บทที่ 12

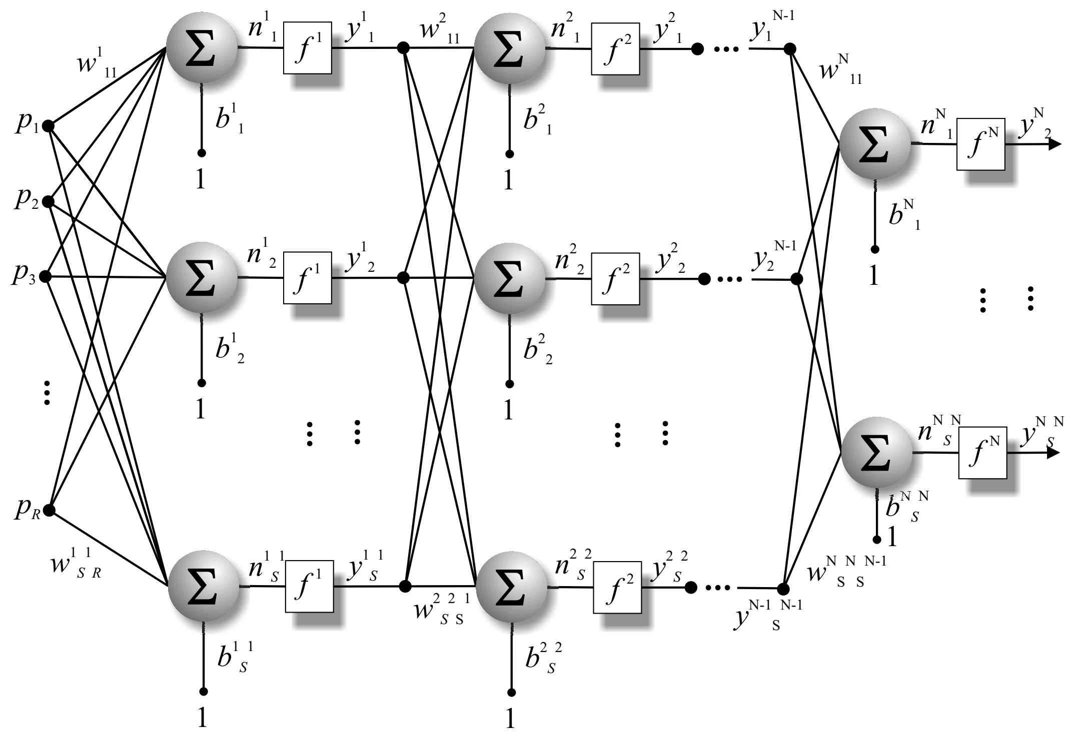
การเรียนรู้แบบแพร่กลับ
Back-Propagation Learning

กฎการเรียนรู้เพอร์เซ็พตรอนของ Frank Rosenblatt และกฎการเรียนรู้แบบวิโดร์ว-ฮอฟฟ์ของ Bernard Widrow และ Marcian Hoff ใช้ในการฝึกสอนเครือข่ายที่มีโครงสร้างเพียงชั้นเดียว (แต่มีหลายนิวรอนได้) เครือข่ายดังกล่าวล้วนแต่มีข้อจำกัดที่สามารถแก้ปัญหาที่แบ่งแยกได้เชิงเส้น (linearly separable) เท่านั้น ทั้ง Rosenblatt และ Widrow ต่างก็ตระหนักถึงข้อจำกัดดังกล่าว และได้นำเสนอเครือข่ายแบบหลายชั้น (multilayer) อย่างไรก็ตาม ทั้งคู่ก็ไม่สามารถพัฒนาอัลกอริทึมในการฝึกสอนเครือข่ายดังกล่าวได้การนำเสนออัลกอริทึมสำหรับฝึกสอนเครือแบบหลายชั้นพบในวิทยานิพนธ์ของ Paul Werbos ในปีคศ. 1974 [Werbos, 1974] แต่ไม่ได้มีการนำเข้าสู่กลุ่มของผู้ใช้และพัฒนาเครือข่ายประสาทเทียม จนกระทั่งปลายทศวรรษที่ 80 ที่มีการนำเอาอัลกอริทึมแพร่กลับ (backpropagation) มาพัฒนาและประยุกต์ใช้กันอย่างแพร่หลายอีกครั้ง ไม่ว่าจะเป็น Parker [Parker, 1982] ในปีค.ศ. 1982 LeCun [LeCun, 1985] ในปีค.ศ.1985 และโดย Rumelhart และคณะ [Rumelhart et al., 1986a][Rumelhart et al., 1986b] ในปีค.ศ. 1986 การนำเสนอของ Rumelhart ถือเป็นจุดที่ทำให้อัลกอริทึมแบบแพร่กลับนี้ได้รับความนิยมเป็นอย่างมากในเวลาต่อมา เครือข่ายไปข้างหน้า (feedforward) แบบหลายชั้นที่มีการฝึกสอน โดยใช้อัลกอริทึมแพร่กลับเป็นเครือข่ายประสาทเทียมที่มีการใช้แพร่หลายที่สุดจนกระทั่งถึง ณ ปัจจุบันนี้

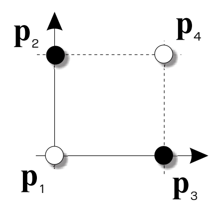
12.1 เครือข่ายหลายชั้น
Multilayer Feedforward Network

รูปที่ 12.1 แสดงเครือข่าย N ชั้น เอาต์พุตของแต่ละชั้นจะเป็นอินพุตให้กับชั้นถัดไป แต่ละชั้นสามารถมีจำนวนนิวรอนแตกต่างกันได้ จากรูปดังกล่าวจำนวนนิวรอนของแต่ละชั้นคือ $R - S^1 - S^2 - \cdots - S^l - \cdots - S^N$ โดยปกติชั้นแรกจะเป็นชั้นอินพุต(input layer) ซึ่งรับอินพุตจากภายนอกเครือข่าย ในชั้นสุดท้ายจะเป็นชั้นเอาต์พุตสำหรับส่งค่าเอาต์พุตออกไปจากเครือข่าย ส่วนชั้นระหว่างอินพุตและเอาต์พุตเรียกว่าเป็น**ชั้นซ่อนเร้น** (hidden layer) เครือข่ายแบบหลายชั้นมีประสิทธิภาพเหนือกว่าแบบชั้นเดียวมาก ทำให้มีการประยุกต์ใช้เครือข่ายแบบหลายชั้นในการแก้ปัญหาหลายๆ อย่างได้อย่างมีประสิทธิภาพ

เครือข่ายหลายชั้นสามารถแก้ปัญหาที่ไม่สามารถแบ่งแยกได้แบบเชิงเส้น พิจารณาตัวอย่างปัญหา XOR ที่มีคู่



รูปที่ 12.1: เครือข่ายไปข้างหน้า N ชั้น



รูปที่ 12.2: รูปแบบเวกเตอร์อินพุตและเอาต์พุตของฟังก์ชัน XOR

อินพุตและเป้าหมายดังนี้ (ดูรูปที่ 12.2 ประกอบ)

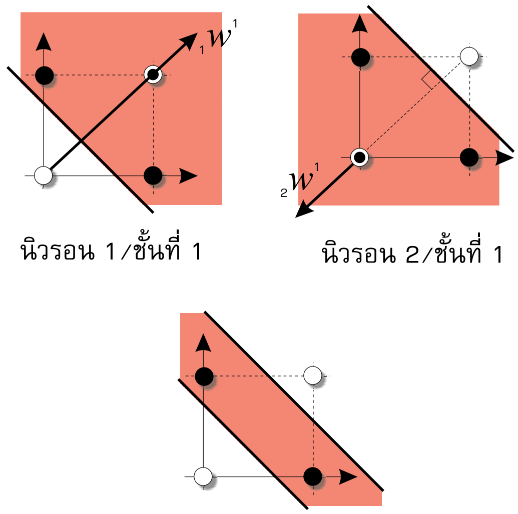
$$\left\{\mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{t}_1 = [0]\right\}$$

$$\left\{\mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{t}_2 = [1]\right\}$$

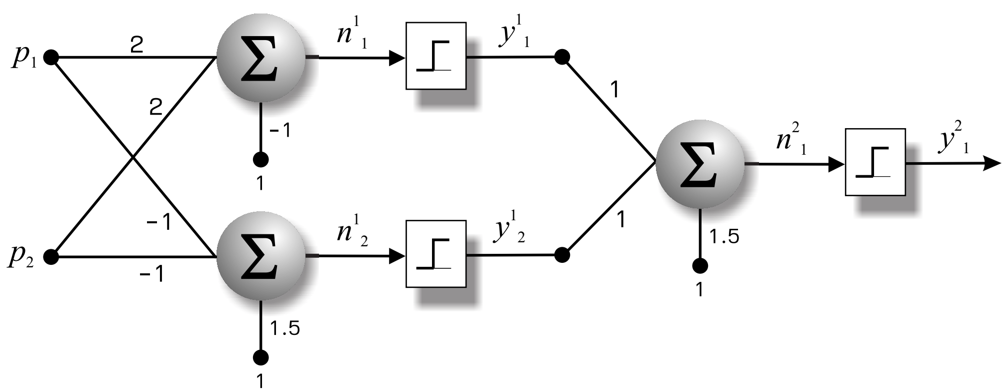
$$\left\{\mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{t}_3 = [1]\right\}$$

$$\left\{\mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{t}_4 = [0]\right\}$$

มีเครือข่ายแบบหลายชั้นหลากหลายโครงสร้างที่สามารถแก้ปัญหา XOR นี้ได้ โดยปกติแล้วเครือข่ายเพียง 2 ชั้นก็สามารถแก้ปัญหาได้ ตัวอย่างหนึ่งก็คือใช้เครือข่าย 2 ชั้นที่ชั้นแรกประกอบไปด้วย 2 นิวรอน เพื่อสร้างเส้นแบ่งพื้นที่2 เส้น เส้นแรกใช้แบ่งอินพุต $\mathbf{p}_1$ ออกจากอินพุตอื่นๆ และเส้นที่สองใช้แบ่งอินพุต $\mathbf{p}_4$ ออก เครือข่ายชั้นที่สองใช้สำหรับรวมเส้นแบ่งพื้นที่จากชั้นแรกเข้าด้วยกันด้วยการกระทำ AND ดังนั้นเครือข่ายชั้นที่สองจึงใช้เพียง



รูปที่ 12.3: เส้นแบ่งพื้นที่ในปัญหา XOR จากเครือข่าย 2 ชั้น



รูปที่ 12.4: ตัวอย่างเครือข่ายไปข้างหน้าขนาด 2 ชั้น (ชั้นละ 2 นิวรอนและ 1 นิวรอน) สำหรับปัญหา XOR

นิวรอนเดียว รูปที่ 12.3 แสดงเส้นแบ่งพื้นที่ที่เกิดจากแต่ละนิวรอนในแต่ละชั้นของเครือข่ายดังกล่าว โครงสร้างของเครือข่ายแสดงในรูปที่ 12.4

12.2 อัลกอริทึมแพร่กลับ Backpropagation Algorithm

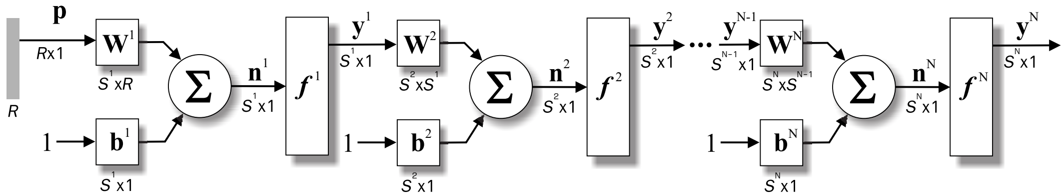
พิจารณาจากเครือข่าย N ชั้น เอาต์พุตของแต่ละชั้นจะเป็นอินพุตให้กับชั้นถัดไป เราสามารถเขียนในรูปความสัมพันธ์ได้ดังนี้

$$\mathbf{y}^{l+1} = \mathbf{f}^{l+1}(\mathbf{W}^{l+1}\mathbf{y}^{l} + \mathbf{b}^{l+1}) \tag{12.1}$$

โดยที่ $\mathbf{y}^l$ $\mathbf{f}^l$ $\mathbf{W}^l$ และ $\mathbf{b}^l$ คือเอาต์พุต ฟังก์ชันถ่ายโอน น้ำหนักประสาทและไบอัสของเครือข่ายชั้นที่ l และ $l = 0, 1, 2, \ldots, N-1$ (ตัวชี้ l ในที่นี้แทนชั้นหรือ layer) ในชั้นแรกนั้นจะเป็นชั้นอินพุตซึ่งรับอินพุตโดยตรงจากภายนอกเครือข่าย กล่าวคือ

$$\mathbf{y}^0 = \mathbf{p} \tag{12.2}$$

ในขณะที่เอาต์พุตในชั้นสุดท้ายคือ $\mathbf{y}^N$ ซึ่งเป็นเอาต์พุตที่ออกมาจากเครือข่ายสู่โลกภายนอก รูปที่ 12.5 แสดงรายละเอียดของเครือข่าย N ชั้น



รูปที่ 12.5: การกำหนดขนาดของพารามิเตอร์ต่างๆ ของเครือข่าย N ชั้น

อัลกอริทึมแพร่กลับใช้หลักการเดียวกันกับอัลกอริทึม LMS ซึ่งเป็นการวิเคราะห์หาค่าความผิดพลาดแบบกำลังสองเฉลี่ย (mean square error) เช่นเดียวกัน ในอัลกอริทึมแพร่กลับ มีการนำเสนอคู่อินพุตและเป้าหมายให้เครือข่ายเรียนรู้ดังนี้

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_q, \mathbf{t}_q\}, \ldots \{\mathbf{p}_Q, \mathbf{t}_Q\} \tag{12.3}$$

เมื่อป้อนแต่ละอินพุตให้กับเครือข่าย เอาต์พุตที่ได้จะถูกนำไปเปรียบเทียบกับเป้าหมาย อัลกอริทึมจะทำการปรับพารามิเตอร์ของเครือข่าย ซึ่งได้แก่น้ำหนักประสาทและไบอัส เพื่อให้ค่าความผิดพลาดแบบกำลังสองเฉลี่ยของเอาต์พุตและเป้าหมายมีค่าน้อยที่สุด จะได้ตัวชี้ประสิทธิภาพ (performance index) คือ (สังเกตว่าเป็นตัวเดียวกันกับอัลกอริทึม LMS ในการเรียนรู้แบบวิโดร์ว-ฮอฟฟ์)

$$F(\mathbf{x}) = E[e^2] = E[(t-y)^2] \tag{12.4}$$

โดยที่ $\mathbf{x} = \begin{bmatrix} \mathbf{W} & \mathbf{b} \end{bmatrix}^T$ เป็นเมตริกซ์รวมของน้ำหนักประสาทและไบอัส (ในกรณีนี้แต่ละชั้นของเครือข่ายไปข้างหน้ามีจำนวนนิวรอนได้มากกว่า 1 นิวรอน ตัวแปร $\mathbf{W}$ จึงแทนเมตริกซ์น้ำหนักประสาท ไม่ใช่แถวของเมตริกซ์น้ำหนักประสาทสำหรับ 1 นิวรอนเหมือนในกรณีของเครือข่าย ADALINE หรือ LA ในบทก่อนหน้านี้) ในกรณีที่เครือข่ายมีเอาต์พุตมากกว่าหนึ่ง จะได้กรณีทั่วไปดังนี้

$$\begin{aligned} F(\mathbf{x}) &= E[\mathbf{e}^T\mathbf{e}] && (12.5)\\ &= E[(\mathbf{t}-\mathbf{y})^T(\mathbf{t}-\mathbf{y})] && (12.6) \end{aligned}$$

เช่นเดียวกันกับอัลกอริทึม LMS เราสามารถประมาณค่าความผิดพลาดกำลังสองเฉลี่ยในรอบการฝึกสอนที่ k ได้ดังนี้

$$\begin{aligned} \hat{F}(\mathbf{x})_k &= (\mathbf{t}(k)-\mathbf{y}(k))^T(\mathbf{t}(k)-\mathbf{y}(k)) && (12.7)\\ &= \mathbf{e}^T(k)\mathbf{e}(k) && (12.8) \end{aligned}$$

จากอัลกอริทึมลงแบบชันสุดจะได้การปรับค่าองค์ประกอบของเมตริกซ์น้ำหนักประสาทและไบอัสของนิวรอนชั้นที่ l ณ รอบการฝึกสอนที่ $k+1$ ด้วยการประมาณค่าความผิดพลาดกำลังสองเฉลี่ยและค่าคงที่การเรียนรู้ α ดังนี้

$$w_{ij}^l(k+1) = w_{ij}^l(k) - \alpha\frac{\partial\hat{F}}{\partial w_{ij}^l} \tag{12.9}$$

และ

$$b_i^l(k+1) = b_i^l(k) - \alpha\frac{\partial\hat{F}}{\partial b_i^l} \tag{12.10}$$

โดยที่ $w_{ij}^l(k+1)$ และ $b_i^l(k+1)$ คือค่าน้ำหนักประสาทและไบอัสของนิวรอนตัวที่ i ในชั้นที่ l ณ รอบการฝึกสอนที่ $k+1$ ตามลำดับ

เมื่อพิจารณาเทียบกับเครือข่ายชั้นเดียวแบบ ADALINE อนุพันธ์ย่อยที่ปรากฏในสมการ (12.9) และ (12.10) สามารถคำนวณหาได้โดยตรง (ในรูปของ $2\mathbf{e}\mathbf{p}^T$ สำหรับเทอม $\frac{\partial\hat{F}}{\partial w_{ij}^l}$ และ $2\mathbf{e}$ สำหรับเทอม $\frac{\partial\hat{F}}{\partial b_i^l}$) อย่างไรก็ดี ในกรณี

ของเครือข่ายหลายชั้นซึ่งมีชั้นซ่อนเร้นนั้นจะไม่สามารถคำนวณค่าความผิดพลาดโดยตรงในรูปของอินพุต $\mathbf{p}$ และเป้าหมาย $\mathbf{t}$ ได้ เนื่องจากอินพุตของชั้นซ่อนเร้นคือเอาต์พุตของชั้นซ่อนเร้นที่อยู่ก่อนหน้า ดังนั้นสำหรับกรณีของชั้นซ่อนเร้น จึงต้องมีการคำนวณเพิ่มเติม พิจารณาการนำเอากฎลูกโซ่ (chain rule) มาช่วยในการคำนวณหาความสัมพันธ์ของอนุพันธ์ระหว่างแต่ละชั้น ดังนั้นจากเทอมอนุพันธ์ย่อยในสมการ (12.9) และ (12.10) จะได้ว่า

$$\frac{\partial \hat{F}}{\partial w_{ij}^l} = \frac{\partial \hat{F}}{\partial n_i^l}\frac{\partial n_i^l}{\partial w_{ij}^l} \tag{12.11}$$

และ

$$\frac{\partial \hat{F}}{\partial b_i^l} = \frac{\partial \hat{F}}{\partial n_i^l}\frac{\partial n_i^l}{\partial b_i^l} \tag{12.12}$$

โดยที่ n_i^l คือเน็ตเอาต์พุตของนิวรอนตัวที่ i ในชั้นซ่อนเร้นที่ l ดังนั้นเทอมที่สองจากทั้งสองสมการข้างต้นสามารถคำนวณได้ เนื่องจากเน็ตเอาต์พุตของเครือข่ายชั้นที่ l ซึ่งได้แก่ n_i^l เป็นฟังก์ชันของค่าน้ำหนักประสาทและไบอัสในชั้นนั้นๆ กล่าวคือ (อินพุตของเครือข่ายชั้นที่ l คือเอาต์พุตของเครือข่ายชั้นที่ $l-1$ ซึ่งได้แก่ y^{l-1})

$$n_i^l = \sum_{j=1}^{S^{l-1}} w_{ij}^l y_j^{l-1} + b_i^l \tag{12.13}$$

ดังนั้น

$$\frac{\partial n_i^l}{\partial w_{ij}^l} = y_j^{l-1} \tag{12.14}$$

และ

$$\frac{\partial n_i^l}{\partial b_i^l} = 1 \tag{12.15}$$

พิจารณาเทอม $\frac{\partial \hat{F}}{\partial n_i^l}$ ในสมการที่ (12.11) และ (12.12) ซึ่งเป็นอัตราการเปลี่ยนแปลงของฟังก์ชันค่าความผิดพลาด $\hat{F}$ เทียบกับเน็ตเอาต์พุต n_i^l โดยกำหนดอัตราการเปลี่ยนแปลงดังกล่าวเป็น**ความไวของค่าความผิดพลาด** (error sensitivity) ของนิวรอนตัวที่ i ในชั้นที่ l ดังนี้

$$\delta_i^l = \frac{\partial \hat{F}}{\partial n_i^l} \tag{12.16}$$

ดังนั้นจากสมการ (12.11) (12.12) (12.14) และ (12.15) จะได้

$$\frac{\partial \hat{F}}{\partial w_{ij}^l} = \delta_i^l y_j^{l-1} \tag{12.17}$$

และ

$$\frac{\partial \hat{F}}{\partial b_i^l} = \delta_i^l \tag{12.18}$$

ในขั้นตอนการนำเอาไปใช้งานจริง เราจำเป็นจะต้องคำนวณหาค่าความไวของค่าความผิดพลาด δ_i^l นี้ พิจารณาเริ่มจากใช้ลูกโซ่อีกครั้งดังนี้

$$\delta_i^l = \frac{\partial \hat{F}}{\partial n_i^l} \tag{12.19}$$

$$= \frac{\partial \hat{F}}{\partial y_i^l}\frac{\partial y_i^l}{\partial n_i^l} \tag{12.20}$$

เนื่องจากเอาต์พุตของเครือข่าย y_i^l เป็นฟังก์ชันโดยตรงของเครือข่ายเอาต์พุต n_i^l กล่าวคือ

$$y_i^l = f(n_i^l) \tag{12.21}$$

โดยที่ $f(\cdot)$ คือฟังก์ชันถ่ายโอนของนิวรอนชั้นที่ l นั้นๆ ดังนั้นจะได้ว่า

$$\frac{\partial y_i^l}{\partial n_i^l} = f'(n_i^l) \tag{12.22}$$

จากความสัมพันธ์ข้างต้นแสดงให้เห็นว่าฟังก์ชันถ่ายโอนที่ใช้ในการฝึกสอนแบบแพร่กลับ จำเป็นจะต้องหาอนุพันธ์ได้ (differentiable) ทำการพิจารณาสมการที่ (12.20) อีกครั้ง โดยพิจารณาเทอม $\frac{\partial \hat{F}}{\partial y_i^l}$ ในที่นี้เราจะต้องแยกพิจารณาเป็น 2 กรณีคือ

1. <u>กรณีชั้นที่ l เป็นชั้นเอาต์พุต ($l = N$)</u> เราสามารถหาอนุพันธ์ของ $\hat{F}$ เทียบโดยตรงกับเอาต์พุต y_i^N ได้เนื่องจาก

$$\hat{F} = (\mathbf{t} - \mathbf{y})^T(\mathbf{t} - \mathbf{y}) = \sum_{j=1}^{S^N} (t_j - y_j^N)^2 \tag{12.23}$$

ดังนั้น

$$\frac{\partial \hat{F}}{\partial y_i^N} = \frac{\partial \sum_{j=1}^{S^N} (t_j - y_j^N)^2}{\partial y_i^N} \tag{12.24}$$

$$= 2(t_i - y_i^N)\frac{\partial (t_i - y_i^N)}{\partial y_i^N} \tag{12.25}$$

$$= -2(t_i - y_i^N) \tag{12.26}$$

ดังนั้นจะได้ค่าความไวของค่าความผิดพลาดในกรณีที่ $l = N$ เป็นชั้นเอาต์พุตคือ

$$\delta_i^N = -2(t_i - y_i^N) f'(n_i^N) \tag{12.27}$$

2. <u>กรณีชั้นที่ l เป็นชั้นซ่อนเร้น ($l \neq N$)</u> ในกรณีนี้เราไม่สามารถหาอนุพันธ์ของ $\hat{F}$ ได้โดยตรง โดยการใช้ลูกโซ่อีกครั้งจะได้

$$\frac{\partial \hat{F}}{\partial y_i^l} = \sum_{l=1}^{S^{l+1}} \frac{\partial \hat{F}}{\partial n_l^{l+1}} \frac{\partial n_l^{l+1}}{\partial y_i^l} \tag{12.28}$$

เนื่องจากเอาต์พุต y_i^l จะกลายเป็นอินพุตให้กับทุกนิวรอนในชั้นที่ $l + 1$ ดังนั้นสมการข้างต้นจึงเป็นผลรวมเอาต์พุตของ S^{l+1} นิวรอนในชั้นที่ $l + 1$ เมื่อพิจารณาจากเทอมหลังของสมการข้างต้นจะได้ว่า

$$\frac{\partial n_l^{l+1}}{\partial y_i^l} = \frac{\partial}{\partial y_i^l}\left(\sum_{r=1}^{S^l} w_{rl}^{l+1} y_r^l + b_r^{l+1}\right) \tag{12.29}$$

$$= \frac{\partial (w_{il}^{l+1} y_i^l)}{\partial y_i^l} \tag{12.30}$$

$$= w_{il}^{l+1} \tag{12.31}$$

แทนค่าจากสมการที่ (12.31) ลงในสมการที่ (12.28) จะได้

$$\frac{\partial \hat{F}}{\partial y_i^l} = \sum_{l=1}^{S^{l+1}} \frac{\partial \hat{F}}{\partial n_l^{l+1}} w_{il}^{l+1} \tag{12.32}$$

จากนิยามค่าความไวของค่าความผิดพลาดของนิวรอนตัวที่ l ในชั้นที่ $l + 1$ จะได้ว่า

$$\delta_l^{l+1} = \frac{\partial \hat{F}}{\partial n_l^{l+1}} \tag{12.33}$$

แทนค่าข้างต้นลงในสมการที่ (12.28) จะได้ว่า

$$\frac{\partial \hat{F}}{\partial y_i^l} = \sum_{l=1}^{S^{l+1}} \delta_l^{l+1} w_{il}^{l+1} \tag{12.34}$$

ดังนั้นค่าความไวของค่าความผิดพลาดของนิวรอนในชั้นที่ l คือ

$$\delta_i^l = f'(n_i^l) \sum_{l=1}^{S^{l+1}} \delta_l^{l+1} w_{il}^{l+1} \tag{12.35}$$

ผลลัพธ์ที่ได้จากสมการข้างต้นแสดงให้เห็นถึงที่มาของชื่ออัลกอริทึม "แพร่กลับ" กล่าวคือค่าความไว δ_i^l ของค่าความผิดพลาดที่ได้ในชั้นที่ l ที่พิจารณาสามารถคำนวณได้จากองค์ประกอบในชั้นถัดไป ได้แก่ δ_i^{l+1} (ค่าความไวเป็นส่วนหนึ่งของการปรับค่าน้ำหนักประสาทและไบอัสเท่านั้น) คำถามที่อาจเกิดขึ้นก็คือ แล้วเราจะรู้ค่าขององค์ประกอบในชั้นถัดไปได้อย่างไร? อย่าลืมว่าเราได้วิเคราะห์และหาค่าต่างๆ ของชั้นเอาต์พุตไว้เรียบร้อยแล้ว เมื่อเริ่มต้นจากชั้นที่ $l = N-1$ เราสามารถนำเอาค่าความไวจากชั้นเอาต์พุต N มาใช้ เมื่อปรับค่าน้ำหนักประสาทและไบอัสในชั้น $N-1$ แล้ว ค่าความไวของชั้นนี้ก็จะถูกนำไปใช้ในการปรับค่าน้ำหนักประสาทและไบอัสในชั้น $N-2$ ไปเรื่อยๆ จนกระทั่งถึงชั้นอินพุต

จากสมการ (12.9) (12.10) (12.17) และ (12.18) เราสามารถสรุปอัลกอริทึมแบบแพร่กลับสำหรับปรับค่าน้ำหนักประสาทได้ดังนี้

$$w_{ij}^l(k+1) = w_{ij}^l(k) - \alpha \delta_i^l(k) y_j^{l-1}(k) \tag{12.36}$$

และ

$$b_i^l(k+1) = b_i^l(k) - \alpha \delta_i^l(k) \tag{12.37}$$

หรือในรูปของเมตริกซ์คือ

$$\mathbf{W}_{new}^l = \mathbf{W}_{old}^l - \alpha \delta^l \mathbf{y}^{l-1} \tag{12.38}$$

และ

$$\mathbf{b}_{new}^l = \mathbf{b}_{old}^l - \alpha \delta^l \tag{12.39}$$

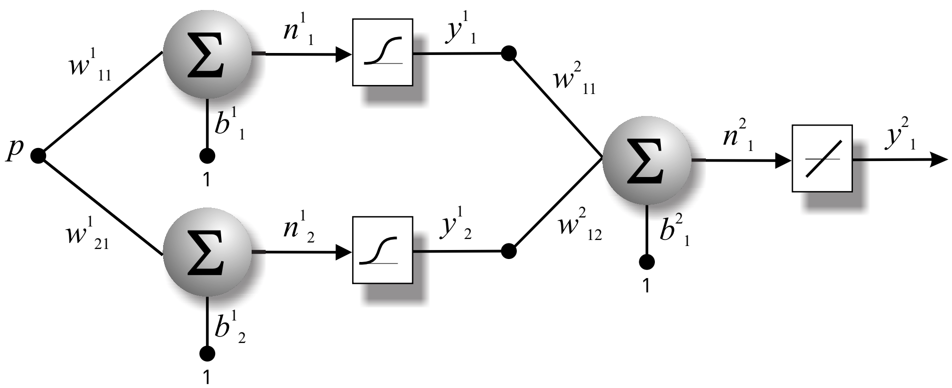
โดยปกติแล้วในทางปฏิบัติเราต้องคำนวณหาเทอม $f'(n)$ ตัวอย่างเช่นฟังก์ชันถ่ายโอนแบบซิกมอยจะได้

$$y = f(n) = \frac{1}{1+e^{-n}} \tag{12.40}$$

และ

$$\begin{aligned}
f'(n) &= \frac{\partial}{\partial n}\left(\frac{1}{1+e^{-n}}\right) && (12.41)\\
&= \left(\frac{-1}{(1+e^{-n})^2}\right)\frac{\partial}{\partial n}\left(1+e^{-n}\right) && (12.42)\\
&= \left(\frac{-1}{(1+e^{-n})^2}\right)(-e^{-n}) && (12.43)\\
&= \left(\frac{1}{1+e^{-n}}\right)\left(\frac{e^{-n}}{1+e^{-n}}\right) && (12.44)\\
&= \left(\frac{1}{1+e^{-n}}\right)\left(1-\frac{1}{1+e^{-n}}\right) && (12.45)\\
&= y(1-y) && (12.46)
\end{aligned}$$

องค์ประกอบฟังก์ชันค่าความไว δ เป็นส่วนที่ทำให้อัลกอริทึมแบบแพร่กลับแตกต่างไปจากอัลกอริทึม LMS การปรับค่าน้ำหนักประสาทและไบอัสยังเป็นไปในแบบลงชันสุด (steepest descent) ซึ่งจะนำเครือข่ายไปในทิศทางที่ทำให้ค่าความผิดพลาดกำลังสองเฉลี่ยมีค่าน้อยที่สุด



รูปที่ 12.6: ตัวอย่างเครือข่ายสำหรับการประมาณค่าฟังก์ชัน

12.3 การประยุกต์ใช้งานการเรียนรู้แบบแพร่กลับ: การประมาณค่าฟังก์ชันด้วยเครือข่ายแบบแพร่กลับ

เครือข่ายแบบไปข้างหน้าสามารถประยุกต์ใช้เป็นเครือข่ายในการประมาณค่าฟังก์ชันใดๆ ได้ ซึ่งการประมาณค่าฟังก์ชันดังกล่าวมีประโยชน์ในงานหลายๆ ด้าน เช่นด้านระบบควบคุม หรือตัวกรองแบบปรับค่าได้ (adaptive filter) เป็นต้น

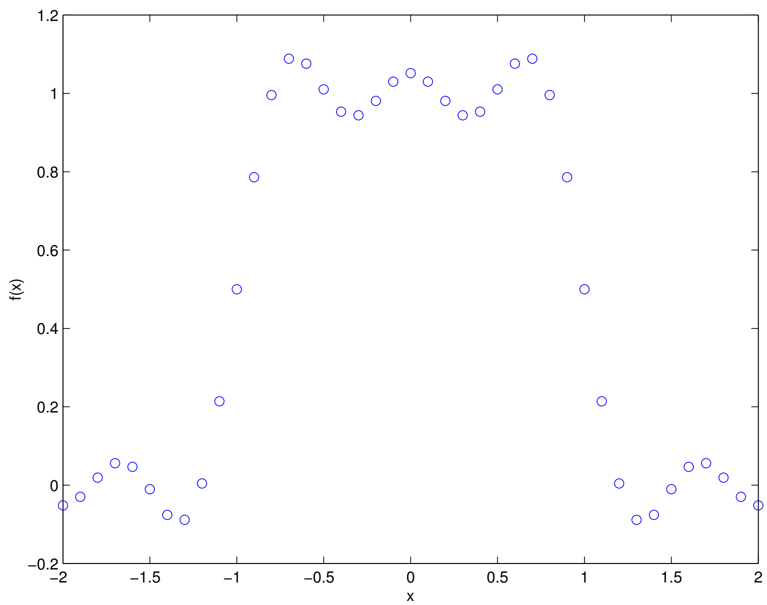
ในการประมาณค่าฟังก์ชัน $y = f(x)$ คู่อินพุต/เป้าหมายที่ใช้ในการฝึกสอนเครือข่ายจะอยู่ในรูป $\{x, y\}$ โดยที่ x เป็นค่าอินพุต และ y เป็นค่าเอาต์พุตหรือเป้าหมายที่ต้องการ เครือข่ายสามารถมีค่าอินพุตได้มากกว่า 1 อินพุต โดยปกติแล้วเครือข่ายไปข้างหน้าที่มี 2 ชั้นเพียงพอที่จะใช้ในการประมาณค่าฟังก์ชันใดๆ ได้ถ้ามีการออกแบบเลือกจำนวนนิวรอนและชนิดของฟังก์ชันถ่ายโอนที่เหมาะสม ยกตัวอย่างเช่นเครือข่ายในรูปที่ 12.6 ซึ่งเป็นเครือข่าย 2 ชั้น ชั้นแรกมีสองนิวรอน แต่ละนิวรอนมีฟังก์ชันถ่ายโอนแบบซิกมอยล็อก ชั้นเอาต์พุตมีเพียง 1 นิวรอนและฟังก์ชันถ่ายโอนคือฟังก์ชันเส้นตรง เครือข่ายมีอินพุตเดียวสำหรับรับค่าข้อมูลตัวอย่างของฟังก์ชันที่ต้องการประมาณค่า พิจารณาฟังก์ชันตัวอย่างที่จะทำการประเมินในรูปที่ 12.7 ตัวอย่างรูปคลื่นดังกล่าวได้จากการประมาณค่าด้วยอนุกรมฟูริเยร์ดังนี้

$$y = f(x) = \frac{1}{2} + \frac{2}{\pi}\left\{\cos\frac{\pi}{2}x - \frac{1}{3}\cos\frac{3\pi}{2}x + \frac{1}{5}\cos\frac{5\pi}{2}x\right\} \tag{12.47}$$

ค่าชักตัวอย่าง (sampling) ที่ใช้อยู่ในช่วง $x = [-2, 2]$ ดังนั้นอินพุตของเครือข่ายคือ x และเป้าหมายของเครือข่ายคือ $f(x)$ ตามความสัมพันธ์ในรูปที่ 12.7

12.3.1 คำสั่ง MATLAB® ในการฝึกสอน

ใน MATLAB Neural Network Toolbox คำสั่งที่ใช้ในการสร้างเครือข่ายคือ net = newff ซึ่งจะทำการสร้างเครือข่ายป้อนไปข้างหน้า (feedforward) เมื่อทำการสร้างเครือข่ายด้วยคำสั่งข้างต้นแล้ว เราสามารถทดสอบเครือข่ายได้ด้วยคำสั่ง y = sim(net, p) โดยที่ตัวแปร net เป็นตัวแปรที่ได้จากการสร้างเครือข่ายด้วยคำสั่ง newff ตัวแปร p เป็นอินพุตที่ต้องการทดสอบและตัวแปร y เป็นเอาต์พุตจากเครือข่ายพิจารณาคำสั่ง newff ซึ่งจะสร้างเครือข่ายที่มีค่าเมตริกซ์น้ำหนักประสาทและไบอัสเริ่มต้นด้วยค่าสุ่ม ดังนั้นเมื่อทดสอบกับอินพุตในขณะนี้ เอาต์พุตที่ได้จึงสามารถคาดเดาได้ว่าจะไม่เป็นไปตามเป้าหมาย (หรือฟังก์ชัน) ที่เราต้องการ ในการฝึกสอนเครือข่ายนั้นจะใช้คำสั่ง net = train(net, p, t) โดยที่ตัวแปร p และ t เป็นคู่อินพุต/เป้าหมายที่ต้องการให้เครือข่ายเรียนรู้ โดยปกติแล้วตัวแปร net ซึ่งเป็นตัวแปรโครงสร้างที่เก็บรายละเอียดของเครือข่ายไว้ จะมีพารามิเตอร์ภายในที่สามารถใช้ปรับแต่งการฝึกสอนตามต้องการได้ ตัวอย่างพารามิเตอร์ดังกล่าวมีเช่น



รูปที่ 12.7: ค่าซักตัวอย่างของฟังก์ชันประมาณของรูปคลื่นสี่เหลี่ยมจากอนุกรมฟูริเยร์

- net.performFcn - ฟังก์ชันประเมินค่า (ค่าปกติคือ 'mse' หรือ mean-squared error)
- net.trainFcn - ฟังก์ชันการฝึกสอน (ค่าปกติคือ 'trainlm' หมายถึงการเรียนรู้แบบแพร่กลับ)
- net.trainParam - ประกอบไปด้วยพารามิเตอร์ย่อยเช่น
  - .epochs - จำนวนรอบสูงสุดในการฝึกสอน (ค่าปกติเท่ากับ 50)
  - .goal - ค่าความผิดพลาดเป้าหมาย (ค่าปกติเท่ากับ 0)
  - .min_grad - ค่าเกรเดียนที่น้อยที่สุด (ค่าปกติเท่ากับ 1×10^{-15})
  - .show - จำนวนรอบที่จะแสดงผลการฝึกสอน (ค่าปกติเท่ากับ 25)

เราสามารถปรับแต่งค่าพารามิเตอร์ต่างๆ เหล่านี้ตามต้องการได้เช่น

$$\text{net.trainParam.epochs} = 100$$

จะทำการปรับให้การฝึกสอนหยุดภายในจำนวนรอบเท่ากับ 100 รอบ (หรือเงื่อนไขอื่นบรรลุก่อนก็ได้) ผู้อ่านสามารถศึกษารายละเอียดเพิ่มเติมได้จาก MATLAB Neural Network Toolbox

12.3.2 ผลการประมาณค่าฟังก์ชัน

หลังจากทำการฝึกสอนเครือข่ายแบบต่างๆ ด้วยการเรียนรู้แบบแพร่กลับ เพื่อให้เครือข่ายเรียนรู้ความสัมพันธ์ระหว่าง x และ $f(x)$ แล้ว ตัวอย่างผลการฝึกสอนสรุปได้ดังตารางที่ 12.1 รูปที่ 12.8 และรูปที่ 12.9 (เครือข่าย 1-a-b-1 คือเครือข่ายที่มี 1 อินพุต ชั้นซ่อนเร้นแรกมี a นิวรอน ชั้นซ่อนเร้นที่สองมี b นิวรอน และมี 1 เอาต์พุต) จากผลการทดลองที่ได้ จะเห็นได้ว่าพารามิเตอร์ของเครือข่าย ได้แก่จำนวนชั้นและจำนวนนิวรอน มีผลโดยตรงต่อประสิทธิภาพของเครือข่าย โดยทั่วไปแล้วเครือข่ายที่มี 1 ชั้นซ่อนเร้นสามารถใช้ประมาณค่าฟังก์ชันต่อเนื่อง (continuous function) ใดๆ ก็ได้ ในขณะที่เครือข่ายที่มี 2 ชั้นซ่อนเร้นสามารถใช้ประมาณค่าฟังก์ชันที่ไม่ต่อเนื่อง (discontinuous function) ได้ รูปที่ 12.10 แสดงการประมาณค่ารูปคลื่นสี่เหลี่ยมด้วยเครือข่าย 1-5-3-1 จำนวนรอบที่ใช้ในการฝึกสอนเท่ากับ 78 รอบ ค่าความผิดพลาดกำลังสองเฉลี่ยจากการฝึกสอนเท่ากับ 3.22447×10^{-28} และค่าเกรเดียนสุดท้ายเท่ากับ 4.55814×10^{-15}

ตารางที่ 12.1: ผลการฝึกสอนเครือข่ายในการประมาณค่าฟังก์ชัน (จำนวนรอบสูงสุด = 500 รอบ, ค่าความผิดพลาดกำลังสองเฉลี่ยเป้าหมาย = 0, ค่าเกรเดียนที่น้อยที่สุด = 1×10^{-15})

เครือข่าย	จำนวนรอบ	ค่าความผิดพลาดกำลังสองเฉลี่ย	ค่าเกรเดียน
1-2-1	27	0.00244106	1.44814e-010
1-4-1	500	0.00239661	2.61261e-006
1-6-1	500	0.000711534	2.61261e-006
1-10-1	500	0.000394703	3.05478e-005
1-5-3-1	500	6.03452e-005	0.00448673
1-6-4-1	500	2.7649e-008	0.000107267

12.4 วิเคราะห์การใช้งานเครือข่ายแบบแพร่กลับ

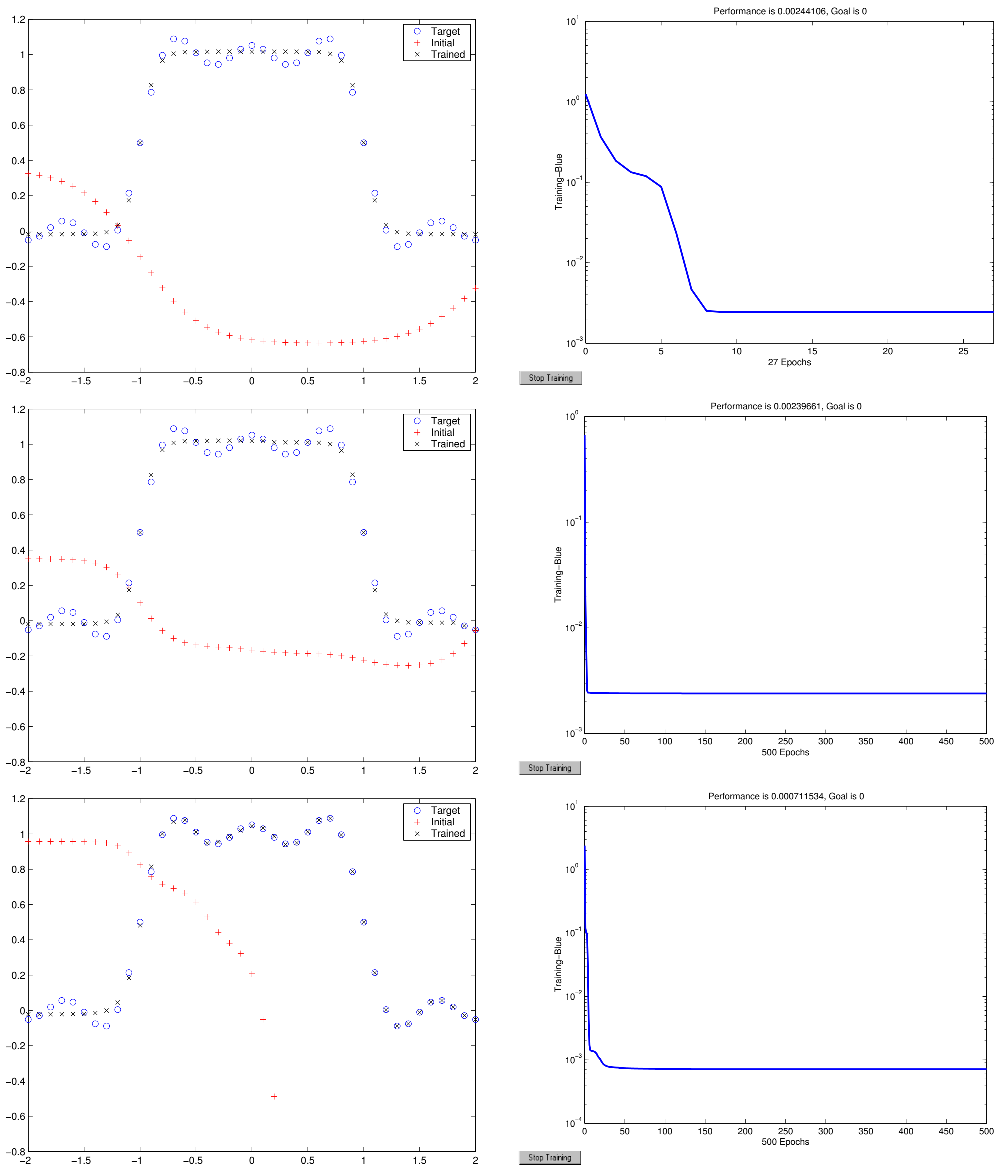
สิ่งสำคัญที่ควรค่าแก่การศึกษาและวิเคราะห์สำหรับการออกแบบใช้งานเครือข่ายประสาทเทียมแบบแพร่กลับ คือการเลือกพารามิเตอร์ของเครือข่าย การลู่เข้าของการฝึกสอน (convergence) และการทำให้เป็นกรณีทั่วไป(generalization) พารามิเตอร์ของเครือข่ายถือเป็นสิ่งแรกที่ผู้ใช้เครือข่ายประสาทเทียมทุกรูปแบบต้องคำนึงถึงก่อนเป็นอันดับแรก การลู่เข้าของอัลกอริทึมการฝึกสอนเป็นส่วนที่จำเป็น เนื่องจากเครือข่ายจะไม่สามารถกล่าวได้ว่ามีการเรียนรู้ ถ้ากระบวนการฝึกสอนไม่มีแนวโน้มลู่เข้าสู่เงื่อนไขที่ต้องการ การทำให้เป็นกรณีทั่วไปถือเป็นคุณสมบัติสำคัญที่เราต้องการและคาดหวังจากเครือข่ายประสาทเทียม ด้วยการฝึกสอนด้วยชุดข้อมูลที่มีจำนวนจำกัด เราคาดหวังที่จะให้เครือข่ายเรียนรู้และสามารถทำงานกับชุดข้อมูลอื่นๆ ที่ไม่เคยใช้ในการฝึกสอนมาก่อน ในหัวข้อต่อไปนี้จะได้นำเสนอรายละเอียดการวิเคราะห์ประสิทธิภาพของเครือข่ายแบบแพร่กลับ ในมุมมองดังกล่าว

12.4.1 การเลือกพารามิเตอร์ของเครือข่าย

จากตัวอย่างเครือข่ายการประมาณค่าฟังก์ชัน จะเห็นได้ว่าโครงสร้างของเครือข่ายมีผลโดยตรงต่อประสิทธิภาพการประมาณค่าฟังก์ชัน โดยมีข้อแม้ว่าจำนวนนิวรอนและจำนวนชั้นซ่อนเร้นจะต้องเหมาะสม ซึ่งโดยทั่วไปแล้วเราไม่สามารถบอกได้ว่าจำนวนดังกล่าวคืออะไร ตัวเลือกดังกล่าวย่อมขึ้นอยู่กับความซับซ้อนของฟังก์ชันที่ต้องการประมาณค่าด้วยดังตัวอย่างแสดงในรูปที่ 12.11 ฟังก์ชันตัวอย่างได้จากอนุกรมฟูริเยร์ในอันดับ 1 จนถึงอันดับ 3 โดยใช้เครือข่าย 1-2-1 จะเห็นได้ว่าสำหรับฟังก์ชันที่มีความซับซ้อนมากขึ้น ระบบจะสามารถประมาณฟังก์ชันได้เพียงบางส่วนเท่านั้น เราสามารถเลือกที่จะปรับพารามิเตอร์ของเครือข่าย ซึ่งได้แก่จำนวนนิวรอนและจำนวนชั้นซ่อนเร้นเพื่อให้ระบบสามารถทำงานได้มีประสิทธิภาพเพิ่มขึ้น อย่างไรก็ตามการเลือกจำนวนดังกล่าวที่มากเกินไปไม่จำเป็นจะต้องให้ผลลัพธ์ที่ดีเสมอไป เนื่องมาจากเครือข่ายที่มีความซับซ้อนมากเกินไปจะทำให้การฝึกสอนยุ่งยากไปด้วย

12.4.2 การลู่เข้า
Convergence

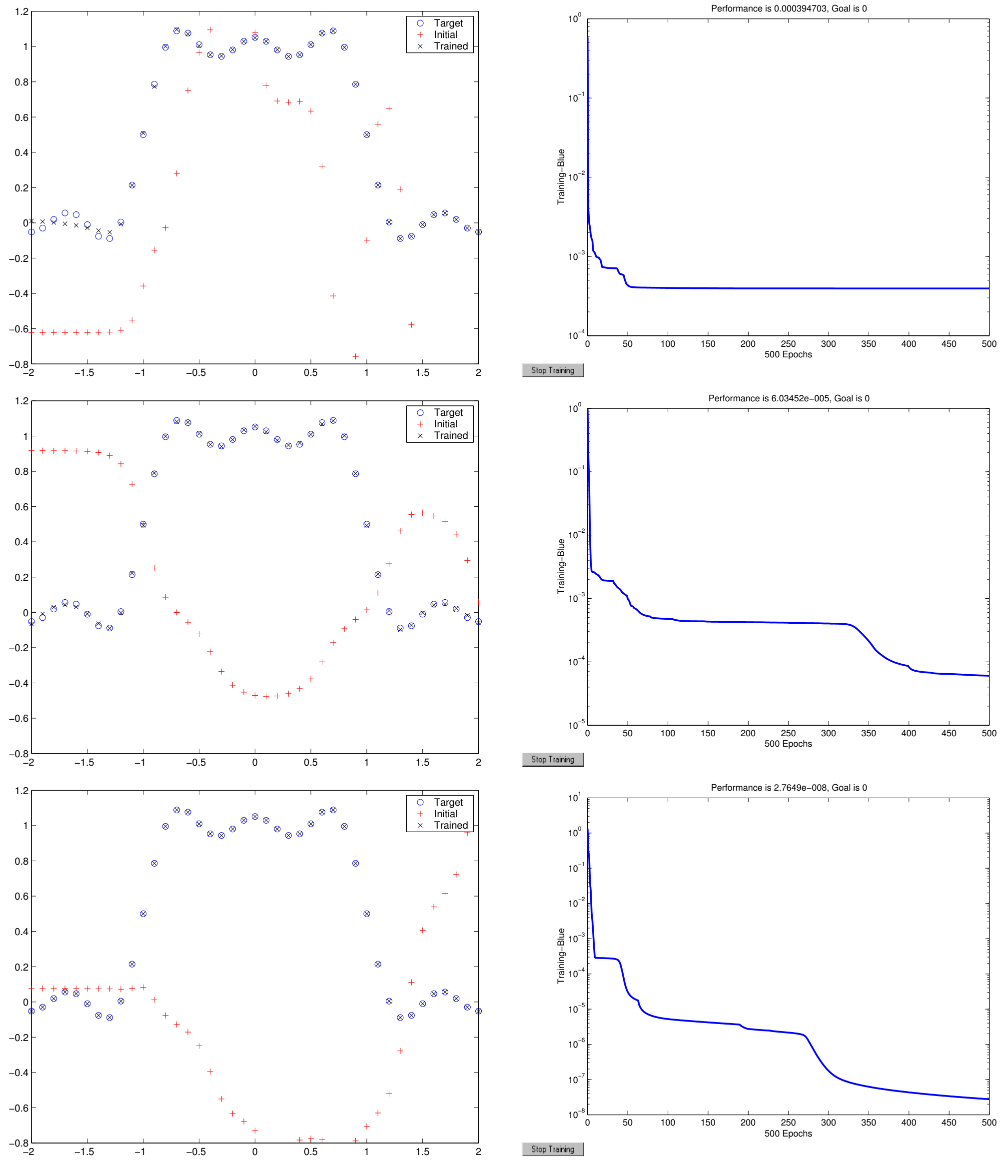
ในเครือข่ายที่มีหลายชั้น (multilayer network) พื้นผิวค่าความผิดพลาดจะมีความซับซ้อนมาก การฝึกสอนของเครือข่ายซึ่งมุ่งเน้นไปที่ทำให้ค่าความผิดพลาดกำลังสองเฉลี่ยมีค่าน้อยที่สุด อาจจะลู่เข้าสู่คำตอบของค่าเมตริกซ์น้ำหนักประสาทและไบอัสที่เป็นแบบวงแคบเฉพาะถิ่นได้ ดังนั้นการฝึกสอนที่ลู่เข้า อาจจะไม่ได้หมายถึงคำตอบที่ดีที่สุดแบบวงกว้างก็ได้ เราสามารถสังเกตพฤติกรรมของค่าเกรเดียนได้ ซึ่งในระหว่างการฝึกสอน ค่าเกรเดียนจะมีแนวโน้มที่ลดลง และจะสิ้นสุดเมื่อค่าเกรเดียนมีค่าประมาณศูนย์ ผลลัพธ์ที่ได้อาจจะไม่ใช่ผลลัพธ์ที่ดีที่สุดก็ได้



(ก) ผลการประมาณค่าฟังก์ชัน (ข) ค่าความผิดพลาดระหว่างการฝึกสอน

รูปที่ 12.8: ตัวอย่างผลการฝึกสอนด้วยเครือข่าย 1-2-1 1-4-1 และ 1-6-1 ตามลำดับ (สัญลักษณ์วงกลม o แทนค่าชักตัวอย่างจากฟังก์ชันประมาณของรูปคลื่นสี่เหลี่ยมจากอนุกรมฟูริเยร์ สัญลักษณ์ x แทนค่าประมาณฟังก์ชันที่ได้จากเครือข่ายแบบแพร่กลับ และสัญลักษณ์ + แทนค่าความผิดพลาดระหว่างฟังก์ชันจากอนุกรมฟูริเยร์และจากการประมาณของเครือข่าย)

เนื่องจากเกรเดียนเป็นศูนย์ ณ จุดที่เป็นคำตอบแบบวงแคบเฉพาะถิ่น โดยทั่วไปแล้วอัลกอริทึมแบบแพร่กลับจะลู่เข้าสู่คำตอบได้ เมื่อค่าคงที่การเรียนรู้มีค่าน้อยเพียงพอที่จะนำไปสู่คำตอบที่ต้องการได้ ค่าคงที่การเรียนรู้ที่มีค่าน้อยทำให้การปรับค่าน้ำหนักประสาทและไบอัสของเครือข่าย เป็นไปอย่างช้าๆ (เปลี่ยนค่าทีละน้อยๆ) แน่นอนว่าความเร็วในการเรียนรู้ก็จะช้าตามไปด้วย

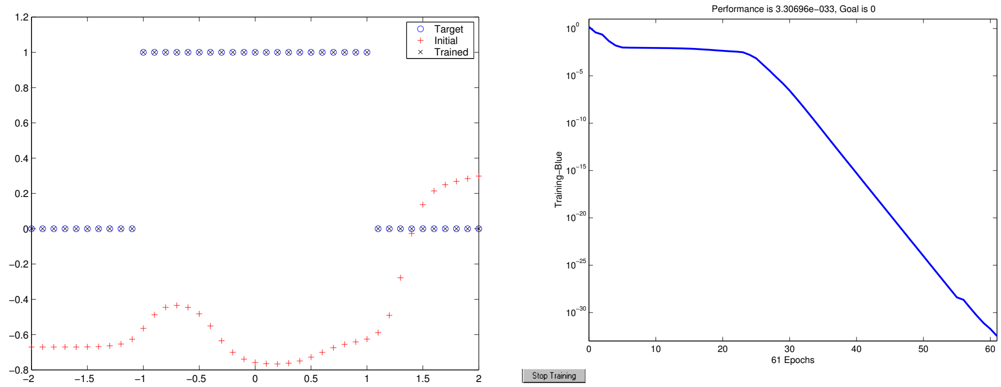


(ก) ผลการประมาณค่าฟังก์ชัน (ข) ค่าความผิดพลาดระหว่างการฝึกสอน

รูปที่ 12.9: ตัวอย่างผลการฝึกสอนด้วยเครือข่าย 1-10-1 1-5-3-1 และ 1-6-4-1 ตามลำดับ (สัญลักษณ์วงกลม o แทนค่าชักตัวอย่างจากฟังก์ชันประมาณของรูปคลื่นสี่เหลี่ยมจากอนุกรมฟูริเยร์ สัญลักษณ์ x แทนค่าประมาณฟังก์ชันที่ได้จากเครือข่ายแบบแพร่กลับ และสัญลักษณ์ + แทนค่าความผิดพลาดระหว่างฟังก์ชันจากอนุกรมฟูริเยร์และจากการประมาณของเครือข่าย) (ต่อ)

12.4.3 การทำให้เป็นกรณีทั่วไป
Generalization

โดยปกติเราใช้คู่อินพุต/เป้าหมาย (input/target) จากชุดข้อมูลจริง ซึ่งมีจำนวนจำกัดจำนวนหนึ่ง จำนวนข้อมูลที่ใช้ฝึกสอนดังกล่าวมีความจำเป็นและแตกต่างกันออกไปตามความซับซ้อนของปัญหา ปัญหาที่มีความซับซ้อนมากยกตัวอย่างเช่นการจดจำใบหน้ามนุษย์ด้วยภาพดิจิตอล อาจจะต้องใช้ข้อมูลภาพหน้ามนุษย์หลายพันหลายหมื่นภาพ เนื่องมาจากภาพใบหน้ามนุษย์มีลักษณะความแตกต่างกันอย่างมากมายนับไม่ถ้วน ในความเป็นจริงแล้ว คู่



(ก) ผลการประมาณค่าฟังก์ชัน (ข) ค่าความผิดพลาดระหว่างการฝึกสอน

รูปที่ 12.10: ตัวอย่างผลการฝึกสอนด้วยเครือข่าย 1-5-3-1 ในการประมาณค่ารูปคลื่นสี่เหลี่ยม (สัญลักษณ์วงกลม o แทนค่าชักตัวอย่างจากฟังก์ชันประมาณของรูปคลื่นสี่เหลี่ยมจากอนุกรมฟูริเยร์ สัญลักษณ์ x แทนค่าประมาณฟังก์ชันที่ได้จากเครือข่ายแบบแพร่กลับ และสัญลักษณ์ + แทนค่าความผิดพลาดระหว่างฟังก์ชันจากอนุกรมฟูริเยร์และจากการประมาณของเครือข่าย)

อินพุต/เป้าหมายนี้เป็นเพียงข้อมูลตัวแทนของคู่อินพุต/เป้าหมายที่เป็นไปได้ทั้งหมด ซึ่งมีขนาดข้อมูลที่ใหญ่กว่ามากๆ ดังนั้นสิ่งสำคัญอย่างหนึ่งในการใช้งานเครือข่ายแบบแพร่กลับ คือความสามารถในการเป็นกรณีทั่วไปได้ นั่นคือจากการฝึกสอนด้วยจำนวนตัวอย่าง (คู่อินพุต/เป้าหมาย) จำนวนที่จำกัด เครือข่ายสามารถทำงานเป็นกรณีทั่วไปได้ โดยครอบคลุมคู่อินพุต/เป้าหมายที่ไม่ใช่ตัวอย่างในชุดข้อมูลที่ใช้ฝึกสอนได้อย่างถูกต้อง

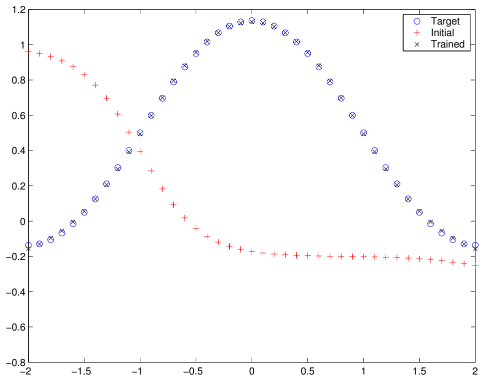
12.5 การปรับแต่งอัลกอริทึมการเรียนรู้แบบแพร่กลับ

ก่อนหน้านี้การประยุกต์ใช้งานอัลกอริทึมแบบแพร่กลับกับระบบในทางปฏิบัติอาจจะต้องใช้เวลาในการฝึกสอนนานมาก ได้มีนักวิจัยมากมายได้นำเสนอวิธีการปรับปรุงอัลกอริทึมแบบแพร่กลับให้ลู่เข้าสู่คำตอบได้เร็วขึ้น ซึ่งวิธีในการปรับปรุงดังกล่าวแบ่งได้เป็นสองกลุ่มใหญ่ๆ คือ

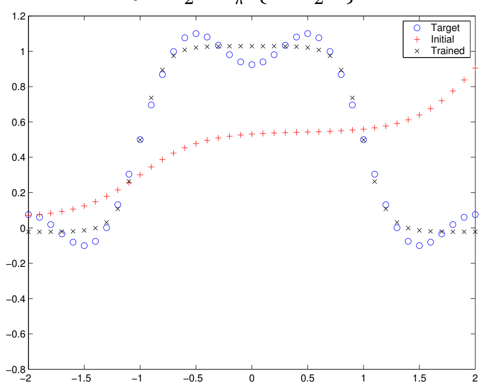
1. **กลยุทธแบบชาญฉลาด** (heuristic strategy) - เป็นการปรับปรุงรายละเอียดของอัลกอริทึมเช่น การปรับค่าการเรียนรู้ หรือโมเมนตัม เป็นต้น หรือใช้เทคนิคปัญญาเชิงคำนวณอย่างอื่นมาช่วยในการฝึกสอนเครือข่าย (หรือค้นหาน้ำหนักประสาทและไบอัส) เทคนิคปัญญาเชิงคำนวณที่สามารถหาค่าเหมาะที่สุด (optimization) ได้ สามารถนำมาประยุกต์ใช้ฝึกสอนเครือข่าย ไม่ว่าจะเป็นจีนเนติกอัลกอริทึม การค้นหาแบบตาบู อัลกอริทึมอบอ่อนจำลอง เป็นต้น รายละเอียดปัญญาเชิงคำนวณแบบผสมนี้จะได้กล่าวถึงในหัวข้อที่เกี่ยวข้องต่อไป

2. **เทคนิคเชิงตัวเลข** (numerical method) - เป็นการปรับปรุงอัลกอริทึมในการคำนวณหาค่าที่เหมาะสมที่สุด เช่นอัลกอริทึมคอนจูเกตเกรเดียน หรืออัลกอริทึม Lavenberg-Marquardt เป็นต้น เทคนิควิธีดังกล่าวมีจุดประสงค์หลักเพื่อให้อัลกอริทึมทำการปรับค่าน้ำหนักประสาทและไบอัส ได้อย่างรวดเร็วและได้ค่าน้ำหนักประสาทและไบอัสที่เหมาะที่สุด

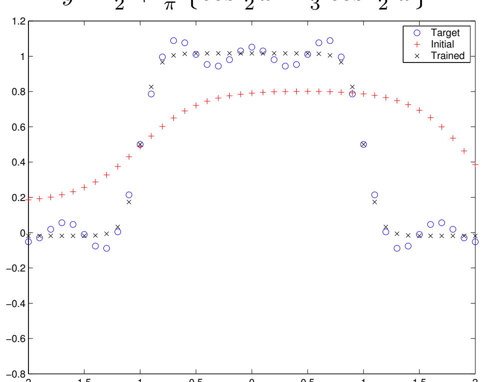
ค่าน้ำหนักประสาทและไบอัสที่เป็นไปได้ทั้งหมดสามารถนำมาพิจารณาในรูปของพื้นผิวค่าความผิดพลาดได้ ดังนั้นประสิทธิภาพของเครือข่ายสามารถวิเคราะห์ได้จากพื้นผิวค่าความผิดพลาดได้เช่นกัน ดังรายละเอียดที่นำเสนอในหัวข้อต่อไปนี้



$$y = \frac{1}{2} + \frac{2}{\pi}\left\{\cos\frac{\pi}{2}x\right\}$$

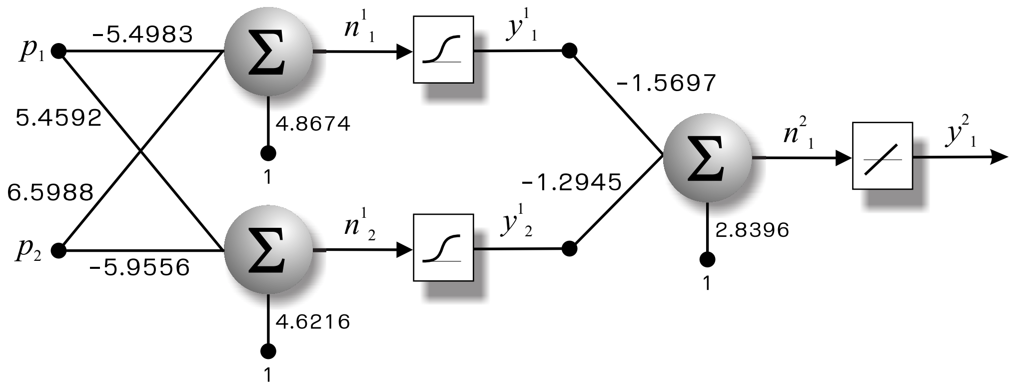


$$y = \frac{1}{2} + \frac{2}{\pi}\left\{\cos\frac{\pi}{2}x - \frac{1}{3}\cos\frac{3\pi}{2}x\right\}$$



$$y = \frac{1}{2} + \frac{2}{\pi}\left\{\cos\frac{\pi}{2}x - \frac{1}{3}\cos\frac{3\pi}{2}x + \frac{1}{5}\cos\frac{5\pi}{2}x\right\}$$

รูปที่ 12.11: การประมาณค่าฟังก์ชันที่มีความซับซ้อนต่างกันด้วยเครือข่าย 1-2-1 (สัญลักษณ์วงกลม o แทนค่าชักตัวอย่างจากฟังก์ชันประมาณของรูปคลื่นสี่เหลี่ยมจากอนุกรมฟูริเยร์ สัญลักษณ์ x แทนค่าประมาณฟังก์ชันที่ได้จากเครือข่ายแบบแพร่กลับ และสัญลักษณ์ + แทนค่าความผิดพลาดระหว่างฟังก์ชันจากอนุกรมฟูริเยร์และจากการประมาณของเครือข่าย)



รูปที่ 12.12: ตัวอย่างเครือข่าย 1-2-1 สำหรับปัญหา XOR

12.5.1 พื้นผิวค่าความผิดพลาด Error Surface

เนื่องจากอัลกอริทึมแบบแพร่กลับใช้หลักวิธีแบบลงชันสุด(steepest descent) เช่นเดียวกันกับในอัลกอริทึม LMS อย่างไรก็ตามในกรณีที่เครือข่ายเป็นแบบหลายชั้น ฟังก์ชันค่าความผิดพลาดกำลังสองเฉลี่ยจะไม่อยู่ในรูปของฟังก์ชันควอดดราติกเหมือนในกรณีของเครือข่ายชั้นเดียว (ADALINE) ซึ่งมีจุดที่ให้ค่าความผิดพลาดกำลังสองเฉลี่ยน้อยที่สุดแบบวงกว้างเพียงจุดเดียว ฟังก์ชันค่าความผิดพลาดของเครือข่ายแบบหลายชั้นมีพื้นผิวที่ซับซ้อนซึ่งสามารถมีจุดที่ให้ค่าความผิดพลาดกำลังสองเฉลี่ยที่น้อยที่สุดแบบวงแคบเฉพาะถิ่นได้หลายจุด รวมไปถึงลักษณะทางกายภาพหลายๆ อย่างที่อาจจะทำให้อัลกอริทึมแพร่กลับแบบธรรมดาไม่สามารถลู่เข้าสู่จุดที่น้อยที่สุดแบบวงกว้างได้ ในหัวข้อนี้จะทำการศึกษารายละเอียดเพิ่มเติมของพื้นผิวค่าความผิดพลาด ซึ่งในที่นี้จะใช้ค่าความผิดพลาดกำลังสองเฉลี่ยของเครือข่ายสำหรับปัญหา XOR ซึ่งมีค่าตัวอย่างของเมตริกซ์น้ำหนักประสาทและไบอัสคือ (ดูรูปที่ 12.12)

$$\mathbf{W}^1 = \begin{bmatrix} -5.4983 & 6.5988 \\ 5.4592 & -5.9556 \end{bmatrix}$$

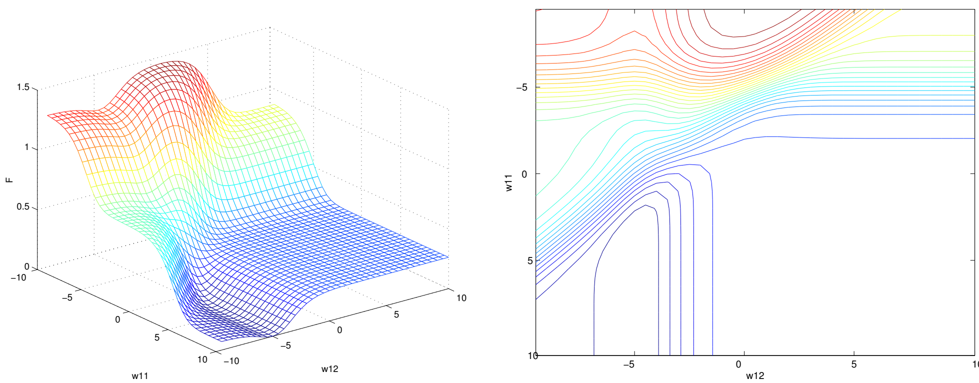
$$\mathbf{W}^2 = \begin{bmatrix} -1.5697 \\ -1.2945 \end{bmatrix}$$

$$\mathbf{b}^1 = \begin{bmatrix} 4.8674 \\ 4.6216 \end{bmatrix}$$

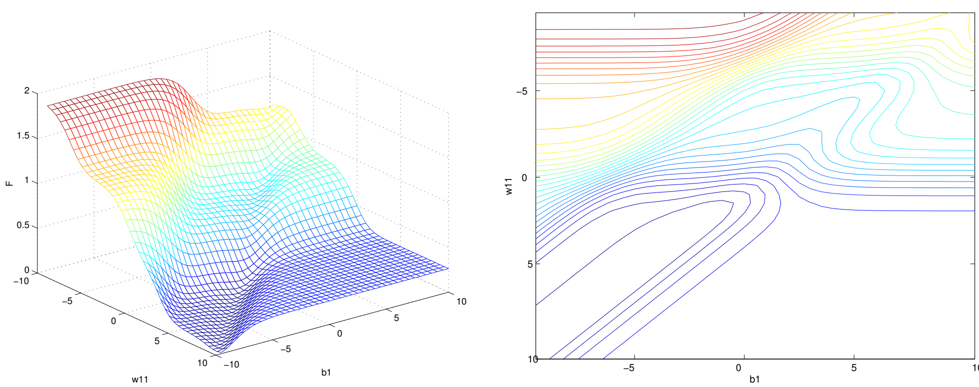
$$\mathbf{b}^2 = [2.8396]$$

เนื่องจากการพล๊อตกราฟความสัมพันธ์ทำได้จำกัดเพียง 2 มิติ เราจะทำการคงค่าพารามิเตอร์ต่างๆ ไว้แล้วทำการปรับค่าเพียง 2 ค่าในแต่ละครั้งเช่นค่า w^1_{11} และ w^1_{12} เป็นต้น ค่าที่ปรับในที่นี้อยู่ในช่วง $[-10, 10]$ รูปที่ 12.13 แสดงผลพื้นผิวค่าความผิดพลาดกำลังสองเฉลี่ยเทียบกับ w^1_{11} และ w^1_{12} และรูปที่ 12.14 แสดงผลพื้นผิวค่าความผิดพลาดกำลังสองเฉลี่ยเทียบกับ w^1_{11} และ b^1_1 จากตัวอย่างพื้นผิวค่าความผิดพลาดกำลังสองเฉลี่ยข้างต้น จะเห็นได้ว่าพื้นผิวไม่ได้เป็นฟังก์ชันควอดราติก นอกไปจากนั้นแล้วลักษณะทางกายภาพของพื้นผิวดังกล่าวยังมีความแตกต่างกันในเทอมของความชัน ดังนั้นจึงไม่ใช่เรื่องง่ายที่จะเลือกค่าคงที่การเรียนรู้ให้เหมาะสมกับพื้นผิวค่าความผิดพลาดกำลังสองในขณะทำการฝึกสอนเครือข่าย นั่นคือพื้นผิวช่วงที่ราบเรียบควรจะมีค่าคงที่การเรียนรู้ที่มาก ในขณะที่พื้นผิวที่มีความชันสูงขึ้นควรมีค่าคงที่การเรียนรู้ที่น้อย

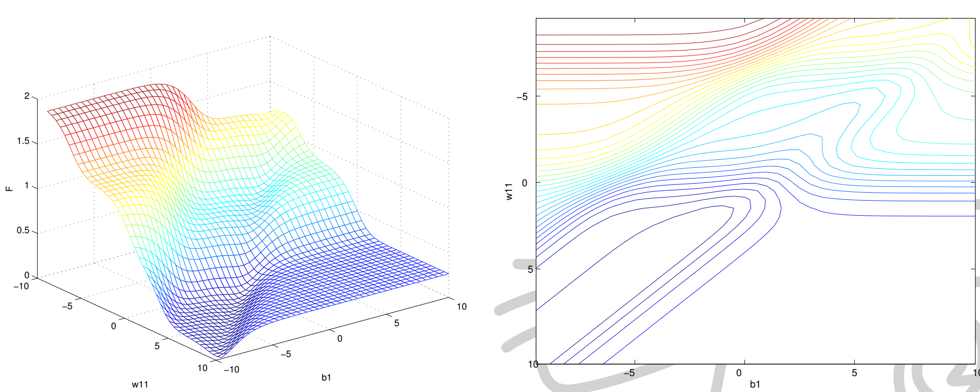
พิจารณาพื้นผิวค่าความผิดพลาดกำลังสองเฉลี่ยในรูปที่ 12.15 ซึ่งเป็นการเปรียบเทียบระหว่าง w^1_{11} และ w^1_{21} จะเห็นได้ว่ามีจุดที่เป็นค่าน้อยที่สุดแบบวงแคบเฉพาะถิ่นอยู่ด้วย ดังนั้นค่าเริ่มต้นของเมตริกซ์น้ำหนักประสาทและ



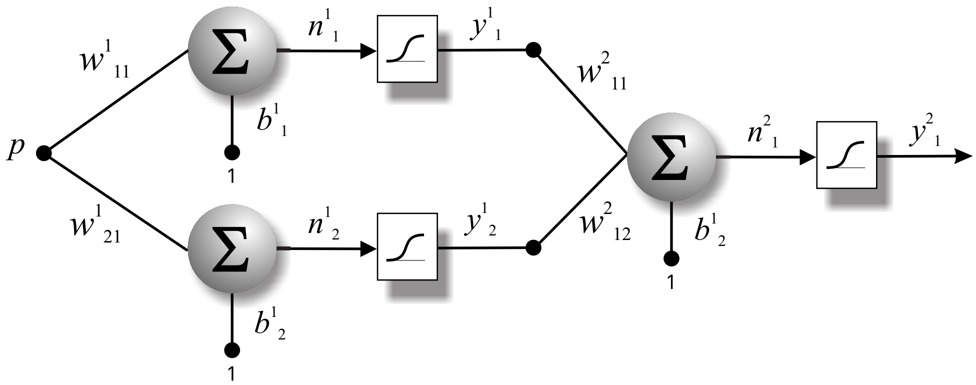
รูปที่ 12.13: พื้นผิวและเส้นโครงร่างของค่าความผิดพลาดกำลังสองเฉลี่ยเทียบกับ w_{11}^1 และ w_{12}^1



รูปที่ 12.14: พื้นผิวและเส้นโครงร่างของค่าความผิดพลาดกำลังสองเฉลี่ยเทียบกับ w_{11}^1 และ b_1^1



รูปที่ 12.15: พื้นผิวและเส้นโครงร่างของค่าความผิดพลาดกำลังสองเฉลี่ยเทียบกับ w_{11}^1 และ w_{21}^1



รูปที่ 12.16: เครือข่าย 1-2-1 สำหรับใช้ในการวิเคราะห์การลู่เข้าสู่คำตอบของเครือข่าย

ไบอัส ซึ่งปกติจะใช้เป็นค่าสุ่มจะมีผลต่อการลู่เข้าของอัลกอริทึม ตำแหน่งเริ่มต้นของเมตริกซ์น้ำหนักประสาทและไบอัสที่อยู่ใกล้จุดที่น้อยที่สุดแบบวงแคบเฉพาะถิ่นจะมีโอกาสในการลู่เข้าไปยังจุดนั้นได้ ในทางปฏิบัติแล้วเรามักจะทดลองใช้ค่าเริ่มต้นของเมตริกซ์น้ำหนักประสาทและไบอัสแตกต่างกันหลายๆ ค่าเพื่อให้แน่ใจว่าเครือข่ายมีการลู่เข้าสู่คำตอบที่เป็นแบบวงกว้าง

พิจารณาเครือข่ายในรูปที่ 12.16 และค่าพื้นผิวในรูปที่ 12.17 และ 12.18 (ดูโปรแกรมสาธิตใน MATLAB เวอร์ชัน 6.0 ขึ้นไป โดยเรียกคำสั่ง nnd12sd1) รูปที่ 12.19 และ 12.20 แสดงการลู่เข้าของการฝึกสอนสำหรับค่าเริ่มต้นที่ตำแหน่งต่างๆ จะเห็นได้ว่าไม่ใช่ทุกค่าเริ่มต้นที่ให้การลู่เข้าสู่คำตอบที่ต้องการได้ รูปที่ 12.21-(ข) แสดงการลู่เข้าสู่คำตอบที่เป็นวงแคบเฉพาะถิ่น

12.5.2 การปรับปรุงอัลกอริทึมการเรียนรู้แบบแพร่กลับด้วยโมเมนตัม

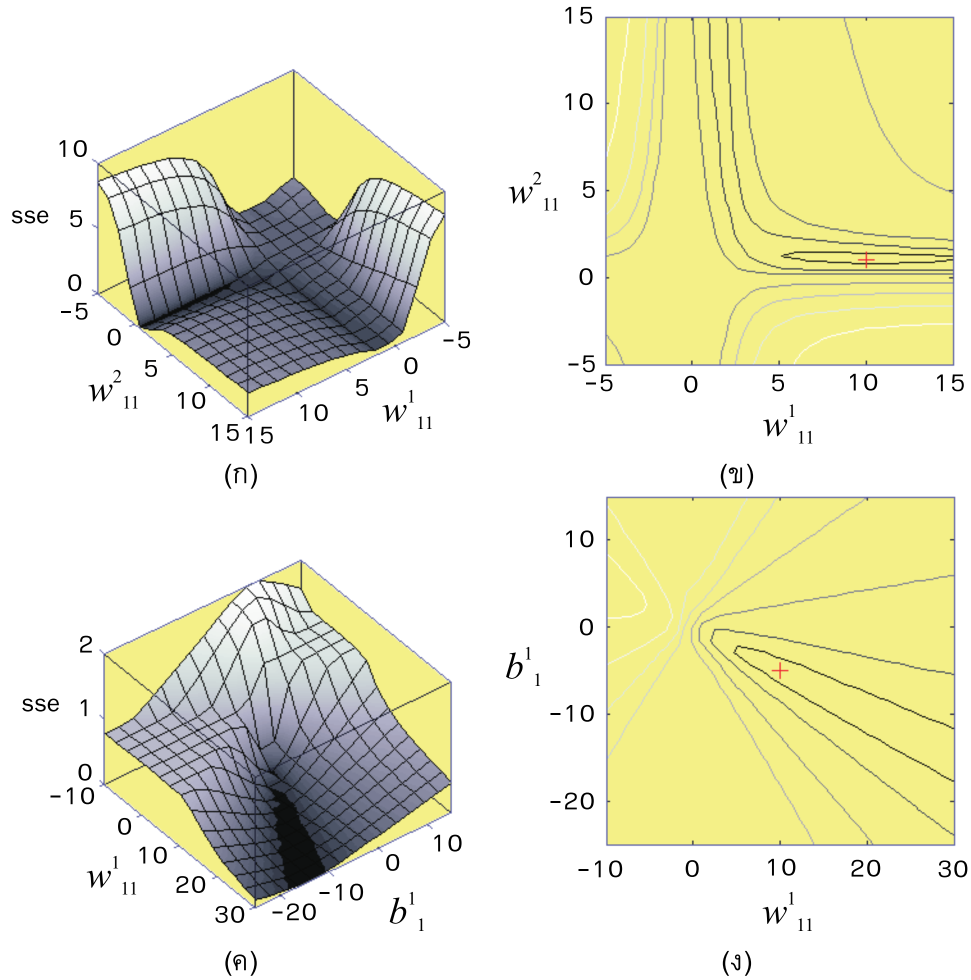
การปรับปรุงอัลกอริทึมแบบแพร่กลับด้วยโมเมนตัมเป็นกลยุทธที่ชาญฉลาดวิธีหนึ่ง จากการศึกษาพฤติกรรมของการลู่เข้าของอัลกอริทึมแพร่กลับของพื้นผิวค่าความผิดพลาด เราจะเห็นได้ว่าในช่วงที่พื้นผิวค่าความผิดพลาดราบเรียบ ค่าคงที่การเรียนรู้ควรจะมีค่ามากเพื่อให้อัลกอริทึมสามารถผ่านช่วงพื้นที่ดังกล่าวไปได้อย่างรวดเร็ว ในขณะเดียวกันในช่วงที่พื้นผิวมีความชันสูง (ลักษณะเป็นหน้าผา) ค่าคงที่การเรียนรู้ควรจะต้องน้อย เพื่อไม่ให้เกิดการแกว่งรอบๆ พื้นที่ดังกล่าว ในการป้องกันไม่ให้เกิดการแกว่งรอบๆ พื้นที่ดังกล่าว เราอาจทำให้การลู่เข้าของอัลกอริทึมเป็นไปอย่างราบเรียบ (smooth) โดยการใช้ตัวกรองความถี่ต่ำผ่าน (low-pass filter) เพื่อเพิ่มประสิทธิภาพในการลู่เข้าหาคำตอบของอัลกอริทึม ตัวอย่างของตัวกรองความถี่ต่ำผ่านมีแสดงในสมการที่ (12.48)

$$y(k) = \eta y(k-1) + (1-\eta)x(k) \tag{12.48}$$

โดยที่ $x(k)$ คืออินพุตของตัวกรองและ $0 \leq \eta \leq 1$ คือค่าคงที่โมเมนตัม รูปที่ 12.22 แสดงตัวอย่างการกรองที่มีค่าคงที่โมเมนตัมเท่ากับ 0.2 0.4 0.7 และ 0.9 ตามลำดับ จากรูปจะเห็นได้ว่าค่าคงที่โมเมนตัมมากจะให้เอาต์พุตของตัวกรองที่เรียบกว่าด้วย

เมื่อพิจารณาการปรับค่าเมตริกซ์น้ำหนักประสาทและไบอัส ที่ซึ่งในบางกรณีเกิดการแกว่ง (หรือไม่เรียบ) ทำให้การลู่เข้าของการฝึกสอนไม่มีเสถียรภาพ การใช้ตัวกรองความถี่ต่ำผ่านสามารถนำมาแก้ปัญหาดังกล่าวได้ในระดับหนึ่ง ดังนั้นกฎการเรียนรู้ในการปรับค่าเมตริกซ์น้ำหนักประสาทแบบแพร่กลับสามารถถูกปรับปรุงด้วยการใช้โมเมนตัม เรียกว่าเป็นอัลกอริทึมแพร่กลับพร้อมโมเมนตัม โดยมีการปรับค่าดังนี้ (พิจารณาเปรียบเทียบกับสมการที่ (12.36) และ (12.37) ของการฝึกสอนแบบแพร่กลับดั้งเดิม)

$$w_{ij}^{l}(k+1) = \eta w_{ij}^{l}(k) - (1-\eta)\alpha\delta_i^{l}(k)y_j^{l-1}(k) \tag{12.49}$$



รูปที่ 12.17: (ก)-(ข) พื้นผิวและเส้นโครงร่างของค่าความผิดพลาดผลรวมกำลังสองระหว่าง w^1_{11} และ w^2_{11} (ค)-(ง) พื้นผิวและเส้นโครงร่างของค่าความผิดพลาดผลรวมกำลังสองระหว่าง w^1_{11} และ b^1_1 (จุดเครื่องหมาย + แสดงจุดที่พื้นผิวมีค่าน้อยที่สุด จุดดังกล่าวถือว่าเป็นจุดที่เครือข่ายควรต้องปรับค่าการเรียนรู้ให้ลู่เข้าหาในที่สุด)

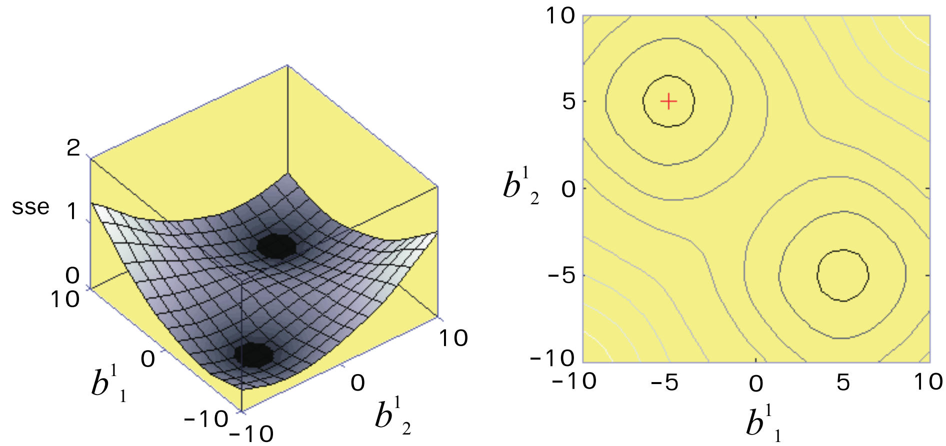
และ

$$b^l_i(k+1) = \eta b^l_i(k) - (1-\eta)\alpha\delta^l_i(k) \tag{12.50}$$

อัลกอริทึมแพร่กลับพร้อมโมเมนตัมทำให้การเรียนรู้ของเครือข่าย (นั่นคือการปรับค่าเมตริกซ์น้ำหนักประสาทและไบอัส) มีความเรียบ ลดการแกว่งรอบๆ คำตอบ ทำให้การลู่เข้าของเครือข่ายมีประสิทธิภาพดีกว่าอัลกอริทึมแพร่กลับแบบลงชันสุด รูปที่ 12.23 แสดงผลของการเพิ่มโมเมนตัมให้กับอัลกอริทึมแพร่กลับแบบลงชันสุด สังเกตว่าค่าของโมเมนตัมที่เหมาะสมเท่านั้นที่นำไปสู่การลู่เข้าอย่างถูกต้องได้

12.5.3 อัตราการเรียนรู้ที่ปรับค่าได้
Variable Learning Rate

ค่าคงที่การเรียนรู้มีผลโดยตรงต่อประสิทธิภาพการลู่เข้าของการฝึกสอน ถ้าพื้นผิวค่าความผิดพลาดมีความราบเรียบ ค่าคงที่การเรียนรู้ควรจะมีค่ามาก เมื่อพื้นผิวมีความชันมากขึ้น ค่าคงที่การเรียนรู้ควรจะน้อยลง จึงได้มีการ



(จ) (ฉ)

รูปที่ 12.18: (ต่อ) (จ)-(ฉ) พื้นผิวและเส้นแสดงรูปร่างค่าความผิดพลาดผลรวมกำลังสองระหว่าง b_1^1 และ b_2^1 (จุดเครื่องหมาย + แสดงจุดที่พื้นผิวมีค่าน้อยที่สุด)

ออกแบบอัลกอริทึมในการปรับให้ค่าคงที่การเรียนรู้เปลี่ยนแปลงไปตามพื้นผิวค่าความผิดพลาดดังกล่าว วิธีในการปรับค่าคงที่การเรียนรู้มีอยู่หลายวิธี เราจะศึกษาวิธีการตรงในการปรับค่าคงที่การเรียนรู้ โดยใช้ค่าวัดประสิทธิภาพของเครือข่ายเป็นตัวชี้ว่าควรปรับค่าคงที่การเรียนรู้อย่างไร ดังรายละเอียดต่อไปนี้

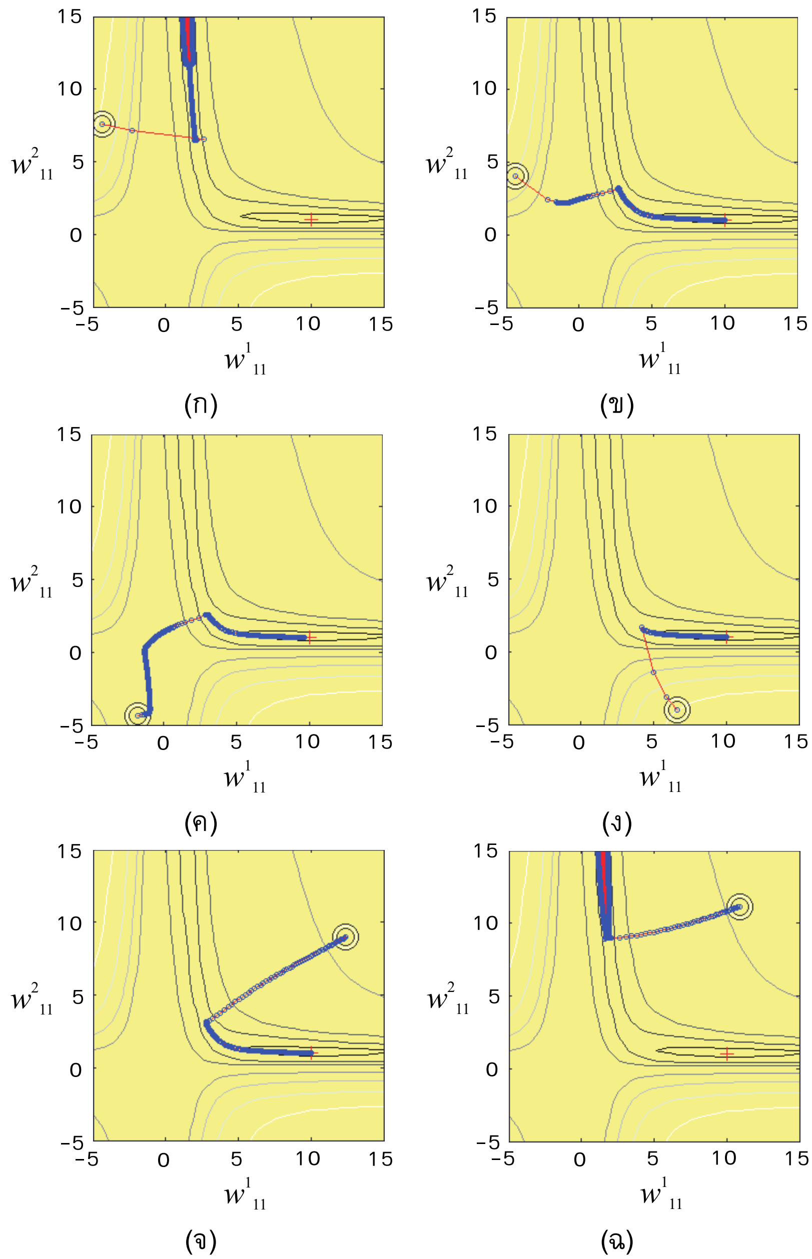
▷ **อัลกอริทึมปรับค่าอัตราการเรียนรู้**

1. หลังการปรับค่าน้ำหนักประสาทและไบอัส ถ้าค่าความผิดพลาดรวมของคู่อินพุต-เป้าหมายทั้งชุดมีค่าเพิ่มขึ้นเกิน ζ เปอร์เซ็นต์ ให้**งด**การปรับค่าน้ำหนักประสาทและไบอัส ค่าคงที่การเรียนรู้ α จะถูกลดค่าลงโดยการคูณกับตัวประกอบ $0 < \sigma < 1$ ให้ค่าคงที่โมเมนตัมเท่ากับศูนย์
2. ถ้าค่าความผิดพลาดลดลงหลังจากปรับค่าน้ำหนักประสาทและไบอัสแล้ว ให้คงการปรับค่าน้ำหนักประสาทและไบอัสนั้นไว้ พร้อมกับเพิ่มค่าคงที่การเรียนรู้โดยการคูณด้วยตัวประกอบ $\sigma > 1$ ปรับค่าคงที่โมเมนตัมกลับสู่สถานะเดิม
3. ถ้าค่าความผิดพลาดเพิ่มขึ้นแต่ไม่มากกว่า ζ เปอร์เซ็นต์ ให้คงการปรับค่าน้ำหนักประสาทและไบอัสไว้ แต่ไม่ต้องปรับค่าคงที่การเรียนรู้และค่าคงที่โมเมนตัม ◁

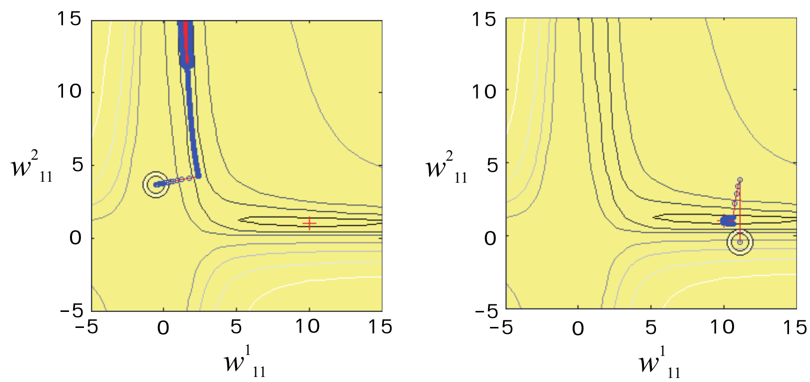
ค่าเปอร์เซ็นต์ ζ เปรียบเสมือนการป้องกันการพุ่งเกิน (overshoot) ของการเรียนรู้ โดยทำการลดค่าคงที่การเรียนรู้ลงในขณะที่การปรับน้ำหนักประสาทและไบอัสจะถูกละเว้น การเพิ่มค่าคงที่การเรียนรู้จะเกิดขึ้นในกรณีที่ค่าผิดพลาดมีขนาดลดลง ซึ่งเรามั่นใจแล้วว่าการเรียนรู้เป็นไปในทิศทางที่ถูกต้อง การเพิ่มค่าคงที่การเรียนรู้จึงเป็นการเพิ่มความเร็วในการลู่เข้านั่นเอง

12.6 สรุป

ในบทนี้เราได้ทำการศึกษาเครือข่ายไปข้างหน้าแบบหลายชั้น รวมไปถึงกฎการเรียนรู้แบบแพร่กลับ เครือข่ายหลายชั้นถือว่าเป็นเครือข่ายที่มีประสิทธิภาพเหนือกว่าเครือข่ายชั้นเดียว โดยเฉพาะการแก้ปัญหาที่แบ่งแยกได้แบบไม่

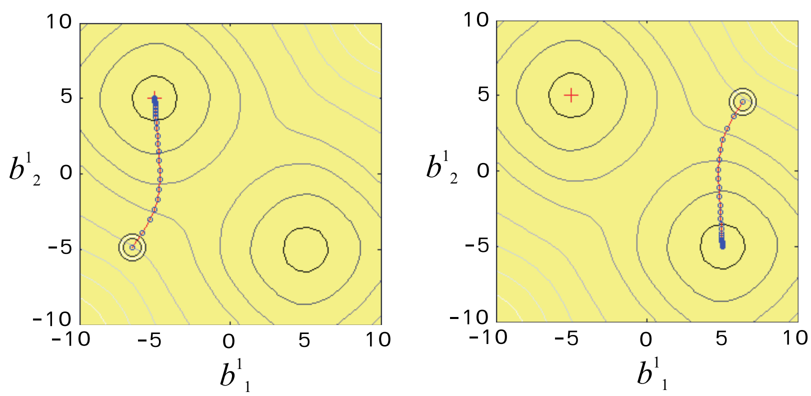


รูปที่ 12.19: การลู่เข้าของค่าน้ำหนักประสาทเริ่มต้นจากตำแหน่งต่างๆ (วงกลมสองวงซ้อนกันในรูปเป็นจุดเริ่มต้นที่ได้จากการสุ่มค่าน้ำหนักประสาทก่อนทำการฝึกสอน สายวงกลมเล็กๆ เป็นตำแหน่งของน้ำหนักประสาทที่ผ่านการฝึกสอนในแต่ละรอบ ทิศทางของสายวงกลมเล็กๆ นี้แสดงทิศทางการปรับตัวของน้ำหนักประสาทที่ได้จากการฝึกสอนเครือข่าย โดยมีเส้นสีแดงแสดงแทนทิศทางดังกล่าว)



(ช) (ซ)

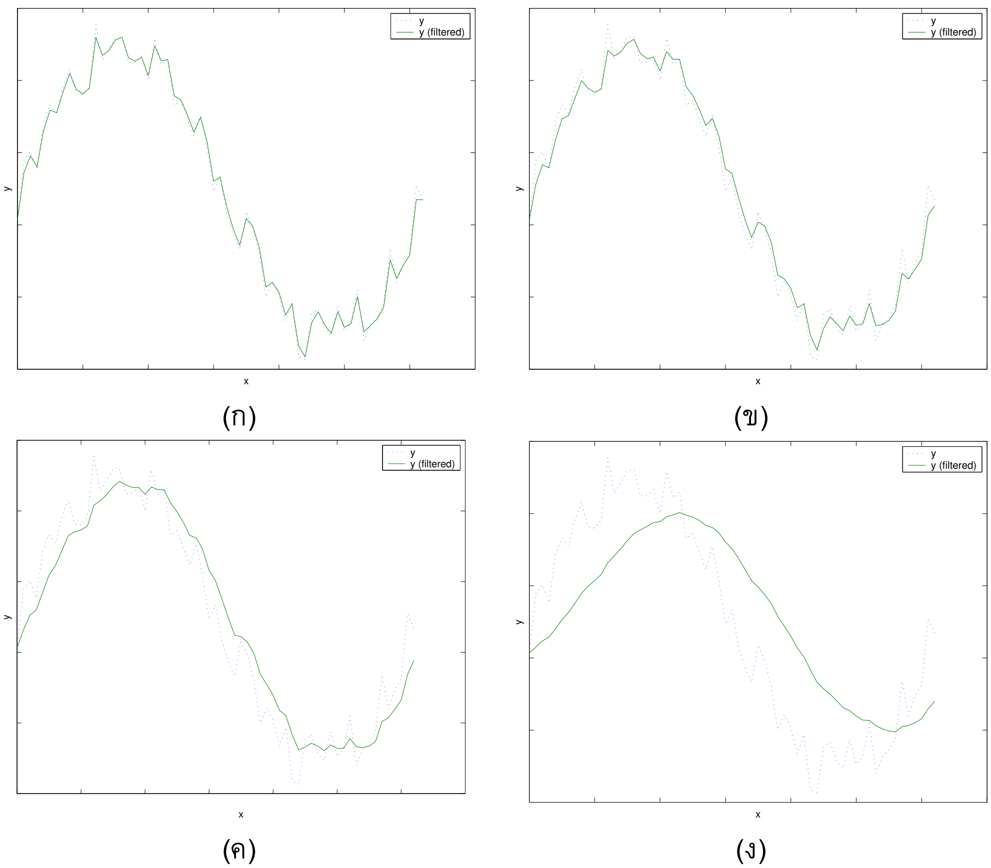
รูปที่ 12.20: (ต่อ) การลู่เข้าของค่าเริ่มต้นจากตำแหน่งต่างๆ



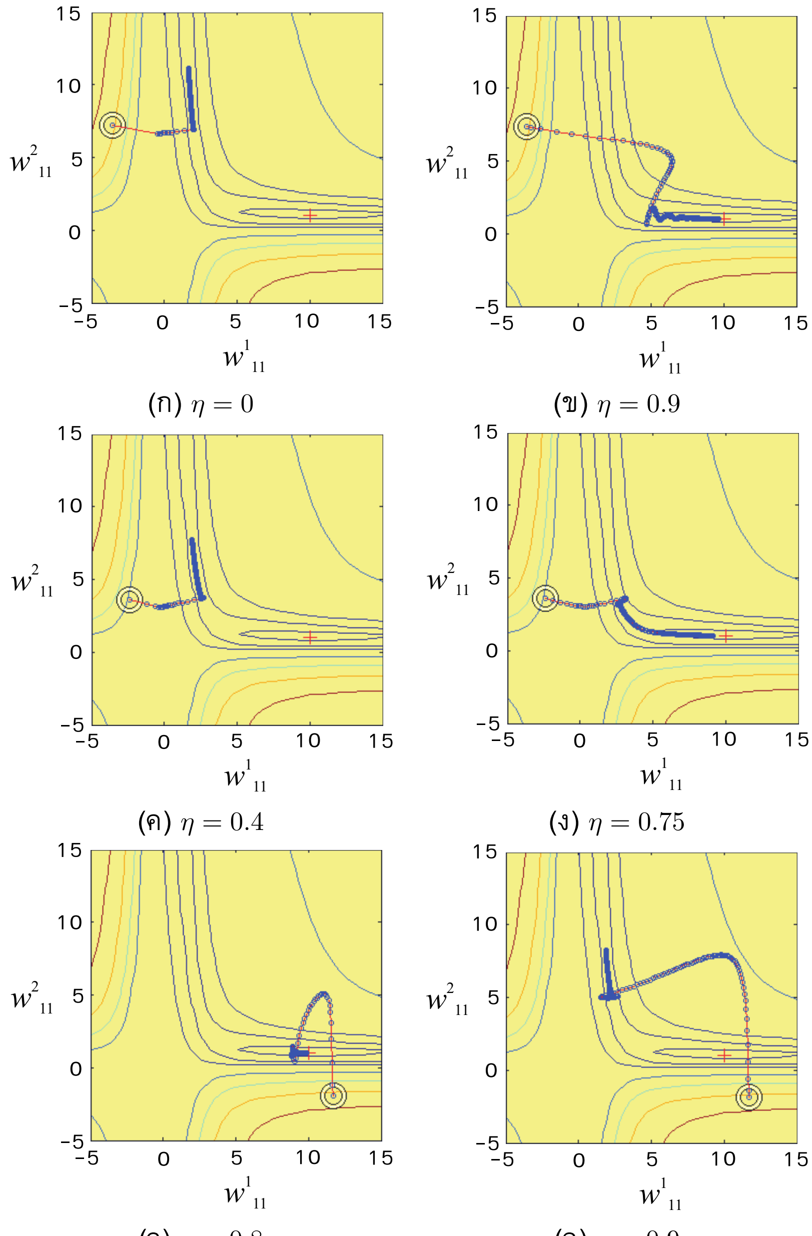
(ก) (ข)

รูปที่ 12.21: การลู่เข้าของค่าเริ่มต้นจากตำแหน่งต่างๆ ของพื้นผิวระหว่าง b^1_1 และ b^1_2 (วงกลมสองวงซ้อนกันในรูปเป็นจุดเริ่มต้นที่ได้จากการสุ่มค่าไบอัสก่อนทำการฝึกสอน สายวงกลมเล็กๆ เป็นตำแหน่งของไบอัสที่ผ่านการฝึกสอนในแต่ละรอบ ทิศทางของสายวงกลมเล็กๆ นี้แสดงทิศทางการปรับตัวของไบอัสที่ได้จากการฝึกสอนเครือข่าย โดยมีเส้นสีแดงแสดงแทนทิศทางดังกล่าว)

เป็นเชิงเส้น ที่ซึ่งไม่สามารถจัดการได้ด้วยเครือข่ายเพียงชั้นเดียว การเรียนรู้แบบแพร่กลับถือเป็นอัลกอริทึมแรกที่พัฒนาต่อยอดมาจากอัลกอริทึม LMS และสามารถใช้กับเครือข่ายแบบหลายชั้นได้อย่างมีประสิทธิภาพ ชื่อการเรียนรู้แบบแพร่กลับได้มาจากลักษณะการแพร่กลับของค่าความผิดพลาด จากชั้นเอาต์พุตของเครือข่าย กลับไปยังชั้นแรกสุดของเครือข่าย จนกระทั่งค่าความผิดพลาดมีค่าในย่านที่ต้องการ ทั้งการเรียนรู้แบบแพร่กลับและ LMS ใช้หลักของอัลกอริทึมลงแบบชันสุด เพื่อลดค่าความผิดพลาดแบบกำลังสองเฉลี่ยของเครือข่าย จนกระทั่งลู่เข้าสู่คำตอบที่เหมาะที่สุด เทคนิคการปรับแต่งอัลกอริทึมแบบแพร่กลับได้ถูกนำเสนออย่างมากมาย โดยมีจุดประสงค์หลักเพื่อที่จะทำให้ความเร็วของการเรียนรู้เพิ่มขึ้น และสามารถลู่เข้าสู่คำตอบที่เหมาะที่สุดแบบวงกว้าง ทำให้เครือข่ายแบบแพร่กลับกลายเป็นเครื่องมือสำคัญในการแก้ปัญหาต่างๆ ได้อย่างทรงประสิทธิภาพที่สุดอย่างหนึ่ง



รูปที่ 12.22: เอาต์พุตของตัวกรองความถี่ต่ำผ่านที่มีค่าคงที่โมเมนตัมเท่ากับ (ก) 0.2 (ข) 0.4 (ค) 0.7 และ (ง) 0.9 (เส้นประแสดงกราฟของฟังก์ชันก่อนทำการปรับด้วยโมเมนตัม และเส้นทึบแสดงกราฟของฟังก์ชันที่ผ่านการปรับด้วยโมเมนตัม)



(ก) $\eta = 0$ (ข) $\eta = 0.9$
(ค) $\eta = 0.4$ (ง) $\eta = 0.75$
(จ) $\eta = 0.8$ (ฉ) $\eta = 0.9$

รูปที่ 12.23: การลู่เข้าของอัลกอริทึมแพร่กลับแบบลงชันสุดพร้อมด้วยโมเมนตัม (วงกลมเล็กซ้อนกันสองวงเป็นจุดเริ่มต้นของคำตอบ สายวงกลมเล็กๆ แทนตำแหน่งของค่าน้ำหนักประสาทและไบอัสที่ได้รับการปรับตามการเรียนรู้แบบแพร่กลับ เครื่องหมาย + แสดงค่าน้อยที่สุดของพื้นผิวค่าความผิดพลาดของเครือข่าย ซึ่งเป็นตำแหน่งที่ต้องการของค่าน้ำหนักประสาทและไบอัสที่ผ่านการฝึกสอน)

โจทย์คำถาม

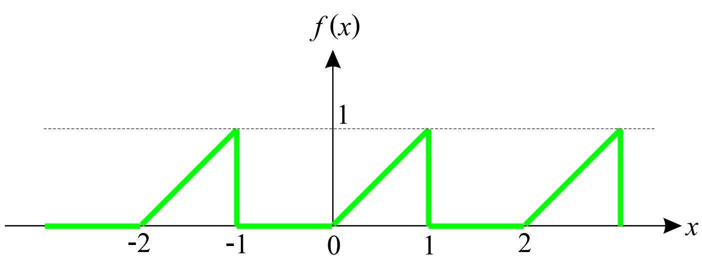
12.1. การประมาณค่าฟังก์ชันแบบไม่เป็นเชิงเส้น ถือเป็นการประยุกต์ใช้งานที่มีการนำเอาเครือข่ายแบบแพร่กลับมาใช้มากที่สุดอย่างหนึ่ง จงออกแบบเครือข่ายแบบแพร่กลับเพื่อทำการประมาณค่าฟังก์ชันต่อไปนี้

- $f(x) = \frac{1}{x^2}$ โดยที่ $x \in [-1, 1]$
- $f(x, y) = ax^2 + by^2$ โดยที่ a และ b เป็นค่าคงที่แบบจำนวนจริงบวกใดๆ และ $x, y \in [-1, 1]$
- $f(x) = \frac{\sin x}{x}$ โดยที่ $x \in [-2, 2]$
- $f(x, y, z) = ax^2 + by^2 + cz^2$ โดยที่ a b และ c เป็นค่าคงที่แบบจำนวนจริงบวกใดๆ และ $x, y, z \in [-1, 1]$

สำหรับแต่ละฟังก์ชันข้างต้น พิจารณาวิเคราะห์เครือข่ายดังรายละเอียดต่อไปนี้

- เลือกพารามิเตอร์ของเครือข่ายที่แตกต่างกัน เช่นชนิดของฟังก์ชันถ่ายโอน จำนวนนิวรอนหรือจำนวนชั้น ฯลฯ
- ใช้จำนวนการชักตัวอย่างของฟังก์ชันที่แตกต่างกัน พร้อมเปรียบเทียบผลที่ได้
- ทดสอบการทำให้เป็นกรณีทั่วไปของเครือข่าย โดยเลือกใช้ข้อมูลการชักตัวอย่างจากฟังก์ชันที่ไม่ได้เป็นข้อมูลที่ใช้ฝึกสอนเครือข่าย

12.2. จงออกแบบเครือข่ายพร้อมฝึกสอนเครือข่ายให้เรียนรู้การประมาณค่าฟังก์ชันในรูปที่ 12.24 โดยใช้ค่าชักตัวอย่างจากอนุกรมฟูริเยร์ (ประมาณ 2-3 เทอมแรก) และใช้ค่าชักตัวอย่างจากรูปคลื่นสามเหลี่ยมโดยตรง อธิบายรายละเอียดการออกแบบ ทดลองปรับพารามิเตอร์ของเครือข่าย วิเคราะห์และสรุปผลการทดลองที่ได้



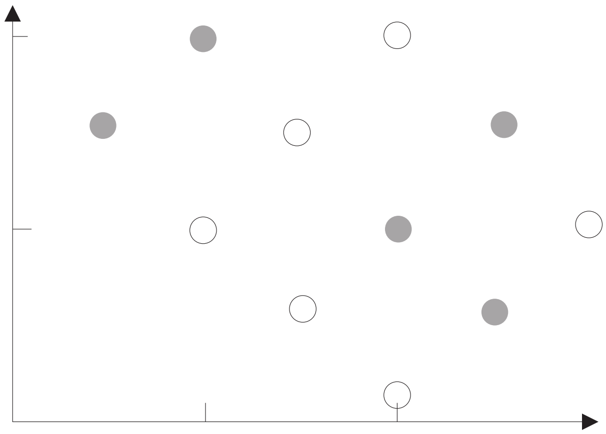
รูปที่ 12.24: รูปคลื่นสามเหลี่ยม

12.3. จากปัญหาการประมาณค่าฟังก์ชันข้างต้น จงเปรียบเทียบและวิเคราะห์ผลจากการใช้ฟังก์ชันถ่ายโอนชนิดต่างๆ (โดยเฉพาะฟังก์ชันฟังก์ชันฐานรัศมี ฟังก์ชันซิกมอยแบบลอการิทึมและแบบเส้นสัมผัสไฮเปอร์โบลาร์)

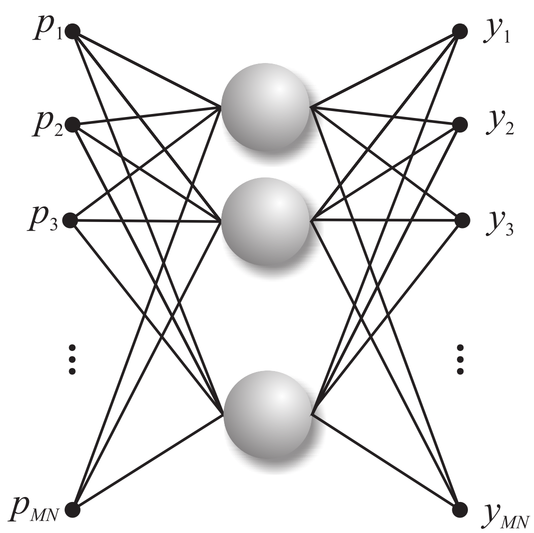
12.4. จงออกแบบเครือข่ายแบบแพร่กลับสำหรับแก้ปัญหาตัวปฏิบัติการ XOR พร้อมทั้งวิเคราะห์เครือข่ายดังนี้

- เลือกโครงสร้างของเครือข่ายแบบต่างๆ ด้วยจำนวนชั้นและจำนวนนิวรอนที่แตกต่างกัน
- เลือกพารามิเตอร์ของเครือข่ายพร้อมทั้งแสดงผลลัพธ์และเส้นแบ่งพื้นที่ที่ได้
- แสดงพื้นผิวค่าความผิดพลาดของตัวแปรค่าน้ำหนักประสาทและไบอัสที่ใช้
- ปรับเปลี่ยนอัลกอริทึมการฝึกสอน เปรียบเทียบประสิทธิภาพของอัลกอริทึมที่ใช้

12.5. เครือข่ายหลายชั้นสามารถรองรับปัญหาที่ไม่สามารถแบ่งแยกได้แบบเชิงเส้น (nonlinearly separable) จงออกแบบเครือข่ายพร้อมฝึกสอนเพื่อทำการคัดแยกรูปแบบในรูปที่ 12.25 อภิปรายและวิเคราะห์ผลที่ได้จากเครือข่ายที่ออกแบบ



รูปที่ 12.25: รูปแบบที่ไม่สามารถแบ่งแยกได้เชิงเส้น



รูปที่ 12.26: รูปแบบที่ไม่สามารถแบ่งแยกได้เชิงเส้น

12.6. เครือข่ายประสาทเทียมแบบแพร่กลับสามารถนำมาประยุกต์ใช้ในการบีบอัดข้อมูลภาพได้ พิจารณาออกแบบเครือข่ายแบบแพร่กลับสำหรับใช้บีบอัดข้อมูลภาพดังรายละเอียดต่อไปนี้ (ในรูปที่ 12.26)

- ชั้นอินพุตรับข้อมูลภาพที่ผ่านการแปลงข้อมูล 2 มิติให้อยู่ในรูปเวกเตอร์ 1 มิติ ดังนั้นสำหรับข้อมูลภาพขนาด $M \times N$ จุดภาพ (M และ N คือขนาดความกว้างและความสูงของภาพในหน่วยจุดภาพ) จะถูกแปลงเป็นข้อมูลเวกเตอร์ขนาด $MN \times 1$
- ชั้นเอาต์พุตมีจำนวนนิวรอนเท่ากับชั้นอินพุต
- นิวรอนในชั้นซ่อนเร้นมีจำนวนน้อยกว่าในชั้นอินพุตหรือเอาต์พุต
- ค่าน้ำหนักประสาทในชั้นซ่อนเร้นจะเป็นตัวแทนการบีบอัดของข้อมูลภาพ กล่าวคือข้อมูลภาพถูกบีบอัดให้กลายเป็นค่าน้ำหนักประสาทของเครือข่าย ซึ่งมีจำนวนน้อยกว่าขนาดของภาพต้นฉบับ
- การเรียนรู้ของเครือข่ายมีจุดประสงค์เพื่อให้เอาต์พุตมีค่าใกล้เคียงอินพุตให้ได้มากที่สุด จำนวนนิวรอนในชั้นซ่อนเร้นที่น้อย อัตราส่วนการบีบอัดข้อมูลจะมาก (นั่นคือรูปภาพที่เอาต์พุตจะเหมือนกับรูปภาพอินพุตน้อยลง)
- วิเคราะห์และอภิปรายผลการบีบอัดข้อมูลภาพ

บรรณานุกรม

Y. LeCun. Une procedure d'apprentissage pour reseau á seuil asymmetrique. In *In Cognitiva 85: A la Frontiere de l'Intelligence Artificielle des Sciences de la Connaissance des Neurosciences, CESTA*, volume 85, pages 599--604, Paris, 1985.

D. B. Parker. Learning - logic. Invention report: S81-64, file 1, Stanford University, Office of Technology Licensing, Stanford, CA, October 1982.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, Cambridge, 1986a. MA: M.I.T.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagation errors. In *Nature*, volume 323, pages 533--536, 1986b.

P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, 1974.