บทที่ 3

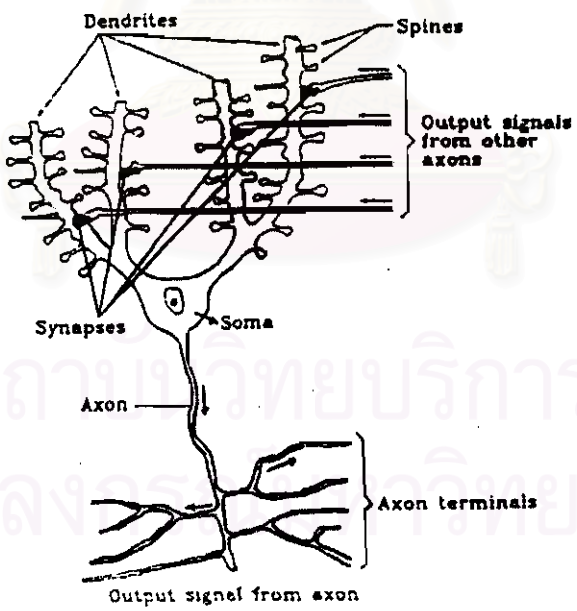
นิวรอลเนตเวิร์กและ Backpropagation Algorithm

3.1 กล่าวนำ

ในทางประสาทวิทยาทราบว่าสมองของมนุษย์ประกอบด้วยเซลล์ประสาท (neuron cell) จำนวน 10^{10} ถึง 10^{11} เซลล์ [5] ซึ่งโครงสร้างของเซลล์ประสาทประกอบด้วย

1. Soma เป็นตัวเซลล์ประสาท
2. Axon เป็นเส้นใยประสาทที่ต่อออกจากเซลล์ประสาทโดยทำหน้าที่เป็นด้านออกของเซลล์ประสาทเพื่อส่งต่อไปยังเซลล์ประสาทอื่น
3. Dendrites ทำหน้าที่เป็นด้านเข้าของเซลล์ประสาทเพื่อรับสัญญาณจากเซลล์ประสาทตัวอื่น โดยผ่าน axon ซึ่งต่อกับ synapes เพื่อส่งไปยัง soma
4. Synapses เป็นตัวต่อ dendrite กับ axon จากเซลล์ประสาทอื่น

โครงสร้างของเซลล์ประสาทนี้แสดงได้ดังรูปที่ 3.1

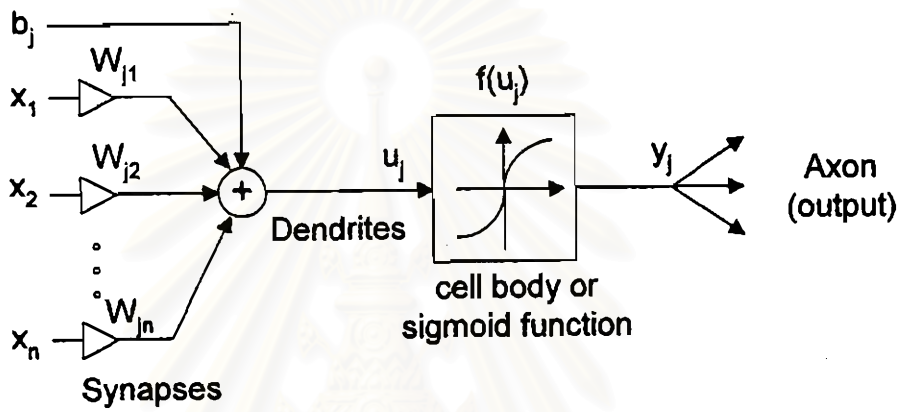


รูปที่ 3.1 โครงสร้างของเซลล์ประสาททางชีวภาพ

การทำงานของโครงข่ายประสาทจะทำงานกันแบบขนานไปพร้อม ๆ กัน ในแต่ละเซลล์ประสาท จึงทำให้โครงข่ายประสาทสามารถทำการประมวลผลได้อย่างรวดเร็ว ด้วยเหตุนี้จึงได้มีการพยายามจำลองโครงข่ายประสาทขึ้นเพื่อใช้ในการแก้ปัญหาต่าง ๆ

3.2 แบบจำลองโครงข่ายประสาทเทียม

นิวรอลเนตเวิร์กหรือโครงข่ายประสาทเทียม (Artificial neural networks) คือโครงข่ายที่สร้างขึ้นเพื่อเลียนแบบการทำงานของโครงข่ายประสาทจริง ๆ กล่าวโดยทั่วไปแล้วโครงข่ายประสาทเทียมคือระบบประมวลสัญญาณที่ประกอบด้วยตัวประมวลผลอย่างง่าย ๆ จำนวนมาก ซึ่งเรียกหน่วยประมวลผลเหล่านี้ว่า "นิวรอน (neuron)" มาต่อเข้าด้วยกันเป็นโครงข่าย ซึ่งการทำงานของโครงข่ายนี้จะกระทำแบบขนานไปพร้อม ๆ กันในแต่ละนิวรอนเพื่อแก้ปัญหาที่ต้องการ โดยแบบจำลองพื้นฐานของนิวรอนแสดงได้ดังรูปที่ 3.2



รูปที่ 3.2 แบบจำลองพื้นฐานของเซลล์ประสาทเทียม

จากรูปที่ 3.2 จะเห็นว่าค่าถ่วงน้ำหนัก (weight , w_{ji}) หน้าที่เสมือนกับ synapses เพื่อต่อด้านเข้า x_i เข้าสู่ตัวรวมซึ่งทำหน้าที่เหมือนกับ dendrites เพื่อส่งสัญญาณเข้าสู่ตัว activation function ($f(u_j)$) ซึ่งเปรียบเสมือนกับ soma ส่วน b_j (bias) เป็นค่าระดับอ้างอิงที่ป้อนจากภายนอก ผลของการประมวลผล y_j จะถูกส่งออกที่ด้านออกของตัวขยายแบบไม่เป็นเชิงเส้นซึ่งเปรียบเสมือนกับ axon สามารถเขียนสมการได้เป็น

$$y_j = f(\sum_{i=1}^{n} w_{ji}x_i + b_j) \qquad (3.1)$$

ค่าระดับอ้างอิงจากภายนอก b_j สามารถเขียนในรูปของค่าถ่วงน้ำหนักได้เป็น $w_{j0} = b_j$ และ $x_0 = 1$ จึงทำให้สมการที่ (3.1) เขียนใหม่ได้เป็น

$$y_j = f(\sum_{i=0}^{n} w_{ji}x_i) \qquad (3.2)$$

3.3 การจำแนกประเภทของนิวรอลเนตเวิร์กตามลักษณะการเรียนรู้

การเรียนรู้ของนิวรอลเนตเวิร์ก จะมีประสิทธิภาพเพียงใดนั้นขึ้นอยู่กับค่าถ่วงน้ำหนักของโครงข่าย ซึ่งการฝึก (Training) โครงข่ายก็คือการหาค่าถ่วงน้ำหนักที่เหมาะสมให้กับโครงข่ายนั้น ๆ วิธีการฝึกนิวรอลเนตเวิร์กมีอยู่ 2 แบบด้วยกันคือ

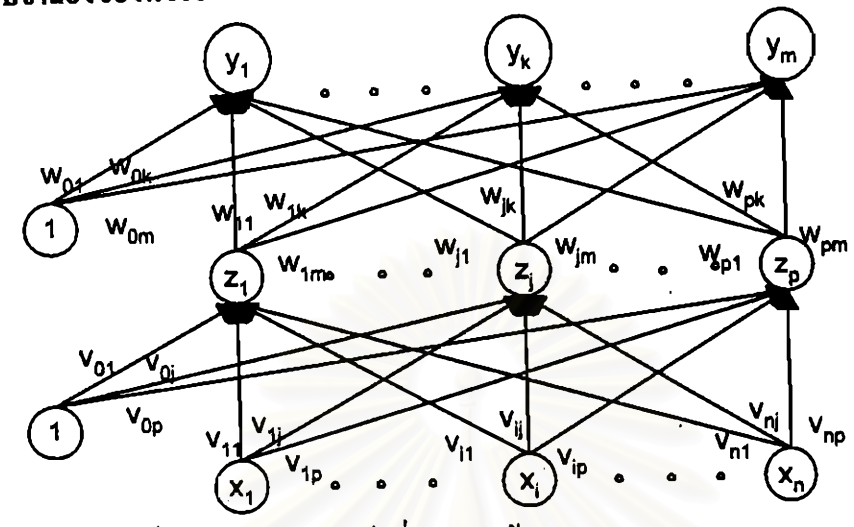
1. การเรียนรู้แบบชี้นำหรือดูแล (Supervised Learning) การเรียนรู้โดยวิธีนี้จะกำหนดเซตของการฝึกให้กับโครงข่ายซึ่งเซตนี้ประกอบด้วยอินพุตและเอาต์พุตที่ต้องการ เมื่อป้อนอินพุตให้กับโครงข่าย โครงข่ายก็จะทำการประมวลผลจนได้คำตอบและค่าถ่วงน้ำหนักออกมาชุดหนึ่ง สำหรับคำตอบที่ได้จะถูกนำมาคำนวณค่าความผิดพลาดโดยวัดเป็นระยะทางว่ามีความห่างจากคำตอบที่ต้องการของอินพุตในชุดเดียวกันมากน้อยเพียงใด ถ้ายังมีความผิดพลาดสูงอยู่ก็จะมีการปรับค่าถ่วงน้ำหนักและทำการฝึกต่อไปจนกว่าค่าความผิดพลาดมีค่าน้อยพอที่จะยอมรับได้จึงจะหยุดการฝึก

2. การเรียนรู้แบบไม่มีการชี้นำหรือไม่มีการดูแล (Unsupervised Learning) การเรียนรู้โดยวิธีนี้จะป้อนอินพุตเข้าสู่โครงข่าย และภายในโครงข่ายจะมีเอาต์พุตโนดอยู่หลายโนดด้วยกันโดยแต่ละโนดจะแทนกลุ่มของข้อมูลที่มีคุณสมบัติเหมือนกัน เมื่อป้อนอินพุตเข้าสู่โครงข่าย โครงข่ายจะคำนวณค่าความสัมพันธ์ที่มีภายในเซตของอินพุตโดยอาศัยค่าถ่วงน้ำหนักเป็นตัวแยกความแตกต่างของอินพุตไปเก็บไว้ในโนดเอาต์พุตของโครงข่าย การเรียนรู้โดยวิธีนี้จะไม่สามารถระบุได้ว่าเอาต์พุตโนดใดเป็นของข้อมูลกลุ่มไหนซึ่งผู้ใช้จะต้องกำหนดเอง

วิทยานิพนธ์ฉบับนี้เสนอนิวรอลเนตเวิร์กที่มีโครงสร้างแบบ Feedforward (Multilayer Perceptron) โดยใช้ Backpropagation Algorithm ซึ่งมีการเรียนรู้แบบมีการชี้นำ ในการจัดสรรแบนด์วิดท์สมมูลสำหรับแหล่งกำเนิดวิวิธพันธุ์ให้กับผู้ใช้ในโครงข่าย ATM ซึ่งต้องการการตอบสนองที่เร็วในขณะเซตอัปการต่อ และจากคุณสมบัติของนิวรอลเนตเวิร์กที่มีการประมวลผลแบบขนาน (parallel processing) ไปพร้อม ๆ กันในนิวรอนแต่ละตัว จึงทำให้สามารถประมวลผลได้อย่างรวดเร็ว ดังนั้นจึงนำนิวรอลเนตเวิร์กโดยใช้ Backpropagation Algorithm มาช่วยแก้ไขปัญหาที่ไม่เป็นเชิงเส้นในการหาค่าแบนด์วิดท์สมมูลในโครงข่ายเอทีเอ็มที่ต้องการความเป็นเวลาจริง (realtime)

3.4 หลักการพื้นฐานของ Backpropagation Algorithm

3.4.1 แบบจำลองของนิวรอลเนตเวิร์กที่มีโครงสร้างแบบ Feedforward



รูปที่ 3.3 นิวรอลเนตเวิร์กที่มีโครงสร้างแบบ Feedforward

Multilayer Neural Network ที่มี 1 hidden layer แสดงดังรูปที่ 3.3 โดยที่ตัวแปร x แทนอินพุตโนด ,ตัวแปร z แทนฮิดเดนโนด และตัวแปร y แทนเอาต์พุตโนด จากรูป ฮิดเดนโนดและเอาต์พุตโนดจะมี bias คือ bias ที่เอาต์พุตโนด y_k กำหนดเป็น w_{0k} และ bias ที่ฮิดเดนโนด z_j กำหนดเป็น v_{0j} ตามลำดับ

3.4.2 อัลกอริทึม

จากที่ได้กล่าวมาแล้วเกี่ยวกับโครงข่ายเซลล์ประสาททั้งที่เป็นแบบจำลองและเป็นโครงข่ายของเซลล์ประสาทจริงของมนุษย์ จะมีการต่อถึงกันของโนดในลักษณะของโครงข่ายอย่างหนาแน่น เพื่อให้โครงข่ายสามารถเรียนรู้และสามารถจดจำสิ่งที่เรียนรู้มาแล้วได้ เราต้องมีอัลกอริทึมการฝึก (training algorithm) ให้กับโครงข่ายโดยแบ่งเป็น 3 ขั้นตอนคือ [6]

1. Feedforward of input training pattern
2. Backpropagation of error
3. ปรับค่าถ่วงน้ำหนัก (Adjustment of weights)

ก่อนที่จะกล่าวถึงอัลกอริทึมการฝึกอย่างละเอียดจะกำหนดตัวแปรที่ใช้ดังต่อไปนี้

x Input training vector

$x = (x_1,\ldots,x_i,\ldots,x_n)$

t Output target vector

$t = (t_1,...,t_k,...,t_m)$

δ_k Error term ของเอาต์พุตโนด เพื่อนำไปปรับค่าถ่วงน้ำหนักระหว่างฮิดเดนเลเยอร์ไปเอาต์พุตเลเยอร์

δ_j Error term ของฮิดเดนโนด เพื่อนำไปปรับค่าถ่วงน้ำหนักระหว่างอินพุตเลเยอร์ไปฮิดเดนเลเยอร์

λ อัตราการเรียนรู้ (learning rate)

v_{0j} bias ของฮิดเดนโนด (z_j)

z_j ฮิดเดนโนดที่ j โดยกำหนดอินพุตเป็น z_in_j

$$z\_in_j = v_{0j} + \sum_i x_i v_{ij}$$

และค่าเอาต์พุตเมื่อผ่าน activation function โดยกำหนดเอาต์พุตเป็น z_j

$$z_j = f(z\_in_j)$$

w_{0k} bias ของเอาต์พุตโนด (y_k)

y_k เอาต์พุตโนดที่ k โดยกำหนดอินพุตเป็น y_in_k

$$y\_in_k = w_{0k} + \sum_j z_j w_{jk}$$

และค่าเอาต์พุตเมื่อผ่าน activation function โดยกำหนดเอาต์พุตเป็น y_k

$$y_k = f(y\_in_k)$$

3.4.3 Activation function

ในที่นี้จะกล่าวถึง activation function ที่นิยมใช้ 3 ชนิด [6] คือ

1. Identity function

$$f(x) = x \qquad for \quad all \quad x$$
$$f'(x) = 1 \qquad (3.3)$$

ฟังก์ชันนี้แสดงดังรูปที่ 3.4(ก)

2. Binary sigmoid function

เอาต์พุตมีค่าอยู่ระหว่างช่วง (0,1) กำหนดเป็น

$$f_1(x) = \frac{1}{1+\exp(-x)} \qquad (3.4)$$

$$f_1'(x) = f_1(x)[1 - f_1(x)]$$

ฟังก์ชันนี้แสดงดังรูปที่ 3.4(ข)

3. Bipolar sigmoid function

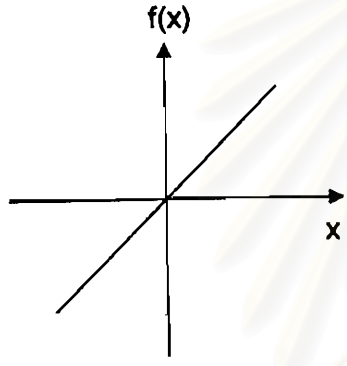
เอาต์พุตมีค่าอยู่ระหว่างช่วง (-1,1) กำหนดเป็น

$$f_2(x) = 2f_1(x) - 1$$

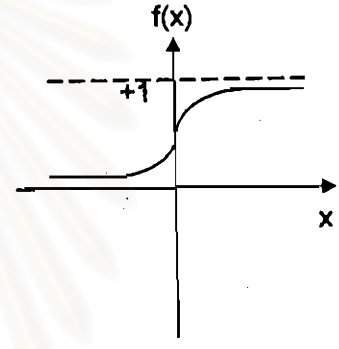
$$= \frac{2}{1+\exp(-x)} - 1 \qquad (3.5)$$

$$f_2'(x) = \frac{1}{2}[1 + f_2(x)][1 - f_2(x)]$$

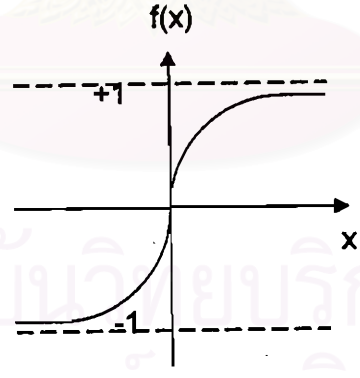
ฟังก์ชันนี้แสดงดังรูปที่ 3.4(ค)



(ก)



(ข)



(ค)

รูปที่ 3.4 Activation function (ก) Identity function (ข) Binary sigmoid function (ค) Bipolar sigmoid function

อัลกอริทึมการฝึก (training algorithm)

จากที่กล่าวมาแล้วว่าอัลกอริทึมการฝึกแบ่งเป็น 3 ขั้นตอนคือ feedforward, backpropagation of error และ update weight and bias ดังต่อไปนี้

- Initial weight และ bias
- Feedforward

  กำหนดให้อินพุตโนด $(x_i, i = 1,...,n)$

  กำหนดให้ฮิดเดนโนด $(z_j, j = 1,...,p)$

  จะได้อินพุตที่เข้าสู่ฮิดเดนโนดเป็น

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \tag{3.6}$$

  และเอาต์พุตจากฮิดเดนโนดเป็น

$$z_j = f(z\_in_j) \tag{3.7}$$

  ในทำนองเดียวกันกำหนดให้เอาต์พุตโนด $(y_k, k = 1,...,m)$

  จะได้อินพุตที่เข้าสู่เอาต์พุตโนดเป็น

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \tag{3.8}$$

  และเอาต์พุตจากเอาต์พุตโนดเป็น

$$y_k = f(y\_in_k) \tag{3.9}$$

- Backpropagation of error

  Error term ของเอาต์พุตโนด $(y_k, k = 1,...,m)$ เป็น

$$\delta_k = (t_k - y_k) f'(y\_in_k) \tag{3.10}$$

นำไปปรับค่าถ่วงน้ำหนักและ bias จะได้

$$\Delta w_{jk} = \lambda \delta_k z_j$$
$$\Delta w_{0k} = \lambda \delta_k \qquad (3.11)$$

ในทำนองเดียวกัน

Error term ของฮิดเดนโนด $(z_j, j = 1,\ldots,p)$ เป็น

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$$
$$\delta_j = \delta\_in_j f'(z\_in_j) \qquad (3.12)$$

นำไปปรับค่าถ่วงน้ำหนักและ bias จะได้

$$\Delta v_{ij} = \lambda \delta_j x_i$$
$$\Delta v_{0j} = \lambda \delta_j \qquad (3.13)$$

- Update weight and bias

จากเอาต์พุตโนด $(y_k, k = 1,\ldots,m)$ จะปรับค่าถ่วงน้ำหนักและ bias ของ w_{jk} โดยที่ $(j = 0,\ldots,p)$ ได้เป็น

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \qquad (3.14)$$

จากฮิดเดนโนด $(z_j, j = 1,\ldots,p)$ จะปรับค่าถ่วงน้ำหนักและ bias ของ v_{ij} โดยที่ $(i = 0,\ldots,n)$ ได้เป็น

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij} \qquad (3.15)$$

เนื่องจากวัตถุประสงค์ในการฝึกนิวรอลเนตเวิร์กก็เพื่อหาค่าถ่วงน้ำหนักที่เหมาะสมกับระบบเพื่อให้เกิดความแตกต่างของค่าเอาต์พุตที่ได้จากโครงข่ายและค่าเอาต์พุตเป้าหมาย t_k น้อยที่สุด ข้อกำหนดที่นิยมคือให้ค่าผลรวมของค่าความผิดพลาดกำลังสอง (Sum-Squared error: SSE) ของระบบมีค่าต่ำที่สุด เมื่อกำหนดให้ p เป็น pattern ของอินพุตดังนั้นหา SSE ได้จาก

$$E=\frac{1}{2}\sum_{p}\sum_{k}(t_k-y_k)^2 \qquad (3.16)$$

จะเห็นได้ว่าค่าความผิดพลาด (E) นี้คิดรวมทุก ๆ เอาต์พุตโนดและคิดจากทุก ๆ pattern ของอินพุต

ในการฝึกนิวรอลเนตเวิร์กโดยใช้ Backpropagation Algorithm นั้น ค่าความผิดพลาด (E) ที่ได้จะถูกคำนวณเป็นสัญญาณค่าหนึ่งซึ่งจะถูกส่งไปปรับค่าถ่วงน้ำหนักที่ link ต่าง ๆ โดยย้อนกลับจากเอาต์พุตเลเยอร์, ฮิดเดนเลเยอร์ ไปจนกระทั่งถึงอินพุตเลเยอร์ โดยที่ทุก link ในเนตเวิร์กจะได้รับการปรับค่าถ่วงน้ำหนักใหม่ก่อนที่จะป้อนข้อมูลอินพุตชุดต่อไปแบบ feedforward อีกครั้งหนึ่ง ทั้ง 2 กระบวนการนี้จะดำเนินไปจนค่าความผิดพลาดต่ำกว่าค่าที่กำหนดไว้จึงจะหยุดการฝึก

3.5 ปัจจัยที่ทำให้ประสิทธิภาพการฝึกโครงข่ายโดยใช้ Backpropagation Algorithm เพิ่มมากขึ้น

3.5.1 การกำหนดค่าเริ่มต้นให้กับเมทริกซ์ถ่วงน้ำหนัก

ก่อนที่จะทำการฝึกโครงข่าย multilayer perceptron โดยใช้ Backpropagation Algorithm จำเป็นต้องกำหนดค่าเริ่มต้นให้กับเมทริกซ์ค่าถ่วงน้ำหนักที่เชื่อมโยงระหว่างชั้นทุกชั้น โดยค่านี้จะเป็นเลขจำนวนจริงแบบสุ่มที่มีค่าน้อย ๆ ค่าเริ่มต้นของเมทริกซ์เชื่อมโยงจะมีผลต่อเวลาที่ใช้ในการฝึกและอาจจะส่งผลถึงค่าถ่วงน้ำหนักหลังจากที่ได้ฝึกโครงข่ายไปแล้วว่ามีศักยภาพมากน้อยเพียงใดในการแปลงข้อมูลอินพุตไปสู่เอาต์พุต

3.5.2 อัตราเร็วในการเรียนรู้

ปกติอัตราการเรียนรู้ (λ) ขึ้นอยู่กับความเร็วที่ต้องการในการปรับค่าถ่วงน้ำหนัก ถ้าต้องการให้เร็วควรเพิ่มค่า λ แต่ไม่ควรเพิ่มมากเกินไปเพราะจะทำให้ค่าถ่วงน้ำหนักแกว่งจนไร้เสถียรภาพได้ ในที่นี้จะให้ค่าอัตราเร็วในการเรียนรู้ปรับค่าได้ตามค่าความผิดพลาดเฉลี่ยของระบบ

ในการฝึกแต่ละรอบ ค่าความผิดพลาด (E) จะถูกคำนวณจากอัตราการเรียนรู้ในรอบนั้น ๆ ถ้าความผิดพลาดในรอบที่ t+1 มีค่ามากกว่าความผิดพลาดในรอบที่ t คือมากกว่าด้วยอัตราส่วน=1.04 แล้ว ค่าถ่วงน้ำหนัก, bias และค่าความผิดพลาดที่คำนวณได้จะไม่นำมาใช้ และอัตราการเรียนรู้จะลดลงเป็น 0.7λ

ในทางตรงกันข้าม ถ้าความผิดพลาดในรอบที่ t+1 มีค่าน้อยกว่าความผิดพลาดในรอบที่ t เราจะเก็บค่าถ่วงน้ำหนักและ bias ไว้ และจะเพิ่มอัตราการเรียนรู้เป็น 1.05λ (อัตราการเพิ่มหรือลดของอัตราการเรียนรู้ ผู้ใช้ต้องกำหนดเอง [6] ในที่นี้ได้มาจากการทดลอง)

อัตราการเรียนรู้ที่ปรับค่าได้ (adaptive learning rate) จะช่วยให้เวลาที่ใช้ฝึกลดลง

3.5.3 โมเมนตัม (Momentum)

โมเมนตัม (μ) มีไว้เพื่อช่วยในการเร่งให้ค่าความผิดพลาดลู่เข้าสู่ 0 ได้เร็วขึ้น และช่วยป้องกันการแกว่งของระบบ โดยค่านี้จะสัมพันธ์กับค่าอัตราเร็วในการเรียนรู้คือถ้าอัตราเร็วในการเรียนรู้มีค่ามากแต่ค่าโมเมนตัมมีค่าน้อยจะทำให้ระบบโครงข่ายเกิดการแกว่ง ส่วนกรณีอื่น ๆ โครงข่ายจะไม่เกิดการแกว่งแต่จะมีผลต่อเวลาที่ใช้ในการฝึก โดยการนำค่าการเปลี่ยนแปลงของค่าถ่วงน้ำหนักครั้งล่าสุดมาใช้ในการปรับค่าถ่วงน้ำหนักปัจจุบันด้วยจะได้

$$\Delta w_{jk} = \lambda \delta_k z_j + \mu \Delta w_{jk}(old) \qquad (3.17)$$

เพราะฉะนั้นจะได้ค่าถ่วงน้ำหนักใหม่จากฮิดเดนโนดไปยังเอาต์พุตโนดเป็น

$$w_{jk}(new) = w_{jk}(old) + \lambda \delta_k z_j + \mu \Delta w_{jk}(old) \qquad (3.18)$$

และในทำนองเดียวกัน

$$\Delta v_{ij} = \lambda \delta_j x_i + \mu \Delta v_{ij}(old) \qquad (3.19)$$

เพราะฉะนั้นจะได้ค่าถ่วงน้ำหนักใหม่จากอินพุตโนดไปยังฮิดเดนโนดเป็น

$$v_{ij}(new) = v_{ij}(old) + \lambda \delta_j x_i + \mu \Delta v_{ij}(old) \qquad (3.20)$$

ส่วนการเลือกจำนวนของฮิดเดนโนดไม่มีกฎเกณฑ์หรือทฤษฎีที่แน่นอนหากเลือกให้มีค่าน้อยเกินไปจะไม่ทนต่อ noise และอาจจะไม่สามารถรู้จำแบบรูป (pattern) ที่เปลี่ยนไปได้ หากมากเกินไประบบจะคำนวณช้าทำให้ต้องฝึกนาน ดังนั้นการกำหนดจำนวนของฮิดเดนโนดที่เหมาะสมสามารถทำได้โดยการลองผิดลองถูก (trial and error)