บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงเนื้อหาที่ได้ทบทวนซึ่งแบ่งเนื้อหาเป็นส่วนต่าง ๆ ดังนี้

ปัญญาประดิษฐ์ (Artificial Intelligence : AI)

วิธีการหาค่าเหมาะสม (Optimization Method)

งานวิจัยเกี่ยวกับอัลกอริทึมฝึกสอนโครงข่ายประสาทเทียม

ปัญญาประดิษฐ์ (Artificial Intelligence : AI)

ในปัจจุบันนี้ได้มีการนำเอาคอมพิวเตอร์เข้ามาช่วยในการประมวลผลอย่างกว้างขวาง โดยเฉพาะในกระบวนการหาคำตอบเหมาะสม (Optimization) ของแต่ละปัญหา ส่งผลให้การศึกษาในเรื่องปัญญาประดิษฐ์ (Artificial Intelligence) สาขาต่างๆ นั้นได้รับความนิยมเพิ่มขึ้นอย่างสูง ซึ่งสามารถนำไปประยุกต์ใช้ได้กับงานหลากหลายสาขา ทั้งสาขาทางด้านเทคโนโลยีสารสนเทศที่เป็นสาขาที่มีการศึกษาเรื่องนี้โดยตรง ทั้งยังสามารถประยุกต์ใช้ได้กับสาขาวิทยาศาสตร์การแพทย์ สาขาวิศวกรรมศาสตร์ รวมไปถึง สาขาการเกษตร ซึ่งหลักการการทำงานในเรื่องของปัญญาประดิษฐ์นั้น จะทำงานในลักษณะการค้นหาคำตอบเพื่อให้ได้คำตอบที่ดีที่สุดและเหมาะสมกับปัญหานั้น ๆ ดังนั้น จึงมีวิธีการเกี่ยวกับการค้นหาคำตอบที่ดีที่สุดมากมายที่ได้ถูกคิดค้นขึ้นและพัฒนามาอย่างต่อเนื่อง โดยปกติวิธีที่ได้รับความนิยมเลือกใช้เพื่อค้นหาคำตอบที่ดีที่สุดวิธีหนึ่ง คือ วิธีการค้นหาคำตอบแบบสุ่ม หรือ Stochastic Search Methods ซึ่งแยกได้เป็น Simulated Annealing (SA), Genetic Algorithm (GA), Neural Network (NN), Ant Colony Optimization (ACO), Particle Swarm Optimization(PSO) ซึ่งวิธีเหล่านี้จะประยุกต์ใช้หลักการความน่าจะเป็นแบบสุ่ม (Stochastic Process) มาอธิบายการทำงานและถูกนำไปประยุกต์ใช้แก้ปัญหาต่างๆ มากมาย ซึ่งขึ้นอยู่กับความเหมาะสมกับลักษณะของปัญหานั้น

ในงานทางด้านวิศวกรรมนั้น มีการนำวิธีการค้นหาคำตอบแบบสุ่มมาใช้ เพื่อหาคำตอบที่เหมาะสมของปัญหาต่างๆ ซึ่งวิธีที่หนึ่งที่ได้รับความนิยมอย่างมาก คือ โครงข่ายประสาทเทียม (Neural Network : NN) จากการทบทวนเอกสารที่ผ่านมา พบว่า การใช้วิธีโครงข่ายประสาทเทียมเพื่อค้นหาค่าเหมาะสมเพื่อแก้ปัญหาใดๆ นั้น ได้มีอัลกอริทึมฝึกสอนโครงข่ายประสาทเทียมที่ถูก

คิดค้นและพัฒนาขึ้นจำนวนมากในปัจจุบันเพื่อนำไปประยุกต์ใช้กับงานหลายหลายประเภท แต่โดยทั่วไปแล้ว พบว่า อัลกอริทึมที่ถูกพัฒนาขึ้นมานั้น มักจะมีคุณลักษณะและสมรรถนะที่แตกต่างกัน ยิ่งอัลกอริทึมเหล่านั้นสามารถทำงานได้ด้วยประสิทธิภาพมากขึ้น ความต้องการหน่วยความจำสำรองและความเร็วของหน่วยประมวลผลที่ใช้สำหรับประมวลผลด้วยกระบวนการของอัลกอริทึมเหล่านั้นก็ยิ่งเพิ่มขึ้น ส่งผลให้ เป็นการสิ้นเปลืองทรัพยากรในการประมวลผลของแต่ละอัลกอริทึม เมื่อต้องนำโครงข่ายประสาทเทียมมาประยุกต์ใช้ในงานทางด้านวิศวกรรมที่ต้องการนำโครงข่ายประสาทเทียมมาดำเนินการด้วยอุปกรณ์ประมวลผลที่มีข้อจำกัดเรื่องหน่วยความจำสำรอง ซึ่งมีจำนวนไม่มาก เช่น ไมโครคอนโทรลเลอร์ พบว่า ไม่สามารถบรรจุกระบวนการการทำงานทั้งหมดของอัลกอริทึมเหล่านั้นลงในหน่วยความจำของไมโครคอนโทรลเลอร์ได้ ดังนั้น งานวิจัยฉบับนี้ ผู้วิจัยจึงได้ทำการศึกษาและพัฒนาอัลกอริทึมที่มีขั้นตอนในการประมวลผลไม่ซับซ้อน ซึ่งเหมาะต่อการนำไปประยุกต์ใช้กับอุปกรณ์ประมวลผลที่มีหน่วยความจำขนาดเล็กหรือที่มีทรัพยากรการประมวลผลที่จำกัด

วิธีการหาค่าเหมาะสม (Optimization Method)

เหตุผลหลักของการนำวิธีการหาค่าเหมาะสมมาประยุกต์ใช้เพื่อแก้ปัญหาใดๆ นั้น คือในปริภูมิ (Space) ของปัญหาต่างๆ ที่เราพิจารณานั้น เราไม่สามารถทราบได้ล่วงหน้าเลยว่าคำตอบที่เหมาะสมนั้นมีค่าเท่าใดและอยู่ที่ใดในปริภูมิของปัญหา หรือมีเงื่อนไขเป็นอย่างไรบ้าง เพื่อให้ได้มาซึ่งคำตอบที่เหมาะสมของปัญหานั้นๆ ดังนั้น ในการหาคำตอบที่เหมาะสมของปัญหาต่างๆ นั้น จึงจำเป็นต้องใช้วิธีการค้นหาคำตอบแบบสุ่ม ซึ่งมีหลายวิธี เช่น Simulated Annealing (SA), Genetic Algorithm (GA), Neural Network (NN), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) ซึ่งแต่ละวิธีนั้นจะถูกนำไปประยุกต์ใช้กับปัญหาที่แตกต่างกัน ในวิทยานิพนธ์นี้ จะกล่าวถึงวิธีการหาคำตอบเหมาะสมของวิธีการเหล่านี้อย่างคราวๆ แต่จะเน้นเนื้อหาในส่วนของ Neural Network (NN) เนื่องจากผู้วิจัยได้ทำการศึกษาและพัฒนาอัลกอริทึมเกี่ยวกับการฝึกสอน Neural Network (NN)

1. Genetic Algorithm (GA)

การทำงาน ของเจเนติกอัลกอริทึม เป็นวิธีการหาคำตอบโดยอาศัยหลักการทำงานจากการคัดสรรพันธุกรรมจากธรรมชาติ ซึ่งคำตอบที่ดีที่สุดจะสามารถอยู่รอดในสิ่งแวดล้อมนั้นและได้รับการถ่ายทอดไปยังรุ่นต่อไป (Goldberg, 1989) การทำงานด้วย Genetic Algorithm นั้นเริ่มต้นจะมีการสุ่มสร้างประชากร (Population) ขึ้นมาเป็นคำตอบจำนวนหนึ่ง โดยกำหนดให้แต่ละคำตอบที่สุ่มขึ้นมานั้น เรียกว่า โครโมโซม แต่ละโครโมโซมนั้น จะประกอบไปด้วยหน่วยเล็กย่อยลง

ไปอีก เรียกว่า ยีน (gene) จากนั้นแต่ละโครโมโซมจะถูกนำไปผ่านกระบวนการทางพันธุกรรม ซึ่งมีอยู่ 2 ประเภท คือ การสลับสายพันธ์ (Crossover) และการกลายพันธ์ (Mutation) ซึ่งจะเลือกทำอย่างใดอย่างหนึ่ง หรือว่าทั้งสองอย่างก็ได้ แล้วแต่ความเหมาะสม หลังจากผ่านขั้นตอนทางพันธุกรรมแล้ว จะทำการประเมินค่าความเหมาะสมของแต่ละโครโมโซม โดยที่โครโมโซมที่มีค่าความเหมาะสมมากสุดจะมีโอกาสอยู่รอดมากที่สุด และผ่านเข้าไปสู่กระบวนการคัดสรร โดยในขั้นตอนการคัดสรรนี้ หากโครโมโซมใดที่สามารถผ่านกระบวนการคัดสรรได้ จะทำให้โครโมโซมนั้นถูกเลือกเพื่อถ่ายทอดลักษณะทางพันธุกรรมให้แก่โครโมโซมในรุ่นต่อๆ ไป

ส่วนการตรวจสอบขั้นตอนการทำงานของเจเนติกอัลกอริทึมนั้น จะตรวจสอบโดยการตรวจสอบเงื่อนไขหยุดการทำงาน ซึ่งขึ้นอยู่กับว่าผู้ใช้จะกำหนดเงื่อนไขไว้อย่างไร อาจจะกำหนดโดยระบุจำนวนรุ่น (Generation) ของประชากรที่ต้องการหรือเมื่อคำตอบมีค่าใกล้เคียงกับคำตอบเป้าหมายมากที่สุด

2. Ant Colony Optimization (ACO)

Ant colony Optimization หรือ ACO เป็นวิธีการหาค่าเหมาะสมโดยจำลองการทำงานมาจากพฤติกรรมการหาอาหารของมด เพื่อหาเส้นทางการค้นหาอาหารที่สั้นที่สุดระหว่างรังของมดและแหล่งอาหาร โดยธรรมชาติการค้นหาอาหารของมดนั้น มดแต่ละตัวจะเลือกเส้นทางการค้นหาอาหารแบบสุ่ม และกลับมาที่รังเมื่อค้นหาอาหารเจอ โดยในขณะที่เดินทางไปหาอาหารนั้นจะทิ้งหลักฐานไว้ตลอดทางเดิน เรียกว่า ฟีโรโมน (Pheromone) โดยที่มดตัวต่อไปจะเลือกเดินตามทางที่มีความหนาแน่นของ ฟีโรโมนสูง ซึ่งเป็นเส้นทางสั้นที่สุดที่สามารถพบแหล่งอาหารได้ เหตุผลที่ทำให้เชื่อว่า เส้นที่มีความหนาแน่นของ ฟีโรโมนสูง นั้น เป็นเส้นทางที่สั้นที่สุดที่จะพบอาหารได้ คือ การระเหยได้ของ ฟีโรโมนดังนั้น เส้นทางยาวๆ จะมีการปล่อยฟีโรโมนไว้เป็นเวลานาน จึงทำให้ความหนาแน่นของ ฟีโรโมน ลดลง เมื่อเปรียบเทียบกับเส้นทางที่สั้นกว่าที่มีการปล่อยฟีโรโมนไว้เพียงเวลาสั้น ๆ และมีการปล่อย ฟีโรโมนซ้ำกันไปมา จึงทำให้ง่ายต่อการหาเส้นทางที่สั้นที่สุดและเหมาะสมที่สุด (Marco Dorigo, et al.,1996)

3. Particle swarm optimization (PSO)

PSO ถือว่าเป็นอีกเทคนิคของ Evaluation Computation เหมือนกับ Genetic Algorithm โดยขั้นตอนการทำงานของอัลกอริทึมนี้ จะเริ่มต้นจากการสุ่มสร้างประชากร โดยเรียกประชากรที่สร้างขึ้นมาแต่ละตัวนั้นว่าพาทิเคิล (Particle) ในแต่ละพาทิเคิลนั้น จะมีพารามิเตอร์ที่กำหนดคุณลักษณะเฉพาะของพาทิเคิลแต่ละตัว ทำให้แต่ละพาทิเคิลมีคุณลักษณะที่แตกต่างกันในขั้นตอนเริ่มต้นของการคำนวณค้นหาคำตอบ แต่ละพาทิเคิลจะค้นหาคำตอบให้คลอบคลุมพื้นที่ทั้งหมดของปัญหา โดยที่แต่ละพาทิเคิลนั้น ไม่ทราบมาก่อนว่าคำตอบที่ดีที่สุดนั้นอยู่ที่ตำแหน่งใด

ของพื้นที่ปัญหานั้น แต่จะจำค่าความเหมาะสมของพาทิเคิลที่ผ่านมาก่อนหน้านั้น เพื่อเป็นประสบการณ์ในการปรับตัวเพื่อหาคำตอบดีที่สุด โดยกำหนดให้ค่าความเหมาะสมของพาทิเคิลนั้นว่า ค่า pbest จากนั้นจะนำค่าความเหมาะสมของแต่ละพาทิเคิลหรือค่า pbest มาเปรียบเทียบกับค่าความเหมาะสมของพาทิเคิลอื่นๆ โดยค่าที่ดีที่สุด ที่ได้รับหลังจากการเปรียบเทียบค่าความเหมาะสมของพาทิเคิลทั้งหมด นั้นจะเรียกว่า ค่า gbest จากนั้นจะนำค่า gbest ที่ได้จากการเปรียบเทียบของพาทิเคิลทั้งหมด ไปปรับค่าความเร็วและตำแหน่งของแต่ละพาทิเคิลเพื่อให้ แต่ละพาทิเคิลนั้น สามารถค้นหาคำตอบได้ถูกต้องและเหมาะสม (Kennedy and Eberhart, 1995)

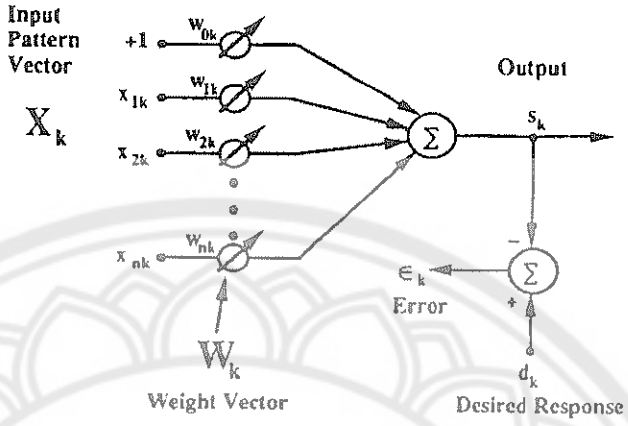
ถึงแม้ว่าวิธีของ PSO จะมีการทำงานที่คล้ายคลึงกับ GA แต่พบว่า มีบางขั้นตอนที่แตกต่างกับ GA อย่างชัดเจน คือ การทำงานของ PSO จะไม่มีการสลับสายพันธ์ (Crossover) และการกลายพันธ์ (Mutation) ซึ่งเป็นขั้นตอนที่สำคัญในการทำงานด้วยวิธีของ GA

4. Neural Networks (NN)

โครงข่ายประสาทเทียมเป็นการทำงานที่จำลองพฤติกรรมการทำงานมาจากเครือข่ายประสาทในสมองของมนุษย์ โดยลักษณะเด่นของการทำงานของโครงข่ายประสาทในสมองของมนุษย์ คือ การเรียนรู้จากประสบการณ์ที่ผ่านมาและสามารถใช้ประสบการณ์นั้นๆ มาสร้างความรู้ใหม่ได้ มีการจำลองการทำงานของสมองในลักษณะการทำงานแบบขนาน ซึ่งประกอบด้วยเซลล์ประสาทจำนวนมาก

ในการศึกษาการทำงานของโครงข่ายประสาทเทียมนั้น จะต้องกล่าวถึงแนวคิดพื้นฐานขององค์ประกอบในโครงข่ายประสาทเทียม คือ Adaptive Linear Combiner เรียกย่อๆ ว่า Adaline (B. Widrow and M. A. Lehr, 1990) จากนั้นนำ Adaline หลายๆ ตัวมาเชื่อมต่อกันเป็นโครงข่ายหลายๆ ชั้น เรียกว่า โครงข่ายประสาทเทียมแบบป้อนผลการคำนวณไปข้างหน้า (Aritificial Feedforward neural networks)

โครงสร้างทั่วไปของ Adaline และส่วนประกอบของแต่ละ Adaline แสดงในภาพ 1



ภาพ 1 โครงสร้าง Adaptive Linear Combiner (Adaline)

Adaline แสดงในภาพ 1 เอาท์พุทที่ได้จาก Adaline จะเป็นค่าเอาท์พุทเชิงเส้น (Linear Output) ของอินพุทและค่าน้ำหนัก เมื่อ กำหนดให้เวกเตอร์อินพุท (Input Vector : $\mathbf{x}_k$) มีค่าดังสมการ (1)

$$\mathbf{x}_k = [1 \quad x_{1k} \quad x_{2k} \quad \dots \quad x_{nk}]^T \tag{1}$$

เวกเตอร์ค่าน้ำหนัก (Weight Vector = $\mathbf{w}_k$) มีค่าดังสมการ (2)

$$\mathbf{w}_k = [w_{1k} \quad w_{2k} \quad \dots \quad w_{nk}]^T \tag{2}$$

เอาท์พุทเชิงเส้น (Linear Output $= s_k$) ระหว่างเวกเตอร์อินพุทและเวกเตอร์ค่าน้ำหนักมีค่าดังสมการ (3)

$$s_k = \mathbf{x}_k^T \mathbf{w}_k \tag{3}$$

ตัวแปรในภาพ 1 แทนค่าต่าง ๆ ดังต่อไปนี้

d_k คือ ค่าเอาท์พุทที่ต้องการ (Desired Output)

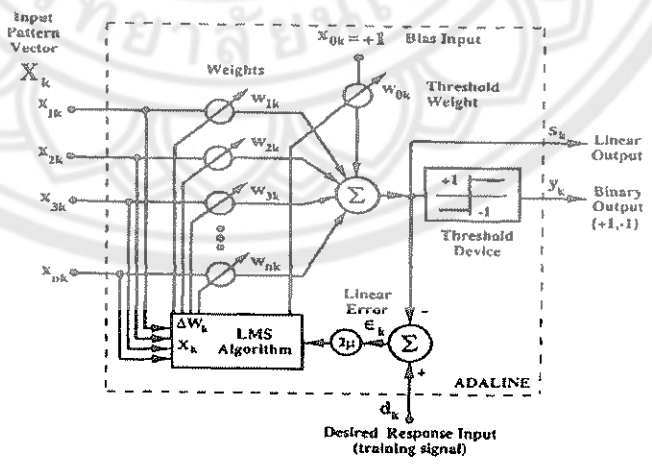
w_{ok} คือ ค่าน้ำหนักไบแอส w_{ok} เป็นค่าคงที่ มีค่าเท่ากับ 1 เพื่อปรับค่าเทรชโฮลด์ของเอาท์พุทโครงข่าย

ในกระบวนการฝึกสอนโครงข่ายโดยทั่วไป เมื่ออินพุทฝึกสอนและค่าเอาท์พุทฝึกสอนโครงข่ายเข้าสู่โครงข่าย จะได้เอาท์พุทโครงข่ายออกมาค่าหนึ่ง เรียกว่า เอาท์พุทเชิงเส้น โดยที่อัลกอริทึมปรับค่าน้ำหนักโครงข่ายแต่ละอัลกอริทึม จะปรับค่าน้ำหนักโครงข่ายเพื่อให้เอาท์พุทโครงข่าย มีค่าใกล้เคียงเอาท์พุทฝึกสอนมากที่สุด

โดยทั่วไปโครงข่ายประสาทเทียม จะมีลักษณะคล้ายคลึงกับโครงสร้างของ Adaline ดังแสดงในภาพ 1 แต่เมื่อถูกนำมาประยุกต์เพื่อแก้ปัญหาทางตรรกศาสตร์ เพื่อให้ค่าเอาท์พุทโครงข่ายมีความสอดคล้องกับเอาท์พุทฝึกสอน ซึ่งส่วนมากจะเป็นค่าระดับ ไบนารี (binary) ดังนั้น Adaline จึงจำเป็นต้องเชื่อมต่อกับตัวแบ่งระดับ (Threshold)

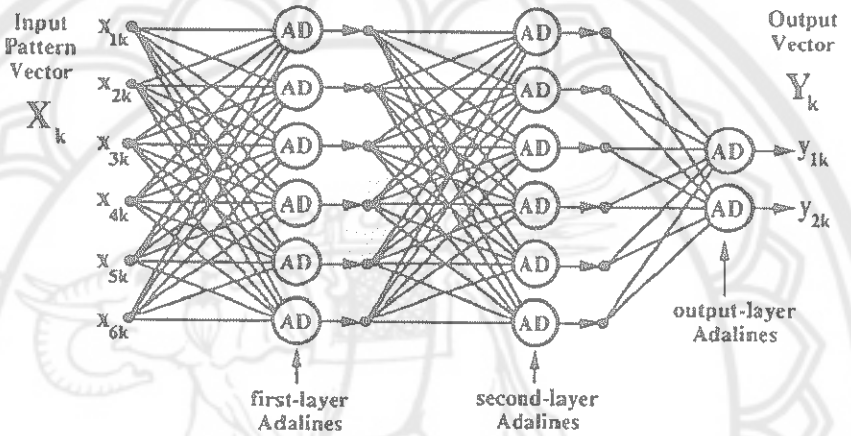
ภาพ 2 เป็นตัวอย่างการใช้ Adaline เชื่อมต่อกับตัวแบ่งระดับแบบ hard-limiting ซึ่งใช้สำหรับการสร้างเอาท์พุทจากโครงข่าย มีค่าเป็น ± 1 ซึ่งตัวแบ่งระดับนี้ สามารถเขียนแทนในรูปแบบสัญลักษณ์ของฟังก์ชั่นได้เป็น sgn (ซิกนั่มฟังก์ชั่น) ดังนั้น เพื่อให้ได้เอาท์พุตโครงข่ายที่สอดคล้องกับเอาท์พุทฝึกสอน จึงต้องนำเอาท์พุทโครงข่ายเชื่อมต่อผ่านกับฟังก์ชั่นนี้ ดังแสดงในสมการ (4)

$$y_k = sgn(s_k) \tag{4}$$



ภาพ 2 โครงสร้างของ Adaline เชื่อมต่อกับตัวแบ่งระดับแบบ hard-limiting

จากที่ผ่านมาได้ศึกษาถึงลักษณะทั่วไปของ Adaline ซึ่งถือว่าเป็นหน่วยประมวลผลพื้นฐานของโครงข่ายประสาทเทียม เมื่อมีการนำ Adaline เชื่อมต่อกันหลายชั้น เรียกว่า โครงข่ายประสาทเทียม (multi-element neural network) โดยทั่วไปจะเรียกโครงข่ายเหล่านี้ว่า โครงข่ายแบบหลายชั้น (multilayer neural network) หรือโครงข่ายประสาทเทียมหลายชั้นแบบไปข้างหน้า (Aritificial Feedforwork Neural Network) ภาพ 3 แสดงตัวอย่างโครงข่ายประสาทเทียมแบบไปข้างหน้าที่มี 3 ชั้น



ภาพ 3 โครงข่ายประสาทเทียมแบบไปข้างหน้า (Aritificial Feedforward Neura Network) แบบ 3 ชั้น

งานวิจัยเกี่ยวกับการฝึกสอนโครงข่ายประสาทเทียม

ตัวอย่างของงานวิจัยที่ศึกษาเกี่ยวกับอัลกอริทึมฝึกสอนโครงข่ายประสาทเทียม ซึ่งดำเนินการโดยอนาล็อกคอมพิวเตอร์

Yutaka MAEDA, et al. (1991) ได้นำเสนอกฎการเรียนรู้เพื่อฝึกสอนโครงข่ายประสาทเทียมแบบป้อนผลการคำนวณไปข้างหน้า โดยใช้ค่าผลต่างค่าความผิดพลาดกำลังสอง เพื่อปรับค่าน้ำหนักโครงข่าย ดังสมการปรับค่าน้ำหนักของโครงข่ายที่ (5)

$$w_i^{t+1} = w_i^t - \Delta w_i^t \qquad (5)$$

เมื่อ

w_i^t คือ ค่าน้ำหนักโนดที่ i ถูกรบกวนด้วยค่าการรบกวนค่าคงที่ c $(w_1^t, \dots, w_i^t + c, \dots, w_n^t)^T$

w_i^{t+1} คือ ค่าน้ำหนักโนดที่ i ที่ถูกปรับค่า

Δw_i^t คือ ผลต่างระหว่างค่าความผิดพลาดกำลังสองของค่าน้ำหนัก w_i^t และ w^t

เมื่อ Δw_i^t คำนวณได้จากสมการ (6)

$$\Delta w_i^t = \alpha \frac{\mathrm{E_p}(w_i^t) - \mathrm{E_P}(w^t)}{c} \tag{6}$$

เมื่อ

E_p คือ ค่าเฉลี่ยความผิดพลาดกำลังสองระหว่างค่าน้ำหนัก w_i^t และ w^t

w_i^t คือ เวกเตอร์ค่าน้ำหนัก โดยที่ค่าน้ำหนักโนดที่ i ถูกรบกวนด้วยค่าการรบกวนค่าน้ำหนัก c $(w_1^t, \dots, w_i^t + c, \dots, w_n^t)^T$

w^t คือ เวกเตอร์ค่าน้ำหนัก $(w_1^t, w_2^t, w_3^t, w_4^t, \dots, w_n^t)^T$

c คือ ค่าการรบกวนค่าน้ำหนักค่าน้อย ๆ (A perturbation) มีค่ามากกว่าศูนย์ (c>0)

α คือ ค่าอัตราการเรียนรู้

o_{pj}^{out} คือ ค่าเอาท์พุทโครงข่ายของนิวรอลที่ j pattern ที่ p ในชั้น $output$

t_{pj} คือ ค่าเอาท์พุทฝึกสอนโครงข่ายที่ j pattern ที่ p

เมื่อ

$$\mathrm{E_P}(w^t) = \sum_{\mathrm{j}} \left(t_{pj} - o_{pj}^{out}\right)^2 \tag{7}$$

เมื่อ $i = 1,2,3, \dots, n$

ค่ารบกวนค่าน้ำหนัก c จะเป็นค่ารบกวนน้อยๆ พบว่าเมื่อ c มีค่าน้อยๆ ยิ่งทำให้ผลลัพธ์ที่ได้มีค่าใกล้เคียงกับค่าเกรเดียนท์มากขึ้นด้วย

Marwan Jabri and Barry Flower (1992) นำเสนองานวิจัยนี้ เนื่องจากถึงแม้ว่าอัลกอริทึมฝึกสอนโครงข่าย Back-propagation จะมีประสิทธิภาพสูงและสามารถทำงาน โดยโครงข่ายที่มีหลายชั้นได้ดี แต่พบว่า การกำหนดอัลกอริทึม Back-propagation ลงบนชิป (Chip) จำเป็นต้องใช้ทรัพยากรในการประมวลผลจำนวนมาก เช่น ต้องมีหน่วยความจำสำรองและความเร็วในการประมวลผลสูง ซึ่งมากเกินความสามารถที่ชิปตัวๆ หนึ่งจะทำได้ เพื่อแก้ปัญหาดังกล่าว Marwan Jabri and Barry Flower จึงได้ใช้การประมาณค่าเกรเดียนท์แทนการคำนวณค่าเกรเดียนท์โดยตรง โดยใช้หลักการการรบกวนค่าน้ำหนักแบบสุ่ม (Weight Perturbation) เพื่อนำมาประยุกต์ใช้กับงานทางด้านอนาล็อก วีแอลเอสไอ (VLSI : Very Large Scale Integrated) ใช้กับโครงข่ายเพอเซฟตรอนแบบหลายชั้นและประมวลผลด้วยอัลกอริทึมฝึกสอนโครงข่ายประสาทเทียม Back-propagation โดยเรียกวิธีดังกล่าวว่า "การรบกวนค่าหนัก (Weight Perturbation)" ซึ่งพบว่า การประมาณด้วยวิธีนี้ใช้เพียงขั้นตอนในการคำนวณไปข้างหน้าอย่างเดียวโดยที่ไม่มีการคำนวณย้อนกลับ ทำให้ลดขั้นตอนการประมวลผลลงได้ ซึ่งเป็นข้อดีสำหรับการนำอัลกอริทึมนี้ไปทดสอบการทำงานบนเครื่องมืออนาล็อกวีแอลเอสไอ (VLSI) ที่มีการประมวลผลเชิงขนาน (Parallel Analog) ได้ และเหมาะสมเมื่อใช้กับโครงข่ายประสาทเทียมแบบย้อนกลับ (Recurrent Networks)

โดยสมการที่ปรับค่าน้ำหนักที่ได้นำเสนอ เป็นดังสมการที่ (8) กำหนดให้

$$\Delta w_{ij} = G(pert_{ij})\Delta E(w_{ij}, pert_{ij}) \tag{8}$$

เมื่อ

$$G(pert_{ij}) = \frac{-\beta}{pert_{ij}} \tag{9}$$

และ

$$\Delta E(w_{ij}, pert_{ij}) = E(w_{ij} + pert_{ij}) - E(w_{ij}) \tag{10}$$

เมื่อ

ΔE คือ ผลต่างของค่าความผิดพลาดกำลังสองระหว่างค่าน้ำหนักที่ถูกรบกวนและค่าน้ำหนักที่ไม่ถูกรบกวน

$pert_{ij}$ คือ ค่าคงที่ เพื่อใช้รบกวนค่าน้ำหนัก

β คือ ค่าคงที่ใช้ปรับค่าน้ำหนัก

w_{ij} คือ ค่าน้ำหนักโครงข่าย

$E(w_{ij})$ คือ ค่าความผิดพลาดกำลังสองของค่าน้ำหนักที่ไม่ถูกรบกวน

$E(w_{ij} + pert_{ij})$ คือ ค่าความผิดพลาดกำลังสองของค่าน้ำหนักที่ถูกรบกวน

Δw_{ij} คือ ผลต่างค่าน้ำหนักโครงข่ายที่ใช้ปรับค่าน้ำหนักโครงข่าย

ซึ่งขั้นตอนการประมวลผล สามารถกำหนดได้ดังนี้

```
For each pattern p {
        E = ForwardPass ( )
        ClearDeltaWeight ( )
        For each weight w_ij do {
                Epert = ApplyPerturbate (w_ij)
                DeltaErrorr = Epert – E
                DeltaW[i][ j] = −β * DeltaError/Perturbation
                RemovePerturbation (w_ij)
                        }
}
```

ภาพ 4 ขั้นตอนการประมวลผล

จากบทความนี้ เราสามารถ สรุปได้ว่า การหาค่าเกรเดียนท์จากหลักการ การรบกวนค่าน้ำหนัก (Weight Perturbation) นั้น ทำให้โครงข่ายประสาทเทียมที่ได้มีขนาดเล็กเพียงพอที่ทำการทดลองและจำลองการทำงานของอัลกอริทึมลงในชิปตัวหนึ่งได้ ซึ่งเป็นอุปกรณ์ที่มีหน่วยความจำและความเร็วในการประมวลผลที่จำกัด แทนการหาค่าเกรเดียนท์จากการหาอนุพันธ์อันดับหนึ่งหรือสองซึ่งมีขั้นตอนการประมวลผลที่ซับซ้อนเกินความสามารถที่จะบรรจุขั้นตอนการทำงานเหล่านั้นลงชิปตัวหนึ่งๆ ได้

Kenichi Hirotsu and Martin A. Brooke (1993) พบว่า อัลกอริทึมฝึกสอนโครงข่ายประสาทเทียมส่วนใหญ่ที่ใช้ดิจิตอลคอมพิวเตอร์ (Digital Computer) ในการดำเนินการนั้น ใช้เวลาในการประมวลผลมาก จึงเป็นเรื่องยากที่จะนำไปประยุกต์ใช้กับปัญหาที่เกิดขึ้นจริงในทางปฏิบัติ ดังนั้น จึงได้เสนออัลกอริทึมที่สามารถดำเนินการด้วยชิป VLSI เพื่อนำไปประยุกต์ใช้กับปัญหาจริง โดยได้เรียกวิธีดังกล่าวว่า Random Weight Change Learning Algorithm ซึ่งมีหลักการ คือ ค่าน้ำหนัก (w_{ij}) จะถูกเปลี่ยนไปในลักษณะสุ่ม จากสถานะเริ่มต้นด้วยค่าสุ่มน้อยๆ มีค่าอยู่ระหว่าง

$\pm\delta$ ถ้าผลรวมค่าความผิดพลาดระหว่างจุดเอาท์พุตโครงข่ายและเอาท์พุตที่ต้องการมีค่าลดลง โดยใช้ค่าน้ำหนักที่ถูกเปลี่ยนแปลงไปนั้น ค่าน้ำหนักตัวดังกล่าวจะถูกใช้ฝึกโครงข่ายต่อไปจนกระทั่งค่าความผิดพลาดมีค่าเพิ่มขึ้น และเมื่อค่าความผิดพลาดมีค่าเพิ่มขึ้นแล้ว ค่าน้ำหนักตัวดังกล่าวจะถูกเปลี่ยนแปลงด้วยค่าสุ่มน้อยๆ อีกครั้งหนึ่ง

ดังสมการปรับค่าน้ำหนักที่นำเสนอดังสมการ (12)

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n+1) \tag{11}$$

เมื่อ

$\Delta w_{ij}(n+1) = \Delta w_{ij}(n)$ $\quad : if E(n+1) < E(n)$

$\Delta w_{ij}(n+1) = \delta \cdot Rand(n)$ $\quad : if E(n+1) \geq E(n)$

$Rand(n)$ คือ ฟังก์ชั่นสุ่ม มีค่าสองระดับ คือ ± 1

δ คือ ค่าคงที่ควบคุมลักษณะการกระจายตัวของฟังก์ชั่นสุ่ม $Rand(n)$

Gert Gauwenberghs (1993) ได้นำเสนอกฎการเรียนรู้ โดยใช้หลักการการเรียนรู้แบบ Stochastic Error-Descent เพื่อหาเหมาะสมของค่าน้ำหนักในโครงข่ายที่กำหนดและทำให้ค่าความผิดพลาดของโครงข่ายมีค่าลดลง

ฟังก์ชั่นค่าเฉลี่ยความผิดพลาดกำลังสอง คำนวณได้จากสมการ (12)

$$\varepsilon(p) = \frac{1}{2}\sum_{\alpha}\sum_{k}(x_k^{T(\alpha)} - x_k^{(\alpha)})^2 \tag{12}$$

เมื่อ

$x_k^{T(\alpha)}$ คือ เอาท์พุทที่ต้องการ

$x_k^{(\alpha)}$ คือ เอาท์พุทโครงข่าย

α คือ ลำดับครั้งในการฝึกสอนโครงข่าย

k คือ ลำดับ pattern ที่ใช้ฝึกสอนโครงข่าย

p คือ ค่าคงที่ซึ่งทำให้ฟังก์ชั่นค่าเฉลี่ยความผิดพลาดกำลังสองมีค่าน้อยสุด

เมื่อ กำหนดให้

$$\hat{p} = p + \pi \tag{13}$$

เมื่อ

$$\hat{\varepsilon} = \varepsilon(\hat{p}) - \varepsilon(p) \tag{14}$$

โดยที่

$$\Delta p = -\mu \hat{\varepsilon} \pi \tag{15}$$

เมื่อ

$\hat{\varepsilon}$ คือ ผลต่างค่าความผิดพลาดกำลังสองระหว่างค่าคงที่ $\hat{p}$ และค่าคงที่ p

π คือ ค่าการรบกวนซึ่งเลือกจากการกระจายตัวในลักษณะสุ่ม (Random Distribution)

$\hat{p}$ คือ ค่าคงที่ ซึ่งบวกด้วยค่าการรบกวน (π) ซึ่ง $\hat{p} = p + \pi$

μ คือ ค่าคงที่น้อยๆ มีค่าเป็นบวกเสมอ

Paul W.Hollis and John J.Paulos (1994) ได้นำเสนอแนวคิดเพื่อหาค่าเหมาะสมเมื่อดำเนินการด้วยอุปกรณ์ VLSI ซึ่งการปรับค่าน้ำหนักประมาณได้จากการเปลี่ยนแปลงค่าความผิดพลาดที่ได้จากการรบกวนค่าน้ำหนักเพื่อหาค่าเกรเดียนท์ในแต่ละจุด (Local Gradien)

โดยทั่วไป สมการที่ใช้ปรับค่าน้ำหนักแสดงดังสมการที่ (16)

$$\Delta w_{ji} = -n \frac{E_{norm} - E_{pert}}{w_{ji_{nom}} - w_{ji_{pert}}} = -n \frac{\Delta E}{\uparrow w_{ji}} \tag{16}$$

เมื่อ

Δw_{ji} คือ ค่าน้ำหนักที่ถูกเปลี่ยนไป ในกระบวนการคำนวณจากชั้น i ไปที่ชั้น j

n คือ อัตราการเรียนรู้

$\uparrow w_{ji}$ คือ ค่าผลต่างระหว่างค่าน้ำหนักเฉลี่ย ($w_{ji_{nom}}$) และค่าน้ำหนักที่ใช้รบกวนค่าน้ำหนักในโครงข่าย ($w_{ji_{pert}}$)

ΔE คือ ผลต่างระหว่างค่าความความผิดพลาดกำลังสองของค่าน้ำหนักเฉลี่ยและค่าหนักที่ใช้รบกวนค่าน้ำหนักในโครข่าย ($E_{norm} - E_{pert}$) ซึ่งเป็นผลจากการเปลี่ยนแปลงค่าน้ำหนักด้วยค่า $\uparrow w_{ji}$

Gert Gauwenberghs (1994) ได้นำเสนอกฎการเรียนรู้แบบมีผู้ฝึกสอน (Supervisor Training) ในโครงข่ายประสาทเทียม โดยทดลองในชิปอนาล็อกที่มีการเปลี่ยนแปลงตลอดเวลา (Dynamical Features) กับโครงข่ายแบบย้อนกลับ (Recurrent Network)

สมการปรับค่าน้ำหนักเป็นดังสมการ (17)

$$p^{(k+1)} = p^{(k)} - \mu \cdot \hat{\varepsilon}^{(k)} \pi^{(k)} \tag{17}$$

และ

$$\hat{\varepsilon}^{(k)} = \frac{1}{2}\left(\varepsilon\left(p^{(k)} + \pi^{(k)}\right) - \varepsilon\left(p^{(k)} - \pi^{(k)}\right)\right) \quad (18)$$

เมื่อ

$p^{(k+1)}$ คือ ค่าคงที่ในโครงข่าย ที่ถูกปรับค่าแล้ว

$p^{(k)}$ คือ ค่าคงที่ ก่อนจะถูกปรับค่า

μ คือ อัตราการเรียนรู้ที่ใช้ในโครงข่าย

$\hat{\varepsilon}^{(k)}$ คือ ค่าเฉลี่ยความผิดพลาดของการรบกวนค่าน้ำหนักทั้งสองด้าน

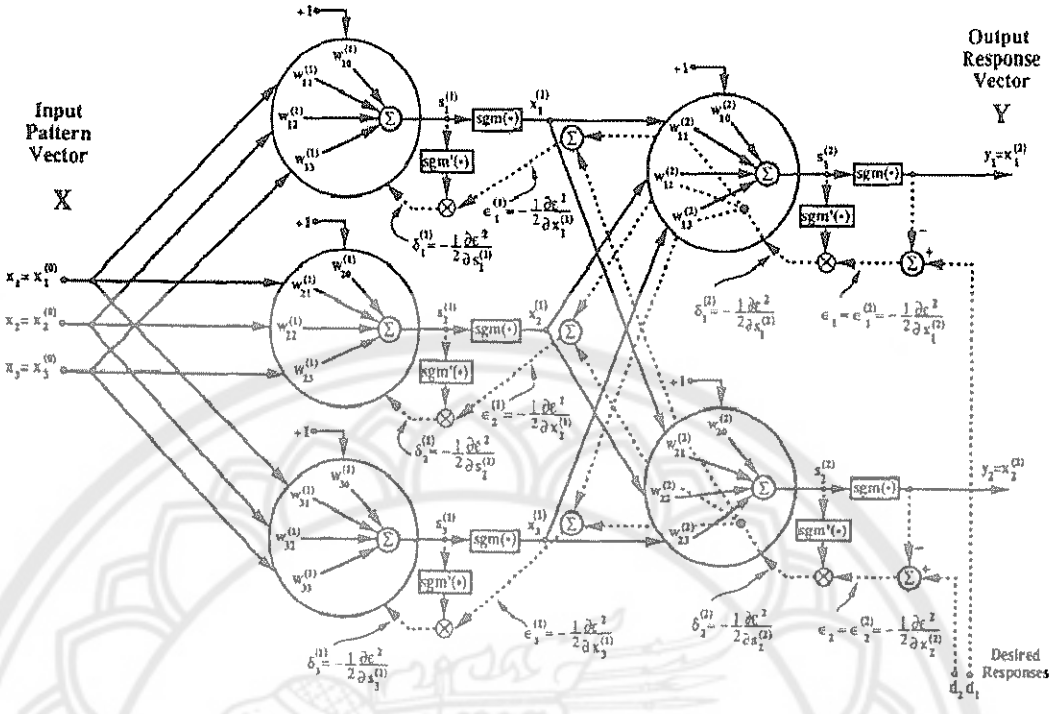
$\pi^{(k)}$ คือ ค่าการรบกวนแบบสุ่มของตัวแปร $p^{(k)}$ โดยกำหนดขนาด

เป็นค่าคงที่ โดยทั่วไปมีค่าเท่ากับค่าความแปรปรวนของตัวแปรสุ่ม $p^{(k)}$

การปรับค่าน้ำหนักแบบนี้อาศัยการรบกวนค่าน้ำหนักทั้งสองด้าน (Two Side Perturbation) เมื่อกำหนดให้ขนาดของค่าการรบกวน คือ $\pi_i^{(k)}$ มีค่าเท่ากับค่าความแปรปรวนให้กับตัวแปร $p^{(k)}$ หรือ $\pm\sigma$

ตัวอย่างของงานวิจัยที่ศึกษาเกี่ยวกับอัลกอริทึมฝึกสอนโครงข่ายประสาทเทียมที่ประมวลผลได้รวดเร็วและมีประสิทธิภาพสูง

Bernard Widrow and Michael A. LEHR (1960) นำเสนอกระบวนการ การประมวลผลในโครงข่ายแบบการแพร่ค่าย้อนกลับ (Backpropagation for Network) เมื่อนำไปประยุกต์ใช้กับโครงข่ายที่เชื่อมต่อกันเป็นโครงข่าย ดังแสดงในภาพ 5



ภาพ 5 ตัวอย่างโครงข่าย 2 ชั้น (Layer) เมื่อใช้กับการเรียนรู้แบบแพร่ค่าย้อนกลับ

โครงข่ายในภาพ 5 จะทำการปรับค่าน้ำหนักโครงข่าย ด้วยหลักการแพร่ค่าย้อนกลับ (Backpropagation) ในทิศทางตรงกันข้ามโดยใช้ค่าเกรเดียนท์ชั่วขณะ (Instantaneous Gradient) ซึ่งคำนวณได้ตามสมการ (19)

$$\widehat{\nabla}_{\mathrm{k}} = \frac{\partial \varepsilon_k^2}{\partial \mathrm{W_k}} = \begin{matrix} \frac{\partial \varepsilon_k^2}{\partial W_{1k}} \\ \vdots \\ \frac{\partial \varepsilon_k^2}{\partial W_{mk}} \end{matrix} \qquad (19)$$

เมื่อ $\mathrm{W_k}$ เป็นเวกเตอร์ค่าน้ำหนักโครงข่าย มีจำนวนองค์ประกอบทั้งหมดในเวกเตอร์เท่ากับ m ค่าเขียนแทนได้เป็น $\mathrm{W_k} = [w_{1k}, w_{2k}, \dots, w_{mk}]$ เมื่อ k เป็นค่าคงที่ ดัชนีบอกลำดับครั้งของการคำนวณ

ผลรวมค่าความผิดพลาดกำลังสอง (Sum Square Error) หรือ ε_k^2 ของแต่ละเอาท์พุทคำนวณได้ตามสมการ (20)

$$\varepsilon_{\mathrm{k}}^2 = \sum_{i=1}^{N_y} \varepsilon_{ik}^2 \qquad (20)$$

เมื่อ

i คือ จำนวนโนดเอาท์พุทโครงข่าย ในภาพ 4 พบว่า $i = 1,2$ เนื่องจากมีโนดเอาท์พุตจำนวน 2 โนด

K คือ ดัชนีบอกลำดับครั้งของการคำนวณ

N_y คือ จำนวนโนดเอาท์พุต ในชั้นเอาท์พุต ซึ่งในภาพ 5 $N_y = 2$

จากตัวอย่างโครงข่ายภาพ 5 ผลรวมค่าความผิดพลาดกำลังสอง (Sum Square Error) ของแต่ละโนดเอาท์พุทโครงข่าย คำนวณได้ตามสมการ (21)

$$\varepsilon_k^2 = (d_1 - y_1)^2 + (d_2 - y_2)^2 \qquad (21)$$

เมื่อ

d_1 คือ เอาท์พุตฝึกสอนโครงข่าย โนดเอาท์พุต ที่ 1

y_1 คือ เอาท์พุตโครงข่าย โนดเอาท์พุต ที่ 1

d_2 คือ เอาท์พุตฝึกสอนโครงข่าย โนดเอาท์พุต ที่ 2

y_2 คือ เอาท์พุตโครงข่าย โนดเอาท์พุต ที่ 2

กระบวนการฝึกสอนโครงข่ายแบบแพร่ค่าย้อนกลับ เริ่มต้นจาก นำเวกเตอร์อินพุทฝึกสอน (X) เข้าสู่โครงข่าย สร้างเอาท์พุทโครงข่าย (y) และคำนวณค่าความผิดพลาดของแต่ละเอาท์พุท จากนั้นนำค่าเอาท์พุทโครงข่าย (y) เปรียบเทียบกับค่าเอาท์พุทฝึกสอน (d_k) เพื่อคำนวณค่าความผิดพลาดกำลังสอง ε_k^2 จากนั้นนำค่าความผิดพลาดกำลังสองไปคำนวณหาค่าเกรเดียนท์ โดยการหาอนุพันธ์ค่าความผิดพลาดกำลังสอง (Square Error Derivative) เปรียบเทียบกับค่าน้ำหนักโครงข่าย ($\frac{\partial \varepsilon_k^2}{\partial W_k}$) ซึ่งแทนด้วยสัญลักษณ์ δ ในแต่ละเอาท์พุท จากนั้น ปรับค่าน้ำหนักโครงข่ายด้วยค่าเกรเดียนท์ที่คำนวณได้ และนำเวกเตอร์อินพุทฝึกสอนโครงข่ายตัวถัดไปเข้าสู่โครงข่าย และทำตามกระบวนการแบบเดิม การกำหนดค่าน้ำหนักในตอนเริ่มต้นการฝึกสอนโครงข่ายโดยทั่วไปแล้วจะกำหนดค่าน้ำหนักโครงข่ายในช่วงเริ่มต้นเป็นค่าสุ่มน้อยๆ

การเรียนรู้แบบแพร่ค่าย้อนกลับในภาพ 5 วงกลมใหญ่ทั้ง 5 วงกลมเปรียบเสมือน Adaptive Linear Combiner (Adaline) แต่ละตัว โดยที่สัญลักษณ์ที่เป็นเส้นทึบ แทน เส้นทางการคำนวณไปข้างหน้าภายในโครงข่าย และสัญลักษณ์เส้นประ แทน การคำนวณย้อนกลับตลอดโครงข่าย ซึ่งจะเกี่ยวข้องกับการคำนวณค่าเกรเดียนท์ (δ) โดยการหาอนุพันธ์

จากที่กล่าวมาก่อนหน้านั้น เมื่อเวกเตอร์อินพุทฝึกสอน (X) เข้าสู่โครงข่าย จะผ่านการคำนวณค่าความผิดพลาดกำลังสอง (Square Error) โดยเปรียบเทียบระหว่างเอาท์พุทโครงข่ายและเอาท์พุทฝึกสอนโครงข่าย ขั้นตอนต่อไป คือ การแพร่ค่าเกรเดียนท์ที่คำนวณได้ย้อนกลับเข้ามาในโครงข่ายเพื่อปรับค่าน้ำหนัก เรียกว่า Back-Propagation

การคำนวณอนุพันธ์ค่าความผิดพลาดกำลังสองของ Adaline ลำดับที่ j ในชั้นที่ l นิยามดังสมการ (22)

$$\delta_j^{(l)} = -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial s_j^{(l)}} \tag{22}$$

จากสมการ (22) เราสามารถคำนวณอนุพันธ์ค่าความผิดพลาดกำลังสองของ Adaline ลำดับที่ 1 ในชั้น 2 ของโครงข่าย ซึ่งจะเขียนแทนได้ คือ $\delta_1^{(2)}$ ดังสมการ (23)

$$\delta_1^{(2)} = -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial s_1^{(2)}} \tag{23}$$

เมื่อ

$s_1^{(2)}$ คือ เอาท์พุตโครงข่ายของ Adaline ลำดับที่ 1 ชั้นที่ 2 ในโครงข่าย

แทนค่า ε^2 จากสมการ (21) ลงในสมการ (23) ได้ตามสมการ (24) และ (25)

$$\delta_1^{(2)} = -\frac{1}{2}\frac{\partial (d_1 - y_1)^2 + (d_2 - y_2)^2}{\partial s_1^{(2)}} \tag{24}$$

แทนค่า $y_1 = sgm\left(s_1^{(2)}\right)$ และ $y_2 = sgm\left(s_2^{(2)}\right)$ ลงในสมการ (24) ได้สมการ (25)

$$= -\frac{1}{2}\frac{\partial \left(d_1 - sgm\left(s_1^{(2)}\right)\right)^2}{\partial s_1^{(2)}} - \frac{1}{2}\frac{\partial \left(d_2 - sgm\left(s_1^{(2)}\right)\right)^2}{\partial s_1^{(2)}} \tag{25}$$

สังเกตในพจน์ที่สองของสมการ (25) มีค่าเท่ากับศูนย์ ได้สมการ (26)

$$\delta_1^{(2)} = -\frac{1}{2}\frac{\partial\left(d_1 - sgm\left(s_1^{(2)}\right)\right)^2}{\partial s_1^{(2)}} \tag{26}$$

จากสมการ (26) ตัวแปร d_1 และ $s_1^{(2)}$ เป็นตัวแปรอิสระไม่เกี่ยวข้องกัน ดังนั้น เมื่อทำการอนุพันธ์แล้วได้ดังสมการ (27)

$$\delta_1^{(2)} = -\left(d_1 - sgm\left(s_1^{(2)}\right)\right)\frac{\partial\left(-sgm\left(s_1^{(2)}\right)\right)^2}{\partial s_1^{(2)}} \tag{27}$$

$$= \left(d_1 - sgm\left(s_1^{(2)}\right)\right) sgm'\left(s_1^{(2)}\right) \tag{28}$$

เมื่อ $\left(d_1 - sgm\left(s_1^{(2)}\right)\right)$ สามารถเขียนแทนได้ด้วย $\varepsilon_1^{(2)}$ ดังนั้น อนุพันธ์ค่าความผิดพลาดกำลังสองของ Adaline ลำดับที่ 1 ชั้นที่ 2 ในโครงข่าย ที่สอดคล้องกับสมการที่ (28) จะกลายเป็น

$$\delta_1^{(2)} = \varepsilon_1^{(2)} sgm'\left(s_1^{(2)}\right) \tag{29}$$

จากภาพ 5 กำหนดให้ $\delta_1^{(1)}$ แทน อนุพันธ์ค่าความผิดพลาดกำลังสองของ Adaline ลำดับที่ 1 ชั้นที่ 1 ในโครงข่าย เขียนแทนได้ดังสมการ (30)

$$\delta_1^{(1)} \equiv -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial s_1^{(1)}} \tag{30}$$

เมื่อใช้คุณสมบัติของ กฎลูกโซ่ (Chain Rule) โดยสังเกตจากค่า ε^2 จะถูกกำหนดค่าในพจน์ของ $s_1^{(2)}$ และ $s_2^{(2)}$ เขียนแทนได้ตามสมการ (31)

$$\delta_1^{(1)} = -\frac{1}{2}\left(\frac{\partial \varepsilon^2}{\partial s_1^{(2)}}\frac{\partial s_1^{(2)}}{s_1^{(1)}} + \frac{\partial \varepsilon^2}{\partial s_2^{(2)}}\frac{\partial s_2^{(2)}}{s_1^{(1)}}\right) \tag{31}$$

เมื่อ

$s_1^{(2)}$ คือ เอาท์พุตโครงข่ายของ Adaline ลำดับที่ 1 ชั้นที่ 2 ในโครงข่าย

$s_2^{(2)}$ คือ เอาท์พุตโครงข่ายของ Adaline ลำดับที่ 2 ชั้นที่ 2 ในโครงข่าย

จากภาพ 5 จะได้ความสัมพันธ์ของ $s_1^{(2)}$, $s_2^{(2)}$ และ $\delta_1^{(1)}$, $\delta_2^{(2)}$

แทนค่า $s_1^{(2)} = (w_{10}^{(2)} + \sum_{i=1}^{3} w_{1i}^{(2)} sgm(s_i^{(1)}))$ และ $s_2^{(2)} = (w_{20}^{(2)} + \sum_{i=1}^{3} w_{2i}^{(2)} sgm(s_i^{(1)})$

ลงในสมการ (31) จะได้สมการ (32) และ (33) ตามลำดับ

เมื่อ

$w_{10}^{(2)}$ คือ ค่าน้ำหนักไบแอสของ adaline ลำดับที่ 1 ชั้นที่ 2 ในโครงข่าย (เมื่อ 0 แสดว่าเป็นค่าน้ำหนักไปแอส)

$w_{20}^{(2)}$ คือ ค่าน้ำหนักไบแอสของ adaline ลำดับที่ 2 ชั้นที่ 2 ในโครงข่าย (เมื่อ 0 แสดงว่าเป็นค่าน้ำหนักไปแอส)

i คือ จำนวน adaline ทั้งหมดในชั้นที่ 1 ดังนั้น

$w_{1i}^{(2)}$ คือ เวกเตอร์ค่าน้ำหนักของ adaline ลำดับที่ 1 ชั้นที่ 2 ในโครงข่าย สามารถเขียนแทนได้ $w_{1i}^{(2)} = [w_{11}^{(2)}, w_{12}^{(2)}, w_{13}^{(2)}]$

$s_i^{(1)}$ คือ เวกเตอร์เอาท์พุททั้งหมดของ adaline ชั้นที่ 2 ในโครงข่าย สามารถเขียนแทนได้ $s_i^{(1)} = [s_1^{(1)}, s_2^{(1)}, s_3^{(1)}]$

$$\delta_1^{(1)} = \delta_1^{(2)} \frac{\partial s_1^{(2)}}{s_1^{(1)}} + \delta_2^{(2)} \frac{\partial s_2^{(2)}}{s_1^{(1)}}) \tag{32}$$

$$= \delta_1^{(2)} \frac{\partial}{s_1^{(1)}} (w_{10}^{(2)} + \sum_{i=1}^{3} w_{1i}^{(2)} sgm(s_i^{(1)}))$$

$$+ \delta_2^{(2)} \frac{\partial}{s_1^{(1)}} (w_{20}^{(2)} + \sum_{i=1}^{3} w_{2i}^{(2)} sgm(s_i^{(1)}) \tag{33}$$

เมื่อ

สังเกตที่ชั้นซ่อน (l) , $\frac{\partial [sgm(s_i^{(l)})]}{\partial s_j^{(l)}} = 0$ เมื่อ $i \neq j$ จะได้ดังสมการ (34)

$$\delta_1^{(1)} = \delta_1^{(2)} w_{11}^{(2)} sgm'(s_1^{(1)}) + \delta_2^{(2)} w_{21}^{(2)} sgm'(s_1^{(1)})$$

$$\equiv \left[\delta_1^{(2)} w_{11}^{(2)} - \delta_2^{(2)} w_{21}^{(2)}\right] sgm'\left(s_1^{(1)}\right) \tag{34}$$

เรานิยาย $\varepsilon_1^{(1)}$ ตามสมการ (35)

$$\varepsilon_1^{(1)} \equiv \delta_1^{(2)} w_{11}^{(2)} - \delta_2^{(2)} w_{21}^{(2)} \tag{35}$$

ดังนั้น $\delta_1^{(1)}$ จากสมการ (35) จะกลายเป็น

$$\delta_1^{(1)} = \varepsilon_1^{(1)} sgm'\left(s_1^{(1)}\right) \tag{36}$$

จากภาพ 5 เราทราบได้ว่า การหาค่าอนุพันธ์ค่าความผิดพลาดกำลังสอง ของ Adaline ลำดับที่ 1 ชั้นที่ 1 ของโครงข่าย ซึ่งเขียนแทนได้เป็น $\delta_1^{(1)}$ สามารถคำนวณได้ตามสมการ (36)

หลังจากคำนวณค่าเกรเดียนท์ (δ) โดยการหาค่าอนุพันธ์ของแต่ละ Adaline ในโครงข่ายแล้ว ใช้ค่าเกรเดียนท์ที่คำนวณได้ ไปปรับค่าน้ำหนักในโครงข่าย โดยกำหนดให้เวกเตอร์ค่าน้ำหนักโครงข่ายมีค่าเป็น $\mathbf{W_k}$ และเวกเตอร์อินพุตโครงข่ายมีค่าเป็น $\mathbf{X_k}$

ดังนั้น ค่าเอาท์พุทโครงข่ายย่อย คำนวณได้ตามสมการ (37)

$$s_k = \mathbf{W_k^T X_k} \tag{37}$$

ค่าเกรเดียนท์ของแต่ละ Adaline ลำดับที่ k คำนวณได้ดังสมการ (38)

$$\hat{\nabla}_k = \frac{\partial \varepsilon_k^2}{\partial \mathbf{W_k}} \tag{38}$$

เราสามารถเขียนได้เป็นอีกรูปแบบหนึ่ง ตามสมการ (39)

$$\hat{\nabla}_k = \frac{\partial \varepsilon_k^2}{\partial \mathbf{W_k}} = \frac{\partial \varepsilon_k^2}{\partial s_k} \frac{\partial s_k}{\partial \mathbf{W_k}} \tag{39}$$

สังเกตว่า ค่า $\mathbf{W_k}$ และ $\mathbf{X_k}$ เป็นตัวแปรอิสระ ดังนั้น หลังจากทำการอนุพันธ์จะได้ดังสมการ (40)

$$\frac{\partial s_k}{\partial \mathbf{W_k}} = \frac{\partial \mathbf{W_k^T X_k}}{\partial \mathbf{W_k}} = \mathbf{X_k} \tag{40}$$

ดังนั้น จะได้เป็นค่าเกรเดียนท์เป็น ดังสมการ (41)

$$\hat{\nabla}_{\mathrm{k}} = \frac{\partial \varepsilon_k^2}{\partial s_k} \mathbf{X_k} \tag{41}$$

จะได้ δ_k ตามสมการ (42)

$$\delta_k = -\frac{1}{2}\frac{\partial \varepsilon_{\mathrm{k}}^2}{\partial s_{\mathrm{k}}} \tag{42}$$

แทนค่า δ_k จากสมการ (42) ลงในสมการ (41) จะได้ค่าเกรเดียนท์ตามสมการ (43)

$$\hat{\nabla}_{\mathrm{k}} = -2\delta_k \mathbf{X_k} \tag{43}$$

จากนั้นปรับค่าน้ำหนักโครงข่ายด้วยวิธีของ Steepest descent โดยปรับค่าน้ำหนักในทิศทางตรงข้ามด้วยค่าเกรเดียนท์ที่คำนวณได้ สมการปรับค่าน้ำหนักโครงข่ายแสดงดังสมการ (44)

$$\mathbf{w_{k+1}^{(2)}} = \mathbf{w_k^2} + \mu\left(-\hat{\nabla}_k\right) = \mathbf{w_k^2} + \mu\delta_k^{(2)} \mathbf{X_k^{(1)}} \tag{44}$$

จากนั้นปรับค่าน้ำหนักโครงข่ายโดย ด้วยวิธีของ Steepest descent เอาท์พุทโครงข่ายตัวแรกในชั้นเอาท์พุท (y_1) แทนค่าเกรเดียนท์ที่คำนวณได้สมการ (30) $\delta_1^{(2)} = \varepsilon_1^{(2)} sgm'\left(s_1^{(2)}\right)$ ลงในสมการ (44) จะได้สมการปรับค่าน้ำหนักของเอาท์พุท y_1 ในภาพ 5 ได้ตามสมการ (45)

$$\mathbf{w_{k+1}^{(2)}} = \mathbf{W_k} + \mu\left(-\hat{\nabla}_k\right) = \mathbf{w_k^2} + \mu\varepsilon_1^{(2)} sgm'\left(s_1^{(2)}\right) \mathbf{X_k^{(1)}} \tag{45}$$

Amir Abolfazl Suratgar, et al. (2005) ได้เสนออัลกอริทึมฝึกสอนโครงข่ายประสาทเทียมที่มีการลู่เข้าที่รวดเร็วและมีประสิทธิภาพสูง ซึ่งพัฒนามาจาก Levenberg-Marquardt Algorithm โดยใช้การคำนวณหาค่าเมตริกซ์อนุพันธ์อันดับสอง เรียกว่า Hessian Matrix เพื่อใช้ปรับค่าน้ำหนักโครงข่าย ซึ่งแทนด้วยสัญลักษณ์ $\mathbf{H}$ แต่เนื่องจากการคำนวณค่าของเฮเซี่ยนเมตริกซ์นั้น

มีความยุ่งยาก อีกทั้งกระบวนการคำนวณนั้นมีความซับซ้อน ดังนั้น Amir Abolfazl Suratgar, et al. จึงได้ทำการประมาณค่าของ $\mathbf{H}$ จากการคำนวณค่าจาโคเบียนแทน ซึ่งใน Levenberg-Marquardt Algorithm จะประมาณค่าของ $\mathbf{H}$ ดังสมการ (46) เมื่อ $\mathbf{J}$ เรียกว่า ค่าจาโคเบียน (Jacobian)

$$\mathbf{H} = \mathbf{J}^{\mathbf{T}}\mathbf{J} \tag{46}$$

เช่นเดียวกับค่าเกรเดียนท์คำนวณได้จากการนำค่าจาโคเบียน ($\mathbf{J}$) คูณด้วยค่าเวกเตอร์ค่าความผิดพลาด ($\mathbf{e}$) คำนวณได้ตามสมการ (47)

$$\mathbf{g} = \mathbf{J}^{\mathbf{T}}\mathbf{e} \tag{47}$$

เมื่อ

$\mathbf{H}$ คือ Hessian Matrix

$\mathbf{J}$ คือ Jacobian Matrix

$\mathbf{e}$ คือ เวกเตอร์ค่าความผิดพลาด

นิยาม กำหนดให้ $\mathbf{X}_k = [x_1 \;\; x_2 \;\; \ldots \;\; x_n]^T$และ $f(\mathbf{X})$ เป็นสเกลาร์ฟังก์ชันของ x ดังนั้น อนุพันธ์ลำดับที่ 2 ของ $f(\mathbf{X})$ เทียบกับ x ซึ่งเรียกว่า Hessian matrix หรือ Hessian นิยามโดย

$$\mathbf{H}_{\mathbf{f}} = \nabla^2\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2}f(x) & \frac{\partial^2}{\partial x_1 x_2}f(x) & \cdots & \frac{\partial^2}{\partial x_1 x_n}f(x) \\ \frac{\partial^2}{\partial x_2 x_1}f(x) & \frac{\partial^2}{\partial x_2^2}f(x) & \cdots & \frac{\partial^2}{\partial x_2 x_n}f(x) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2}{\partial x_n x_1}f(x) & \frac{\partial^2}{\partial x_n x_2}f(x) & \cdots & \frac{\partial^2}{\partial x_n^2}f(x) \end{bmatrix} \tag{48}$$

$\mathbf{J}$ คือ Jacobian ซึ่งสามารถนิยามได้ดังต่อไปนี้

นิยาม กำหนดให้ $\mathbf{X}_k = [x_1 \;\; x_2 \;\; \ldots \;\; x_n]^T$ และ $f(\mathbf{X})$ เป็นสเกลาร์ฟังก์ชันของ x โดยที่ $f(\mathbf{X}) = [f_1(x) \;\; f_2(x) \;\; \ldots \;\; f_n(x)]^T$ ดังนั้นอนุพันธ์ของ $f(\mathbf{X})$เทียบกับ x ซึ่งเรียกว่า Jacobian

$$\mathbf{J}_{\mathbf{f}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{49}$$

$\mathbf{g}$ คือ gradient ซึ่งสามารถนิยามได้ดังต่อไปนี้

นิยาม กำหนดให้ $\mathbf{X}_k = [x_1 \;\; x_2 \;\; \ldots \;\; x_n]^T$ และ $f(\mathbf{X})$ เป็นสเกลาร์ฟังก์ชันของ x ดังนั้นอนุพันธ์ ของ $f(\mathbf{X})$ เทียบกับ x ซึ่งเรียกว่า เกรเดียนท์เวกเตอร์ หรือ เกรเดียนท์

$$\nabla \mathbf{f}(\mathbf{x}) = [\frac{\partial}{\partial x_1} f(x) \quad \frac{\partial}{\partial x_2} f(x) \quad \cdots \quad \frac{\partial}{\partial x_m} f(x)]^T \tag{50}$$

$\mathbf{e}$ คือ เวกเตอร์ค่าความผิดพลาดของโครงข่าย มีค่าเป็นดังสมการ (51)

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_n \end{bmatrix}^{\mathbf{T}} \tag{51}$$

สมการที่ใช้ปรับค่าน้ำหนักของโครงข่าย มีค่าเป็นดังสมการ (52)

$$\mathbf{x}_{\mathbf{k+1}} = \mathbf{x}_{\mathbf{k}} - [\mathbf{J}^{\mathbf{T}}\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}^{\mathbf{T}}\mathbf{e} \tag{52}$$

ตัวอย่างของงานวิจัยที่ศึกษาเกี่ยวกับการหาค่าที่เหมาะสมโดยอาศัยการรบกวนแบบสุ่ม

Bernard Widrow and Samuel D.Stearns (1985) ได้ศึกษาการหาค่าเกรเดียนในลักษณะสุ่ม และใช้ค่าเกรเดียนท์ที่ได้นั้น ปรับค่าน้ำหนักโครงข่าย โดยเรียกวิธีดังกล่าวว่า Linear Random-Search algorithm (LRS)

โดยสมการที่ใช้ในการ ปรับค่าน้ำหนักที่นำเสนอ มีค่าเป็นดังสมการ (53)

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \frac{\mu}{\sigma^2}[\xi\ (\mathbf{W}_k) - \xi\ (\mathbf{W}_k + \mathbf{U}_k)]\mathbf{U}_k \tag{53}$$

ว
วฤ
235
ต149ก
2554
25 ก.ค. 2554
สำนักหอสมุด
23 ธ.ค. 2554
1.5532417

เมื่อ

U_k คือ ค่าสุ่มน้อยมาก

$\xi(W_k)$ คือ ค่าเฉลี่ยความผิดพลาดกำลังสองของค่าน้ำหนัก

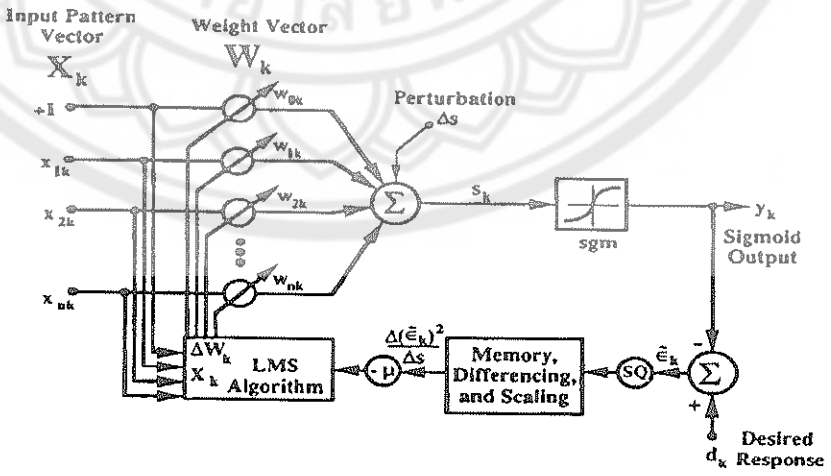
$\xi(W_k + U_k)$ คือ เป็นค่าเฉลี่ยความผิดพลาดกำลังสองของค่าน้ำหนักๆ ที่บวกด้วยค่าสุ่มน้อยมาก

μ คือ ค่าคงที่มีผลต่ออัตราการลู่เข้า

σ^2 คือ ค่าคงที่ที่มีผลต่อเสถียรภาพและช่วงขั้น (step) ของการลู่เข้า

วิธีการปรับค่าน้ำหนักด้วยวิธีนี้คล้ายคลึงกับวิธีของ Least Mean Square ซึ่งมีอัตราการลู่เข้าค่าน้ำหนักเหมาะสมอย่างช้าๆ แต่มีเสถียรสูง บางครั้งถูกนำไปใช้ฝึกสอนโครงข่ายแทนวิธี Least Mean Square ในกรณีที่ไม่สามารถหาค่าเกรเดียนท์ได้โดยตรงเนื่องจากการเปลี่ยนแปลงค่าความผิดพลาดกำลังสองนั้นมีการเปลี่ยนแปลงน้อยมาก ดังนั้น จึงใช้หลักการสุ่มเพื่อรบกวนค่าน้ำหนัก และนำค่าน้ำหนักที่ถูกเปลี่ยนแปลงไปปรับค่าน้ำหนักโครงข่าย

Bernard Widrow and Michael A. LEHR (1960) ได้นำเสนอกฎ Madaline Rule III หรือ MR III เพื่อเป็นตัวอย่างของโครงข่ายที่มีการปรับโครงข่ายแบบไม่เป็นเชิงเส้น (Sigmoid Adaline) จะเห็นได้ว่า ไม่มีการหาค่าอนุพันธ์ของ Sigmoid Function โดยตรง แต่แทนการคำนวณค่าอนุพันธ์ด้วย การรบกวนโนด (Node Perturbation) ด้วยสัญญาณค่ารบกวนน้อยมากๆ (a small perturbation signal) เรียกว่า Δs โดยการบวกค่าน้อยๆ เข้ากับเอาท์พุทเชิงเส้น (s_k) ที่โนดเอาท์พุทเขียนแทนได้เป็น $s_k + \Delta s$, เอาท์พุตโครงข่ายที่ได้มีค่าเป็น y_k และค่าความผิดพลาดของโครงข่ายมีค่าเป็น ε_k



ภาพ 6 การประมวลผลด้วยอัลกอริทึม MRIII

จากภาพ 6 สามารถคำนวณค่าเกรเดียนท์ประมาณชั่วขณะ (instantaneous estimated gradient) ได้จากสมการ (54)

$$\hat{\nabla}_k = \frac{\partial(\hat{\varepsilon}_k)^2}{\partial \mathbf{W_k}} = \frac{\partial(\hat{\varepsilon}_k)^2 \partial s_k}{\partial s_k \partial \mathbf{W_k}} = \frac{\partial(\hat{\varepsilon}_k)^2}{\partial s_k}\mathbf{X}_k \tag{54}$$

เนื่องจาก ค่า Δs มีค่าน้อยมาก จึงทำการประมาณค่าเกรเดียนท์ชั่วขณะได้ดังสมการ (55)

$$\hat{\nabla}_\mathrm{k} = \frac{\Delta(\hat{\varepsilon}_\mathrm{k})^2}{\Delta s_\mathrm{k}}\mathrm{X}_k \tag{55}$$

อีกวิธีหนึ่ง ของการประมาณค่าเกรเดียนท์ชั่วขณะ คือ การอนุพันธ์สมการ (54) ได้ดังสมการ (56)

$$\hat{\nabla}_\mathrm{k} = \frac{\partial(\hat{\varepsilon}_\mathrm{k})^2}{\partial s_\mathrm{k}}\mathrm{X}_k = 2\hat{\varepsilon}_\mathrm{k}\frac{\partial \hat{\varepsilon}_\mathrm{k}}{\partial s_\mathrm{k}}\mathrm{X}_\mathrm{k} \approx 2\hat{\varepsilon}_\mathrm{k}\frac{\Delta \hat{\varepsilon}_\mathrm{k}}{\Delta s_\mathrm{k}}\mathrm{X}_k \tag{56}$$

การหาค่าเกรเดียนท์ของอัลกอรึทึม MRIII ที่ใช้สำหรับโครงข่ายแบบ Sigmoid Adaline โดยอาศัยพื้นฐานของ Steepest Descent จะสามารถปรับค่าน้ำหนักได้ตามสมการ (57)

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu\frac{\Delta(\hat{\varepsilon}_k^2)}{\Delta s}\mathbf{X}_k \tag{57}$$

หรือ

$$\mathrm{W}_{k+1} = \mathrm{W}_k - 2\mu\frac{\Delta(\hat{\varepsilon}_k)}{\Delta s}\mathrm{X}_k \tag{58}$$

เมื่อพิจารณา ค่าเฉลี่ยกำลังสองของฟังก์ชั่นซิกมอยด์ (mean square of the sigmoid) มีค่าดังสมการ (59)

$$\varepsilon_k = d_k - y_k = d_k - sgm(s_k) \tag{59}$$

การเพิ่ม Δs เข้าไปใน Adaline ทำให้ค่า ε_k กลายเป็น

$$\Delta\hat{\varepsilon}_k = -\Delta y_k \tag{60}$$

ดังนั้น สมการในการปรับค่าน้ำหนัก มีค่าเป็นไปตามสมการ (61)

$$\mathbf{W}_{k+1} = \mathbf{W}_k - 2\mu\frac{\Delta y_k}{\Delta \mathbf{s}}\mathbf{X}_k \tag{61}$$

เนื่องจากค่า Δs มีค่าน้อยมาก ดังนั้น อัตราการเพิ่มขึ้นของค่าน้ำหนัก อาจจะแทนด้วยค่าที่เพิ่มขึ้นได้ด้วยการหาอนุพันธ์ของเอาท์พุตที่เปลี่ยนแปลงไป จะกลายเป็น

$$\mathbf{W}_{k+1} = \mathbf{W}_k - 2\mu\hat{\varepsilon}_k\frac{\partial y_k}{\partial \mathbf{s}}\mathbf{X}_k \tag{62}$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k - 2\mu\hat{\varepsilon}_k \mathrm{sgm}(s_k)\mathbf{X}_k \tag{63}$$

อัลกอริทึมนี้จะเทียบเท่ากับวิธีของอัลกอริทึม Backpropagation of the Sigmoid Adaline ถ้าค่าการรบกวน (Perturbation) Δs มีค่าที่น้อยมากๆ ซึ่งทำให้อัลกอริทึมนี้มีความคงทน (Robust) แม้ว่าจะนำไปใช้งานกับเครื่องมือแบบอนาล็อก

Akaraphunt Vongkunghae (2005) ในเรื่องการหาค่าที่เหมาะสม (Optimization) นั้น ได้นำหลักการ การรบกวนแบบสุ่ม (random perturbation) มาประยุกต์ใช้โดยกำหนดให้ ค่าการรบกวน มีคุณลักษณะการกระจายตัวแปรสุ่มแบบเกาซ์เซี่ยน (Gaussian Distribution) รบกวนค่าน้ำหนักของโครงข่ายเพื่อค้นหาค่าเหมาะสม
โดยสมการที่ใช้ปรับค่าน้ำหนัก เป็นดังนี้

$$\Delta e = e(W_n + v\Delta P) - e(W_n - v\Delta P) \tag{64}$$

$$W_{n+1} = W_n - \mu \cdot \Delta e \cdot v\Delta P \tag{65}$$

เมื่อ

W_{n+1} คือ ค่าน้ำหนักที่ปรับค่าแล้ว

W_n คือ ค่าน้ำหนักตัวเดิม

ΔP คือ ค่าการรบกวน มีลักษณะการกระจายตัวแปรสุ่มเกาซ์เซี่ยน และมีค่าเบี่ยงเบนมาตรฐาน เท่ากับ 1.0

v คือ เบี่ยงเบนมาตรฐานควบคุมการกระจายตัวของค่าการรบกวน

$e(W_n + v\Delta P)$ คือ เฉลี่ยความผิดพลาดกำลังสองของค่าน้ำหนักที่บวกค่าการบกวน

$e(W_n - v\Delta P)$ คือ เฉลี่ยความผิดพลาดกำลังสองของค่าน้ำหนักที่ลบค่าการบกวน

Δe คือ ผลต่างของค่าเฉลี่ยความผิดพลาดกำลังสองระหว่างค่าน้ำหนักบวกด้วยการค่าการรบกวนและค่าน้ำหนักลบด้วยค่าการรบกวน

μ คือ ค่าอัตราการเรียนรู้

ซึ่งในวิทยานิพนธ์เรื่องนี้ ได้ใช้หลักการการหาค่าเหมาะสมจาก Akaraphunt Vongkunghae (2005) เพื่อใช้ฝึกสอนโครงข่ายประสาทเทียมแบบป้อนผลการคำนวณไปข้างหน้าเพื่อเปรียบเทียบการทำงานกับอัลกอริทึมของ Levenberg-Marquardt backpropagation (LM), Gradient descent with adaptive learning rule backpropagation (GDA) และ Gradient descent backpropagation (GD)