิตทั่วไปกลับทำได้อย่างง่ายๆ ตัวอย่างเช่นพวกเรามีสามารถที่จะจดจำหลายสิ่งหลายอย่างที่เกิดขึ้นในอดีต รวมทั้งคาดการณ์สิ่งต่างๆได้จากประสบการณ์ที่ได้เรียนรู้มา โดยอยู่ในหลักการของเหตุและผล นักวิทยาศาสตร์มากมายพยายามที่จะจำลองระบบการคำนวณที่มีความสามารถทางด้านนี้เหมือนกับสิ่งที่มนุษย์สามารถทำได้ โดยพยายามจำลองระเบียบวิธีการคิดของมนุษย์ และพยายามเสริมเพิ่มเติมความสามารถบางอย่างในส่วนที่เป็นข้อจำกัดของมนุษย์ออกไป

ระบบประสาทของสิ่งมีชีวิต ประกอบไปด้วยส่วนที่ต่อกันของเส้นใยประสาท ที่มีโครงสร้างแบบเครือข่าย แบบจำลองโครงข่ายใยประสาทเทียมคือระบบการคำนวณที่จำลองการทำงานของระบบประสาทของสิ่งมีชีวิต โดยประกอบด้วยส่วนประกอบหน่วยเล็กๆ ซึ่งทำหน้าที่ส่งถ่ายข้อมูล วิธีการเหล่านี้สร้างความสัมพันธ์ระหว่างข้อมูลด้วยขนาดของหน่วยน้ำหนักซึ่งสามารถปรับเปลี่ยนไปได้เรื่อยๆตลอดกระบวนการเรียนรู้กับชุดข้อมูล แบบจำลองนี้เรียกว่า 'neural net' หรือ 'artificial neural network'

การพัฒนาทางด้านโครงข่ายใยประสาทเทียม เริ่มตั้งแต่ปี พ.ศ. 2483 โดย Mc-Culloch และ Pitts ซึ่งเป็นผู้ที่นำเสนอหลักการทำงานของโครงข่ายใยประสาทเทียมอย่างง่าย ต่อมาในปี พ.ศ. 2503 Widrow และ Hoff ได้พัฒนากฎการเรียนรู้สำหรับโครงข่ายใยประสาทขึ้นมาใหม่ สำหรับใช้กับโครงข่ายใยประสาทเทียมอย่างง่ายที่ประกอบด้วยยูนิตเชิงเส้นเป็นหลัก ซึ่งยังมีใช้อยู่ในปัจจุบัน เรียกว่า Widrow-Hoff learning rule หลังจากนั้นมีการคิดค้นพัฒนาเครือข่ายแบบใหม่ขึ้นมากมาย เพื่อแก้ปัญหาที่ซับซ้อนมากขึ้น แต่การปรับปรุงการเรียนรู้ให้ดีขึ้นก็ยังไม่ประสบผลสำเร็จ เช่น ปี พ.ศ. 2515 Kohonen ได้คิดค้นเครือข่ายที่เรียกว่า Kohonen network ปี พ.ศ. 2519 Grossberg ได้คิดค้นเครือข่ายที่สามารถเรียนรู้ได้ตัวเอง (Self-organizing network) ต่อมาเมื่อวิทยาการด้านคอมพิวเตอร์พัฒนาไปมากขึ้น เครื่องคอมพิวเตอร์มีความเร็วในการปรับสอนสูงขึ้น มีการค้นพบที่สำคัญคือการคิดค้นกระบวนการเรียนรู้แบบแพร่กระจายความผิดพลาดย้อนกลับ (Error back-propagation learning rule) ซึ่งในปี พ.ศ. 2523 ได้นำวิธีคิดนี้ไปใช้ได้กับโครงข่ายใยประสาทแบบหลายชั้น

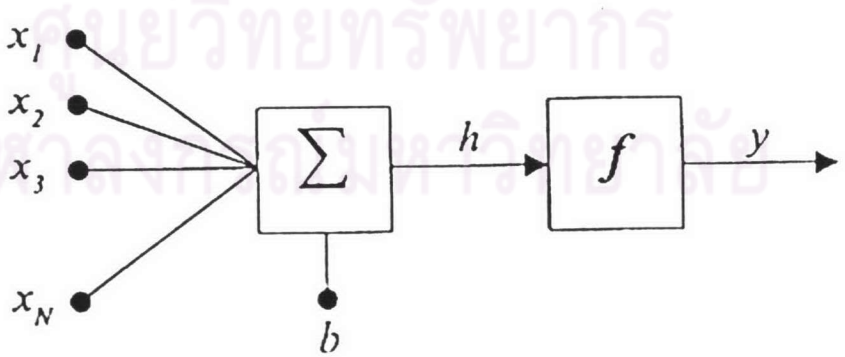
3.1 ระบบประสาทของสิ่งมีชีวิต



รูปที่ 3.1 เซลล์ประสาทของสิ่งมีชีวิต

เซลล์ประสาท (neuron) ประกอบไปด้วยส่วนประกอบต่างๆ ได้แก่ เดนไดรต์ (dendrites) ซึ่งทำหน้าที่ในการรับความรู้สึกต่างๆ ที่ส่งเข้ามา ดังรูปที่ โดยที่เดนไดรต์แต่ละกิ่งจะรับรู้ได้ด้วยค่าถ่วงน้ำหนัก (weight) ที่แตกต่างกัน และแทนค่าถ่วงน้ำหนักด้วยความแข็งแรงของแต่ละไซแนปส์ (synapse) จากนั้นตัวเซลล์ (cell body) ก็จะรวบรวมสิ่งที่รับรู้ผ่านแอกซอน (axon) ซึ่ง แอกซอนจะส่งสัญญาณออกไป โดยสัญญาณออกจะเป็นฟังก์ชันของผลรวมของสิ่งที่ได้รับรู้จากตัวเซลล์

3.2 แบบจำลองโครงข่ายใยประสาทเทียม



รูปที่ 3.2 เซลล์ประสาทเทียม

จากกลไกการทำงานของเซลล์ประสาท สามารถสร้างแบบจำลองของเซลล์ประสาทได้ ซึ่งโดยทั่วไปเรียกว่าเซลล์ประสาทเทียม (artificial neuron) ดังรูปที่ 3.2

จากรูปที่ 3.2 ตัวแปรด้านเข้า x_i โดยที่ i = 1,2,3,...,N จะถูกส่งผ่านไปรวมกันด้วยค่าถ่วงน้ำหนักที่ไม่เท่ากัน จะได้ผลรวมเป็น

$$h_k = \sum_{i=1}^{N} w_{ki} x_i + b_k \tag{3.1}$$

$$h_k = w_{k1} x_1 + w_{k2} x_2 + ... + w_{kN} x_N + b_k \tag{3.2}$$

จากนั้นส่งสัญญาณ h_k ผ่านแอกติเวชันฟังก์ชัน (activation function) เพื่อคำนวณค่าตัวแปรด้านออก y_k

$$y_k = f(\sum_{i=1}^{N} w_{ki} x_i + b_k) \tag{3.3}$$

โดยที่ h_k คือค่าผลรวมของตัวแปรด้านเข้าของยูนิตที่ k

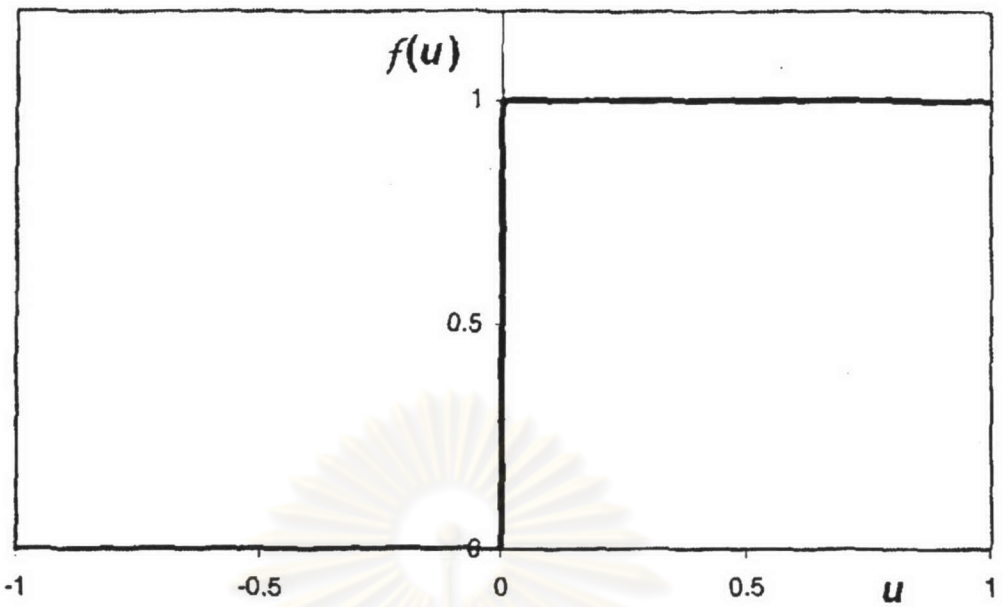
w_{ki} คือค่าถ่วงน้ำหนักระหว่างยูนิตที่ i และ k

b_k คือค่าไบอัส (bias) ของยูนิตที่ k

แอกติเวชันฟังก์ชันมี 3 ชนิดที่ใช้กันทั่วไป

1) Threshold หรือ step function ตามรูปที่ 3.3 มีฟังก์ชันก์คือ

$$f(u) = \begin{cases} 1 & \text{ถ้า } u \geq 0 \\ 0 & \text{ถ้า } u < 0 \end{cases}$$

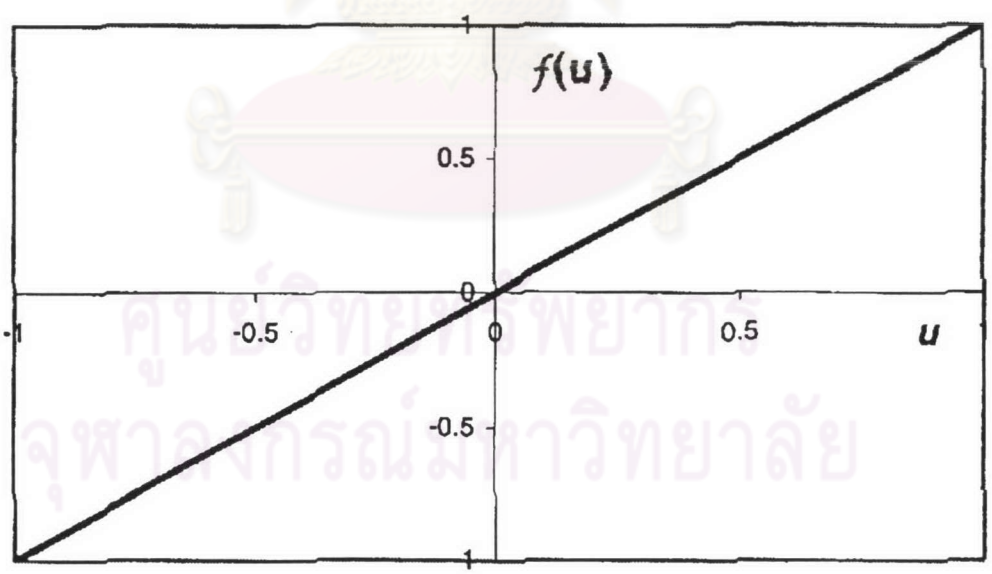


รูปที่ 3.3 Threshold หรือ Step function

2) ฟังก์ชันเชิงเส้น (Linear function) ฟังก์ชันที่แสดงตามรูปที่ 3.4

$$f(u) = w \times o + b \tag{3.4}$$

w คือ ค่าความชันของเส้นตรง และ b คือ ค่าตัดแกนของ f(u)

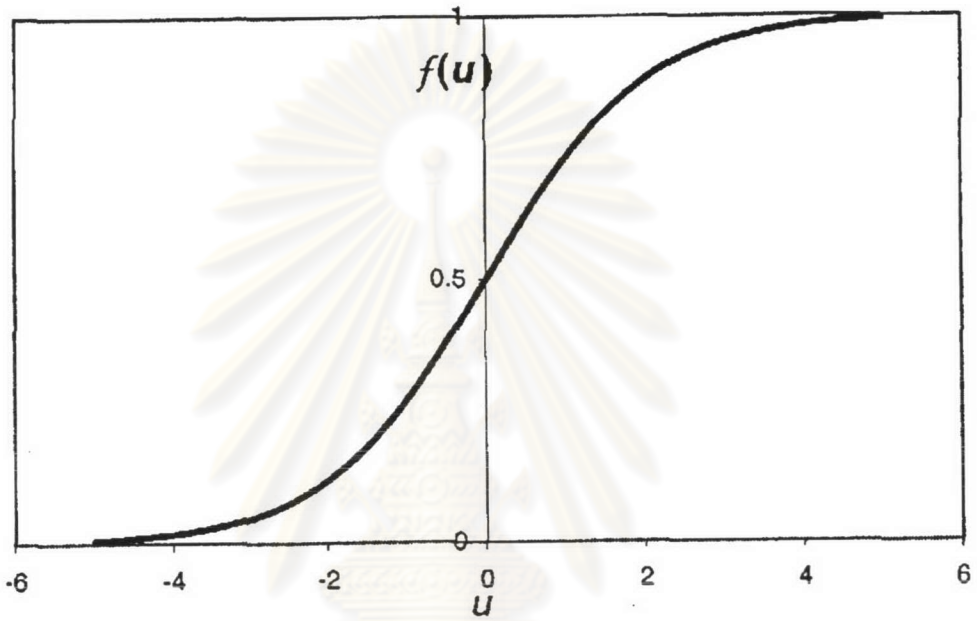


รูปที่ 3.4 Liner function

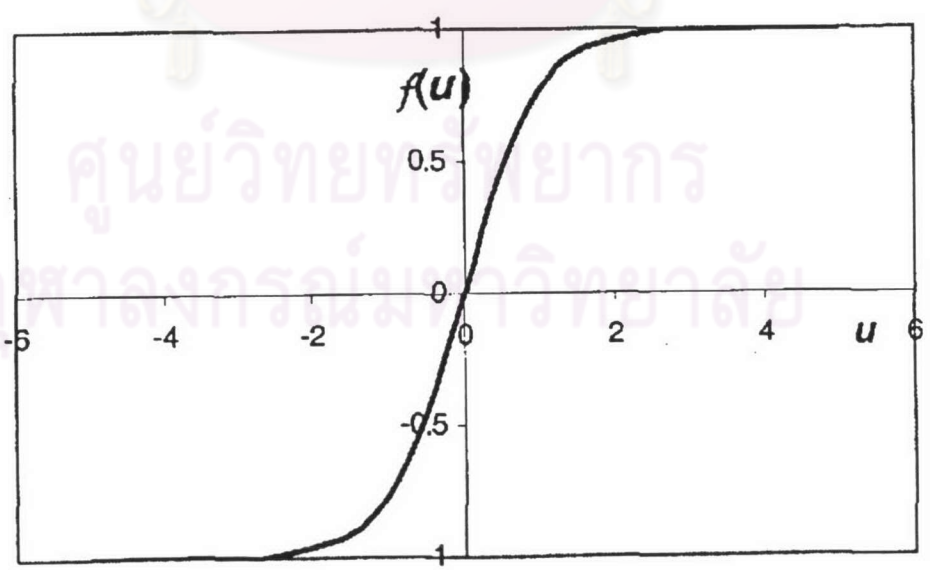
3) ฟังก์ชัน Sigmoid (Sigmoid function) เป็นฟังก์ชันที่ได้รับความนิยมอย่างมากในแบบจำลองโครงข่ายใยประสาทเทียม ฟังก์ชันนำเสนอในรูป sigmoid และ tangent sigmoid กราฟของ sigmoid มีรูปร่างคล้ายตัว S ซึ่งมีสมการคือ

$$f(u) = \frac{1}{1 + e^{-wx}} \tag{3.5}$$

w คือ ค่าความชันของฟังก์ชัน และ ฟังก์ชันมีค่าต่ำสุดที่ 0 และค่าสูงสุดที่ 1 ในรูปที่ 3.5
ฟังก์ชัน sigmoid ที่มีชื่อว่า tangent sigmoid ในตัวแปรด้านออกมีค่าอยู่ระหว่าง -1 ถึง 1 ดังรูปที่ 3.6

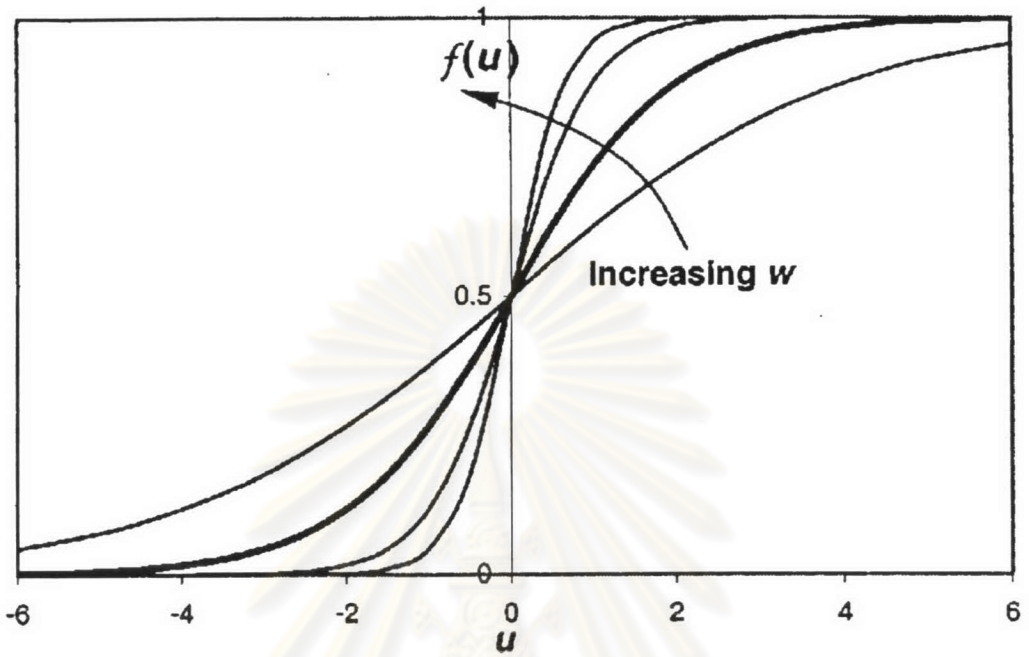


รูปที่ 3.5 Sigmoid function



รูปที่ 3.6 Tangent sigmoid function

ข้อดีของ sigmoid ฟังก์ชัน คือ เป็นฟังก์ชันที่ต่อเนื่อง มีความเรียบที่ไม่ซ่องว่างหรือขอบ มุม และยังเป็นฟังก์ชันที่สามารถหาอนุพันธ์ได้ทุกๆที่บนฟังก์ชัน นอกจากนั้นการเปลี่ยนแปลงของค่าความชันของฟังก์ชัน w ที่แสดงในรูปที่ 3.7



รูปที่ 3.7 Sigmoid function ในค่าความชัน (w) แตกต่างกัน

ส่วนหนึ่งที่ได้รับจาก sigmoid ฟังก์ชันในความชันที่แตกต่างกัน คือความสามารถในการสร้างความสัมพันธ์กับข้อมูลที่แตกต่างกัน

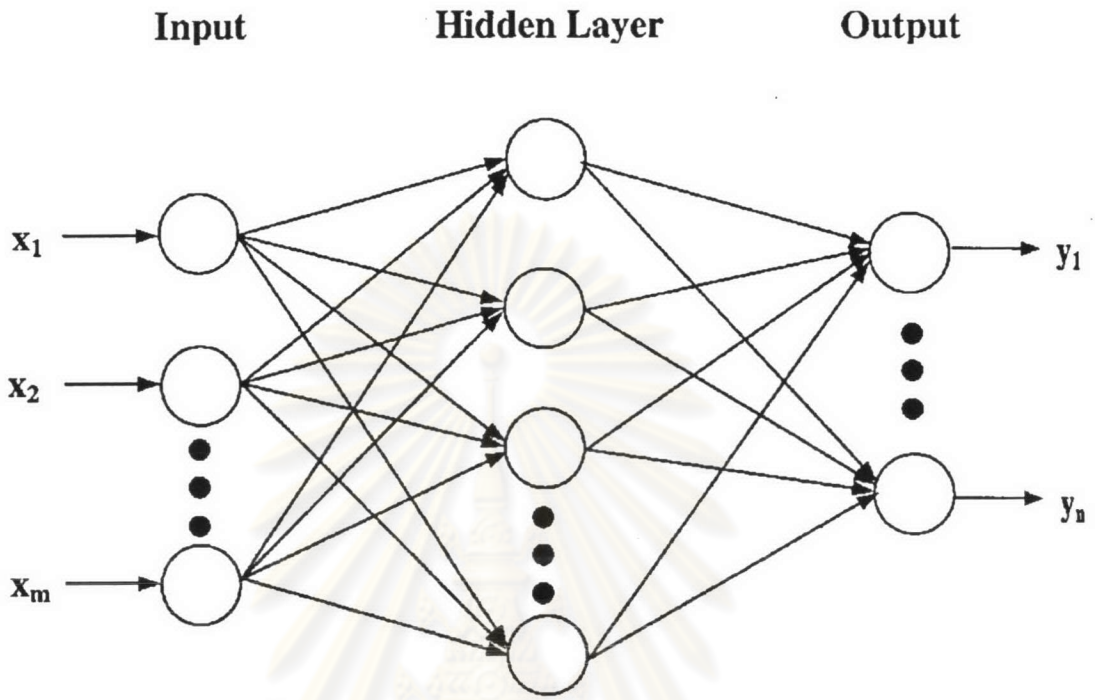
3.3 ชนิดของแบบจำลองโครงข่ายใยประสาทเทียม

แบบจำลองโครงข่ายใยประสาทเทียมสามารถจำแนกตามลักษณะทางโครงสร้างออกได้เป็น 2 ส่วน คือ โครงสร้างแบบ feedforward และ โครงสร้างแบบ recurrent

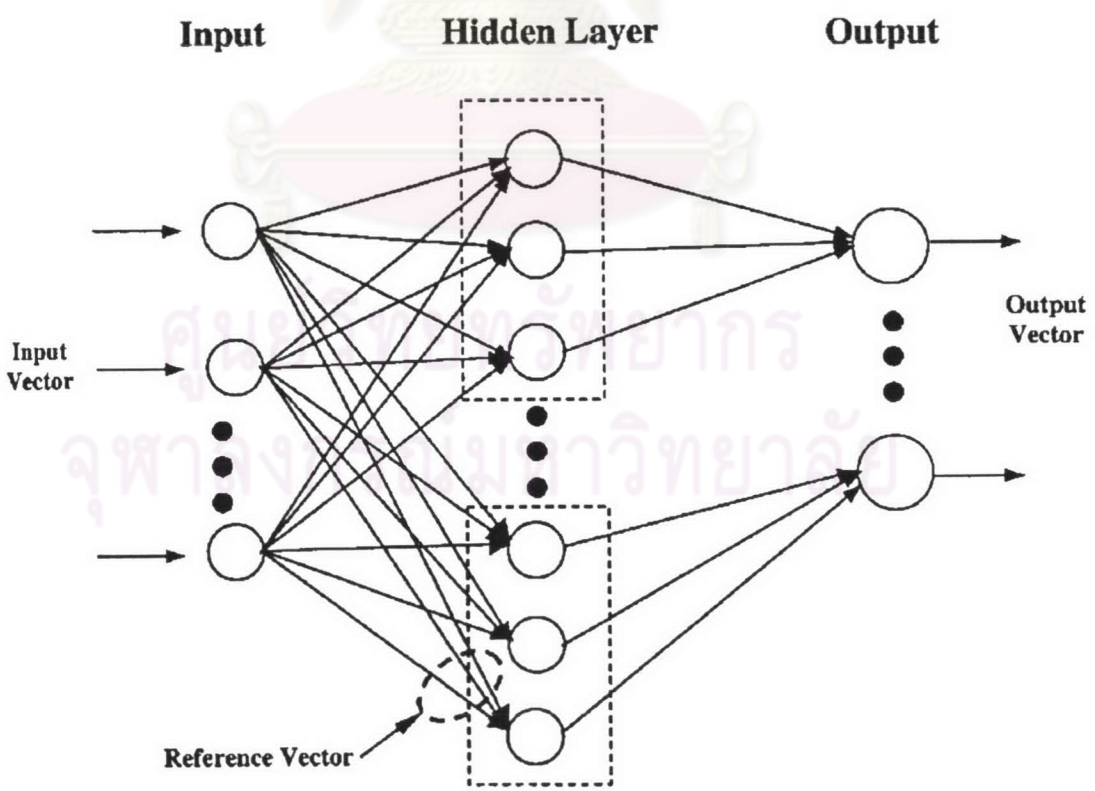
3.3.1 โครงสร้างแบบ feedforward

ในโครงสร้างแบบ feedforward หน่วยประสาทถูกจัดแบ่งออกเป็นชั้นๆ โดยส่งผ่านค่าผ่านชั้นตัวแปรด้านเข้าไปสู่ตัวแปรด้านออก โครงสร้างของแบบจำลองมีความสัมพันธ์กับชั้นก่อนหน้าและชั้นถัดไป แต่จะไม่มีความสัมพันธ์กันภายในชั้น ตัวอย่างของโครงสร้างแบบ feedforward คือ multi-layer perceptron (MLP), learning vector quantization (LVQ), the cerebellar model articulation control (CMAC) และ group method of date handling

(GMDH) โครงข่ายแบบ feedforward สามารถเขียนแทนด้วยภาพได้ดังแสดงในรูปที่ 3.8 แสดงถึงตัวอย่างของโครงข่ายใยประสาทเทียมแบบหลายชั้น feedforward :ซึ่งเป็นหนึ่งในหลายชนิดของแบบจำลองโครงข่ายใยประสาทเทียมที่นิยมนำไปใช้

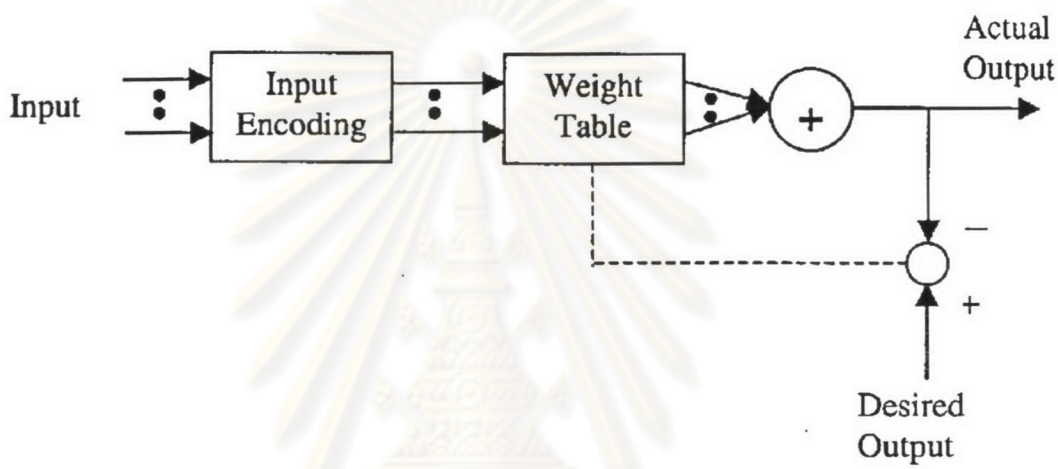


รูปที่ 3.8 Feedforward network (Multi-layer perceptron)



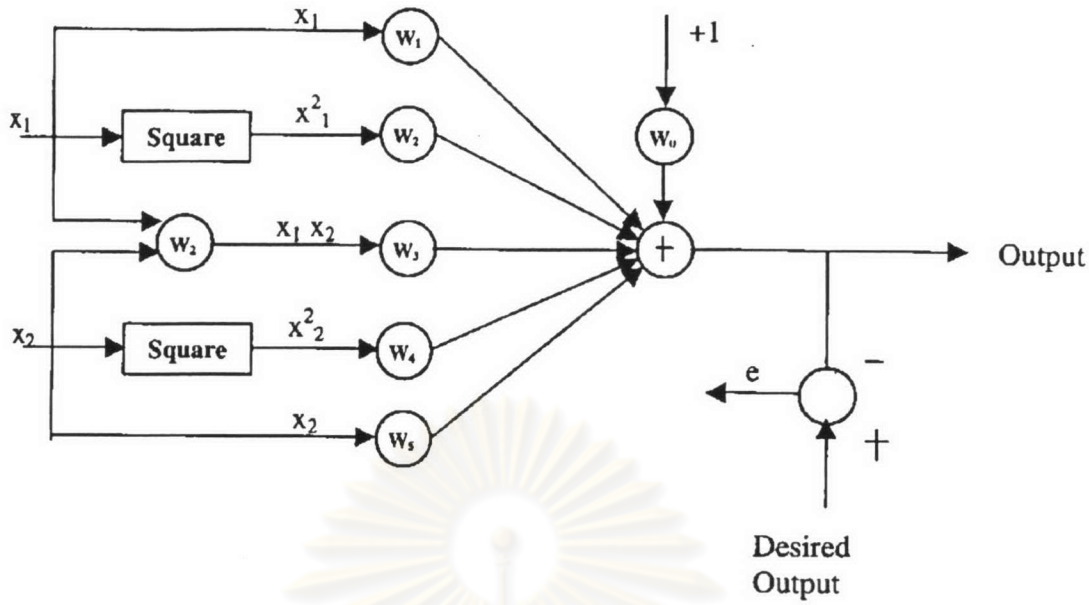
รูปที่ 3.9 Learning vector quantization network

รูปที่ 3.9 แสดงภาพของ learning vector quantization (LVQ), ซึ่งประกอบไปด้วย 3 ชั้นของหน่วยประสาท คือ ชั้น buffer ของตัวแปรด้านเข้า, ชั้นซ่อน และชั้นตัวแปรด้านออก โครงสร้างของแบบจำลองมีการเชื่อมโยงกันแบบทั่วถึงระหว่างชั้นตัวแปรด้านเข้ากับชั้นซ่อน แต่มีการเชื่อมโยงแบบไม่ทั่วถึงของชั้นซ่อนกับตัวแปรด้านออก และแต่ละตัวแปรด้านออกเชื่อมโยงกันเป็นกลุ่มๆ ในชั้นซ่อน ค่าถ่วงน้ำหนักระหว่างชั้นซ่อนกับตัวแปรด้านออกมีค่าคงที่เท่ากับ 1 สำหรับค่าถ่วงน้ำหนักของตัวแปรด้านเข้ากับชั้นซ่อนอยู่ในรูปแบบของ reference vector ซึ่งถูกใช้ในแต่ละชั้นซ่อน ถูกปรับค่าระหว่างกระบวนการสร้างแบบจำลองโดยมีค่าเป็นเลขฐานสอง คือ 0 กับ 1



รูปที่ 3.10 CMAC module

รูปที่ 3.10 cerebellar model articulation control (CMAC) จัดอยู่ในกลุ่มโครงข่ายใยประสาทเทียมแบบ feedforward ที่มีโครงสร้าง และหน้าที่เหมือนสมองมนุษย์ในส่วนที่เราเรียกว่า cerebellum



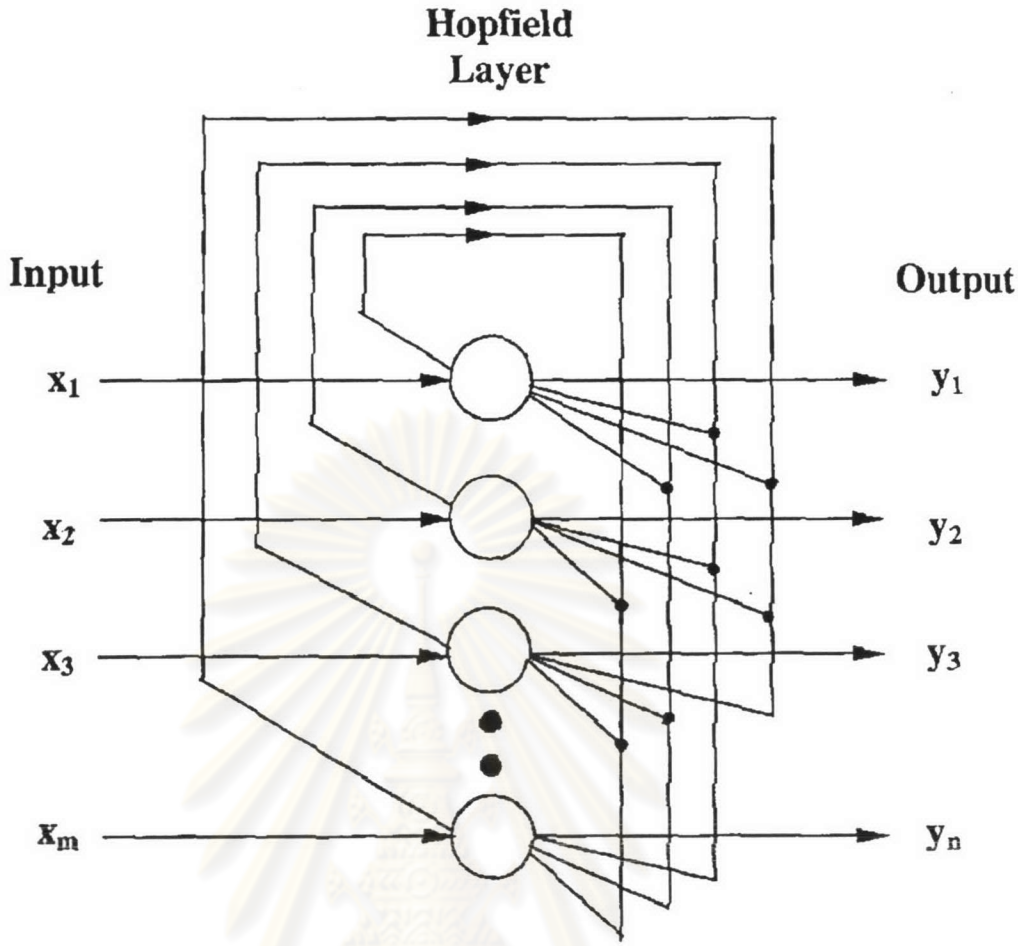
รูปที่ 3.11 GMDH network

รูปที่ 3.11 เป็นโครงสร้างที่ไม่เหมือนกับ feedforward ที่กล่าวมาก่อนหน้านี้ ซึ่งมีลักษณะโครงสร้างที่คงที่ แต่ GMDH มีโครงสร้างซึ่งจำนวนชั้นของโครงสร้างสามารถเพิ่มขึ้นระหว่างกระบวนการสร้างแบบจำลอง แต่ละหน่วยประสาทใน GMDH โดยปกติจะมี 2 ตัวแปรด้านเข้า x_1 และ x_2 ซึ่งสร้าง ตัวแปรด้านออก y โดยที่เป็นการรวมกับของสมการกำลังสองของแต่ละตัวแปรด้านเข้า

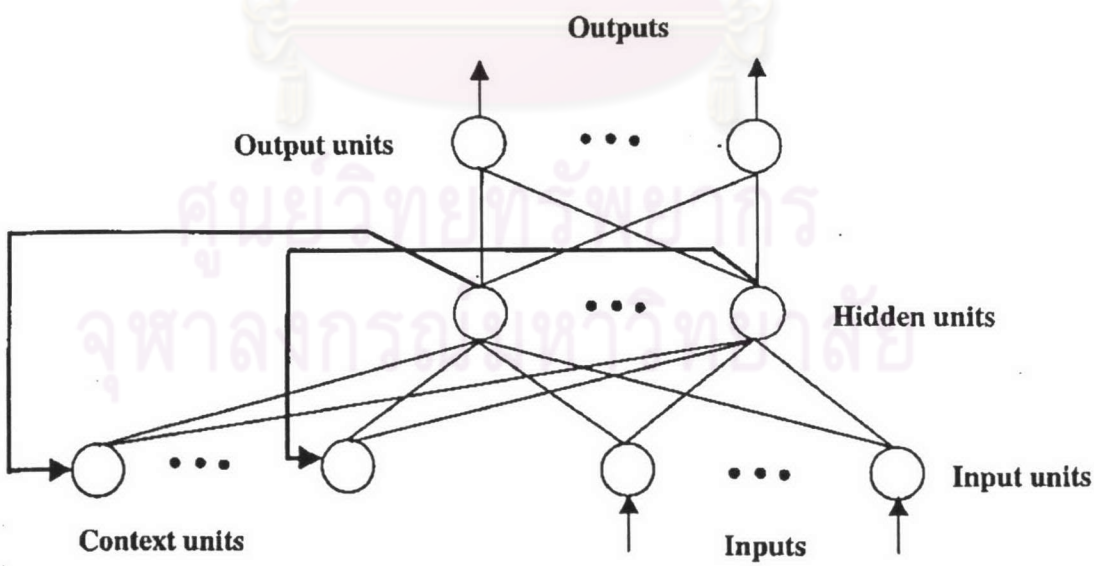
3.3.2 โครงสร้างแบบ recurrent

ในโครงสร้างแบบ recurrent ตัวแปรด้านออกของบางหน่วยประสาทจะป้อนค่ากลับไปสู่ตัวหน่วยประสาทเดียวกัน หรือหน่วยประสาทชั้นก่อนหน้า ดังนั้นค่าจะถูกส่งผ่านในแบบจำลองทั้งในทิศทางที่ไปข้างหน้าและทิศทางย้อนกลับ ตัวอย่างของ โครงสร้างแบบ recurrent คือ Hopfield network, Elman network และ Jordan network โครงสร้างแบบ recurrent มักมีหน่วยความจำแบบ dynamic คือ ตัวแปรด้านออกที่ให้ค่ากับตัวแปรด้านเข้าในปัจจุบัน มีค่าพอๆกับตัวแปรด้านเข้าและตัวแปรด้านออกก่อนหน้า

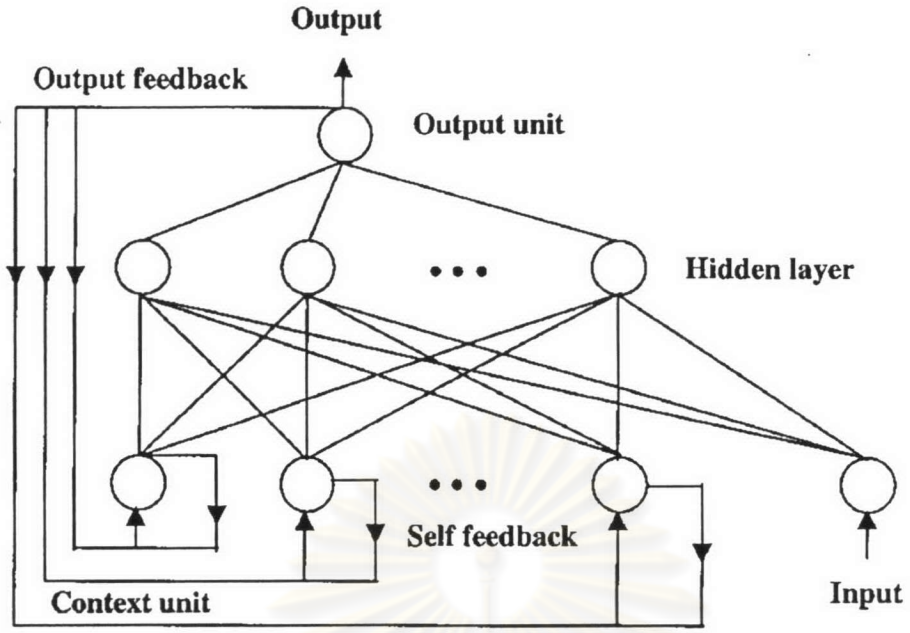
รูปที่ 3.12 แสดงค่าของ Hopfield network ซึ่งมีหน่วยประสาทชั้นเดียว (ไม่มีชั้นซ่อน) และแต่ละหน่วยประสาทเชื่อมโยงกับหน่วยประสาทอื่นๆ ซึ่งแบบโครงสร้างแบบ recurrent



รูปที่ 3.12 Hopfield network



รูปที่ 3.13 Elman network



รูปที่ 3.14 Jordan network

รูปที่ 3.13 และ รูปที่ 3.14 แสดงภายของแบบจำลอง Elman และ Jordan ตามลำดับ แบบจำลองโครงข่ายใยประสาทเทียมมีโครงสร้างแบบหลายชั้นคล้ายๆกับ โครงสร้างของ multi-layer perceptron โดยทั้งคู่มีการเพิ่ม ชั้นซ่อนและชั้นซ่อนพิเศษที่เรียกว่า context or state layer ชั้นเหล่านี้ได้ส่งผ่านค่าจากชั้นซ่อนปกติ หลังจากนั้นตัวแปรด้านออกใน Jordan network มีการเชื่อมโยงจากหน่วยประสาทในชั้น context กลับสู่ตัวเอง และตัวแปรด้านออกในหน่วยประสาทใน context ถูกส่งผ่านค่าไปยังชั้นซ่อน

สำหรับแบบจำลองโครงข่ายใยประสาทเทียม multi-layer feedforward ได้รับความนิยมสูงสุดกว่าแบบจำลองชนิดอื่นๆ เพราะ ตัวแบบจำลองที่ไม่ซับซ้อนเมื่อเปรียบเทียบกับแบบจำลองอื่นๆ นอกจากนั้น multi-layer feedforward ประสบความสำเร็จในการใช้งานในหลายๆ สาขาของวิศวกรรม

3.4 การเรียนรู้ของแบบจำลองโครงข่ายใยประสาทเทียม

แบบจำลองโครงข่ายใยประสาทเทียมสามารถแบ่งตามชนิดของกระบวนการเรียนรู้ได้หลายๆวิธี การเรียนรู้ของโครงข่ายใยประสาทใช้วิธีการเรียนรู้ (learning rule) เพื่อสอนให้

เครือข่ายทำการคำนวณหาผลลัพธ์ในการทำงานบางอย่าง โดยการปรับเปลี่ยนค่าถ่วงน้ำหนักของเครือข่าย กฎการเรียนรู้ของโครงข่ายใยประสาทมีหลายแบบ แต่ที่ใช้กันทั่วไปมี 2 แบบ คือ

3.4.1 การเรียนรู้แบบมีการควบคุม (supervised learning) การเรียนรู้แบบนี้ต้องมีชุดตัวอย่างในการปรับสอน (training set) เพื่อแสดงวัตถุประสงค์ที่แท้จริงในการทำงานของเครือข่ายนั้นๆ ตัวอย่างเช่น

$$\{x_1, t_1\}, \{x_2, t_2\}, \{x_3, t_3\}, \ldots, \{x_P, t_P\}$$

โดยที่ x_i คือตัวแปรด้านเข้าของโครงข่ายใยประสาท (input variable)

t_i คือผลลัพธ์ที่ต้องการ (target)

P คือจำนวนรูปแบบ (patterns) ที่ใช้ในการปรับสอนทั้งหมด

เมื่อป้อนตัวแปรด้านเข้า (input variable) ให้แก่โครงข่ายใยประสาท เครือข่ายจะคำนวณหาผลลัพธ์คือตัวแปรด้านออก (output variable) แล้วนำผลลัพธ์จากโครงข่ายใยประสาทนี้ไปเปรียบเทียบกับผลลัพธ์ที่ต้องการจากนั้นนำค่าผิดพลาดที่เกิดขึ้นไปใช้ปรับค่าถ่วงน้ำหนักของเครือข่ายโดยอาศัยกฎการเรียนรู้ และวิธีการทำซ้ำ (iterative method) เพื่อให้ตัวแปรด้านออกมีค่าเข้าใกล้ผลลัพธ์ที่ต้องการ

3.4.2 การเรียนรู้แบบไม่มีการควบคุม (unsupervised learning) การเรียนรู้แบบนี้ค่าถ่วงน้ำหนักจะถูกปรับเพื่อให้สอดคล้องกับตัวแปรด้านเข้าเท่านั้น ไม่ต้องใช้ค่าผลลัพธ์ที่ต้องการ ส่วนใหญ่การเรียนรู้แบบนี้จะใช้ในงานจำแนกประเภทของข้อมูลด้านเข้า (clustering operation or network classifier) ถ้าข้อมูลด้านเข้าเป็นประเภทเดียวกัน เครือข่ายจะให้ผลลัพธ์อย่างเดียวกัน ตัวอย่างกฎการเรียนรู้แบบนี้คือ adaptive resonance theory และ Kohonen self-organizing map

3.5 กระบวนการเรียนรู้แบบเพอร์เซพตรอน

กระบวนการเรียนรู้แบบเพอร์เซพตรอนนี้เป็นวิธีการเรียนรู้แบบมีการควบคุมวิธีหนึ่งที่ใช้ในการแก้ปัญหาที่สามารถแยกออกจากกันได้เชิงเส้น (linearly separable) ซึ่งมีคำตอบและวิธีการคำนวณค่าถ่วงน้ำหนักที่เหมาะสม โดยจะปรับค่าถ่วงน้ำหนักไปทีละยูนิตตามตัวแปรด้านออกที่เกิดขึ้น ถ้าตัวแปรด้านออกมีเครื่องหมายเหมือนกับผลลัพธ์ที่ต้องการแล้วจะไม่ปรับค่า

ถ่วงน้ำหนัก แต่ถ้าตัวแปรด้านออกมีเครื่องหมายตรงกันข้ามจะต้องทำการปรับค่าถ่วงน้ำหนัก โดยเป็นสัดส่วนเดียวกับผลคูณของตัวแปรด้านเข้ากับผลลัพธ์ที่ต้องการ ดังสมการ

$$w_{ki}^{new} = w_{ki}^{old} + \Delta w_{ki} \tag{3.6}$$

$$\Delta w_{ki} = (t_k - y_k)x_k \tag{3.7}$$

$$b_k^{new} = b_k^{old} + (t_k - y_k) \tag{3.8}$$

โดยที่ t_k คือค่าผลลัพธ์ที่ต้องการจากยูนิตที่ k

y_k คือค่าตัวแปรด้านออกของยูนิตที่ k

3.6 กระบวนการเรียนรู้แบบเกรเดียนต์เดสเซนต์ (Gradient descent learning)

การเรียนรู้แบบเกรเดียนต์เดสเซนต์ เป็นการเรียนรู้พื้นฐานของเครือข่ายที่ประกอบด้วยยูนิตที่ฟังก์ชันเชิงเส้น หรือฟังก์ชันที่ไม่เป็นเชิงเส้นเช่น ฟังก์ชันซิกมอยด์ต่างๆ ที่มีค่าต่อเนื่อง (continunous functions) เป็นแอกติเวชันฟังก์ชัน

$$E = 1/2\sum_k (t_k - y_k)^2 \tag{3.9}$$

$$E = 1/2\sum_k (t_k - f(\sum_i w_{ki}x_i + b_k))^2 \tag{3.10}$$

ฟังก์ชันค่าผิดพลาด (E) เป็นสมการกำลังสองในรูปของค่าถ่วงน้ำหนักกับค่าผิดพลาด มีพื้นผิวเป็นพาราโบลาและประกอบด้วยค่าต่ำสุดค่าหนึ่ง กระบวนการเรียนรู้แบบเกรเดียนต์เดสเซนต์จะปรับเวกเตอร์ถ่วงน้ำหนักในทิศทางของเวกเตอร์ตัวแปรด้านเข้า โดย E จะมีค่าเป็นบวกเสมอ ถ้า w_{ki} และ b_k ถูกปรับเข้าสู่ค่าที่ต้องการ ค่าของฟังก์ชันค่าผิดพลาดจะน้อยลงในลักษณะที่ลู่เข้าสู่ศูนย์เมื่อผ่านกระบวนการทำซ้ำไปเรื่อยๆ

การปรับค่าถ่วงน้ำหนักและค่าไบอัสทำได้โดยการปรับค่าฟังก์ชันลงมายังจุดต่ำสุดบนพื้นผิวที่ถูกนิยามใน space ในกรณีวิธีเกรเดียนต์เดสเซนต์จะปรับเปลี่ยนค่า w_{ki} และค่า b_k แต่ละตัวด้วยค่า Δw_{ki} และ Δb_k ซึ่งเป็นสัดส่วนกับค่าเกรเดียนต์ของฟังก์ชันค่าผิดพลาดที่นิยามไว้

$$\Delta w_{ki} = -\alpha \frac{\partial E}{\partial w_{ki}} \tag{3.11}$$

$$\Delta w_{ki} = \alpha \sum_{P} (t_k - y_k) x_i \tag{3.12}$$

$$\Delta b_k = -\alpha \frac{\partial E}{\partial b_k} \tag{3.13}$$

$$\Delta b_k = \alpha \sum_{P} (t_k - y_k) \tag{3.14}$$

โดยที่ P คือจำนวนรูปแบบ (patterns) ทั้งหมดที่ใช้ในการปรับสอน

α คือค่าอัตราการเรียนรู้ (learning rate parameter)

ถ้าสามารถปรับเปลี่ยนค่าถ่วงน้ำหนักได้อย่างอิสระ ได้ทีละรูปแบบของตัวแปรด้านเข้า จะได้กำหนดให้

$$\delta_k = t_k - y_k \tag{3.15}$$

จะได้

$$\Delta w_{ki} = \alpha \delta_k x_i \tag{3.16}$$

และ

$$\Delta b_k = \alpha \delta_k \tag{3.17}$$

สมการ (3.15) และ (3.17) มีชื่อเรียกว่า Delta rule สังเกตได้ว่าสมการทั้งหมดเป็นผลมาจากวิธีเกรเดียนต์เดสเซนต์ ซึ่งง่ายต่อการนำไปใช้กับเครือข่ายที่มีหลายชั้น แต่ต้องใช้กับฟังก์ชันแอกติเวชันที่ต่อเนื่องหรือสามารถหาอนุพันธ์ได้ และค่า α ที่เหมาะสมจะทำให้การเปลี่ยนแปลงของค่าถ่วงน้ำหนัก เป็นไปในทางที่ทำให้ค่าความผิดพลาดลดลง หรือทำให้กระบวนการทำซ้ำลู่เข้า โดย α มีค่าน้อยการคำนวณจะใช้เวลานาน ในทางตรงกันข้ามถ้า α มีค่ามากการคำนวณจะเร็วขึ้น แต่อาจทำให้เกิดการแกว่งได้

3.7 กระบวนการเรียนรู้แบบแพร่กระจายความผิดพลาดย้อนกลับ

ความสามารถที่ยอดเยี่ยมของระบบโครงข่ายใยประสาทเทียมคือความสามารถในการเรียนรู้จากข้อมูลตัวอย่างและเพิ่มประสิทธิภาพจากการเรียนรู้ ด้วยการสร้างความสัมพันธ์ระหว่างตัวแปรด้านเข้าไปสู่ตัวแปรด้านออกโดยผ่านวิธีการปรับเปลี่ยนค่าถ่วงน้ำหนัก (weight)

และค่าไบอัส (bias) ตามการเรียนรู้แบบเกรเดียนต์เดสเซนต์ ยิ่งเพิ่มความสามารถในการเรียนรู้มากขึ้นในทุกๆรอบของการคำนวณ

โดยทั่วไปการเรียนรู้แบบ back-propagation ประกอบด้วย 2 ส่วนหลัก คือ ส่วนการเรียนรู้ไปข้างหน้าและส่วนการกระจายค่าความผิดพลาดย้อนกลับ ในส่วนการเรียนรู้ไปข้างหน้า ตัวแปรด้านเข้า (input) จะถูกใช้ส่งผ่านค่าไปสร้างค่าน้ำหนัก และไบอัส ในแบบจำลองผ่านไปทีละชั้น จนในที่สุดได้ผลลัพธ์ออกมาทางตัวแปรด้านออก (output) ของแบบจำลอง ค่าผิดพลาดในแต่ละรอบของการคำนวณ คือค่าของความแตกต่างระหว่างตัวแปรด้านออกกับค่าผลลัพธ์ที่ต้องการ ค่าความผิดพลาดที่เกิดขึ้นจะถูกส่งไปยังส่วนกระจายค่าความผิดพลาดกลับไปสู่แบบจำลองเพื่อปรับค่าถ่วงน้ำหนักและค่าไบอัสในแบบจำลองทีละชั้น

แบบจำลองโครงข่ายใยประสาทเทียมประกอบไปด้วยการเชื่อมโยงกันแบบทั่วถึงของเครือข่าย หมายความว่าเครือข่ายในทุกๆชั้นของแบบจำลองเชื่อมโยงกับเครือข่ายทั้งหมดของชั้นก่อนหน้า มีการส่งผ่านค่าทางเครือข่ายไปข้างหน้าที่ละชั้นๆ ในแต่ละรอบของการคำนวณจะมีการปรับค่าน้ำหนักของแต่ละหน่วยประสาท j หน่วยประสาท j จะได้รับค่าที่สร้างมาจากชั้นก่อนหน้า ตัวแปรด้านออกของหน่วยประสาท j จะถูกแทนด้วย $y_j(n)$ ตัวแปรด้านออกที่ได้จะถูกเปรียบเทียบกับ ค่าผลลัพธ์ที่ต้องการ $t_j(n)$ ค่าผิดพลาดที่เกิดขึ้นคือ $e_j(n)$

$$e_j(n) = t_j(n)-y_j(n) \tag{3.18}$$

ค่าผิดพลาด $e_j(n)$ ที่ได้จะถูกส่งผ่านกระจายย้อนกลับไปปรับค่าถ่วงน้ำหนักในชั้นก่อนหน้า การปรับค่าถ่วงน้ำหนักนี้จะสร้างให้ตัวแปรด้านออก $y_j(n)$ มีค่าผลลัพธ์ที่ได้ใกล้เคียงกับค่าผลลัพธ์ที่ต้องการ $t_j(n)$ มากยิ่งขึ้น กระบวนการนี้นำไปสู่การลดค่าผิดพลาดลงเรื่อยๆของแบบจำลองโครงข่ายใยประสาท $E_o(n)$ คือค่าของความผิดพลาดที่ถูกส่งกลับ

$$E_o(n) = 1/2e_j^2(n) \tag{3.19}$$

ค่าของความผิดพลาดทั้งหมดจะถูกคำนวณจากทุกหน่วยประสาทในแบบจำลองในชั้นสุดท้าย

$$E(n) = 1/2\sum_{j\in C} e_j^2(n) \tag{3.20}$$

ที่ซึ่ง C คือหน่วยประสาทชั้นนอกของเครือข่าย

ค่าเฉลี่ยของค่าผิดพลาดสร้างจากผลรวมของ E(n) หารด้วย จำนวนตัวอย่างข้อมูลทั้งหมดในกลุ่มสร้างแบบจำลอง

$$E_{av}(n) = 1/N \sum_{n=1}^{N} E(n) \tag{3.21}$$

ค่าผิดพลาด E(n) และค่าความผิดพลาดเฉลี่ย E_{av} คือฟังก์ชันของค่าถ่วงน้ำหนักและค่าไบอัส สำหรับชุดข้อมูลในแบบจำลอง E_{av} เป็นตัวแทนของความสามารถในการเรียนรู้ของแบบจำลอง ด้วยการปรับค่าถ่วงน้ำหนักและค่าไบอัสเพื่อให้มีค่า E_{av} น้อยที่สุด

หน่วยประสาท j ถูกส่งผ่านค่าไปที่ละชั้น โดยพยายามสร้างผลลัพธ์ซึ่งมาจากตัวแปรด้านเข้าผ่านมายังฟังก์ชันแอกติเวชัน (activate function)

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) + b_j(n) \tag{3.22}$$

ซึ่ง m คือจำนวนทั้งหมดของตัวแปรด้านเข้า

$b_j(n)$ คือ ไบอัสที่มาจากหน่วยประสาท j

$$y_j(n) = f_j((v_j(n)) \tag{3.23}$$

error back-propagation ใช้สัดส่วนของการปรับแก้ค่าถ่วงน้ำหนัก $\Delta w_{ji}(n)$ กับค่าถ่วงน้ำหนัก $w_{ji}(n)$ จากอนุพันธ์ของ $\partial E(n)/\partial w_{ji}(n)$ ตามสมการ

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{3.24}$$

อนุพันธ์ของสมการที่ (3.20) เทียบกับ $e_j(n)$

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{3.25}$$

อนุพันธ์ของสมการที่ (3.18) เทียบกับ $y_j(n)$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{3.26}$$

อนุพันธ์ของสมการที่ (3.23) เทียบกับ $v_j(n)$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = f_j^{'}(v_j(n)) \tag{3.27}$$

สุดท้าย อนุพันธ์ของสมการที่ (3.22) เทียบกับ $w_{ji}(n)$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{3.28}$$

แทนค่าสมการ (3.25) , (3.26) , (3.27) , (3.28) ลงในสมการ (3.24)

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n) \cdot f_j'(v_j(n)) \cdot y_i(n) \tag{3.29}$$

ค่าปรับแก้ค่าถ่วงน้ำหนัก $\Delta w_{ji}(n)$ ตามกฎของ delta rule

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{3.30}$$

โดยค่า η คือ learning-rate parameter ของ back-propagation เป็นไปตามกฎของความต่อเนื่องของค่าถ่วงน้ำหนักของเครือข่ายที่ลดลงตามค่าความแตกต่าง (delta) ระหว่างตัวแปรด้านออกกับค่าผลลัพธ์ที่ต้องการ โดยที่กฎการเปลี่ยนแปลงค่าถ่วงน้ำหนักเป็นไปในทิศทางเดียวกับการลดลงของ mean square error (E(n)) ของเครือข่ายพร้อมกับที่ค่าความผิดพลาดจะถูกส่งกลับไปในชั้นก่อนของเครือข่าย จนกระทั่งย้อนกลับไปถึงชั้นแรกสุดของเครือข่าย

การใช้เครื่องหมายลบในสมการ ได้มาจาก gradient descent ของค่าถ่วงน้ำหนักน้ำหนักพร้อมกับทิศทางการลดลงของค่าผิดพลาด E(n)

จากสมการ (3.29) , (3.30)

$$\Delta w_{ji} = \eta \cdot \delta_j(n) \cdot y_i(n) \tag{3.31}$$

โดยที่ local gradient $\delta_j(n)$ คือ

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_{ji}(n)} = -e_j(n) \cdot f_j'(v_j(n)) \tag{3.32}$$

local gradient แสดงถึงการเปลี่ยนแปลงของค่าน้ำหนัก ตามสมการ (3.32) local gradient $\delta_j(n)$ สำหรับตัวแปรด้านออก j คือความสัมพันธ์ระหว่างค่าผิดพลาด $e_j(n)$ ของแต่ละหน่วยประสาทกับค่าของอนุพันธ์ $f_j'(v_j(n))$ ของฟังก์ชันแอกติเวชัน (activation function) ปัจจัยสำคัญที่เกี่ยวกับ $\Delta w_{ji}(n)$ คือค่าผิดพลาด $e_j(n)$ ที่ตัวแปรด้านออก j สามารถแบ่งออกได้เป็น 2 กรณี ขึ้นกับตำแหน่งของหน่วยประสาท j กรณีที่ 1 หน่วยประสาท j คือหน่วยประสาทชั้นนอก ในกรณีนี้ง่ายที่จะคำนวณเพราะค่าตัวแปรด้านออกของเครือข่ายเทียบกับค่าผลลัพธ์ที่ต้องการได้โดยตรง จึงสามารถคำนวณค่าผิดพลาดได้ กรณีที่ 2 หน่วยประสาท j อยู่ในชั้นซ่อน (hidden layer) ค่าความผิดพลาดที่มาจากหน่วยประสาทในชั้นซ่อนไม่สามารถคำนวณได้โดยตรง จึงต้องคำนวณค่าความผิดพลาดเทียบกับตัวแปรด้านออกของเครือข่าย

<u>กรณี 1</u> หน่วยประสาทอยู่ที่ชั้นนอกสุดของเครือข่าย

เมื่อหน่วยประสาทตั้งอยู่ที่ชั้นนอกสุดของเครือข่าย สามารถเทียบกับผลลัพธ์ที่ต้องการได้ สามารถคำนวณค่าของความผิดพลาดได้โดยตรงจากสมการ (3.18) และคำนวณ local gradient $\delta_j(n)$ จากสมการ (3.32)

<u>กรณี 2</u> หน่วยประสาทอยู่ในชั้นซ่อนของเครือข่าย

เมื่อหน่วยประสาทตั้งอยู่ที่ชั้นซ่อนของเครือข่าย จึงไม่สามารถหาค่าที่ถูกต้องเพื่อคำนวณความผิดพลาดได้ ค่าผิดพลาดที่อยู่ในชั้นซ่อนจึงจะถูกคำนวณโดยใช้ค่าผิดพลาดทั้งหมดที่ชั้นซ่อนได้เชื่อมต่อโดยตรง ที่ได้อธิบายก่อนหน้านี้ local gradient ต้องการจากการเปลี่ยนแปลงของน้ำหนักตามสมการ ตัว local gradient $\delta_j(n)$ ที่ถูกคำนวณมาจากชั้นซ่อนเท่ากับค่าผิดพลาด $e_j(n)$ ของหน่วยประสาทและอนุพันธ์ $f'_j(v_j(n))$ ของ activation function

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = -\frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \cdot f'_j(v_j(n)) \qquad (3.33)$$

ที่ซึ่งหน่วยประสาท j คือหน่วยประสาทในชั้นซ่อน

$$E(n) = \frac{1}{2}\sum_{k \in C} e_k^2(n) \qquad (3.34)$$

ที่ซึ่งหน่วยประสาท k คือหน่วยประสาทชั้นนอกสุด (k ใช้แทน j เพื่อไม่ให้สับสน) หาอนุพันธ์สมการ เทียบกับ $y_j(n)$ จะได้

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \qquad (3.35)$$

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)} \qquad (3.36)$$

อย่างไรก็ตาม ค่าผิดพลาดของตัวแปรด้านออก คือ

$$e_k(n) = t_k(n) - y_k(n) = t_k(n) - f_k(v_k(n)) \qquad (3.37)$$

ที่ซึ่ง k คือ หน่วยประสาทชั้นนอกสุด หาอนุพันธ์ทั้ง 2 ข้างเทียบกับ $v_k(n)$

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -f'_k(v_k(n)) \qquad (3.38)$$

หน่วยประสาท k ทำให้เกิด

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) \cdot y_j(n) + b_k \qquad (3.39)$$

ซึ่ง m คือ จำนวนตัวแปรทั้งหมดและ b_k คือ ค่าไบอัสที่ถูกใช้กับหน่วยประสาท k โดยหาอนุพันธ์เทียบกับ $y_i(n)$

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{3.40}$$

แทนค่าสมการ (3.38) และ (3.40) ใน (3.36)

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n) \cdot f_k'(v_k(n)) \cdot w_{kj}(n) = -\sum_k \delta_k(n) \cdot w_{kj}(n) \tag{3.41}$$

จากนิยามของ local gradient δ_k ที่ใช้ในสมการ (3.33) โดยใช้ k แทนใน j สุดท้ายแทน (3.41) ใน (3.32) จะได้สูตร back-propagation สำหรับ local gradient $\delta_j(n)$ คือ

$$\delta_j(n) = f_j'(v_j(n)) \cdot \sum_k \delta_k(n) \cdot w_{kj}(n) \tag{3.42}$$

โดย หน่วยประสาท j คือหน่วยประสาทชั้นซ่อน

ตัวแปร $f_j'(v_j(n))$ ใช้ในการคำนวณหา local gradient $\delta_j(n)$ เนื่องมาจาก activated function ในชั้นซ่อน j ตัวแปรที่ใช้ในการคำนวณหา local gradient $\delta_j(n)$ มี 2 ชุดคือ $\delta_k(n)$ คือ local gradient ของชั้นนอก และ $w_{kj}(n)$ ค่าถ่วงน้ำหนักของตัวแปรระหว่างชั้น j และ k

วิธีการคำนวณแบบ back-propagation สามารถสรุปได้ง่ายๆ คือ ใช้ค่าปรับแก้ค่าถ่วงน้ำหนัก $\Delta w_{ji}(n)$ ระหว่างหน่วยประสาท i กับ j โดยใช้ delta rule

(ค่าปรับแก้ความผิดพลาด)=(ค่าเรียนรู้).(local gradient).(ตัวแปรด้านเข้า)

ค่าของ local gradient $\delta_j(n)$ขึ้นกับ ตำแหน่งของหน่วยประสาท j ว่าอยู่ในชั้นซ่อนหรือชั้นนอก

1. ถ้าหน่วยประสาท j คือหน่วยประสาทชั้นนอกสุด $\delta_j(n)$ จะเท่ากับผลจากอนุพันธ์ของ $f_j'(v_j(n))$ และค่าความผิดพลาด $e_j(n)$ ทั้งคู่ที่เกี่ยวพันกับชั้น j
2. ถ้าหน่วยประสาท j อยู่ในชั้นซ่อน $\delta_j(n)$ จะเท่ากับอนุพันธ์ของ $f_j'(v_j(n))$ และผลรวมของน้ำหนักของ local gradient ที่คำนวณมาจากหน่วยประสาทในชั้นซ่อนถัดไปหรือชั้นนอกสุดที่เชื่อมโยงกับหน่วยประสาท j

3.8 จำนวนของชั้นซ่อนในแบบจำลองโครงข่ายใยประสาทเทียม

โดยทั่วๆ ไป จะไม่มีวิธีที่ถูกต้องที่ใช้หาจำนวนที่เหมาะสมของหน่วยประสาทในชั้นซ่อนในแต่ละแบบจำลอง ปัญหานี้กลายเป็นปัญหาที่ซับซ้อนมากยิ่งขึ้นเมื่อจำนวนของชั้นซ่อนมีค่ามากยิ่งขึ้น จากวิธีการคำนวณของแบบจำลองเอง ปรากฏว่าการเพิ่มขึ้นของจำนวนหน่วยประสาทในชั้นซ่อนนำไปสู่ความสามารถในการสร้างความสัมพันธ์ได้ดียิ่งขึ้น เพราะมันจะทำให้สร้างฟังก์ชันที่ซับซ้อนมากยิ่งขึ้นได้ในแบบจำลอง อย่างไรก็ตาม จำนวนที่มากของหน่วยประสาท

ในชั้นซ่อนสามารถชี้นำคำตอบได้ในขณะที่สร้างความสัมพันธ์เพื่อให้คำตอบเข้าใกล้เป้าหมายมากที่สุด จำนวนที่มากขึ้นของหน่วยประสาทในชั้นซ่อนนำไปสู่การคำนวณที่ช้าลงของแบบจำลองในระหว่างการสร้างแบบจำลอง เนื่องจากต้องใช้เวลานานในการเข้าสู่จุดต่ำสุด นอกจากนี้สัญญาณความผิดพลาดอาจยุ่งเหยิง เมื่อการกระจาย (propagation) ของค่าถ่วงน้ำหนักมากเกินไป อาจทำให้เกิดจุดต่ำสุดในบริเวณ (local minima) เพิ่มขึ้นได้ ดังนั้นจึงมักนิยมใช้โครงข่ายเพียง 2 ชั้น

เพื่อให้ได้ค่าที่ดีที่สุดของแบบจำลองจำเป็นต้องจัดทำการทดสอบแบบจำลอง การทดสอบควรเริ่มมาจากชุดข้อมูลจำนวนหนึ่ง ซึ่งเป็นชุดข้อมูลของข้อมูลชุดใหญ่ โดยแบ่งออกเป็น 2 กลุ่ม กลุ่มชุดข้อมูลสำหรับสร้างแบบจำลอง (training set) กับกลุ่มชุดข้อมูลสำหรับทำการทดสอบแบบจำลอง (testing set) แบบจำลองที่มีจำนวนหน่วยประสาทที่แตกต่างกันจะถูกนำมาทดสอบหาจำนวนหน่วยประสาทที่เหมาะสมที่สุดสำหรับแบบจำลองโดยดูจากค่าของความผิดพลาดที่แสดงออกมา และแบบจำลองจะถูกทดสอบอีกครั้งด้วยชุดข้อมูลตรวจสอบความถูกต้องของแบบจำลอง (validation set) เพื่อนำไปใช้งาน

3.9 จำนวนของกลุ่มตัวอย่างที่ใช้ในแบบจำลอง

ปัจจัยที่สำคัญซึ่งแสดงความสามารถของการเรียนรู้ของกลุ่มตัวอย่าง มาจากเครือข่ายสร้างแบบจำลองจากชุดข้อมูลสร้างแบบจำลอง ตัวแปรที่ใช้กับค่าผลลัพธ์ที่ต้องการ สิ่งที่สำคัญอย่างยิ่ง คือข้อมูลในกลุ่มสร้างแบบจำลองต้องครอบคลุมในทุกขอบเขตของคำตอบของปัญหาที่จะใช้วิเคราะห์ ทั่วไปเมื่อเพิ่มชุดข้อมูลมากขึ้น แบบจำลองสามารถสร้างความสัมพันธ์ได้ดียิ่งขึ้นและลดค่าของความผิดพลาดที่เกิดขึ้นของชุดข้อมูลเองได้ด้วย นำไปสู่การสร้างความสัมพันธ์ที่ถูกต้องมากยิ่งขึ้นของแบบจำลอง

ในขั้นตอนแรก กลุ่มข้อมูลสำหรับสร้างแบบจำลอง ควรจะนำไปสู่โครงสร้างที่ดีของแบบจำลอง (จำนวนของชั้นซ่อน,จำนวนหน่วยในชั้นซ่อน และจำนวนรอบของการคำนวณ) ข้อมูลชุดแรกอาจจะใช้หาความสัมพันธ์เล็กๆ ของแบบจำลอง การทดสอบนี้ใช้เพื่อศึกษาหาตัวแปรที่เกี่ยวข้องเพื่อตรวจสอบแบบจำลองก่อนจะเริ่มใช้กับชุดข้อมูลกลุ่มใหญ่

การกระจายตัวของชุดข้อมูลในแบบจำลอง ตัวแปรต่างๆจะต้องอยู่ภายในชุดข้อมูลทั้งหมด จะมีผลสำคัญกับการเรียนรู้และประสิทธิภาพของแบบจำลอง เนื่องจากแบบจำลองไม่สามารถทำนายในส่วนที่นอกเหนือจากขอบเขตข้อมูลที่เรียนรู้ไว้ได้ ชุดของข้อมูลที่ใช้สำหรับ

สร้างแบบจำลองน่าจะต้องครอบคลุมทุกช่วงของข้อมูล ซึ่งกลุ่มข้อมูลชุดสร้างแบบจำลองจะต้องกระจายทั่วถึง

Flood และ Kartam (1994) แนะนำว่ากระบวนการสร้างแบบจำลองที่ดีกลุ่มของข้อมูลในชุดสร้างแบบจำลองจะต้องไม่แคบเกินไป ปัญหานี้สามารถที่จะแก้ไขได้โดยวิธีการปรับแก้ขอบเขตของข้อมูล (normalize) ข้อมูลชุดสร้างแบบจำลองในแต่ละตัวแปรที่ใส่เข้าไป เพิ่มตามสัดส่วนการเพิ่มขึ้นของขอบเขตข้อมูล วิธีการปรับแก้ขอบเขตข้อมูลเป็นไปตามชนิดของฟังก์ชัน เช่น sigmoid ฟังก์ชัน ตัวแปรด้านเข้าและตัวแปรด้านออกควรจะอยู่ในช่วง 0 ถึง 1 หรือ -1 ถึง 1 สำหรับ sigmoid ฟังก์ชันและ tangent sigmoid ฟังก์ชันตามลำดับ แนะนำว่าควรทำการ normalize ตัวแปรด้านเข้าและขาออกก่อนจะทำการสร้างแบบจำลอง และ normalize สามารถที่จะเพิ่มความเร็วในการคำนวณแบบจำลองได้ (Rafig M. et. Al., 2001)

3.10 การสร้างความสัมพันธ์ของข้อมูลที่เกินจริง

เนื่องจากการใช้แบบจำลองโครงข่ายใยประสาทเทียมเรียนรู้และวิเคราะห์หาผลลัพธ์ที่จะเกิดขึ้นกับข้อมูลชุดใหม่ จะต้องให้แบบจำลองเรียนรู้จากข้อมูลที่เตรียมไว้เพื่อสร้างความสัมพันธ์แล้วนำไปวิเคราะห์กับข้อมูลชุดใหม่ ถึงแม้ว่าแบบจำลองโครงข่ายใยประสาทจะสามารถสร้างความสัมพันธ์ระหว่างตัวแปรที่ซับซ้อนได้อย่างมากมาย แต่ความสามารถของมันก็กลับกลายเป็นจุดอ่อนที่สำคัญที่สุดด้วย จากการที่สร้างความสัมพันธ์ที่ไม่ถูกต้องออกมาจากข้อมูลที่เรียนรู้ที่มากเกินไป นำไปสู่การชี้นำของผลลัพธ์ ปัญหาที่เรียกว่า (overfitting) คือปัญหาสำคัญของการใช้แบบจำลองเครือข่าย มี 3 วิธีที่ใช้เพื่อหลีกเลี่ยงการเกิดปัญหา overfitting คือ เพิ่มปริมาณของข้อมูลที่นำมาใช้สร้างแบบจำลอง จำกัดปริมาณชั้นซ่อนและจำนวนหน่วยประสาทในชั้นซ่อน และจำกัดปริมาณรอบการคำนวณของแบบจำลอง

3.10.1 เพิ่มปริมาณข้อมูลที่นำมาใช้สร้างแบบจำลอง

การเพิ่มปริมาณของข้อมูลที่นำมาใช้สร้างแบบจำลอง เหมือนกับการเพิ่มขึ้นของข้อมูลที่แบบจำลองใช้เรียนรู้สร้างความสัมพันธ์ระหว่างตัวแปรด้านเข้าไปสู่ตัวแปรด้านออก และเพื่อลดการชี้นำจากกลุ่มของข้อมูลบางส่วนที่มีความผิดพลาดปะปน

3.10.2 จำกัดจำนวนชั้นซ่อนและหน่วยประสาทในชั้นซ่อน

ในแต่ละค่าถ่วงน้ำหนักเป็นเหมือนค่าพารามิเตอร์ตัวหนึ่งที่เกี่ยวกับความสามารถของแบบจำลองเพราะมันจะส่งค่าไปยัง activated function เพื่อสร้างความสัมพันธ์ของกลุ่มข้อมูล ในอีกความหมายหนึ่งจำนวนของค่าถ่วงน้ำหนัก คือระดับขั้นที่บอกความสามารถในการสร้างความสัมพันธ์ที่ซับซ้อนจากตัวแปรด้านเข้าไปสู่อีกตัวแปรด้านออก ที่นำไปสู่การสร้างความสัมพันธ์หาผลลัพธ์ที่ต้องการ ความสามารถของแบบจำลองจะถูกจำกัดลงถ้าจำนวนค่าถ่วงน้ำหนักถูกจำกัดลง การควบคุมจำนวนค่าถ่วงน้ำหนักมาจากการควบคุมจำนวนตัวแปรด้านเข้ากับจำนวนของชั้นซ่อน จำนวนของตัวแปรด้านเข้ามีความสำคัญการใช้จำนวนตัวแปรด้านเข้าหลายๆตัวแปรสร้างแบบจำลองเพิ่มความสามารถสร้างความสัมพันธ์ที่ซับซ้อนและมีความถูกต้องมากยิ่งขึ้น แต่ในอีกด้านหนึ่ง ตัวแปรด้านเข้าที่มากเกินไปกลายเป็นแบบจำลองที่มีความซับซ้อนในอีกความหมายของ overfitting การสร้างความสัมพันธ์ที่มากเกินจริง และเป็นความสัมพันธ์ที่ไม่ถูกต้อง จำนวนของตัวแปรด้านเข้าส่งผลถึงจำนวนของค่าถ่วงน้ำหนักภายในแบบจำลอง โดยจะเป็นการสร้างตัวแปรส่วนเกิน (redundant) เพิ่มขึ้นมาอย่างมากมายโดยไม่มีความหมาย ถ้าจำนวนตัวแปรด้านเข้ามากเกินไปโดยขาดความเข้าใจในพฤติกรรมและความสัมพันธ์ของแต่ละตัวแปร ความสัมพันธ์ที่ได้จากการสร้างแบบจำลองอาจจะไม่ถูกต้อง

3.10.3 จำกัดจำนวนรอบของการคำนวณ

ระหว่างการคำนวณ การสร้างความสัมพันธ์จะซับซ้อนมากยิ่งขึ้น โดยผ่านจุดที่เรียกว่าสร้างความสัมพันธ์ที่ดีที่สุดในแต่ละโครงสร้างของแบบจำลองและหลังจากนั้นแบบจำลองจะนำไปสู่ overfitting ถ้าใช้รอบการคำนวณที่มากเกินไป สามารถที่จะหยุดการสร้างแบบจำลองก่อนที่จะเกิดการ overfitting ได้ และเลือกใช้แบบจำลองที่ให้ค่าผิดพลาดต่ำที่สุด โดยการเกิด overfitting สามารถตรวจสอบได้จากชุดข้อมูลสำหรับตรวจสอบแบบจำลอง