บทที่ 2

ทฤษฎีเครือข่ายประสาทเทียม

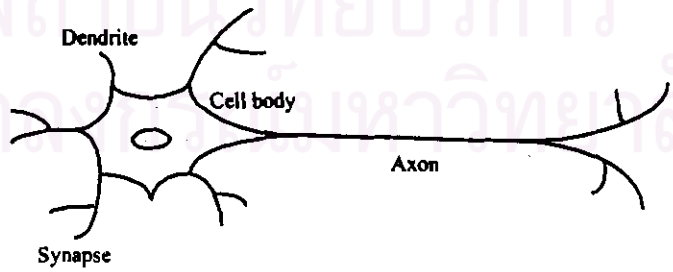
ปัจจุบันคอมพิวเตอร์เข้ามามีส่วนร่วมในการทำงานแทบทุกด้าน อย่างไรก็ตามคอมพิวเตอร์ยังคงมีขีดความสามารถที่จำกัดอยู่เมื่อเทียบกับสมองของมนุษย์ในด้านความสามารถในการจดจำ เรียนรู้ และพัฒนาจากประสบการณ์ในอดีต ซึ่งสามารถนำมาปรับใช้กับเหตุการณ์ในปัจจุบัน ดังนั้นจึงได้มีการศึกษาการทำงานของสมอง แล้วนำมาพัฒนาเป็นแบบจำลองของเซลล์ประสาท (Artificial neuron) เครือข่ายประสาทเทียม (Artificial neural network) รวมทั้งการศึกษากระบวนการเรียนรู้ (Learning algorithm) เพื่อนำมาประยุกต์ใช้กับคอมพิวเตอร์ ด้วยเหตุดังกล่าวจึงมีผู้คิดค้นพัฒนาทฤษฎีเครือข่ายประสาทเทียมขึ้น ซึ่งทฤษฎีเครือข่ายประสาทเทียมนี้เป็นแขนงหนึ่งในสาขาวิทยาการคอมพิวเตอร์ ซึ่งต้องการให้คอมพิวเตอร์สามารถทำงานในลักษณะเลียนแบบการทำงานของสมองมนุษย์ .โดยเฉพาะในงานที่คอมพิวเตอร์ยังทำไม่ได้หรือทำได้ไม่ดี เช่น การจดจำรูปแบบ การจัดหมวดหมู่ เป็นต้น

การพัฒนาทางด้านเครือข่ายประสาทเทียม [5] เริ่มตั้งแต่ปี พ.ศ.2483 โดย Mc Culloch และ Pitts ซึ่งเป็นผู้ที่นำเสนอหลักการทำงานของเครือข่ายประสาทเทียมอย่างง่าย ต่อมาในปี พ.ศ.2503 Widrow และ Hoff ได้พัฒนากฎการเรียนรู้สำหรับเครือข่ายประสาทขึ้นมาใหม่ สำหรับใช้กับเครือข่ายประสาทเทียมอย่างง่ายที่ประกอบด้วยยูนิตเชิงเส้นเป็นหลัก ซึ่งยังมีใช้อยู่ในปัจจุบัน เรียกว่า Widrow-Hoff learning rule หลังจากนั้นมีผู้คิดค้นพัฒนาเครือข่ายแบบใหม่ขึ้นมากมาย เพื่อแก้ปัญหาที่มีความซับซ้อนมากขึ้น แต่ยังไม่ประสบผลสำเร็จในการปรับปรุงกฎการเรียนรู้ให้ดีขึ้น เช่น ปี พ.ศ.2515 Kohonen ได้คิดค้นเครือข่ายที่เรียกว่า Kohonen network ปี พ.ศ.2519 Grossberg ได้คิดค้นเครือข่ายที่สามารถเรียนรู้ได้ด้วยตัวเอง (Self-organizing network) ต่อมาเมื่อวิทยาการด้านคอมพิวเตอร์พัฒนาไปมากขึ้น เครื่องคอมพิวเตอร์มีความเร็วในการคำนวณสูงขึ้น ส่งผลให้มีการคิดค้นกระบวนการเรียนรู้ใหม่ๆ ที่มีความสามารถในการปรับสอนสูงขึ้น ที่สำคัญคือการคิดค้นกระบวนการเรียนรู้แบบแพร่กระจายความผิดพลาดกลับ (Error back-propagation learning rule) ในปี พ.ศ.2523 ซึ่งนำไปใช้ได้กับเครือข่ายประสาทแบบหลายชั้น และเป็นการเรียนรู้ชนิดที่จะนำมาใช้ในวิทยานิพนธ์นี้

ปัจจุบันได้มีการนำทฤษฎีเครือข่ายประสาทเทียมมาประยุกต์ใช้กับงานจริงกันอย่างกว้างขวาง เช่น งานทางด้านโทรคมนาคม นำมาใช้ในการบีบข้อมูลและภาพ (Image and data compression) การแปลภาษาพูด (Real time translation of spoken language) การจดจำเสียง (Voice recognition) หรือเรียกโดยรวมว่าการจดจำรูปแบบ (Pattern recognition) ซึ่งมีการสอนให้เครือข่ายประสาทเทียมจดจำรูปแบบเดิมไว้ก่อน แล้วสามารถจำรูปแบบนั้นๆได้เมื่อพบซ้ำอีกครั้ง นอกจากนี้ยังมีงานในด้านอื่นๆ อีกคือ การประมาณค่าฟังก์ชัน (Function approximation) การค้นหาเป้าหมาย (Trajectory control) การวิเคราะห์ตลาด (Market analysis) การวิเคราะห์ดัชนีราคาหุ้น (Stock index analysis) นอกจากนี้ยังมีการนำทฤษฎีดังกล่าวมาประยุกต์ใช้กับงานทางด้านไฟฟ้ากำลังด้วย เช่น การพยากรณ์ความต้องการใช้ไฟฟ้าล่วงหน้าระยะสั้น (Short-term load forecasting) [6-10] การคำนวณโหลดโฟลว์ [11] และในวิทยานิพนธ์นี้จะนำทฤษฎีเครือข่ายประสาทเทียมมาประยุกต์ใช้ในการพัฒนากระบวนการเรียนรู้ของเครื่องคอมพิวเตอร์ เพื่อสร้างแบบจำลองในรูปของโปรแกรมคอมพิวเตอร์ที่สามารถนำไปใช้ในการพยากรณ์โหลดของหม้อแปลงไฟฟ้าได้อย่างถูกต้องแม่นยำ ดังจะกล่าวในบทต่อไป

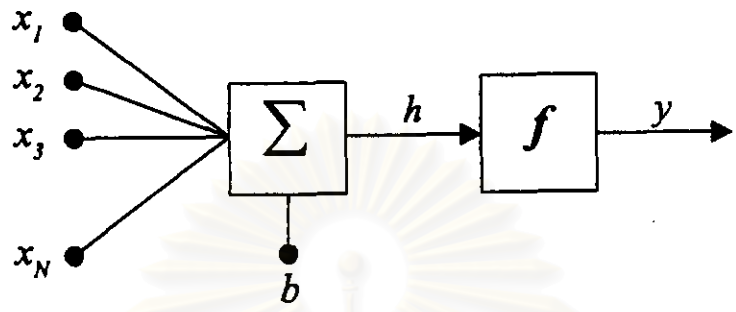
2.1 การจำลองเซลล์ประสาท [5,12]

เซลล์ประสาท (Neuron) ประกอบไปด้วยส่วนประกอบต่างๆได้แก่ เดนไดรต์ (Dendrites) ซึ่งทำหน้าที่ในการรับความรู้สึกต่างๆที่ส่งเข้ามา โดยที่เดนไดรต์แต่ละกิ่งจะรับรู้ได้ด้วยค่าถ่วงน้ำหนัก (Weight) ที่แตกต่างกัน และแทนค่าถ่วงน้ำหนักด้วยความแข็งแรงของแต่ละไซแนปส์ (Synapse) จากนั้นตัวเซลล์ (Cell body) ก็จะรวบรวมสิ่งที่รับรู้ผ่านแอกซอน (Axon) ซึ่งแอกซอนจะส่งสัญญาณออกไป โดยสัญญาณออกจะเป็นฟังก์ชันของผลรวมของสิ่งที่ได้รับรู้จากตัวเซลล์



รูปที่ 2.1 เซลล์ประสาทอย่างง่าย

จากกลไกการทำงานของเซลล์ประสาท เราสามารถสร้างแบบจำลองของเซลล์ประสาทได้ ซึ่งโดยทั่วไปเรียกว่าเซลล์ประสาทเทียม (Artificial neuron) ดังรูปที่ 2.2



รูปที่ 2.2 เซลล์ประสาทเทียมอย่างง่าย

จากรูปที่ 2.2 ตัวแปรด้านเข้า x_i โดยที่ i = 1,2,3,...,N จะถูกส่งผ่านไปรวมกันด้วยค่าถ่วงน้ำหนักที่ไม่เท่ากัน จะได้ผลรวมเป็น

$$h_k = \sum_{i=1}^{N} w_{ki} x_i + b_k$$
$$= w_{k1} x_1 + w_{k2} x_2 + \ldots + w_{kN} x_N + b_k \qquad (2.1)$$

จากนั้นส่งสัญญาณ h_k ผ่านแอกติเวชันฟังก์ชัน (Activation function) เพื่อคำนวณค่าตัวแปรด้านออก y_k

$$y_k = f(\sum_{i=1}^{N} w_{ki} x_i + b_k) \qquad (2.2)$$

โดยที่ h_k คือค่าผลรวมของตัวแปรด้านเข้าของยูนิตที่ k

w_{ki} คือค่าถ่วงน้ำหนักระหว่างยูนิตที่ i และ k

b_k คือค่าไบอัส (Bias) ของยูนิตที่ k

แอกติเวชันฟังก์ชันมีอยู่หลายชนิด เช่น

1) ฟังก์ชันฮาร์ดลิมิต (Hard limit function)

โดยที่ f (h) = hardlim (h)

= 1 เมื่อ h ≥ 0

= 0 เมื่อ h < 0 (2.3)

2) ฟังก์ชันเชิงเส้น (Linear function)

โดยที่ $f(h) =$ purelin (h)

$= h$ เมื่อ $-\infty < h < \infty$ (2.4)

3) ฟังก์ชันไบนารีซิกมอยด์ (Binary sigmoid function หรือ Log-sigmoid function)

โดยที่

$$f(h) = \frac{1}{1 + \exp(-h)} \tag{2.5}$$

4) ฟังก์ชันไบโพลาร์ซิกมอยด์ (Bipolar sigmoid function)

โดยที่

$$f(h) = \frac{1}{1+\exp(-h)} - 1 \tag{2.6}$$

5) ฟังก์ชันไฮเพอร์โบลิกเทนเจนท์ (Hyperbolic tangent function หรือ Tan-sigmoid function)

โดยที่

$$f(h) = \frac{1-\exp(-2h)}{1+\exp(-2h)} \tag{2.7}$$

2.2 การเรียนรู้ของเครือข่ายประสาท (Neural network learning)

การเรียนรู้ของเครือข่ายประสาทจะใช้กฎการเรียนรู้ (Learning rule) เพื่อสอนให้เครือข่ายทำการคำนวณหาผลลัพธ์ในการทำงานบางอย่าง โดยการปรับเปลี่ยนค่าถ่วงน้ำหนักของเครือข่าย กฎการเรียนรู้ของเครือข่ายประสาทมีหลายแบบ แต่ที่ใช้กันทั่วไปมี 2 แบบ [5,12] คือ

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 คือตัวแปรด้านเข้าของเครือข่ายประสาท (Input variable)

t_i คือผลลัพธ์ที่ต้องการ (Target)

P คือจำนวนรูปแบบ (Patterns) ที่ใช้ในการปรับสอนทั้งหมด

เมื่อป้อนตัวแปรด้านเข้าให้แก่เครือข่ายประสาท เครือข่ายจะคำนวณหาผลลัพธ์คือตัวแปรด้านออก (Output variable) แล้วนำผลลัพธ์จากเครือข่ายประสาทนี้ไปเปรียบเทียบกับผลลัพธ์ที่ต้องการ จากนั้นนำค่าผิดพลาดที่เกิดขึ้นไปใช้ปรับค่าถ่วงน้ำหนักของเครือข่ายโดยอาศัยกฎการเรียนรู้ และวิธีการทำซ้ำ (Iterative method) เพื่อให้ตัวแปรด้านออกมีค่าเข้าใกล้ผลลัพธ์ที่ต้องการ

2) การเรียนรู้แบบไม่มีการควบคุม (Unsupervised learning)

การเรียนรู้แบบนี้ค่าถ่วงน้ำหนักจะถูกปรับเพื่อให้สอดคล้องกับตัวแปรด้านเข้าเท่านั้น ไม่ต้องใช้ค่าผลลัพธ์ที่ต้องการ ส่วนใหญ่การเรียนรู้แบบนี้จะใช้ในงานจำแนกประเภทของข้อมูลด้านเข้า (Clustering operation or network classifier) ถ้าข้อมูลด้านเข้าเป็นประเภทเดียวกัน เครือข่ายจะให้ผลลัพธ์อย่างเดียวกัน ตัวอย่างกฎการเรียนรู้แบบนี้คือ Adaptive resonance theory และ Kohonen self-organizing map

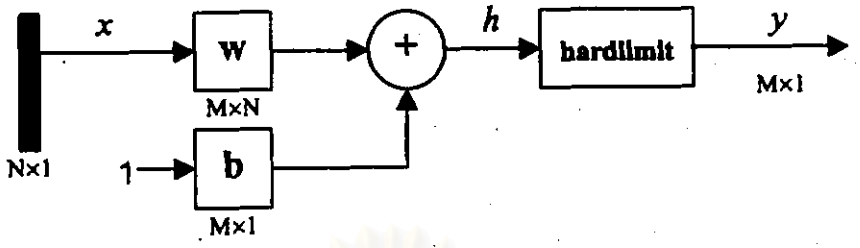
สำหรับในวิทยานิพนธ์นี้ จะใช้วิธีการเรียนรู้แบบมีการควบคุมในการปรับสอนเครือข่ายประสาทเพื่อให้ได้ผลลัพธ์ที่ต้องการ

2.3 เครือข่ายเพอร์เซฟตรอน (Perceptron network) [5,12,13]

เพอร์เซฟตรอนคือรูปแบบของเครือข่ายประสาทอย่างง่ายที่ใช้ฟังก์ชันฮาร์ดลิมิตเป็นแอกติเวชันฟังก์ชันหลัก ซึ่งมีคุณสมบัติสำหรับใช้เพื่อให้เกิดการแยกออกจากกันได้เชิงเส้น (Linearly separable) ของตัวแปรด้านเข้า

2.3.1 โครงสร้างและการทำงานของเครือข่ายเพอร์เซฟตรอน

รูปแบบของเครือข่ายเพอร์เซฟตรอนเป็นดังแสดงในรูปที่ 2.3 ประกอบด้วยเวกเตอร์ของตัวแปรด้านเข้า x_i มีมิติ N×1 เมตริกซ์ของค่าถ่วงน้ำหนัก w มีมิติ M×N เวกเตอร์ของตัวแปรด้านออก y_i มีมิติ M×1 และค่าไบอัส b_i มีมิติ M×1 โดย M คือจำนวนปมประสาท



รูปที่ 2.3 เครือข่ายเพอร์เซพตรอน

ค่าของตัวแปรด้านออกสามารถคำนวณได้จาก

$$y_k = \text{hardlim}\ (\sum_{i=1}^{N} w_{ki} x_i + b_k) \qquad (2.8)$$

โดยที่เมตริกซ์ค่าถ่วงน้ำหนักสามารถเขียนได้เป็น

$$w = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1i} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2i} & \cdots & w_{2N} \\ \cdot & \cdot & & \cdot & & \cdot \\ w_{k1} & w_{k2} & \cdots & w_{ki} & \cdots & w_{kN} \\ \cdot & \cdot & & \cdot & & \cdot \\ w_{M1} & w_{M2} & \cdots & w_{Mi} & \cdots & w_{MN} \end{bmatrix} \qquad (2.9)$$

จากสมการ (2.8) จะเห็นว่าถ้า Inner product ของตัวแปรด้านเข้า กับหลักที่ i^{th} ของเมตริกซ์ค่าถ่วงน้ำหนัก มีค่ามากกว่าหรือเท่ากับ $-b_i$ ตัวแปรด้านออกจะมีค่าเป็น 1 นอกนั้นจะมีค่าเป็น 0 ดังนั้นในแต่ละยูนิตหรือนิวรอน (unit or neuron) ของเครือข่ายเพอร์เซพตรอน จะแบ่งชุดตัวแปรด้านเข้าออกเป็นสองส่วน โดยมีเส้นแบ่งหรือเส้นแสดงขอบเขตการตัดสินใจคือ $(w \cdot x) + b = 0$ ซึ่งเป็นลักษณะสมบัติของเครือข่ายแบบนี้

2.3.2 กฎการเรียนรู้ของเพอร์เซพตรอน (Perceptron learning rule)

กฎการเรียนรู้ของเพอร์เซพตรอนนี้เป็นวิธีการเรียนรู้แบบมีการควบคุมวิธีหนึ่งที่ใช้ในการแก้ปัญหาที่สามารถแยกออกจากกันได้เชิงเส้น (Linearly separable) ซึ่งมีคำตอบและวิธีการคำนวณค่าถ่วงน้ำหนักที่เหมาะสม โดยจะปรับค่าถ่วงน้ำหนักไปทีละยูนิตตามตัวแปรด้านออกที่เกิดขึ้น ถ้าตัว

แปรค้านออกมีเครื่องหมายเหมือนกับผลลัพธ์ที่ต้องการแล้วจะไม่ปรับค่าถ่วงน้ำหนัก ถ้าตัวแปรค้านออกมีเครื่องหมายตรงกันข้ามจะต้องทำการปรับค่าถ่วงน้ำหนัก โดยเป็นสัดส่วนเดียวกับผลคูณของตัวแปรค้านเข้ากับผลลัพธ์ที่ต้องการ ดังสมการ

$$w_{ki}^{new} = w_{ki}^{old} + \Delta w_{ki} \tag{2.10}$$

$$\Delta w_{ki} = (t_k - y_k)x_k \tag{2.11}$$

$$b_k^{new} = b_k^{old} + (t_k - y_k) \tag{2.12}$$

โดยที่ t_k คือค่าผลลัพธ์ที่ต้องการจากยูนิตที่ k

y_k คือค่าตัวแปรค้านออกของยูนิตที่ k

2.4 การเรียนรู้แบบเกรเดียนต์เดสเซนต์ (Gradient descent learning) [5]

ในหัวข้อที่แล้วได้พิจารณาการเรียนรู้ของเครือข่ายเพอร์เซพตรอนที่ใช้ฟังก์ชันแอกติเวชันแบบฟังก์ชันฮาร์ดลิมิต ในหัวข้อนี้จะพิจารณาถึงการเรียนรู้แบบเกรเดียนต์เดสเซนต์ ซึ่งเป็นการเรียนรู้พื้นฐานของเครือข่ายที่ประกอบด้วยยูนิตที่ใช้ฟังก์ชันเชิงเส้น หรือฟังก์ชันที่ไม่เป็นเชิงเส้นเช่นฟังก์ชันซิกมอยด์ต่างๆ ที่มีค่าต่อเนื่อง (Continuous functions) เป็นแอกติเวชันฟังก์ชัน

หากเราเริ่มจากการนิยามค่าผิดพลาดของตัวแปรค้านออกด้วยฟังก์ชันที่แสดงดังสมการ (2.13)

$$\begin{aligned} E &= \frac{1}{2}\sum_k (t_k - y_k)^2 \\ &= \frac{1}{2}\sum_k [t_k - f(\sum_i w_{ki} x_i + b_k)]^2 \end{aligned} \tag{2.13}$$

ฟังก์ชันค่าผิดพลาด (E) เป็นสมการกำลังสองในรูปของค่าถ่วงน้ำหนักกับค่าผิดพลาด มีพื้นผิวเป็นพาราโบลิกและประกอบด้วยค่าต่ำสุดค่าหนึ่ง กฎเกรเดียนต์เดสเซนต์จะปรับเวกเตอร์ถ่วงน้ำหนักในทิศทางของเวกเตอร์ตัวแปรค้านเข้า โดย E จะมีค่าเป็นบวกเสมอ ถ้า w_{ki} และ b_k ถูกปรับเข้าสู่ค่าที่ต้องการ ค่าของฟังก์ชันจะน้อยลงในลักษณะที่ลู่เข้าสู่ศูนย์เมื่อผ่านกระบวนการทำซ้ำไปเรื่อยๆ

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

$$\begin{aligned}\Delta w_{ki} &= -\alpha \frac{\partial E}{\partial w_{ki}} \\ &= \alpha \sum_{P} (t_k - y_k) x_i\end{aligned} \tag{2.14}$$

$$\begin{aligned}\Delta b_k &= -\alpha \frac{\partial E}{\partial b_k} \\ &= \alpha \sum_{P} (t_k - y_k)\end{aligned} \tag{2.15}$$

โดยที่ P คือจำนวนรูปแบบ (Patterns) ทั้งหมดที่ใช้ในการปรับสอน

α คือค่าอัตราการเรียนรู้

ถ้าสามารถปรับเปลี่ยนค่าถ่วงน้ำหนักอย่างอิสระได้ทีละรูปแบบของตัวแปรด้านเข้า จะได้

$$\Delta w_{ki} = \alpha(t_k - y_k) x_i \tag{2.16}$$

$$\Delta b_k = \alpha(t_k - y_k) \tag{2.17}$$

กำหนดให้ $$\delta_k = t_k - y_k \tag{2.18}$$

จะได้ $$\Delta w_{ki} = \alpha \delta_k x_i \tag{2.19}$$

และ $$\Delta b_k = \alpha \delta_k \tag{2.20}$$

สมการ (2.16) ถึงสมการ (2.20) นั้นมีชื่อที่เรียกกันโดยทั่วไปหลายชื่อเช่น Delta rule, Widrow-Hoff rule, Least mean square rule จะสังเกตได้ว่าสมการทั้งหมดเป็นผลมาจากวิธีเกรเดียนต์เดสเซนต์ ซึ่งง่ายต่อการนำไปใช้กับเครือข่ายที่มีหลายชั้น แต่ต้องใช้กับฟังก์ชันแอกติเวชันที่ต่อเนื่องหรือสามารถหาอนุพันธ์ได้ และค่า α จะต้องเป็นค่าเหมาะสมที่จะทำให้การเปลี่ยนแปลงของค่าถ่วงน้ำหนักเป็นไปในทิศทางที่ทำให้ค่าผิดพลาดลดลง หรือทำให้กระบวนการทำซ้ำลู่เข้า [12,13] โดยถ้า

α มีค่าน้อยการคำนวณจะใช้เวลานาน ในทางตรงกันข้ามถ้า α มีค่ามากการคำนวณจะเร็วขึ้น แต่อาจทำให้เกิดการแกว่งได้ ดังนั้นจึงต้องมีกระบวนการที่ใช้ในการเลือกค่า α ที่เหมาะสม ดังจะกล่าวต่อไปในหัวข้อ 2.6.3 และ 2.6.4

2.5 เครือข่ายประสาทแบบหลายชั้น

เครือข่ายประสาทแบบหลายชั้นเป็นเครือข่ายประสาทที่สำคัญแบบหนึ่ง ประกอบด้วยส่วนรับข้อมูล ซึ่งโดยทั่วไปเรียกว่าชั้นตัวแปรด้านเข้า (Input layer) ส่วนการคำนวณ เรียกว่าชั้นซ่อน (Hidden layer) ซึ่งอาจมีมากกว่าหนึ่งชั้นก็ได้ และมีส่วนส่งข้อมูลออกจากส่วนการคำนวณ เรียกว่าชั้นตัวแปรด้านออก (Output layer)

เครือข่ายประสาทแบบหลายชั้นได้นำมาใช้ในการแก้ปัญหาที่ซับซ้อน และปัญหาการไม่ลู่เข้าของเครือข่ายประสาทแบบชั้นเดียวได้ [12] โดยกระบวนวิธีการปรับสอนเครือข่ายที่นิยมใช้กันอย่างกว้างขวางสำหรับงานทางด้านไฟฟ้ากำลัง โดยเฉพาะอย่างยิ่งการพยากรณ์โหลดไฟฟ้าล่วงหน้าระยะสั้น (Short-term load Forecasting) [6-10] และการประมาณค่าฟังก์ชัน (Function approximation) คือ กระบวนวิธีแพร่กระจายความผิดพลาดกลับ (Error back-propagation algorithm) ซึ่งประกอบด้วยการส่งผ่านข้อมูลระหว่างชั้นของเครือข่ายประสาทสองแบบคือ การส่งไปข้างหน้า (Forward pass) และการส่งย้อนกลับ (Backward pass) [12-14] การส่งผ่านไปข้างหน้า ข้อมูลจะถูกป้อนเข้าสู่เครือข่ายประสาททางยูนิตตัวแปรด้านเข้าผ่านไปยังชั้นซ่อนเพื่อทำการคำนวณ และได้ผลลัพธ์ส่งออกมายังชั้นตัวแปรด้านออก โดยค่าถ่วงน้ำหนักที่เชื่อมต่อระหว่างชั้นของตัวแปรในเครือข่ายประสาททั้งหมดยังไม่เปลี่ยนแปลง ส่วนการส่งย้อนกลับจะตรงกันข้ามคือค่าถ่วงน้ำหนักระหว่างชั้นของเครือข่ายประสาทจะถูกปรับเปลี่ยนค่าไปตามกฎการปรับค่าความผิดพลาด (Error correction rule) โดยค่าความแตกต่างระหว่างค่าผลลัพธ์ที่ได้จากเครือข่ายกับค่าผลลัพธ์ที่ต้องการ หรือเรียกว่าค่าผิดพลาด (Error) จะถูกส่งกลับไปยังเครือข่ายตามทิศทางของการเชื่อมต่อ เพื่อนำไปใช้ปรับค่าถ่วงน้ำหนักและค่าไบอัส เพื่อให้ได้ค่าผลลัพธ์ที่ได้จากเครือข่ายเข้าใกล้ผลลัพธ์ที่ต้องการ

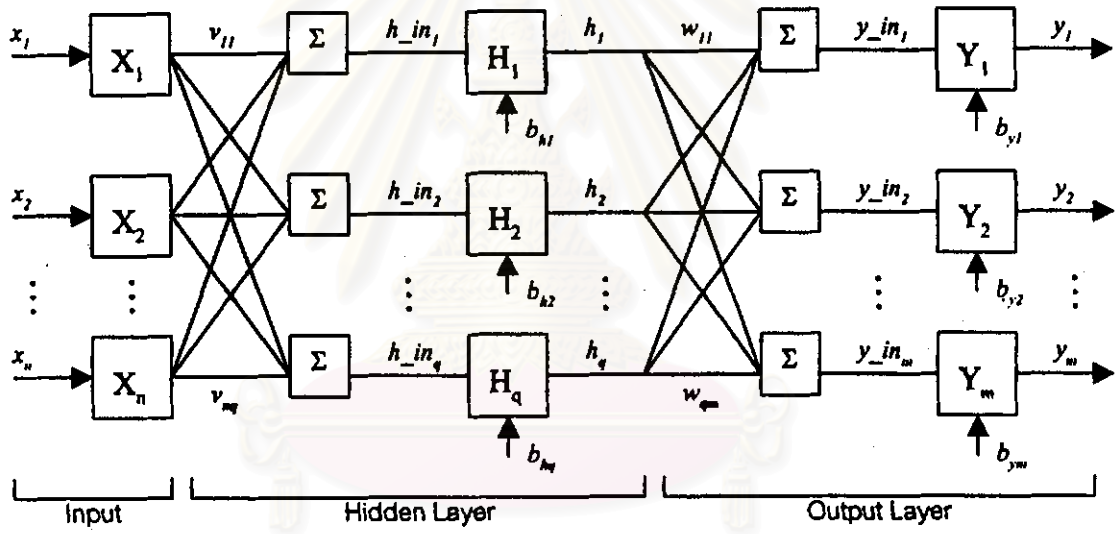
ในวิทยานิพนธ์ฉบับนี้ใช้กระบวนวิธีแพร่กระจายความผิดพลาดกลับนี้ในการปรับสอนเครือข่ายประสาทแบบหลายชั้นให้มีความสามารถในการพยากรณ์โหลดหม้อแปลงได้อย่างถูกต้อง

2.6 กฎการเรียนรู้ที่ใช้กับเครือข่ายประสาทแบบหลายชั้น

2.6.1 กฎการเรียนรู้แบบการแพร่กระจายความผิดพลาดกลับ

(Error back-propagation learning rule) [12-14]

ในที่นี้จะพิจารณาเครือข่ายแบบป้อนไปข้างหน้าที่มีสองชั้น (2-layers feed-forward network) ซึ่งเป็นเครือข่ายประสาทที่นิยมใช้กันโดยทั่วไป ประกอบไปด้วยชั้นของตัวแปรด้านออก 1 ชั้น ชั้นซ่อน 1 ชั้น และยูนิตของตัวแปรด้านเข้าซึ่งไม่นับเป็นชั้น ดังแสดงในรูปที่ 2.4



รูปที่ 2.4 เครือข่ายประสาทแบบป้อนไปสู่ข้างหน้าที่มี 2 ชั้น

โดยกำหนดให้

X_i คือยูนิตของตัวแปรด้านเข้าที่ i โดยที่ i = 1,2,...,n

H_j คือยูนิตซ่อนที่ j โดยที่ j = 1,2,...,q

Y_k คือยูนิตของตัวแปรด้านออกที่ k โดยที่ k = 1,2,...,m

h_in_j คือผลรวมของข้อมูลด้านเข้าของยูนิตซ่อนที่ j

h_j คือข้อมูลด้านออกของยูนิตซ่อนที่ j

y_in_k คือผลรวมของข้อมูลด้านเข้าของยูนิตตัวแปรด้านออกที่ k

y_k คือข้อมูลด้านออกของยูนิตตัวแปรด้านออกที่ k

v_{ij} คือค่าถ่วงน้ำหนักระหว่างยูนิตตัวแปรด้านเข้ากับยูนิตซ่อน

w_{jk} คือค่าถ่วงน้ำหนักระหว่างยูนิตซ่อนกับกับยูนิตตัวแปรด้านออก

b_{hj} คือค่าไบอัสที่ป้อนให้แก่ยูนิตซ่อนที่ j

b_{yk} คือค่าไบอัสที่ป้อนให้แก่ยูนิตของตัวแปรด้านออกที่ k

จะได้ ผลรวมของข้อมูลด้านเข้าของยูนิตซ่อน H_j คือ

$$h\_in_j = \sum_i x_i v_{ij} + b_{hj} \tag{2.21}$$

ข้อมูลด้านออกจากการผ่านแอกติเวชันฟังก์ชันของยูนิตซ่อน H_j คือ

$$h_j = f_h(h\_in_j) \tag{2.22}$$

ผลรวมของข้อมูลด้านเข้าของยูนิตของตัวแปรด้านออก Y_k คือ

$$y\_in_k = \sum_j h_j w_{jk} + b_{yk} \tag{2.23}$$

ข้อมูลด้านออกจากการผ่านแอกติเวชันฟังก์ชันของยูนิตของตัวแปรด้านออก Y_k คือ

$$y_k = f_y(y\_in_k) \tag{2.24}$$

ดังนั้นสมการ (2.13) จะเขียนใหม่ได้เป็น

$$E = \frac{1}{2}\sum_k \left[t_k - f_y\left(\sum_j w_{jk} f_h\left(\sum_i v_{ij} x_i + b_{hj} \right) + b_{yk} \right) \right]^2 \tag{2.25}$$

ด้วยการใช้กฎลูกโซ่ในการหาเกรเดียนต์จะได้

$$\frac{\partial E}{\partial w_{jk}} = -(t_k - y_k) f_y'(y\_in_k) h_j \tag{2.26}$$

$$\frac{\partial E}{\partial b_{yk}} = -(t_k - y_k) f_y'(y\_in_k) \tag{2.27}$$

$$\frac{\partial E}{\partial v_{ij}} = -\sum_k \left[(t_k - y_k) f_y'(y\_in_k) w_{jk}\right] f_h'(h\_in_j) x \tag{2.28}$$

$$\frac{\partial E}{\partial b_{hj}} = -\sum_k \left[(t_k - y_k) f_y'(y\_in_k) w_{jk}\right] f_h'(h\_in_j) \tag{2.29}$$

กำหนดให้

$$\delta_k = (t_k - y_k) f_y'(y\_in_k) \tag{2.30}$$

$$\delta_j = f_h'(h\_in_j) \sum_k (\delta_k w_{jk}) \tag{2.31}$$

ถ้า E เป็นฟังก์ชันที่สามารถหาอนุพันธ์ได้อย่างต่อเนื่องทุกค่าของค่าถ่วงน้ำหนักและค่าไบอัส จะสามารถคำนวณหาค่าถ่วงน้ำหนักและค่าไบอัสที่เหมาะสมโดยใช้กฎเกรเดียนต์เดสเซนต์ตามสมการ (2.14) และสมการ (2.15) ได้ดังนี้

สำหรับการปรับค่าค่าถ่วงน้ำหนักและค่าไบอัสของยูนิตตัวแปรด้านออก จะได้

$$\Delta w_{jk} = \alpha \delta_k h_j \tag{2.32}$$

$$\Delta b_{jk} = \alpha \delta_k \tag{2.33}$$

และสำหรับการปรับค่าค่าถ่วงน้ำหนักและค่าไบอัสของยูนิตซ่อน จะได้

$$\Delta v_{ij} = \alpha \delta_j x_i \tag{2.34}$$

$$\Delta b_{hj} = \alpha \delta_j \tag{2.35}$$

ค่าถ่วงน้ำหนักและค่าไบอัสของแต่ละยูนิตจะถูกปรับด้วยค่า Δw และ Δb ตามลำดับ จนกระทั่งฟังก์ชันค่าผิดพลาดมีค่าน้อยลงถึงจุดที่ยอมรับได้ การที่กระบวนการนี้นำค่าผิดพลาด $(t_k - y_k)$ ในชั้นของตัวแปรด้านออกกลับมาคำนวณหาค่าที่ใช้ปรับค่าถ่วงน้ำหนักและค่าไบอัสของแต่ละยูนิต เริ่มจากยูนิตในชั้นของตัวแปรด้านออก จนถึงยูนิตในชั้นซ่อนชั้นแรกตามลำดับ จึงเรียกกระบวนการนี้ว่า กระบวนการแพร่กระจายความผิดพลาดกลับ (Error back-propagation algorithm)

การนำกฎการเรียนรู้แบบแพร่กระจายความผิดพลาดกลับไปใช้ในการปรับสอนเครือข่ายประสาทสามารถสรุปเป็นขั้นตอนได้ดังนี้

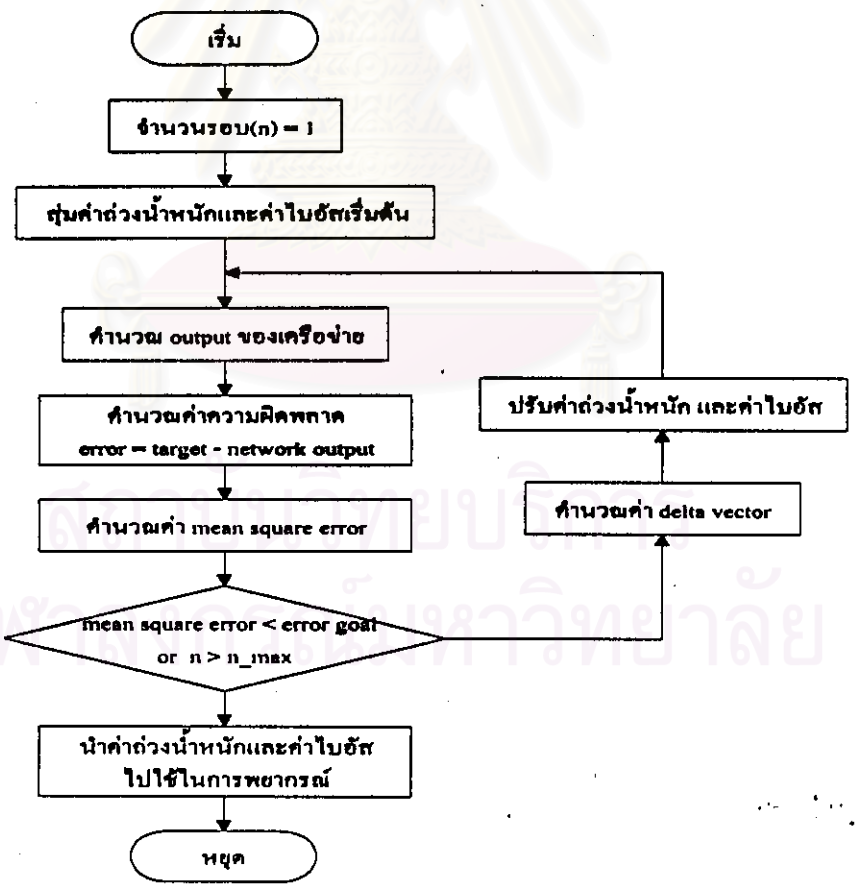
ขั้นที่ 1 คำนวณหาเวกเตอร์ตัวแปรด้านออกจากเวกเตอร์ตัวแปรด้านเข้าที่กำหนด ทั้งนี้หากเป็นการคำนวณรอบแรกจะต้องมีการสุ่มค่าน้อยๆให้แก่ค่าถ่วงน้ำหนักและค่าไบอัส (Bias value) ก่อน

ขั้นที่ 2 ตรวจสอบเงื่อนไขการหยุดทำงาน โดยคำนวณค่าเฉลี่ยของค่าผิดพลาดกำลังสอง (Mean square error) หรือค่าผลรวมของค่าผิดพลาดกำลังสอง (Sum square error) ที่เกิดขึ้นจากเวกเตอร์ที่ใช้ในการปรับสอนทั้งหมด ถ้ามีค่าน้อยกว่าค่าผิดพลาดที่ยอมรับได้ หรือรอบการปรับสอนเกินจำนวนรอบที่กำหนดไว้ ก็ให้หยุดการปรับสอน ไม่เช่นนั้นให้ทำการคำนวณต่อไปในขั้นที่ 3

ขั้นที่ 3 คำนวณค่า δ (Delta vector) ตามกฎการเรียนรู้แบบแพร่กระจายความผิดพลาดกลับ โดยเริ่มจากชั้นตัวแปรด้านออกก่อน แล้วใช้กฎการเรียนรู้แบบแพร่กระจายความผิดพลาดกลับคำนวณค่า δ สำหรับชั้นถัดเข้าไป

ขั้นที่ 4 คำนวณค่าถ่วงน้ำหนักและค่าไบอัสทั้งหมดใหม่ แล้วกลับไปทำขั้นที่ 1 ใหม่

ขั้นตอนทั้งหมดสามารถแสดงในรูปแผนภูมิการทำงานได้ดังนี้



รูปที่ 2.5 แผนภูมิแสดงการทำงานของเครือข่ายประสาทที่ใช้กระบวนการเรียนรู้แบบแพร่กระจายความผิดพลาดกลับ

2.6.2 กฎการเรียนรู้แบบการแพร่กระจายความผิดพลาดกลับร่วมกับโมเมนตัม (Error back-propagation learning rule with momentum) [12,13]

จากการใช้กฎเกรเดียนต์เดสเซนต์ในการปรับค่าถ่วงน้ำหนักและค่าไบอัสในระหว่างการปรับสอนนั้น พบว่าในบางกรณีเครือข่ายจะไม่ลู่เข้าสู่จุดผลตอบที่มีค่าความผิดพลาดต่ำสุด หรือใช้เวลาในการลู่เข้านาน เนื่องมาจากผลของ Local minimum บน Error surface จึงได้มีการนำเอาหลักการของโมเมนตัมเข้ามาช่วย โดย Plaut และคณะเป็นผู้คิดค้นขึ้นในปี ค.ศ.1986 [5] การปรับสอนเครือข่ายร่วมกับโมเมนตัมนี้จะผลอย่างมากในการปรับค่าถ่วงน้ำหนักและค่าไบอัส เพราะว่าเป็นการปรับสอนที่สอนให้เครือข่ายรับรู้ถึงแนวโน้มการเปลี่ยนแปลงค่าถ่วงน้ำหนักและค่าไบอัสที่เป็นมาในอดีต ร่วมกับการปรับสอนที่ใช้กฎเกรเดียนต์เดรสเซนต์ตามปกติ เครือข่ายที่ปรับสอนโดยใช้หลักการของโมเมนตัมนี้จะปฏิบัติตัวเสมือนว่ามี Low pass filter ช่วยกรองไม่ให้เครือข่ายรับรู้ค่าผิดพลาดเล็กๆ บน Error surface จึงเป็นการช่วยแก้ปัญหาในเรื่องของ Local minimum ซึ่งมีส่วนช่วยให้เครือข่ายใช้เวลาในการคำนวณน้อยลงอีกด้วย

กฎการเรียนรู้แบบการแพร่กระจายความผิดพลาดกลับร่วมกับโมเมนตัมมีหลักการปรับค่าถ่วงน้ำหนักและค่าไบอัสดังนี้

จากกฎของเกรเดียนต์เดรสเซนต์ที่ใช้ในการปรับค่าถ่วงน้ำหนักและค่าไบอัสของยูนิตในชั้นตัวแปรด้านออก และของยูนิตซ่อนตามสมการที่ (2.32) ถึงสมการที่ (2.35) กฎการเรียนรู้แบบการแพร่กระจายความผิดพลาดกลับร่วมกับโมเมนตัมจะปรับค่าถ่วงน้ำหนักและค่าไบอัสในรอบการคำนวณปัจจุบัน โดยจะมีการรวมกับผลการปรับค่าถ่วงน้ำหนักและค่าไบอัสในรอบการคำนวณก่อนหน้านี้หนึ่งรอบเข้าไว้ด้วยกัน โดยมีค่าคงที่โมเมนตัม (Momentum constant, *mc*) เป็นตัวกำหนดอัตราส่วนดังสมการ

สำหรับยูนิตของตัวแปรด้านออก

$$\Delta w_{jk}^{new} = mc \cdot \Delta w_{jk}^{old} + (1 - mc)\alpha\delta_k h_j \qquad (2.36)$$

$$\Delta b_{jk}^{new} = mc \cdot \Delta b_{jk}^{old} + (1 - mc)\alpha\delta_k \qquad (2.37)$$

และสำหรับยูนิตซ่อน

$$\Delta v_{ij}^{new} = mc \cdot \Delta v_{ij}^{old} + (1 - mc)\alpha\delta_j x_i \quad (2.38)$$

$$\Delta b_{ij}^{new} = mc \cdot \Delta b_{ij}^{old} + (1 - mc)\alpha\delta_j \quad (2.39)$$

โดยที่ mc คือค่าคงที่โมเมนตัมซึ่งมีค่าอยู่ระหว่าง 0 ถึง 1

การใช้หลักการโมเมนตัมต้องเลือกค่า mc ให้เหมาะสมจึงจะทำให้ค่าความผิดพลาดลดลงได้เร็วกว่าวิธีแบบดั้งเดิม (Standard back-propagation) หากเลือกค่าไม่เหมาะสมจะเกิดการแกว่งไม่ลู่เข้าของค่าถ่วงน้ำหนัก ในกรณีทั่วไปค่า mc จะมีค่าเท่ากับ 0.9 [13]

2.6.3 กฎการเรียนรู้แบบการแพร่กระจายความผิดพลาดกลับร่วมกับกระบวนการปรับเปลี่ยนอัตราการเรียนรู้ (Error back-propagation learning rule with adaptive learning rate algorithm) [12,13]

ถึงแม้ว่าการใช้โมเมนตัมเข้ามาช่วยในการปรับค่าถ่วงน้ำหนักทำให้เครือข่ายใช้เวลาในการคำนวณน้อยลงแล้วก็ตาม แต่การปรับสอนยังคงใช้เวลานานอยู่ จึงมีการวิจัยเพื่อหากระบวนวิธีการปรับสอนใหม่ๆ ซึ่งจากการปรับสอนเครือข่ายแบบดั้งเดิมที่ใช้อัตราการเรียนรู้ (α) คงที่นั้นมีข้อเสียหลายประการ ประการแรกคือ การที่อัตราการเรียนรู้มีค่าคงที่ค่าเดียว ทำให้ความเร็วในการเรียนรู้ของเครือข่ายมีค่าคงที่ด้วย ไม่ขึ้นอยู่กับอัตราการลดลงของค่าผิดพลาด ช่วงที่ค่าผิดพลาดลดลงเร็วการคำนวณเสียจะเวลามาก ช่วงที่ค่าผิดพลาดลดลงช้าการคำนวณจะไม่ละเอียด อีกประการหนึ่งคือ ต้องใช้วิธีการสุ่ม (Trial and error) เพื่อให้ได้ค่าอัตราการเรียนรู้ที่เหมาะสมที่จะทำให้การเปลี่ยนแปลงของค่าถ่วงน้ำหนักเป็นไปในทิศทางที่ทำให้ค่าผิดพลาดลดลง หรือทำให้กระบวนการทำซ้ำลู่เข้า ถ้า α มีค่าน้อยเกินไปการคำนวณจะใช้เวลานาน ในทางตรงกันข้ามถ้า α มีค่ามากการคำนวณจะเร็วขึ้นแต่อาจทำให้เกิดการแกว่งได้ ซึ่งจะเห็นว่าการพิจารณาเลือกใช้ค่า α จะมีผลต่อการคำนวณอย่างมาก

ด้วยเหตุผลดังกล่าวจึงมีผู้พัฒนากระบวนการปรับสอนเครือข่ายแบบปรับเปลี่ยนอัตราการเรียนรู้ได้ขึ้นมาใช้แทนกระบวนการปรับสอนแบบดั้งเดิม โดย Jacobs เป็นผู้นำเสนอแนวคิดนี้ในปี ค.ศ.1988 โดยมีวัตถุประสงค์เพื่อให้เครือข่ายสามารถปรับค่าอัตราการเรียนรู้ให้เหมาะสมกับการคำนวณในแต่ละรอบได้เอง โดยช่วงที่ค่าผิดพลาดมีแนวโน้มที่จะลดลงเร็ว หรือช่วงที่ Error surface มีความชันมาก ก็ให้เพิ่มค่าอัตราการเรียนรู้ และช่วงที่ค่าผิดพลาดมีแนวโน้มที่จะลดลงช้า หรือช่วงที่

Error surface มีความชันน้อย ก็ให้ลดค่าอัตราการเรียนรู้ โดยมีกระบวนการปรับสำหรับรอบการคำนวณที่พิจารณาใดๆ ดังนี้

<u>กรณีที่ 1</u> อัตราส่วนระหว่างที่ค่าผิดพลาดใหม่ที่เกิดจากการคำนวณในรอบที่พิจารณากับค่าผิดพลาดที่เกิดจากการคำนวณในรอบก่อนหน้านี้ มีค่ามากกว่าค่าอัตราส่วนค่าผิดพลาด (*Err_ratio*) ซึ่งปกติจะมีค่าเท่ากับ 1.04 [13]

$$Error^{new} > (err\_ratio) \cdot Error^{old}$$

สำหรับยูนิตของตัวแปรด้านออก ให้ใช้ค่าถ่วงน้ำหนักและค่าไบอัสที่ได้จากการคำนวณในรอบที่แล้ว

$$w_{jk}^{new} = w_{jk}^{old} \tag{2.40}$$

$$b_{yk}^{new} = b_{yk}^{old} \tag{2.41}$$

และสำหรับยูนิตซ่อน ให้ใช้ค่าถ่วงน้ำหนักและค่าไบอัสที่ได้จากการคำนวณในรอบที่แล้วเช่นกัน คือ

$$v_{ij}^{new} = v_{ij}^{old} \tag{2.42}$$

$$b_{hj}^{new} = b_{hj}^{old} \tag{2.43}$$

แต่เปลี่ยนค่าอัตราการเรียนรู้ใหม่ให้ลดลงเป็น

$$\alpha^{new} = lr\_dec \cdot \alpha^{old} \tag{2.44}$$

โดยที่ *lr_dec* คืออัตราการลดลงของค่าอัตราการเรียนรู้ โดยทั่วไปมีค่าเท่ากับ 0.7 [13]

<u>กรณีที่ 2</u> อัตราส่วนระหว่างที่ค่าผิดพลาดใหม่ที่เกิดจากการคำนวณในรอบที่พิจารณากับค่าผิดพลาดที่เกิดจากการคำนวณในรอบก่อนหน้านี้ มีค่าน้อยกว่าค่าอัตราส่วนค่าผิดพลาด (*Err_ratio*) ซึ่งปกติจะมีค่าเท่ากับ 1.04 [13]

$$Error^{new} < (err\_ratio) \cdot Error^{old}$$

สำหรับยูนิตของตัวแปรด้านออกให้ใช้ค่าถ่วงน้ำหนักและค่าไบอัสตามที่คำนวณได้จากกฎการเรียนรู้เดิมคือ

$$w_{jk}^{new} = w_{jk}^{old} + \Delta w_{jk} \tag{2.45}$$

$$b_{yk}^{new} = b_{yk}^{old} + \Delta b_{yk} \tag{2.46}$$

และสำหรับยูนิตซ่อนให้ใช้ค่าถ่วงน้ำหนักและค่าไบอัสตามที่คำนวณได้จากกฎการเรียนรู้เดิมเช่นกัน คือ

$$v_{ij}^{new} = v_{ij}^{old} + \Delta v_{ij} \tag{2.47}$$

$$b_{hj}^{new} = b_{hj}^{old} + \Delta b_{hj} \tag{2.48}$$

แต่เปลี่ยนค่าอัตราการเรียนรู้ใหม่ให้เพิ่มขึ้นเป็น

$$\alpha^{new} = lr\_inc \cdot \alpha^{old} \tag{2.49}$$

โดยที่ lr_dec คืออัตราการเพิ่มของค่าอัตราการเรียนรู้ โดยทั่วไปมีค่าเท่ากับ 1.05 [13]

กระบวนการเรียนรู้แบบนี้ อัตราการเรียนรู้จะถูกปรับเปลี่ยนไปแต่ละยูนิตที่เชื่อมต่อกันในทันที โดยจะเพิ่มอัตราการเรียนรู้ขึ้นไปเรื่อยๆ ตราบเท่าที่ค่าผิดพลาดยังไม่เพิ่มขึ้น และเมื่ออัตราการเรียนรู้มีค่ามากจนทำให้เกิดการแกว่งขึ้นในกระบวนการปรับสอน เครือข่ายก็จะปรับอัตราการเรียนรู้ให้น้อยลงเพื่อให้กระบวนการเรียนรู้กลับสู่สภาวะปกติอีกครั้ง ด้วยเหตุนี้จึงทำให้ค่าความผิดพลาดลดลงเร็วกว่า และใช้เวลาในการคำนวณน้อยกว่ากฎการเรียนรู้แบบดั้งเดิม กระบวนการเรียนรู้แบบปรับเปลี่ยนอัตราการเรียนรู้นี้จึงเป็นวิธีได้รับความนิยมมาก ทั้งนี้ Howard Demuth และ Mark Beale ได้แนะนำค่าอัตราการเรียนรู้เริ่มต้นสำหรับใช้ในกระบวนการปรับสอนรอบแรกสำหรับกรณีทั่วไปไว้โดยให้มีค่าเท่ากับ 0.01 [12,13]

2.6.4 วิธีการปรับสอนแบบ Modified back-propagation (MBP) [12-14]

จากข้อได้เปรียบของวิธีการปรับสอนที่ใช้หลักการของโมเมนตัม และกระบวนการปรับสอนแบบปรับเปลี่ยนอัตราการเรียนรู้ที่ได้กล่าวไว้ในหัวข้อก่อนหน้านี้ ทำให้มีการคิดค้นพัฒนาวิธีการปรับสอนแบบใหม่ที่รวมเอาประโยชน์ของทั้งสองวิธีมารวมกัน เรียกว่ากระบวนการปรับสอนแบบ Modified back-propagation (MBP) โดยมีหลักการปรับสอนเหมือนกับที่กล่าวไว้แล้วในหัวข้อ 2.6.3 ทุกประการ เพียงแต่เปลี่ยนการปรับค่าถ่วงน้ำหนักและค่าไบอัสในสมการที่ (2.45) ถึงสมการที่ (2.48) ไปใช้ค่าตามสมการที่ (2.36) ถึงสมการที่ (2.39) ในหัวข้อ 2.6.2 ตามลำดับ

ในวิทยานิพนธ์ฉบับนี้ใช้กระบวนวิธีแพร่กระจายความผิดพลาดกลับ ร่วมกับหลักการของโมเมนตัมและกระบวนการปรับเปลี่ยนอัตราการเรียนรู้ (MBP) สำหรับใช้ในการปรับสอนเครือข่ายประสาทแบบหลายชั้นให้มีความสามารถในการพยากรณ์โหลดหม้อแปลงได้อย่างถูกต้อง ทั้งนี้วิธีการปรับสอนแบบนี้ได้รวมเอาข้อดีของหลักการโมเมนตัม และการปรับสอนแบบปรับเปลี่ยนอัตราการเรียนรู้เข้าไว้ด้วยกัน และสามารถปรับค่าพารามิเตอร์ต่างๆ ที่ใช้ในการปรับสอนให้มีค่าเหมาะสมได้ ซึ่งค่าพารามิเตอร์ดังกล่าวประกอบด้วย

1) ค่า Momentum constant (*mc*)
2) ค่าอัตราการเรียนรู้เริ่มต้น (α_ini)
3) ค่าอัตราส่วนค่าผิดพลาด (*Err_ratio*)
4) ค่าอัตราการลดลงของค่าอัตราการเรียนรู้ (*lr_dec*)
5) ค่าอัตราการเพิ่มขึ้นของค่าอัตราการเรียนรู้ (*lr_dec*)

เราสามารถปรับค่าพารามิเตอร์ต่างๆ ให้มีความเหมาะสม เพื่อให้กระบวนการปรับสอนเครือข่ายมีความยืดหยุ่นสูงขึ้น กล่าวคือในการปรับสอนเครือข่ายเราสามารถควบคุมผลของโมเมนตัม และการเพิ่มขึ้นหรือลดลงของอัตราการเรียนรู้ได้โดยปรับค่าพารามิเตอร์ดังกล่าว ซึ่งหลักการของโมเมนตัมที่ช่วยแก้ปัญหาด้าน Local minimum และการปรับเปลี่ยนอัตราการเรียนรู้ให้มีค่าเหมาะสมนี้ ส่งผลให้กระบวนการปรับสอนลู่เข้าเร็วขึ้น หรือใช้เวลาในการปรับสอนเครือข่ายน้อยลง

2.7 ปัจจัยที่มีผลต่อการเรียนรู้ของเครือข่ายประสาท [5,14]

การปรับสอนเครือข่ายโดยใช้กฎการเรียนรู้แบบต่างๆ เป็นการปรับค่าถ่วงน้ำหนักและค่าไบอัสในทิศทางที่ทำให้ความแตกต่างระหว่างข้อมูลด้านออกของเครือข่ายเข้าใกล้ผลลัพธ์ที่ต้องการมากขึ้นเรื่อยๆ หรือเรียกว่าทิศทางที่ทำให้ระบบลู่เข้า ซึ่งนอกจากกฎการเรียนรู้แล้วยังมีปัจจัยอื่นๆที่มีผลต่อการลู่เข้าอีกคือ

1) จำนวนชั้นซ่อน ขึ้นอยู่กับความซับซ้อนของงานที่จะนำเครือข่ายประสาทไปใช้ ถ้าปัญหาที่พิจารณามีความซับซ้อนมากก็จะต้องใช้จำนวนชั้นซ่อนหลายชั้น สำหรับการสร้างแบบจำลองเพื่อพยากรณ์โหลดหม้อแปลงในวิทยานิพนธ์ฉบับนี้จะใช้จำนวนชั้นซ่อน 1 ชั้น
2) จำนวนยูนิตในชั้นซ่อน ปัจจุบันยังไม่มีวิธีใดที่สามารถระบุจำนวนยูนิตในชั้นซ่อนที่เหมาะสมได้โดยตรง วิธีที่ใช้กันโดยทั่วไปคือวิธี Trial and error จำนวนยูนิตในชั้นซ่อนที่เหมาะสมควรเป็นค่าที่น้อยที่สุดโดยที่ยังคงมีอัตราการลู่เข้าที่ดี ซึ่งขึ้นอยู่กับชนิดของงานที่จะนำเครือข่ายประสาทไปใช้
3) จำนวนชุดข้อมูลที่จะนำไปใช้ในการปรับสอน ถ้ามีข้อมูลยิ่งมาก การเรียนรู้จะทำให้ได้ผลลัพธ์ถูกต้องยิ่งขึ้น แต่ก็จะเสียเวลาในการปรับสอนเครือข่ายเพิ่มขึ้นด้วย และถ้ามีข้อมูลน้อยเกินไปจะทำให้ผลลัพธ์มีความคลาดเคลื่อนสูง ในงานทั่วไปจำนวนชุดหรือรูปแบบข้อมูลที่เพียงพอสำหรับการปรับสอน (Pattern) คือ

$$P = \frac{W}{e} \tag{2.50}$$

โดยที่ W คือจำนวนค่าถ่วงน้ำหนักที่ใช้ในเครือข่ายทั้งหมด
e คือค่าความผิดพลาดที่ยอมรับได้

และชุดข้อมูลที่มีทั้งหมดควรต้องแบ่งออกเป็นสามส่วน สำหรับใช้เป็นข้อมูลในการปรับสอนเครือข่ายเป็นจำนวนตามสมการที่ (2.47) 2 ส่วน และสำหรับใช้ทดสอบเครือข่ายที่ได้รับการปรับสอนแล้วว่ามีความถูกต้องแม่นยำเพียงใดอีก 1 ส่วน [9]
4) ค่าเริ่มต้นของค่าถ่วงน้ำหนักและค่าไบอัส หากใช้ค่าเริ่มต้นที่ดีจะทำให้จำนวนรอบที่ใช้ในการปรับสอนลดลงหรือระบบลู่เข้าเร็วขึ้น และในทางตรงกันข้ามค่าเริ่มต้นบางค่าอาจทำให้ระบบไม่ลู่เข้าสู่จุดต่ำสุดในสเปซ (Global Minimum) เลยก็ได้ ในทางปฏิบัติจะใช้การสุ่มค่าที่เป็นค่าบวกหรือค่าลบน้อยๆ ระหว่างช่วง –1 ถึง 1