บทที่ 2

แนวความคิดและทฤษฎี

โครงสร้างพื้นฐานทางดนตรีที่สำคัญ (Musical basic material) นั้นเกิดจากตัวแปรที่สำคัญ 2 ตัวที่เกี่ยวข้องคือ ระดับเสียง (Level of tone) และ จังหวะ (rhythm) ซึ่งบอกระยะเวลา ซึ่งตัวแปร 2 ตัวนี้คือ ตัวโน้ต 1 ตัว

ระดับเสียงที่เป็นเสียงเพลงมีระดับแน่นอน เช่นในช่วงคู่แปด จะแบ่งเป็น 7 หรือ 12 หรือ 32 (1,2,3) และจังหวะของทางดนตรี(Musical duration) มีระยะเวลาที่แน่นอนเช่น 8 จังหวะ 4 จังหวะ 2 จังหวะ 1/2 จังหวะ 1/4 จังหวะ 1/8 จังหวะ

ด้วยลักษณะดังกล่าว เราสามารถเขียนในรูปของข้อมูลเป็นตัวเลขแทนทั้งระดับเสียงและจังหวะได้ และสามารถเก็บข้อมูลดังกล่าวไว้ในคอมพิวเตอร์ เมื่อข้อมูลอยู่ในรูปที่ระบบคอมพิวเตอร์เก็บไว้ได้แล้ว ถ้ามีการแก้ไขเพิ่มเติมปรับเสียงให้ดังหรือค่อย การยกระดับของคู่แปด การขยับบันไดเสียง จะสามารถทำได้โดยง่าย

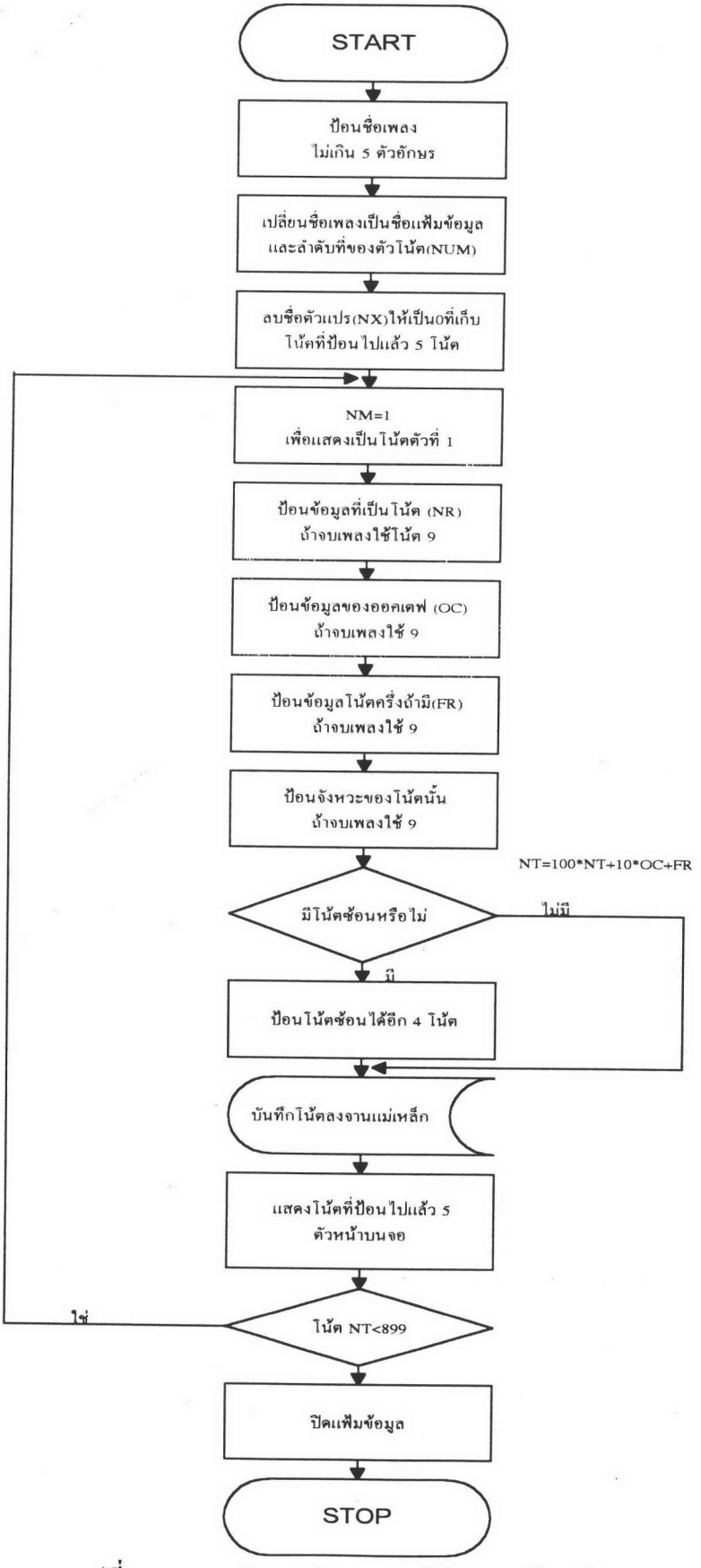
จากความสามารถที่ทำได้ง่ายโดยใช้คอมพิวเตอร์เช่นนี้ เมื่อเราเก็บข้อมูลที่เป็นเพลงต่างๆ ไว้จำนวนมาก จะสามารถนำข้อมูลคอมพิวเตอร์ปรับเสียงให้อยู่ในระดับเดียวกัน และนำข้อมูลดังกล่าวเปรียบกับข้อมูลที่มีอยู่ โดยพิจารณาว่า ถ้าเพลงที่มีท่วงทำนองแบบเดียวกันจะมีคุณลักษณะแบบเดียวกัน

2.1 วิธีการที่ใช้ในการวิเคราะห์ทำนองเพลง

วิธีการแบบเดิม (Traditional approach) (สมชาย ทยานยง, 2528) ประกอบด้วย 7 ขั้นตอน คือ

2.1.1 ตรวจสอบจำนวนข้อมูลและชื่อเพลงที่เก็บในเทป

2.1.2 วิเคราะห์ความห่างของตัวโน้ตที่วางเรียงกันทีละตัว



รูปที่ 2-1 แสดงผังการทำงานของโปรแกรมป้อนข้อมูล

2.1.3 วิเคราะห์ทำนองโดยวิธีทาบชุดของโน้ต(Super Impose) ของ 2 เพลง ชุดละ 10 โน้ต และปรับระดับโน้ต (Transpose) ทั้งโน้ตและจะปรับจังหวะแล้วบันทึกชื่อเพลงทั้ง 2 เพลง และโน้ตจังหวะที่ตรงกันลงม้วนเทปอีก 1 ม้วน ลักษณะการทาบของชุดโน้ต 1 ครั้ง และเลื่อนโน้ตของเพลงที่ถูกเปรียบเทียบไปโน้ตที่ 1 ตัว นอกจากนี้โน้ตที่มีจังหวะยาวมากกว่า 1 จะถูกปรับลงมาเป็น 1 เพราะว่าการมีจังหวะยาวนั้นไม่แน่นอนขึ้นกับลักษณะของเพลง เมื่อมีโน้ตที่ตรงกันมากกว่า 3 ครั้ง จะบันทึกเทปข้อมูลคอมพิวเตอร์อีก 1 ม้วน

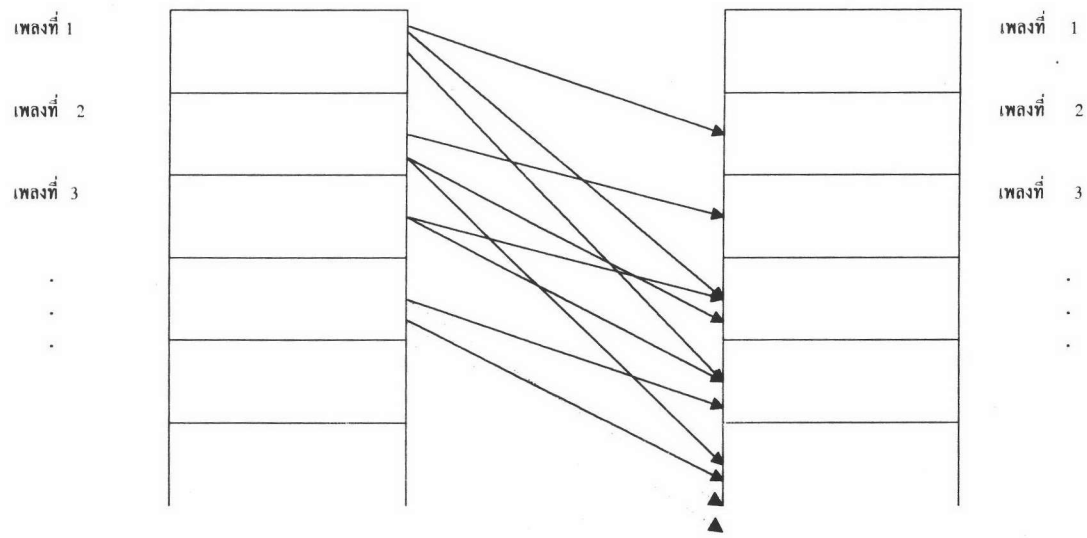
2.1.4 เนื่องจากผลลัพธ์ที่ได้จากข้อ 2.1.3 ได้มาโดยนับจำนวนครั้งที่เกิด เมื่อถึง 4 ครั้งขึ้นไป แต่บางครั้งในจำนวนนั้นมีไม่เท่ากันแทรกอยู่ระหว่างกลางจึงเลือกเอาเฉพาะที่ถึง 4 ครั้งขึ้นไปและต้องต่อเนื่องกันพร้อมกับจำนวนที่เกิด แล้วจึงบันทึกเทปข้อมูลคอมพิวเตอร์ออกมาอีก 1 ม้วน

2.1.5 การปรับตัวโน้ต ผลที่ได้จากข้อ 2.1.4 อาจมีทำนองเดียวกัน แต่โน้ตอยู่คนละระดับ จึงสร้างโปรแกรมปรับระดับข้อมูล(Normalize data) โดยให้ตัวแรกมาอยู่ที่ตัวโทนิค คือ 130 หรือ C ในออคเตฟ 3

2.1.6 การนับจำนวนที่เกิด โดยดูโน้ตและจังหวะทำโดยนำเอาเทปที่ได้จาก ข้อ 2.1.6 มาจัดเรียงลำดับใหม่จากน้อยไปหามาก สร้างโปรแกรมนับจำนวนที่เกิดซ้ำกัน ผลที่ได้บันทึกลงเทป และจัดเรียงโดยดูจากจำนวนที่เกิดจากน้อยไปมาก และพิมพ์ผลออกมาทางเครื่องพิมพ์

2.1.7 การนับจำนวนที่เกิด โดยดูเฉพาะตัวโน้ตทำโดยนำเทปจากข้อ 2.1.5มาจัดเรียงลำดับจากน้อยไปมาก และสร้างโปรแกรมนับจำนวนโน้ตที่เกิดซ้ำซ้อน โดยไม่พิจารณาจังหวะ และจัดพิมพ์ทางเครื่องพิมพ์

การเทียบทำนองกระทำโดยหลักการดังนี้ คือ บันทึกข้อมูลลงเทป 2 ม้วนเหมือนกัน จัดเรียงตามเพลงเหมือนกัน



รูปที่ 2-2 แสดงการเทียบทำนองเพลงแต่ละเพลง

วิธีการทางนิวรอลเนตเวิร์ก (Neural Network Approach)

2.2 นิวรอลเนตเวิร์กเทียม (Artificial Neural Network) (Fausett,1994)

นิวรอลเนตเวิร์กเทียมเป็นระบบการประมวลผลสารสนเทศแบบหนึ่งที่มีลักษณะการทำงานโดยทั่วไปเลียนแบบการทำงานของนิวรอลเนตเวิร์กทางชีววิทยา(biological neural network) นิวรอลเนตเวิร์กเทียมถูกพัฒนาให้เป็นรูปแบบที่ใช้โดยทั่วไป(generalization)จากแบบจำลองทางคณิตศาสตร์ของการรู้จำในมนุษย์(human cognition) หรือ นิวรอลทางชีววิทยา โดยอาศัยสมมติฐานที่ว่า

2.2.1 การประมวลผลสารสนเทศเกิดขึ้นที่หน่วยพื้นฐานหลายๆหน่วยเรียกว่า นิวรอน (neuron)

2.2.2 สัญญานถูกส่งระหว่างนิวรอนผ่านส่วนเชื่อมโยง(connection link)

2.2.3 ส่วนเชื่อมโยงมีความสัมพันธ์กับค่าน้ำหนักคูณ(weight)ด้วยค่าสัญญานที่ส่ง

2.2.4 แต่ละนิวรอนนำค่าแอคติเวชั่นฟังก์ชั่น (activation function) (โดยปกติเป็นฟังก์ชั่นที่ไม่ใช่ฟังก์ชั่นเชิงเส้น(nonlinear function)) ไปใช้กับเนตอินพุต (ผลรวมค่าสัญญานอินพุตที่ถูกถ่วงน้ำหนักแล้ว)เพื่อที่ใช้พิจารณาค่าสัญญานเอาต์พุต

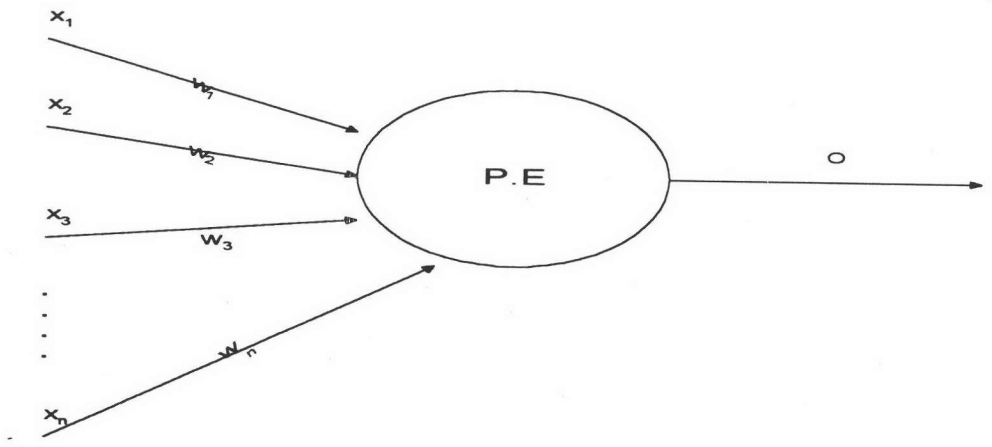
นิวรอลเนตเวิร์กถูกกำหนดลักษณะโดย 1) รูปแบบของการเชื่อมโยงระหว่างนิวรอน(เรียกกันว่าเป็น สถาปัตยกรรมของนิวรอลเนตเวิร์ก) 2) วิธีการให้ค่าน้ำหนักของการเชื่อมโยง (เรียกว่าขั้นตอนวิธีการฝึก หรือการเรียนรู้ (training or learning algorithm)) 3) แอคติเวชั่นฟังก์ชั่น

นิวรอลเนตเวิร์กประกอบด้วยหน่วยประมวลผลเป็นจำนวนมาก เรียก นิวรอน (neuron), หน่วย (unit), เซลล์ (cell) หรือ โหนด (node) แต่ละนิวรอลเชื่อมโยงกับนิวรอลอื่นโดยอาศัยการเชื่อมต่อสื่อสารโดยตรงกับแต่ละนิวรอลด้วยค่าน้ำหนักที่สัมพันธ์กัน ค่าน้ำหนักแสดงสารสนเทศที่ถูกใช้โดยเนตเพื่อที่จะใช้แก้ปัญหา นิวรอลเนตเวิร์กสามารถนำไปใช้แก้ปัญหาหลายประการ เช่น การเก็บระลึกข้อมูลหรือจัดรูปแบบ การแยกแยะรูปแบบ การทำแมปปิ้งจากรูปแบบอินพุตไปยังรูปแบบเอ้าต์พุต การจัดกลุ่มรูปแบบที่คล้ายกัน หรือการหาคำตอบจากปัญหาความคุ้มค่าแบบมีข้อกำหนด (constrained optimization problem)

แต่ละนิวรอนมีสภาวะภายในเรียก แอคติเวชั่น หรือ ระดับแอคติวิตี้ (activation or activity level) ซึ่งเป็นฟังก์ชั่นของอินพุตที่รับเข้ามา โดยรูปแบบแล้วนิวรอนส่งแอ้คติเวชั่นเป็นสัญญานไปยังนิวรอนอื่นๆหลายนิวรอน ข้อสังเกตที่สำคัญคือนิวรอนสามารถส่งเพียงสัญญานเดียวในแต่ละครั้ง ถึงแม้ว่าสัญญานเป็นการแพร่กระจายไปยังนิวรอนอื่นๆหลายนิวรอน

2.3 นิวรอน (neuron)

คือส่วนประกอบพื้นฐานของนิวรอลเนตเวิร์ก ซึ่งประกอบด้วย ส่วนรับข้อมูล (input) ค่าน้ำหนัก(weight) หน่วยประมวลผล (processing element) และ ส่วนส่งข้อมูล (output) ส่วนรับข้อมูลมีจำนวนเท่าไรก็ได้ $(x_1,x_2,x_3,...,x_n)$ หน่วยประมวลผลเป็นแอคติเวชั่นฟังก์ชั่น(Activation function) ของผลรวมของผลคูณระหว่างค่าน้ำหนักกับข้อมูลเข้า และมีหนึ่งส่วนส่งข้อมูลออก ซึ่งก็คือผลที่ได้จากการประมวลผล

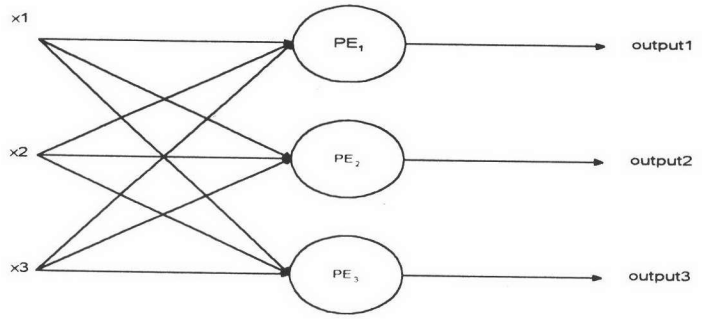


Processing Element $PE = f(\sum_{i=1}^{n} w_i x_i)$

รูปที่ 2-3 แสดงส่วนประกอบของนิวรอล

2.4 นิวรอลเนตเวิร์กชั้นเดียว(Single layer neural network)

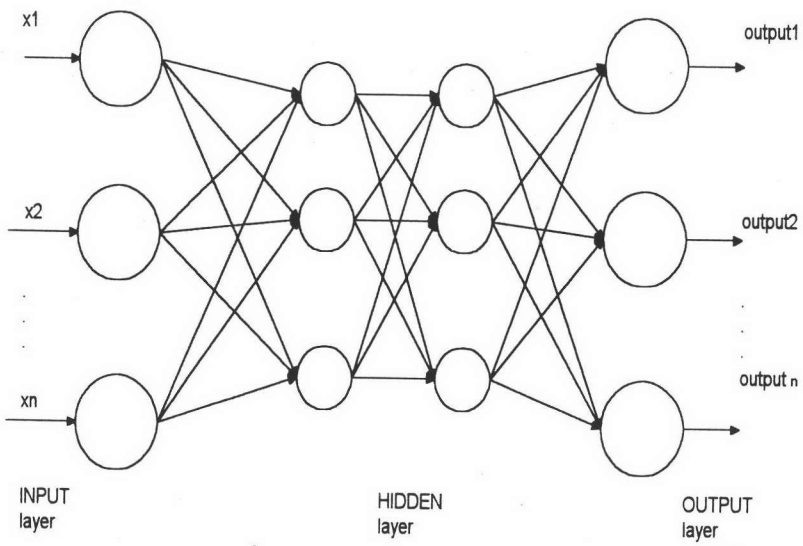
เป็นการนำนิวรอนหลายๆตัวมารวมกันเพื่อให้ข้อมูลผ่านไป ผลลัพธ์คือกลุ่มข้อมูลออก



รูปที่ 2-4 ตัวอย่างนิวรอลเนตเวิร์กแบบชั้นเดียว

2.5 นิวรอลเนตเวิร์กแบบหลายชั้น (Multi layer neural network)

จากข้อ 2.4 เราสามารถนำนิวรอนแบบชั้นเดียวแต่ละชั้นมาต่อกัน โดยที่ชั้นรับข้อมูลเข้าจะเรียกว่า ชั้นข้อมูลเข้า(Input layer) ซึ่งชั้นนี้ก็จะทำหน้าที่เป็นเสมือนบัฟเฟอร์(buffer) สำหรับรับสัญญานเข้าเท่านั้น ส่วนผลลัพธ์ของเนตเวิร์กจะถูกผลิตออกมาทางชั้นข้อมูลออก(Output layer) และชั้นอื่นๆ นอกจากชั้นทั้งสองที่กล่าวมาแล้วจะเรียกว่า ชั้นฮิดเดน Hidden layer ซึ่งเปรียบเสมือนกล่องดำ(black box) ดังแสดงในรูปที่ 2-5



รูปที่ 2-5 ตัวอย่างนิวรอลเนตเวิร์กแบบหลายชั้น

การที่ข้อมูลของแต่ละชั้นถูกส่งไปยังชั้นถัดไปเรื่อยๆเช่นนี้เรียกว่า เนตวอร์คแบบป้อนข้างหน้า (feedforward network) และถ้าข้อมูลออกสามารถถูกดึงนำกลับมาใช้เป็นข้อมูลเข้าเนตเวิร์กได้อีกเรียกว่า เนตเวิร์กแบบป้อนกลับ (feedback network)

2.6 ประวัติโดยย่อของนิวรอลเนตเวิร์ก (Fausette,1994)

2.6.1 ยุคปี ค.ศ. 1940 : เริ่มต้นนิวรอลเนตเวิร์ก (The 1940s:The Beginning of Neural Network)

แมคคูลลอค-พิตส์ นิวรอนส์ (McCulloch-Pitts neurons)

Warren McCulloch และ Pitts ได้ออกแบบนิวรอลเนตเวิร์กเป็นรายแรก นักวิจัยเหล่านี้พบว่า การประกอบนิวรอลนิวรอนแบบง่ายหลายๆนิวรอนเข้าเป็นระบบเป็นแหล่งกำเนิดของการเพิ่มอำนาจการคำนวณ ค่าน้ำหนักในนิวรอนของMcCulloch-Pitts ถูกจัดขึ้นมาเพื่อว่านิวรอนแสดงฟังก์ชั้นเฉพาะทางตรรกะอย่างง่ายด้วยนิวรอลหลายๆนิวรอลทำฟังก์ชั่นต่างๆ นิวรอนสามารถถูกจัดตัวเป็นเนตเพื่อที่จะให้เนตผลิตเอาต์พุต เอาต์พุตที่แสดงถึงการรวมกันของฟังก์ชั่นทางตรรกะ การไหลของสารสนเทศผ่านเนตถือว่าขั้นหน่วยเวลา (unit time step) สำหรับสัญญานที่จะเดินทางจากนิวรอนหนึ่งไปยังนิวรอนถัดไป เวลาหน่วง (time delay) ยอมให้เนตจัดรูปแบบคล้ายกระบวนการทางสรีระดังเช่นการรับรู้ร้อนหนาว

แนวความคิดมาจากจุดเริ่มต้นที่ว่าถ้าเนตอินพุตหนึ่งไปยังนิวรอนหนึ่งใหญ่กว่าค่าเริ่มต้น ดังนั้นหน่วยยิง(unit fire) เป็นความสามารถหนึ่งของนิวรอน McCullloch-Pitts ที่ใช้กันในนิวรอนเทียมในปัจจุบัน อย่างไรก็ตามนิวรอน McCulloch-Pitts ถูกนำมาใช้กว้างขวางมากที่สุดในวงจรตรรกะ(Anderson & Rosenfeld,1988)

งานของ McCullochและPittsที่ตามมา(Pitts&McCulloch,1947)กล่าวถึงหัวข้อที่ยังคงอยู่ในงานวิจัยที่สำคัญในปัจจุบันเช่นการแปลและการหมุนเวียนการรู้จำแบบไม่ผันแปร(translation and rotation invariant pattern recognition)

เฮบบ์ เลินนิ่ง (Hebb Learning)

Donald Hebb นักจิตวิทยาแห่งมหาวิทยาลัย McGill ได้ออกแบบการเรียนรู้สำหรับนิวรอลเนตเวิร์ก (Hebb,1994) เขากล่าวว่า ถ้านิวรอลสองนิวรอลทำงานพร้อมกัน ดังนั้นความแข็ง (strength) ของการเชื่อมต่อระหว่างนิวรอลจะเพิ่มขึ้น มีการขัดเกลาทำให้ข้อความคำกล่าวอยู่ในรูป

ทั่วไปที่นำไปใช้ในการจำลองในคอมพิวเตอร์ (Rochester,Holland,Haibt&Dude,1956) แนวความคิดมีความสัมพันธ์ใกล้เคียงกับการเรียนรู้แมตริกซ์สหสัมพันธ์ (Correlation matrix learning) พัฒนาโดย Kohonen (1972) และ Anderson (1972)

2.6.2 ยุคปี ค.ศ. 1950 และ ค.ศ. 1960:ยุคทองยุคแรกของนิวรอลเนตเวิร์ก(The 1950s and 1960s:The First Golden Age of Neural Networks)

เปอร์เซปตรอนส์ (Perceptrons)

FrankRosenblatt(19858,1959,1962)พร้อมด้วยนักวิจัยอีกหลายท่าน(Block,1962;Minsky& Papert,1988 (พิมพ์ต้นฉบับ ปี ค.ศ. 1969)) เริ่มแนะนำและพัฒนาเพอเซพตรอน(perceptron) เพอเซพตรอนพื้นฐานประกอบด้วยชั้นอินพุต (เรตินา)เชื่อมต่อตามเส้นทางซึ่งกำหนดตายตัวไปยังนิวรอนที่เกี่ยวข้อง ค่าน้ำหนักบนเส้นทางเชื่อมต่อสามารถปรับปรุงได้ กฎการเรียนรู้เพอเซพตรอนใช้การปรับน้ำหนักวนซ้ำที่มีความสามารถมากกว่ากฎของHebb การเรียนรู้เพื่อที่จะเข้าใกล้ค่าน้ำหนักที่ถูกต้อง ถ้ามีค่าน้ำหนักที่จะแก้ปัญหาที่มีอยู่(ตัวอย่าง อนุญาตให้เนตผลิตคู่ทั้งหมดของการฝึกอินพุตและเอาต์พุตเป้าหมาย) งานของ Rosenblatt ในปี 1962 แสดงรูปแบบต่างๆ ของเพอเซพครอน เหมือนกับนิวรอลที่พัฒนาโดย McCulloch และ Pitts และโดย Hebb เพอเซพตรอนใช้ฟังก์ชั่นเอาต์พุตเริ่มต้น

ความสำเร็จของในระยะแรกกับเพอเซพตรอนมีการกล่าวถึงกันอย่างมาก อย่างไรก็ตามคารพิสูจน์การลู่เข้าทางคณิตศาสตร์ของการเรียนรู้ซ้ำ(the mathematical proof of the convergence of iterative learning) ภายใต้สมมติฐานที่เหมาะสมถูกนำไปปฏิบัติโดยการแสดงให้เห็นถึงข้อจำกัดการพิจารณาเพอเซพตรอนแบบใดของเนตสามารถเรียนรู้

อดาลีน (Adaline)

Benard Widrow และ นักศึกษาของเขา Marcian (Ted) Hoff (Widrow & Hoff,1960) พัฒนากฎการเรียนรู้ (ซึ่งปกติไม่ว่าถือตามชื่อหรือถือค่าเฉลี่ยกำลังสองน้อยที่สุด(the least mean square) หรือกฎเดลต้า(the delta rule))ที่มีความสำคัญใกล้เคียงกับกฎการเรียนรู้ของเพอเซพตรอน กฎเพอเซพตรอนปรับค่าน้ำหนักเชื่อมโยงไปยังหน่วยที่เมื่อไรก็ตามการโต้ตอบของหน่วยไม่ถูกต้อง (การโต้ตอบแสดงถึงการจัดชั้นของรูปแบบอินพุตกฎเดลต้าปรับน้ำหนักเพื่อที่จะลดความแตกต่างระหว่างเนตอินพุตไปยังหน่วยอินพุตและเอาต์พุตเป้าหมาย นี้เป็นผลมาจากการทำให้ค่าความ

ผิดพลาดกำลังสองเฉลี่ยน้อยที่สุด ความคล้ายกันของตัวแบบที่พัฒนาในทางจิตวิทยาโดย Rosenblattที่ถูกนำไปพัฒนาในเชิงวิศวกรรมไฟฟ้าโดย Widrow และ Hoff เป็นเครื่องแสดงถึงกฎธรรมชาติที่เกี่ยวพันกัน(Interdisciplinary nature) ของนิวรอลเนตเวิร์ก ความแตกต่างในกฎการเรียนรู้ถึงแม้แตกต่างไม่มากแต่ก็นำไปสู่ความสามารถที่ถูกพัฒนาของเนตที่จะนำไปใช้ในรูปทั่วไป (การโต้ตอบอินพุตที่คล้ายแต่ไม่เหมือนของสิ่งที่ถูกฝึก) กฎการเรียนรู้ของ Widrow-Hoff สำหรับเนตเวิร์กชั้นเดียวเป็นต้นกำเนิดของกฎการเรียนรู้แบบย้อนกลับสำหรับเนตหลายชั้น

งานที่ Widrow และ นักศึกษาของเขาทำ ถูกรายงานในลักษณะของการวิจัยทางนิวรอลเนตเวิร์ก บางครั้งก็เรียกเป็นการปรับตัวเชิงเส้น (Adaptive linear system) ชื่อ ADALINE มาจากไม่เป็น ADAptive Linear neuron ก็เป็น ADAptive Linear system เป็นระบบที่เรียกชื่อตามเนตเหล่านี้ มีหลายแอพพลิเคชั่นที่น่าสนใจของ ADALINE จากนิวรอลเนตเวิร์กเพื่อระบบการปรับตัวเสาอากาศ (Adaptive antenna system) (Widrow,Mantey,Griffith & Goode,1967) การหมุนเวียนรู้จำแบบไม่แปรปรวน (rotation-invariation pattern recognition) ไปสู่ปัญหาการควบคุมต่างๆ เช่น การดุลการกวาด (broom balancing) การต่อท้ายรถบรรทุก (backing up a truck) (Widrow,1987;Tolat&Widrow,1988;Nguyen & Widrow,1989) MADALINEเป็นรูปที่ขยายจากADALINE หลายชั้น

2.6.3 ยุคปี ค.ศ. 1970:ยุคแห่งความเงียบ (The 1970s:The Quiet Years)

โคโฮเนน (Kohonen)

งานในระยะแรกของ Tuevo Kohonen (1972) แห่งHelsinki University of Technology เกี่ยวข้องกับนิวรอลเนตเวิร์กแบบจัดกลุ่มความจำ (associative memory nets) งานล่าสุด (Kohonen,1982) ได้ทำการพัฒนาแมปปรับโครงสร้างตัวเอง (self-organizing maps) ที่ใช้รูปแบบโครงสร้างแบบหน่วยกลุ่ม(cluster unit) เนตเหล่านี้นำไปใช้ในรู้จำคำพูด(เป็นภาษาฟินแลนด์และภาษาญี่ปุ่น(Kohonen, Torkkola, Shozakai, Kangas & Venta,1987;Kohonen,1988)การหาคำตอบของ "ปัญหาการเดินทางของเซลแมน(Traveling Saleman Problem)" (Angeniol,Vubois,&Le Texier,1988)

แอนเดอร์สัน (Anderson)

James Anderson แห่งมหาวิทยาลัยบราวน์ เริ่มงานวิจัยนิวรอลเนตเวิร์กด้วยเนตแบบจัดกลุ่มความจำ(Anderson,1968,1972)เขาพัฒนาแนวความคิดดังกล่าวใน "Brain-State-in-Box" (Anderson,Siverstein,Ritz,&Jones,1977) ตัดเอาต์พุทเชิงเส้นของตัวแบบในระยะต้นที่จะป้องกันเอาต์พุทจากการที่ค่าเริ่มใหญ่เกินไปเมื่อเนตวนซ้ำเพื่อหาคำตอบที่คงตัว ระหว่างขอบข่ายของแอพพลิเคชั่นสำหรับเนตเหล่านี้เป็นการวินิจฉัยโรคทางการแพทย์และการเรียนรู้ตารางการคูณ Anderson และ Rosenfold(1988) และ Anderson,Pellionisz และ Rosenfeld (1990) ทำการรวบรวมบทความพื้นฐานการวิจัยนิวรอลเนตเวิร์ก

กรอสเบอร์ก (Grossberg)

Stephen Grossberg ร่วมกับผู้ร่วมงานและร่วมประพันธ์สร้างผลงานมากมาย Klimasauskas(1989) ทำการรวบรวมบทที่ตีพิมพ์โดย 146 รายการจากปีค.ศ. 1967 ถึง 1988 งานของเขาเป็นที่รู้จักกันอย่างกว้างขวางส่วนมากเป็นงานทางด้านคณิตศาสตร์และชีววิทยา (Grossberg,1976,1981,1987)

คาร์เพนเตอร์ (Carpenter)

Gail Carpenter ร่วมกับ Stephen Grossberg พัฒนาทฤษฎีนิวรอลเนตเวิร์กแบบจัดโครงสร้างตัวเองเรียกว่าทฤษฎีการปรับให้ได้ระดับกัน(Adaptive Resonance Theory ART) (Carpenter & Grossberg,1985,1987a,1987b,1990)

2.6.4 ยุคปี ค.ศ. 1980:ยุคแห่งการเร่งฟื้นฟู (The 1980s:The Renewed Enthusiasm)

แบคโปรปาเกชัน (Backpropagation)

เหตุผล 2 ประการสำหรับยุคแห่งความเงียบของปี ค.ศ. 1970 คือ ความล้มเหลวของเพอเซพตรอนชั้นเดียวในการแก้ปัญหาง่ายๆอย่าง ฟังก์ชั่นXOR และขาดวิธีทั่วไปในการฝึกเนตหลายชั้น วิธีกระจายสารสนเทศเกี่ยวกับความผิดพลาดที่หน่วยเอาต์พุตไปยังหน่วยแฝงถูกค้นพบก่อนหน้านี้(Werbos,1974)แต่ไม่ได้แพร่หลายในวงกว้าง วิธีการนี้ถูกค้นพบอย่างอิสระโดย David Parker (1985) และโดย LeCun (1986)ก่อนที่จะเป็นที่รู้จักกันอย่างกว้างขวาง ซึ่งคล้ายกันอย่างมาก

กับอัลกอริธึมในทฤษฎีการควบคุมความคุ้มค่า(optimal control theory) (Bryson&Ho,1969) งานของParker มาได้รับความสนใจใน Parallel Distributed Processing Group นำโดยนักจิตวิทยาชื่อ David Rumelhart แห่งมหาวิทยาลัย California ที่ San Diego และ James McCelland แห่งมหาวิทยาลัย Carnegie Melon เป็นผู้ที่ทำการปรับปรุงและทำให้เป็นรู้จักแพร่หลาย (Rumelhart,Hinton&Williams,1986a,1986b;McCelland&Rumelhart,1988)

ฮอปฟิลด์ เนตส์ (Hopfeild nets)

ผู้ที่มีความโดดเด่นในการเพิ่มความกระจ่างและสร้างความสนใจในนิวรอนเนตเวิร์กคือผู้ที่ไรับรางวัลโนเบลสาขาฟิสิก John Hopfeild แห่ง California Institute of Technology พร้อมด้วย David Tank นักวิจัยแห่ง AT&T Hopfield พัฒนาจำนวนนิวรอลเนตเวิร์กโดยอาศัยการตรึงค่าน้ำหนักและการปรับการกระตุ้น (adaptive activation) (Hopfeild,1982,1984; Hopfield & Tank,1985,1987) เนตเหล่านี้สามารถรองรับเนตแบบจัดกลุ่มความจำ (associative memory nets) และสามารถใช้แก้ปัญหาความพอใจแบบมีข้อกำหนด (To solve constraint satification problem) ดังเช่น "ปัญหาการเดินทางของเซลล์แมน" บทความใน Scientific American (Tank & Hopfield,1987) ช่วยทำให้นิวรอลเนตได้รับความสนใจ

นีโอคอคนิตรอน (Neocognitron)

Kunihiko Fukushima และเพื่อนร่วมงานในห้องปฏิบัตการ NHK ในโตเกียวได้พัฒนานิวรอลเนตพิเศษเพื่อการรู้จำตัวอักษร ตัวอย่างหนึ่งของเนตเรียก นีโอคอคนิตรอน (neocognitron) เนตแบบจัดโครงสร้างตัวเองเรียก คอคนิตรอน (cognitron) (Fukushima,1975) ไม่สามารถรู้จำตำแหน่งหรือตัวอักษรหมุนบิดเบี้ยว (position or rotation-distorted character) ความบกพร่องถูกปรับปรุงใน นีโอคอคนิตรอน (Fukushima,1988;Fukushima & Miyake,&Ito,1983)

โบลส์มาน มาชีน (Boltzmann machine)

นักวิจัยหลายท่านที่เกี่ยวข้องกับการพัฒนานิวรอลเนตแบบ นันดีเทอมินิสติค(nondeterministic neural net)เนตซึ่งค่าน้ำหนักหรือแอคติเวชั่นถูกปรับปรุงโดยอาศัยพื้นฐานของฟังก์ชั่นความน่าจะเป็นหนาแน่น (probability density function) (Kirkpatrick, Gelatt, &Vecchi, 1983,1987) เนตเหล่านี้รวบรวมแนวความคิดแบบดั้งเดิม (Classical idea) ของการหลอมรวมสิ่งที่ถูกจำลองและทฤษฎีการตัดสินใจแบบเบย์ (simulated annealing and Baysian decision theory)

การใช้ฮาร์ดแวร์ (Hardware Implementation)

อีกเหตผลหนึ่งของการกลับมาสนใจนิวรอลเนตเวิร์ก(อีกทั้งการแก้ปัญหาวิธีการฝึกเนตหลายชั้น) คือการพัฒนาความสามารถในการคำนวณ นิวรอลเนตแบบนำแสง (optical neural net) (Farhat,Psaltis,Prata&Pack,1985) และการสร้าง VLSI ถูกพัฒนาขึ้น (Silvilatti,Mahowald & Mead,1987)

Caver Mead แห่งสถาบันเทคโนโลยีแคลิฟอร์เนีย ผู้ศึกษาการดักความเคลื่อนไหว (motion detection) เป็นผู้ร่วมประดิษฐ์ซอฟต์แวรเพื่อการออกแบบไมโครชิป เป็นผู้ร่วมก่อตั้ง Synaptics,Inc., ผู้นำการศึกษาวงจรนิวรอล

Leon Cooper แห่งมหาวิทยาลัยบราวน์ แนะนำหนึ่งในเนตหลายชั้นรายแรก เครือข่ายลดค่าพลังงานคูลอมบ์ (the reduced coulomb energy network) Cooper เป็นประธานของบริษัท Nestor บริษัทนิวรอนเนตเวิร์กแห่งแรก(Johnson & Brown,1988)

Robert Hetch Nielson และ Todd Gutschow ได้พัฒนานิวโรคอมพิวเตอร์ดิจิตอลที่ TRW,Inc., ระหว่างปี ค.ศ. 1983-1985 การให้เงินทุนนับสนุนโดย Defense Advanced Research Project Agency DARPA(1988) เป็นการรวบรวมนิวรอลเนตเวิร์กที่มีคุณค่าล้ำสมัย(โดยเฉพาะคำนึงถึงแอพลิเตชั่นที่ประสพความสำเร็จ) Hetch-Nielsongเป็นผู้ก่อตั้ง HNC Inc., และยังเป็นศาตราจารย์แห่งมหาวิทยาลัย California SanDiego และเป็นผู้พัฒนา นิวรอลเนตเวิร์กเคาน์เตอร์พรอพเคชั่น (Counterpropagation neural network)

2.7 ลักษณะทางชีววิทยาของเซลล์ประสาท(ทบวงมหาวิทยาลัย,ชีววิทยา,2527)

2.7.1 เซลล์ประสาท(neuron) ประกอบด้วย

2.7.1.1 ตัวเซลล์ (cell body/soma cell) มีส่วนประกอบเหมือนเซลล์ทั่วไป เช่น นิวเคลียส ไมโตคอนเดรีย ไซโตพลาสซึม เป็นต้น

2.7.1.2 ใยประสาท (nerve fiber) คือส่วนประกอบของโปรโตปลาสซึมของเซลล์ที่ยื่นออกไปมี 2 ชนิด คือ แอกซอน และ เดนไดรต์

- เดนไดรต์ (Dendrite) เส้นใยนี้ทำหน้าที่รับความรู้สึกเข้าสู่ตัวเซลล์ประสาท มีแขนงสั้นๆ มีจำนวนมากกว่า 1 แขนง ไม่มีเยื่อไมอีลินหุ้ม เส้นผ่าศูนย์กลางไม่เท่ากันโดยตลอด

- แอกซอน (Axon) เส้ยใยนี้ ทำหน้าที่ส่งกระแสประสาทออกจากตัวเซลล์ มักเป็นแขนงยาวเพียง 1 แขนง เส้นผ่าศูนย์กลางเท่ากันโดยตลอดและมีเยื่อไมอีลินหุ้ม ในคนเราแอกซอนบางเส้นยาวมากกว่า 1 เมตร ส่วนปลาวาฬ อาจยาวถึง 10 เมตร

- เยื่อไมอีลิน (Myelin Sheath) คือเปลือกชั้นในที่หุ้มประสาท(แขนงประสาท) ถือว่าเป็นเซลล์เมมแบรนของเซลล์ชวาน เป็นสารพวกฟอสโฟลิปิด(Phospholipid) มีคุณสมบัติเป็นฉนวนไฟฟ้าที่ดีทำให้กระแสประสาทเคลื่อนที่ได้รวดเร็ว

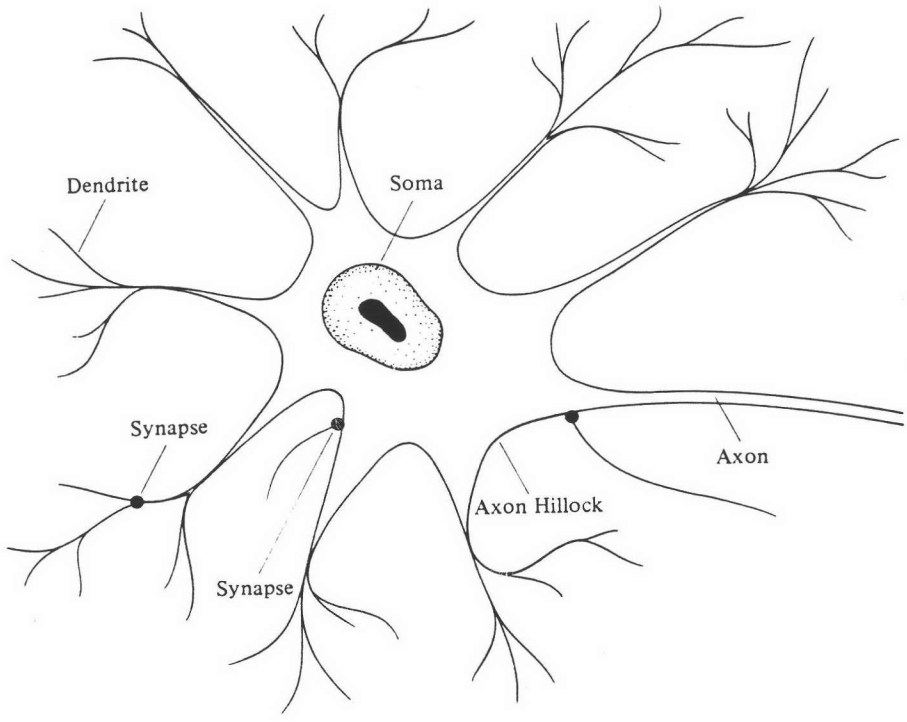
- เซลล์ชวาน (Schwann Cell) ส่วนเปลือกชั้นนอกเป็นเซลล์บางๆ ติดต่อกับเยื่อไมอีลิน ทำหน้าที่สร้างเยื่อไมอีลิน

- โนด ออฟ แรนเวียร์ (Node of Ranvier) หมายถึงรอยคอดของเซลล์ชวานในแอกซอนเป็นบริเวณที่ไม่มีเยื่อไมอีลินหุ้ม ทำหน้าที่ช่วยให้สัญญานประสาท(Nerve Impulse) กระโดดข้ามช่องว่างนี้ ทำให้การส่งสัญญานประสาทเร็วขึ้น การนำกระแสความรู้สึกแบบนี้เรียกว่าการนำกระแสแบบไม่ต่อเนื่อง หรือ แบบกระโดด(Hoping / Salatory Conduction) เซลล์ที่มีขนาดใหญ่และ มีโหนด ออฟแรนเวียร์ห่างกันมาก ก็ยิ่งสามารถนำกระแสความรู้สึกได้เร็วยิ่งขึ้น

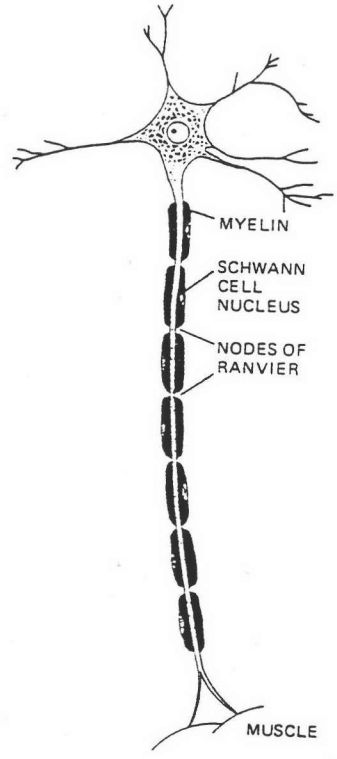
เส้นใยประสาทที่มีเยื่อไมอีลินหุ้มนี้ อาจเทียบได้กับระบบส่งสัญญานโทรทัศน์ ไปยังสถานีที่อยู่ไกลๆ ที่ต้องถ่ายทอดผ่านสถานีถ่ายทอดเป็นช่วงๆ โดยมีโหนด ออฟ แรนเวียร์ ทำหน้าที่เป็นสถานีถ่ายทอด พบว่าระยะห่างระหว่างโหนดหนึ่งๆ ของเซลล์ประสาทอยู่ในช่วง 200-2000 ไมครอน อัตราส่วนระหว่างระยะห่าง ของ โหนดต่อเส้นผ่าศูนย์กลาง มีประมาณ 100 ต่อ 1 ยิ่งเส้นใยประสาทมีเยื่อไมอีลินหุ้มมากก็มีระยะห่างระหว่างโหนดมาก และยิ่งเพิ่มประสิทธิภาพในการนำกระแสความรู้สึกได้ดียิ่งขึ้น

- 2.7.2 ไซแนปส์ (Synapse) คือ บริเวณที่ปลายของเอกซอนเซลของประสาทหนึ่งกับปลายของเซลลีประสาทอื่น มาสัมผัสติดต่อกัน หรือเป็นบริเวณที่อยู่ชิดกันที่สุดระหว่าง เยื่อหุ้มของเซลล์ประสาทด้วยกัน หรือระหว่างเซลล์ประสาทและเอฟเฟคเตอร์(Effector) เช่นกล้ามเนื้อและต่อมต่างๆด้วย

จากการคำนวณพบว่า สมองของคนซึ่งเป็นส่วนที่สลับซับซ้อนที่สุดของระบบประสาท มีเซลล์ประสาทอยู่ถึง 10^{11} เซลล์ (John Herz,Anders Krogh,Richard G.Palmer,1991) มีไซแนปอยู่มากถึงประมาณ 10^{14} แห่งด้วยกันการถ่ายทอดกระแสความความรู้สึกที่ผ่านเซลล์ประสาทหนึ่งไปยังอีกเซลล์ประสาทหนึ่ง อาจอยู่ในรูปของสารเคมี เรียกว่า ไซแนปเคมี (Chemical Synapse) หรือถ่ายทอดในรูปกระแสไฟฟ้าโดยตรง เรียกว่า ไซแนปไฟฟ้า (Electrical Synapse)



รูปที่ 2-6 แสดงเซลล์ประสาททางชีววิทยา 1 (Kosko,1992)



รูปที่ 2-7 แสดงเซลล์ประสาททางชีววิทยา 2 (Dayhoff,1990)

2.8 คุณสมบัติของ นิวรอลเนตเวิร์ก (Phillip D. Wassermann,1989)

นิวรอลเนตเวิร์กนั้นประกอบด้วย โนด จำนวนมากมายซึ่งคล้ายกับเซลสมองจำนวนมากมาย การจัดเรียงตัวของโนด อาจจะเหมือนหรือต่างจากลักษณะทางกายวิภาค ของสมองมนุษย์ แม้ว่าการจัดเรียงตัวของโนดจะมีความเหมือนอย่างผิวเผินกับเซลสมองมนุษย์ แต่นิวรอลเนตเวิร์ก ก็มีคุณสมบัติบางอย่างคล้ายกับสมองมนุษย์ เช่น นิวรอลเนตเวิร์กสามารถเรียนรู้จากตัวอย่างประสบการณ์ สามารถปรับตัวเองเข้ากับสิ่งแวดล้อมได้เป็นอย่างดี และสามารถอนุมานจากสิ่งที่เรียนรู้ไปสู่ สิ่งที่ นิวรอลเนตเวิร์ก ไม่เคยเรียนรู้มาก่อนได้ นั่นคือคุณสมบัติที่เรียกว่า ความสามารถในการบ่งชี้ลักษณะทั่วไป (Generalization)

ดังนั้นสามารถแบ่งคุณสมบัติแบ่งเป็น 2 ประการคือ

2.8.1 การเรียนรู้ (Learning)

นิวรอล เนตเวิร์ก สามารถเรียนรู้จากชุดการสอนที่เราป้อนให้เรียนรู้

2.8.2 การระลึกหรือจดจำได้

นิวรอล เนตเวิร์ก สามารถระลึกได้ทั้ง ชุดการสอน และชุดการทดสอบ หรือชุดทั่วไป (General Set) ได้ดีในระดับที่ยอมรับได้ โดยชุดการทดสอบ จะมีความแตกต่างจากชุดที่นิวรอลเนตเวิร์กได้เรียนรู้อยู่บ้าง ซึ่งความแตกต่างนี้ เรียกว่า มีสิ่งปนเปื้อน หรือสิ่งบิดเบือนของข้อมูลประเภทนั้น ๆ ซึ่งเป็นลักษณะของสภาพความเป็นจริงที่เกิดขึ้น

นิวรอล เนตเวิร์ก นั้นมีความเหมาะสมที่จะใช้กับงานประเภทการรู้จำรูปแบบ(Pattern Recognition) ซึ่งวิธีการทางคอมพิวเตอร์แบบดั้งเดิม ทำไม่ได้ดี ซึ่งวิธีการแบบดั้งเดิมเหมาะกับงานการคำนวณ เช่น งานคำนวณทางบัญชี เป็นต้น

2.9 ส่วนประกอบของนิวรอลเนตเวิร์ก

2.9.1 เซลประสาทเทียม หรือโนด หรือ หน่วยประมวลผล

มีการจัดโครงสร้างในรูปแบบต่าง ๆ เช่น นิวรอลเนตเวิร์กแบบ 3 ชั้น คือประกอบด้วย ชั้นข้อมูลเข้า (Input Layer) ชั้นแอบแฝง (Hidden Layer) ชั้นผลลัพธ์ (Output Layer)

2.9.2 เส้นเชื่อมโยง (Interconnection)

ทำหน้าที่เชื่อมโยงโนดในชั้นต่าง ๆ

2.9.3 ชั้น (Layer)

ประกอบด้วยจำนวนโนดที่ต่าง ๆ กัน ชั้นแบ่งออกเป็น

- ชั้นข้อมูลเข้า คือ ชั้นที่ทำหน้าที่รับข้อมูลเข้ามามีเพียง 1 ชั้น

- ชั้นแอบแฝง คือ ชั้นที่ทำหน้าที่ประมวลผลข้อมูล อาจมีเพียง 1 ชั้น หรือมากกว่านี้ ซึ่งถ้ายิ่งมาก ก็จะยิ่งทำให้ นิวรอล เนตเวิร์ก มีความซับซ้อนยิ่งขึ้น ทำให้เวลาในการคำนวณเพิ่มมากขึ้นในอัตราแบบ เอ็กโปเนนเชียล (Exponential) (Efraim Turban,1992)

- ชั้นผลลัพธ์ คือ ชั้นที่แสดงผลลัพธ์ที่ได้จากชั้นแอบแฝงในชั้นสุดท้าย

2.9.4 โครงสร้างของเนตเวิร์ก (Architecture Of The Network)

เกิดจากการจัดโครงสร้างของโนด เส้นเชื่อมโยง และจำนวนชั้นต่าง ๆ มาประกอบกัน เกิดเป็นโครงสร้างของเนตเวิร์กในรูปแบบ (Model/Paradigm) ที่ต่าง ๆ กัน เช่น Backpropagation Bidirectional Associative Memory (BAM) Hopfield เป็นต้น

2.10 สารสนเทศที่ต้องใช้และผลลัพธ์ที่ได้จาก นิวรอลเนตเวิร์ก

2.10.1 ข้อมูลเข้า (Input)

โดยต้องเป็นค่าตัวเลข ถ้าข้อมูลในรูปเชิงคุณภาพ ต้องแปลงข้อมูลให้อยู่ในรูปเชิงปริมาณที่นิวรอลเนตเวิร์กรับเข้าไปเพื่อเรียนรู้ได้ นั่นคือ กระบวนการเบื้องต้นก่อนประมวลผล (Preprocess)

2.10.2 ชั้นผลลัพธ์ (Output)

คือผลลัพธ์ที่เกิดขึ้นจริง (Actual Output) จากกระบวนการเรียนรู้ของนิวรัล เน็ทเวิร์ค

2.10.3 ค่าน้ำหนัก(Weight)

คือ สิ่งที่ได้จากการเรียนรู้ของนิวรอลเนตเวิร์กหรือเรียกอีกอย่างหนึ่งว่า ความรู้ (Knowledge) ค่าน้ำหนักเป็นสิ่งที่สำคัญมากของนิวรัล เน็ทเวิร์ค ค่าเหล่านี้ จะไม่มีการเปลี่ยนแปลงอีกต่อไป เพื่อนำค่าเหล่านี้ไปใช้ในการระลึกข้อมูลอื่นที่อยู่ในรูปแบบเดียวกันได้ดี

2.10.4 ฟังก์ชันผลรวม (Summation Function)

เป็นผลรวมของค่าข้อมูลเข้า (X_i) และค่าน้ำหนัก (W_i)

2.10.5 ฟังก์ชั่นการแปลงค่า(Transfer function)

เป็นการจำลองการทำงานของโหนดต่างๆ ฟังก์ชั่นที่นิยมใช้กันคือ ฟังก์ชั่นซิกมอยด์

$$f(x) = 1/(1+e^{-x})$$

จุดประสงค์ของการแปลงค่าก็คือ เพื่อให้ได้ผลลัพธ์ที่ได้อยู่ในค่า 0-1 มิฉะนั้นค่าผลลัพธ์จะโตมาก

2.11 กระบวนการเรียนรู้ของนิวรอลเนตเวิร์ก

แบ่งออกเป็น 3 กระบวนการคือ

2.11.1 คำนวณผลลัพธ์

2.11.2 เปรียบเทียบที่เกิดจริง(Desired ouput) และถ้าค่าความผิดพลาด (RMS) อยู่ในระดับที่เรายอมรับได้ ก็จะหยุดการสอนเน็ทเวอร์ค ถ้ายังไม่ยอมรับก็ไปข้อ 2.12.3

2.11.3 ปรับปรุงค่าน้ำหนัก และทำซ้ำข้อ 2.12.1 ใหม่

2.12 วัตถุประสงค์ของการสร้างแบบจำลองเซลล์สมองมนุษย์

มี 2 ประการคือ

2.12.1 เพื่อให้เข้าใจลักษณะทางสรีระศาสตร์ และ จิตตศาสตร์ของระบบเซลล์สมองมนุษย์ (Human Neural system)

2.12.2 เพื่อสร้างสมการคำนวณ เพื่อเลียนแบบกระบวนการทำงานของสมองมนุษย์

2.13 ประเภทของการเรียนรู้ของนิวรอลเนตเวิร์ก

สามารถแบ่งออกเป็น 2 ประเภทใหญ่ๆ คือ

2.13.1 การเรียนรู้แบบมีครู (Supervised Learning)

การเรียนรู้แบบมีครู ต้องการชุดข้อมูลเข้าและชุดข้อมูลเป้าหมายหรือข้อมูลที่ต้องการทั้งชุดข้อมูลขาเข้า และข้อมูลเป้าหมายเรียกว่า ชุดการสอนควบคู่ (Training Pair) โดยปรกติการสอนเนตเวิร์ก จะใช้ชุดการสอนควบคู่หลายชุดในระหว่างการสอนเนตเวิร์ก จะเกิดผลลัพธ์ขึ้นจริงโดยต่างระหว่างผลลัพธ์จริงกับผลลัพธ์เป้าหมาย ก็คือค่าความคลาดเคลื่อน หรือ ค่าความผิดพลาด

นอกจากนี้ยังมีการเรียนรู้แบบมีครูอีกชนิดหนึ่ง คือ การเรียนรู้เชิงบังคับ (Reinforcement Learning) ซึ่งเป็นการเรียนรู้ที่ให้คำตอบว่าผิดหรือถูกแต่ไม่ได้บอกคำตอบที่ถูกคืออะไร(John Herz,Anders Krogh,Richard G. Palmer,1991)

ค่าความผิดพลาดที่เกิดขึ้นจะถูกป้อนกลับสู่เนตเวิร์ก เพื่อให้ค่าลดลงเรื่อยๆ จนถึงระดับที่ยอมรับได้ ในระหว่างการกระบวนการลดค่าความผิดพลาดนั้น ค่าน้ำหนักก็จะถูกปรับตามไปด้วย

2.13.2 การเรียนรู้แบบไม่มีครู(Unsupervised Learning)

การเรียนแบบไม่มีครูนั้น ถูกพัฒนาโดย Kohonen (ค.ศ 1984)และท่านอื่นๆ เพื่อให้ใกล้เคียงกับระบบการเรียนรู้ของสมองมนุษย์มากขึ้น โดยไม่ต้องมีชุดข้อมูลเป้าหมาย มีเพียงข้อมูลชุดเข้า โดยการเรียนรู้เนตเวิร์กจะใช้หลักการทางสถิติ โดยหาค่าทางสถิติของชุดการสอน และจัดข้อมูลเป็นระดับต่างๆและนิวรอลเนตเวิร์ก จะหาค่าผลลัพธ์ของนิวรอลเนตเวิร์กเองโดยเป็นความสัมพันธ์ระหว่างข้อมูลเข้าและผลลัพธ์

2.14 ขั้นตอนวิธีการฝึก(Training algorithm)

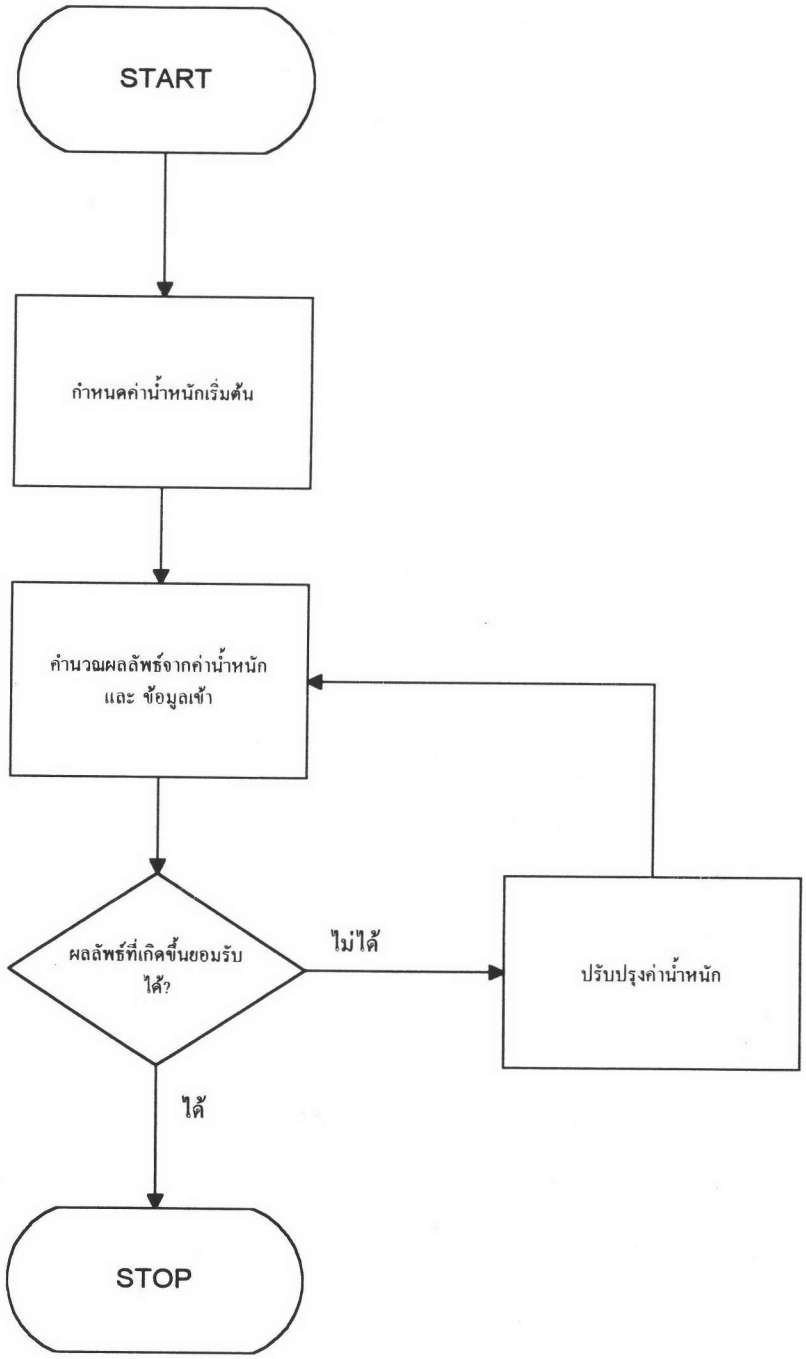
โดยทั่วไปประกอบด้วย 4 ขั้นตอนดังนี้ (เหรียญพงษ์ จูพาวิเศษกุล, 2536)

2.14.1 กำหนดค่าเริ่มต้น ซึ่งอาจกำหนดให้มีค่าเป็นศูนย์ หรือตัวเลขที่มีค่าน้อยได้จากการสุ่ม ทั้งนี้ขึ้นกับรูปแบบของนิวรอลเนตเวิร์ก

2.14.2 คำนวณผลลัพธ์จากค่าน้ำหนักและข้อมูลเข้า

2.14.3 เปรียบเทียบผลลัพธ์ที่เกิดขึ้นกับผลลัพธ์เป้าหมาย(desired output) ถ้ายอมรับได้ก็จะสิ้นสุดการเทรนนิ่ง ถ้ายังไม่สามารถยอมรับได้ ทำขั้นตอน2.15.4

2.14.4 ทำการปรับค่าน้ำหนัก แล้วไปขั้นตอนที่ 2.15.2



รูปที่ 2-8 แสดงขั้นตอนวิธีการเทรนนิ่ง

ในกระบวนการเทรนนิ่ง จะมีการวนซ้ำหลายรอบ (Iteration) เพื่อให้นิวรอลเนตเวิร์กเกิดการเรียนรู้ เมื่อสิ้นสุดการสอนแล้วการเก็บข้อมูลของนิวรอลเนตเวิร์กนั้นเป็นแบบกระจาย (Distributed) และถูกใช้ร่วมกัน (Shared) โดยหลายๆหน่วยประมวลผล(Processing Units) ซึ่งต่างกับแบบเดิม ข้อมูลจะเก็บไว้ที่เฉพาะแห่งเดียวในหน่วยความจำ

ชุดข้อมูลการสอน(Training data sets) ที่ใช้สอนนิวนอลเนตเวิร์กนั้นต้องมีความหลากหลาย โดยมีลักษณะเด่นในการตรวจจับ (Feature-detection) เป็นตัวที่คอยแยกความแตกต่างของคุณลักษณะต่างๆ อยู่ในชั้นของนิวรอน หลังจากที่เนตเวิร์กถูกสอน ชุดข้อมูลการสอนมีผลต่อการเรียนรู้ของเนตเวิร์ก ถ้าเป็นชุดตัวอย่างที่ดี จะทำให้เกิดการเรียนรู้ได้ดีและรวดเร็ว แต่ถ้าเป็นชุดการสอนที่ไม่ดี ทำให้การเรียนรู้ไม่ดีเท่าที่ควร

จากเหตผลข้างต้น คือต้องมีการวนซ้ำหลายๆรอบ และต้องมีชุดการสอนอย่างหลายหลาก นั้นทำให้นิวรอลเนตเวิร์ก ต้องการกำลังในการคำนวณสูงมาก จึงต้องการบอร์ดเร่งการประมวลผล(Accelerator Boards) เพื่อเพิ่มความเร็วในการประมวลผล เช่น ตัวประมวลผลร่วมทางคณิตศาสตร์(Math coprocessor),ตัวประมวลผลเวคเตอร์(Vector processor) หรือตัวประมวลผลแบบขนาน(Parallel processor) อื่นๆ

นิวรอลเนตเวิร์กมีหลายรูปแบบ (Paradigm) เช่น ฮอปฟิลด์เนตเวิร์ก (Hopfield Network), แบคออร์เรอร์โปรปาเกชัน (Back-Error Propagation), เคานเตอร์โปรปาเกชัน(Counter Propagation) เป็นต้นทั้งนี้การที่จะเลือกรูปแบบใดขึ้นกับความเหมาะสมของปัญหา

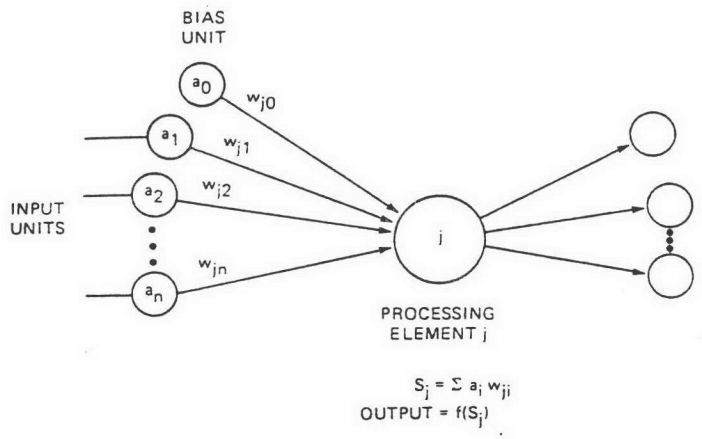
ในการศึกษาวิจัยครั้งนี้จะใช้รูปแบบ Back Propagation เนื่องจากเป็นวิธีการหนึ่งของนิวรอลเนตเวิร์กที่ง่ายต่อการเข้าใจเนื่องจากกระบวนการเรียนรู้และปรับปรุงแก้ไขเป็นไปด้วยตัวมันเอง ถ้าเนตเวิร์กให้คำตอบที่ผิด ดังนั้นค่าน้ำหนัก(weights) จะถูกแก้ไขจนกว่าค่าความผิดพลาดน้อยลง จนอยู่ในเกณฑ์ที่ยอมรับได้(Non-significant) นั่นก็คือค่าที่ได้ในครั้งต่อๆไปจะมีความถูกต้องมากขึ้น

2.15 แบคเออร์เรอร์โปรปาเกชัน พาราดิกม์ (Back-Error Propagation Paradigm (BEP))

BEP สามารถแก้ปัญหาที่ต้องการรูปแบบ(Pattern Mapping) โดยการป้อนรูปแบบเข้าไป (Input Pattern) Network ก็จะผลิตรูปแบบผลลัพธ์(Output Pattern) ที่เกี่ยวข้องกันออกมา (Dayhoff, 1990)

ขั้นตอนการเรียนรู้ของ BEP ประกอบด้วย 2 ขั้นตอน

2.15.1 ขั้นตอนการแพร่ไปข้างหน้า(Forward Propagation)



รูปที่ 2-9 แสดงขั้นตอนการแพร่ไปข้างหน้า(Forward Propagation) (Dayhoff,1990)

จากรูปเป็นการคำนวณผลรวมของผลลัพธ์ที่เข้ามายังหน่วยที่ j ดังสมการ

$$S_j = \sum_i a_i w_{ji} \qquad \text{----- (1)}$$

โดยที่ a_i = ระดับแอคติเวชั่น(Activation Level) ของหน่วยที่ I

w_{ji} ค่า weight จากหน่วยที่ i ไปยังหน่วยที่ j

เมื่อคำนวณได้จนค่า S_j ก็ทำการคำนวณหาค่า $f(S_j)$ อีกครั้ง

โดย $f(x) = 1/(1+e^{-x})$ ซึ่งเป็นสมการ Sigmoid function

ดังนั้น $f(S_j) = 1/(1+e^{-Sj}) = 1/(1+e^{-\sum_i a_i w_{ji}})$ ------(2)

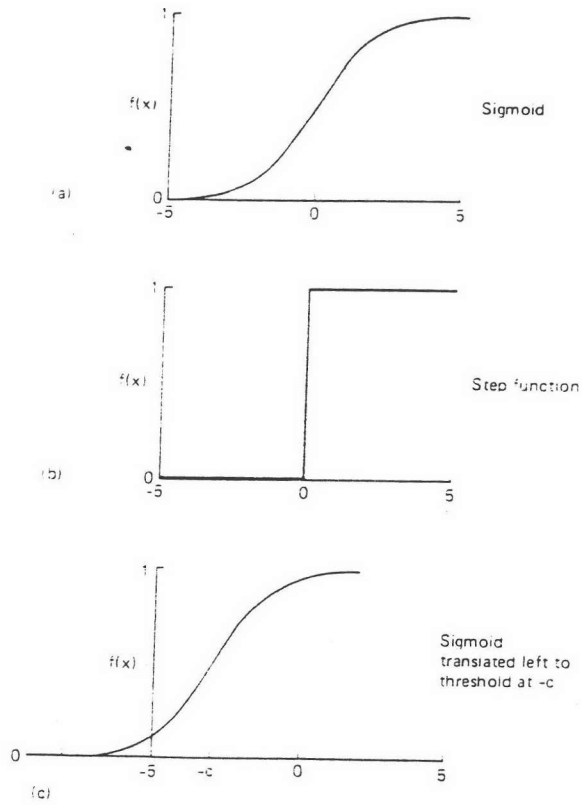
เมื่อได้ค่า $f(S_j)$ แล้วค่า $f(S_j)$ ก็จะกลายเป็นผลลัพธ์ของหน่วยที่ j ซึ่งก็คือ a_j ดังรูป 2-9 โดยจะส่งออกไปทางหน่วยอื่นๆ ด้วยค่า a_j ที่เท่ากันจากรูป 2-9 หน่วยไบแอส (Bias Unit) เป็นค่าคงที่ที่เราใส่เข้าไป เพื่อทำให้การเรียนรู้ของเนตเวอร์คเร็วขึ้นเรียกว่า เวลาคอนเวอร์เจน (Convergence time) เร็วขึ้น นอกจากนี้หน่วยไบแอสยังมีผลต่อ เทรสโฮลด์(Threshold) ด้วย เช่น ให้ค่า C = 5

ทำให้กราฟขยับไปทางซ้าย 5 หน่วยดังรูป

$$C=w_{j0}$$

ให้ $z = \sum_{i=1}^{n} a_i w_{ji}$

ผลรวม = $z+C$

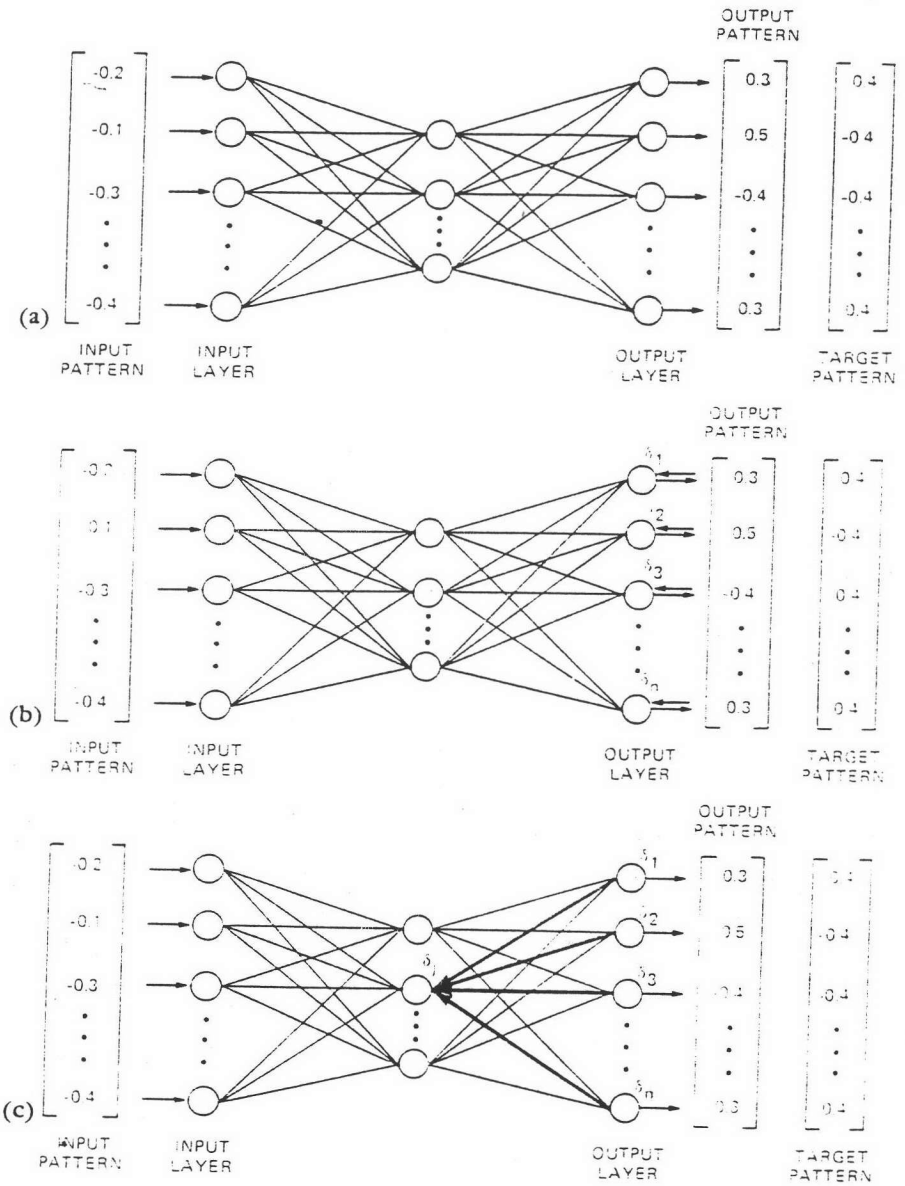


รูปที่ 2-10 แสดง (a)ฟังก์ชั่นซิกมอยด์(sigmoid function) (b)ฟังกชั่นขั้นบันได(step function) (c) ซิกมอยด์ เคลื่อนไป c หน่วยทางซ้าย ไปค่าเทรสโฮลด์(threshold) ที่ -c (Dayhoff,1990)

สาเหตุที่ต้องใช้ฟังก์ชั่นซิกมอยด์

เนื่องจาก ต้องการให้เป็นซอฟต์เทรสโฮลด์(Soft Threshold) มากกว่าฮาร์ดเทรสโฮลด์(Hard Threshold)(ดังเช่น ฟังก์ชั่นขั้นบันไดดังรูป) นั่นคือซิกมอยด์ฟังก์ชั่น ให้ค่าต่อเนื่องกัน

2.15.2. ขั้นตอนการแพร่ย้อนกลับ(Backward Propagation)



รูปที่ 2-11 แสดงการแพร่ย้อนกลับ(Backward Propagation) (Dayhoff, 1990)

จากรูป ค่า δ จะถูกคำนวณเริ่มที่ ชั้นข้อมูลออก(output layer) เมื่อคำนวณเสร็จแล้วค่า weight ก็จะมีการปรับค่า δ_k ดังสมการ

$$\delta_k = (t_k - a_k) f'(S_k) \quad \text{-------(3)}$$

โดยที่ t_k = ค่าเป้าหมายสำหรับหน่วยที่ k

a_k = ค่าผลลัพธ์สำหรับหน่วยที่ k

$f'(x)$ = ค่าอนุพันธ์(Derivative) ของฟังก์ชั่นซิกมอยด์ f

โดยที่ $f(x) = 1/(1+e^{-x})$

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

S_k = ผลรวมน้ำหนักของข้อมูลเข้าไปยัง k

ค่า (t_k-a_k) แสดงถึงค่าความผิดพลาด

ค่า η คือ อัตราการเรียนรู้(Learning rate)

และ $\Delta w_{ji} = \eta \delta_j a_j$ ----------(4)

ส่วนรูป c) ลูกศรทึบ แสดงการคำนวณย้อนหลังเพื่อให้ได้ค่า δ ที่ชั้นฮิดเดน (hidden layer) และเมื่อคำนวณ δ แล้วก็มีการคำนวณปรับค่า น้ำหนัก(weight) ที่เข้ามา

ค่า δ_j ถูกคำนวณดังสมการ

$$\delta_j = \left[\sum \delta_k w_j \right] f'(S_j) \quad \text{-------(5)}$$

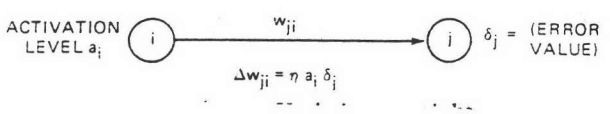
โดย δ_j ได้จากสมการที่ (3)

w_{ji} ได้จากสมการที่ (4)

$f'(x)$ = ค่า อนุพันธ์ของฟังก์ชั่นซิกมอยด์ f

Sj = ผลรวมน้ำหนักของข้อมูลเข้าไปยัง j

และ $\Delta w_{ji} = \eta \delta_j a_i$ ----------(6)



รูปที่ 2-12 แสดงการปรับปรุงค่าน้ำหนัก

สมการปรับปรุงค่าหนักที่ (4) หรือ (6) มีชื่อเรียกว่ากฎเจนเนอรัลไลซ์ δ (generalized δ rule) (Rumelhart & McClelland,1986)

ค่า η โดยปกติอยู่ระหว่าง 0.25-0.75 โดยถูกกำหนดโดยผู้ใช้ ถ้ากำหนดค่า η มากเกินไปจะทำให้เกิดความไม่แน่นอน(Instability) ขึ้นในเนตเวอร์ค และทำให้การเรียนรู้ของเนตเวอร์คไม่ดีเท่าที่ควร หรือถ้ากำหนดค่า η น้อยเกินไปจะทำให้การเรียนรู้ช้ามาก

2.16 การฝึกเนตเวิร์ก (Network training)

การสอน BEP เรียกว่า การสอนแบบมีการควบคุม (Supervised Learning) โดยมีการกำหนดรูปแบบข้อมูลเข้า(input pattern) ควบคู่กับเป้าหมาย(Target pattern) โดยมีการกำหนดชุดการสอน(Training sets) หลายๆรูปแบบ เพื่อให้เนตเวิร์กสามารถเรียนรู้โดยการแมปรูปแบบ (Pattern mapping)

ในกระบวนการสอน เพื่อให้ เนตเวิร์กสามารถเรียนรู้ได้ดีนั้น จะต้องใช้การวนซ้ำหลายๆ รอบ ซึ่งอาจเป็น 100-1,000 รอบก็ได้ ดังนั้นจึงจำเป็นต้องอาศัยฮาร์ดแวร์ที่มีประสิทธิภาพสูงเข้าช่วย เพื่อให้ได้ผลลัพธ์ที่เร็วขึ้น

2.17 การประเมินการฝึกเนตเวิร์ก(Network Evaluation)

การสอนเนตเวิร์ก จะสำเร็จหรือไม่วัดจากค่า Root Mean Square (RMS) ดังสมการ

$$RMS = \sqrt{\frac{\sum\sum(t_{jp} - x_{jp})^2}{n_p n_o}} \qquad \text{------(7)}$$

โดย n_p คือ จำนวนของรูปแบบในชุดการเทรน (Training set)

n_o คิอ จำนวนหน่วย(Units) ในชั้นผลลัพธ์ (Output layer)

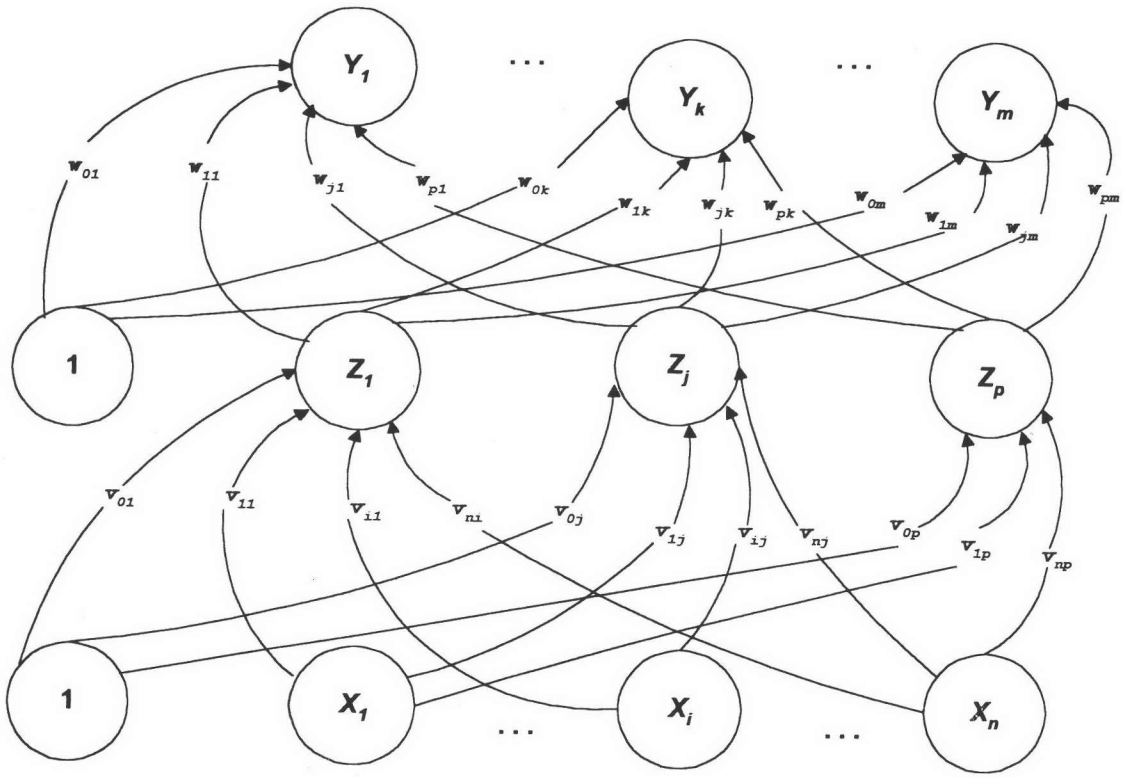
t_{jp} = ค่าเป้าหมาย (target value) ของหน่วยที่ j หลังจากที่เสนอรูปแบบ p

x_{jp} = ค่าผลลัพธ์ที่ออกมา (output value) โดยหน่วยที่ j จากการเสนอรูปแบบ p

โดยปกติ ถ้าค่า RMS ที่ได้ < 0.1 แสดงว่า เนตเวิร์กได้เกิดการเรียนรู้แล้ว

2.18 อัลกอริธึมการฝึกนิวรอลเนตเวิร์กแบบการแพร่ย้อนกลับ(Backpropagation trainning algorithm)

ตามที่กล่าวมาก่อนหน้านี้ การฝึกเนตเวิร์กแบบการแพร่ย้อนกลับเกี่ยวข้องกับ 3 ขั้นตอน : การป้อนไปข้างหน้า(Feedforward)ของรูปแบบการฝึกอินพุต การแพร่ย้อนกลับของค่าความผิดพลาดที่เกี่ยวข้อง และการปรับค่าน้ำหนัก



รูปที่ 2-13 นิวรอลเนตเวิร์กแบบแพร่ย้อนกลับที่มีชั้นแฝงชั้นเดียว

ระยะการป้อนไปข้างหน้าแต่ละหน่วยอินพุต (X_i)รับค่าสัญญานอินพุตและกระจายไปยังแต่ละหน่วยแฝง $Z_1, \ldots, Z_p$ แต่ละหน่วยแฝงคำนวณค่าแอคติเวชั่นและส่งสัญญาน(z_j)ไปยังแต่ละหน่วยเอาต์พุต แต่ละหน่วยเอาต์พุต(Y_k)คำนวณค่าแอคติเวชั่น(y_k)เพื่อที่จะจัดรูปแบบการโต้ตอบของเนตสำหรับรูปแบบอินพุตที่ให้

ระหว่างการฝึก แต่ละหน่วยเอาต์พุตเปรียบเทียบค่าแอคติเวชั่น y_k กับค่าเป้าหมาย t_k เพื่อที่จะหาค่าความผิดผลาดที่เกี่ยวข้องสำหรับรูปแบบที่หน่วยนั้น โดยอาศัยค่าความผิดพลาด แฟคเตอร์ δ_k (k=1, . . . , m) ถูกคำนวณ δ_k ถูกใช้เพื่อกระจายค่าความผิดพลาดที่หน่วยเอาต์พุต Y_k กลับไปยังหน่วยทั้งหมดในชั้นก่อนหน้านี้(หน่วยแฝงที่เชื่อมต่อกับ Y_k) ค่าดังกล่าวก็ถูกใช้(ต่อมา)เพื่อปรับปรุงค่าน้ำหนักระหว่างเอาต์พุตและชั้นแฝง ในลักษณะที่คล้ายกันแฟคเตอร์ δ_j (j=1, . . . ,p) ถูกคำนวณสำหรับแต่ละหน่วยแฝง Z_j ไม่จำเป็นต้องกระจายค่าความผิดพลาดกลับไปยังชั้นอินพุต แต่ δ_j ถูกนำมาใช้เพื่อปรับปรุงค่าน้ำหนักระหว่างชั้นแฝงและชั้นอินพุต

หลังจากค่าแฟคเตอร์ทั้งหมดถูกนำมาพิจารณา ค่าน้ำหนักสำหรับชั้นทั้งหมดถูกปรับพร้อมกัน การปรับค่าน้ำหนัก w_{jk} (จากหน่วยแฝง Z_j จนถึงหน่วยเอาต์พุต Y_k) อาศัยค่าแฟคเตอร์ δ_k

และ ค่าแอคติเวชั่น z_j ของหน่วยแฝง Z_j การปรับค่าน้ำหนัก v_{ij} (จากหน่วยอินพุต X_i ไปยังหน่วยแฝง Z_j) อาศัยค่าแฟคเตอร์ δ_j และค่าแอคติเวชั่น x_i ของหน่วยอินพุต

2.18.1 สัญญลักษณ์ต่างที่ใช้ในอัลกอริธึมการฝึกนิวรอลเนตเวอร์คแบบแพร่ย้อนกลับกำนดให้

x เวคเตอร์การฝึกอินพุต (Input training vector)

$$\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$$

t เวคเตอร์เอาต์พุตเป้าหมาย (Output target vector)

$$\mathbf{t} = (t_1, \ldots, t_i, \ldots, t_n)$$

δ_k ส่วนของค่าความผิดพลาดที่ใช้ในการปรับน้ำหนักสำหรับ w_{jk} ที่เกี่ยวข้องกับหน่วยเอาต์พุต Y_k นอกจากนี้สารสนเทศที่เกี่ยวข้องกับค่าความผิดพลาดที่หน่วย Y_k ถูกกระจายกลับไปยังหน่วยแฝงที่ป้อนเข้าไปยังหน่วย Y_k

δ_j ส่วนของค่าความผิดพลาดที่ใช้ในการปรับน้ำหนักสำหรับ v_{ij} ที่เกี่ยวข้องกับการแพร่ย้อนกลับของสารสนเทศค่าความผิดพลาดจากหน่วยเอาต์พุต ไปยังหน่วยแฝง Z_j

η อัตราการเรียนรู้

X_i หน่วยอินพุต i

สำหรับหน่วยอินพุต สัญญานอินพุตและสัญญานเอาต์พุต ให้ชื่อ x_i

$f'(x)$ ค่าอนุพันธ์ของฟังก์ชั่น $f(x)$

v_{0j} ค่าเอนเอียง(Bias) บนหน่วยแฝง j

Z_j หน่วยแฝง j

ค่าเนตอินพุตไปยัง Zj กำหนดให้เป็น z_in_j:

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

สัญญานเอาต์พุท(แอคติเวชั่น)ของ Z_j ถูกกำหนดโดย z_j:

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

w_{0k} ค่าเอนเอียง(Bias) บนหน่วยเอาต์พุต k

Y_k หน่วยเอาต์พุท k

ค่าเนตอินพุตไปยัง Y_k กำหนดให้เป็น y_in_k:

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

สัญญานเอาต์พุท(แอคติเวชั่น)ของ Y_k ถูกกำหนดโดย y_k :

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

อัลกอริธึม:นิวรอลเนตเวิร์กที่มีชั้นแฝงชั้นเดียว(one hidden layer neural network)

ขั้นที่ 0. เริ่มต้นค่าน้ำหนัก (Intialize weight)
(กำหนดให้เป็นค่าที่เกิดจากการสุ่มตัวเลขน้อยๆ)

ขั้นที่ 1. ขณะที่เงื่อนไขหยุดเป็นเท็จ, ทำขั้นที่ 2-9

ขั้นที่ 2. สำหรับแต่ละคู่การฝึก, ทำขั้นที่ 3-8

Feedforward:

ขั้นที่ 3. แต่ละหน่วยอินพุต (X_i,$i=1, \ldots, n$) รับสัญญานอินพุต x_i และกระจายสัญญานนี้ไปยังทุกหน่วยในชั้นข้างบน(หน่วยแฝง)

ขั้นที่ 4. แต่ละหน่วยแฝง (Z_j,$j=1, \ldots, p$) รวมค่าสัญญานอินพุตที่ถ่วงน้ำหนัก

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

นำแอคติเวชั่นฟังก์ชั่นมาใช้เพื่อคำนวณสัญญานเอาต์พุต

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

และส่งสัญญานไปยังหน่วยทั้งหมดในชั้นข้างบน(หน่วยเอาต์พุต)

ขั้นที่ 5. แต่ละหน่วยเอาต์พุต (Y_k,$k=1, \ldots, m$)รวมค่าสัญญานอินพุตที่ถูกถ่วงน้ำหนัก

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

นำแอคติเวชั่นฟังก์ชั่นมาใช้เพื่อคำนวณสัญญานเอาต์พุต

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

Backpropagation of error:

ขั้นที่ 6. แต่ละหน่วยเอาต์พุต (Y_k,$k=1, \ldots, m$) รับรูปแบบเป้าหมาย(target pattern) ให้สอดคล้องกับรูปแบบการฝึกอินพุต(input training pattern)ทำการคำนวณในรูป

$$\delta_k = (t_k - y_k) f'(y\_in_k)$$

คำนวณเทอร์มการปรับแก้น้ำหนัก(เพื่อใช้ปรับปรุง w_{jk} ภายหลัง)

$$\Delta w_{jk} = \eta \delta_k z_j$$

คำนวณเทอร์มการปรับแก้ไบแอส(เพื่อใช้ปรับปรุง w_{0k} ภายหลัง)

$$\Delta w_{0k} = \eta \delta_k$$

และส่ง δ_k ไปยังหน่วยในระดับล่าง

ขั้นที่ 7. แต่ละหน่วยแฝง ($Z_j, j=1, \ldots, p$) รวมเดลต้าอินพุต (จากหน่วยในชั้นข้างบน)

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$$

คูณด้วยค่าอนุพันธ์ของแอคติเวชั่นฟังก์ชั่นเพื่อที่จะคำนวณค่าเทอร์มความผิดพลาด

$$\delta_j = \delta\_in_j \; f'(z\_in_j)$$

คำนวณค่าเทอร์มการแก้ไขน้ำหนักน้ำหนัก

$$\Delta v_{ij} = \eta \delta_j \, x_i$$

และคำนวณค่าเทอร์มแก้ไขไบแอส(ใช้ปรับปรุง v_{oj} ภายหลัง)

$$\Delta v_{0j} = \eta \delta_j$$

Update weights and biases :

ขั้นที่ 8. แต่ละหน่วยเอาต์พุต ($Y_k, k=1, \ldots, m$) ปรับปรุงค่าน้ำหนักและค่าไบแอส($j=0, \ldots, p$):

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

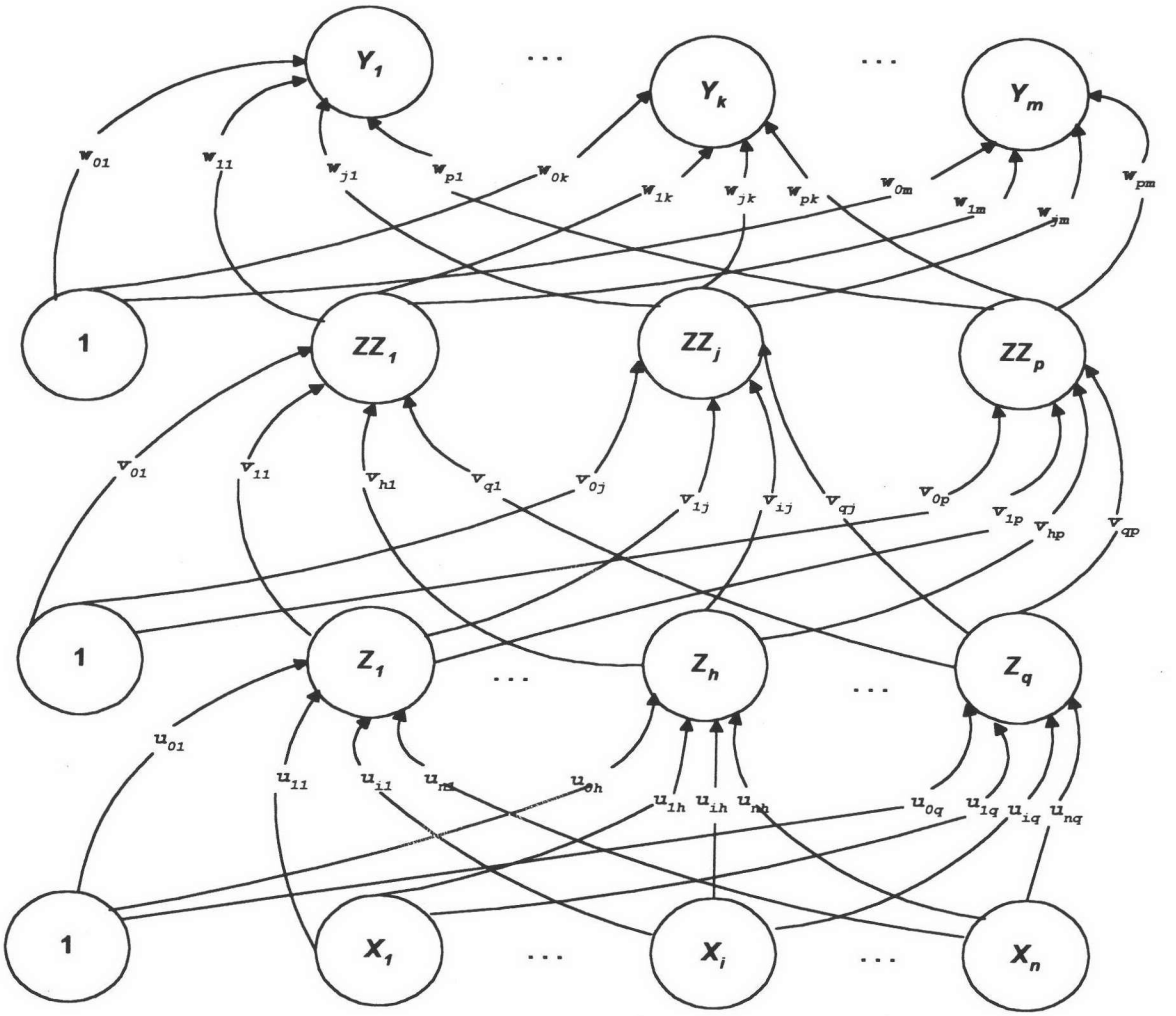
แต่ละหน่วยแฝง ($Z_j, j=1, \ldots, p$) ปรับปรุงค่า

น้ำหนักและค่าไบแอส($i=0, \ldots, n$):

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

ขั้นที่ 9. ทดสอบเงื่อนไขหยุด(Test stopping condition)

อัลกอริธึม:นิวรอลเนตเวิร์กที่มีชั้นแฝงชั้นสองชั้น(two hidden layer neural network)



รูปที่ 2-14 นิวรอลเนตเวิร์กแบบแพร่ย้อนกลับที่มีชั้นแฝง 2 ชั้น

ระหว่างการป้อนข้างหน้า(feedforward) แต่ละหน่วยอินพุตรับสัญญานและส่งสัญญานไปที่หน่วยแฝง $Z_1, \ldots, Z_q$ ในชั้นแรก (X_i เป็นหน่วยอินพุต) แต่ละหน่วยแฝงคำนวณค่าแอคติเวชั่นและส่งสัญญานไปหน่วยแฝง ($ZZ_1, \ldots, ZZ_p$) ในชั้นสอง (Z_h เป็นต้นแบบในชั้นแฝง) ต่อไปแต่ละหน่วยแฝงในชั้นสองคำนวณค่าแอคติเวชั่น (y_k เป็นค่าของหน่วยเอาต์พุตของ Y_k) เพื่อจัดรูปแบบการโต้ตอบของของเนตสำหรับรูปแบบอินพุตที่ให้มา

ระหว่างการฝึกแต่ละหน่วยเอาต์พุตทำการเปรียบเทียบค่าแอกติเวชั่นที่ถูกคำนวณค่า y_k กับค่าเป้าหมาย t_k เพื่อพิจารราค่าความผิดพลาดที่เกี่ยวกับหน่วยนั้นโดยอาศัย แฟคเตอร์ δ_k ถูกคำนวณ ($k = 1, \ldots, m$) δ_k ถูกนำมาใช้เพื่อกระจายสารสนเทศบนค่าความผิดพลาดที่หน่วยเอาต์พุต y_k กลับไปยังหน่วยทั้งหมดในชั้นที่ต่ำลงมา ค่าดังกล่าวยังใช้ต่อมาเพื่อปรับปรุงชั้นเอาต์พุตและชั้นแฝง แฟคเตอร์ δ_j ($j = 1, \ldots, p$)ถูกคำนวณสำหรับหน่วยแฝง ZZ_j และใช้เพื่อกระจายสาร

สนเทศบนค่าความผิดพลาดกลับไปยังทุกหน่วยในชั้นก่อนหน้า (หน่วย $Z_1, \ldots, Z_n, \ldots, Z_q$) ค่าดังกล่าวยังใช้ในภายหลังเพื่อปรับปรุงค่าน้ำหนักระหว่างชั้นแฝงที่สองและชั้นแฝงแรก แฟคเตอร์ δ_h ($h=1, \ldots, q$)ถูกคำนวณสำหรับหน่วยแฝง Z_h ค่าดังกล่าวไม่จำเป็นต้องกระจายกลับไปที่ชั้นอินพุต แต่ δ_h ถูกใช้เพื่อปรับค่าน้ำหนักระหว่างชั้นแฝงแรก (หน่วย $Z_1, \ldots, Z_n, \ldots, Z_p$) และชั้นอินพุต

หลังจากค่าแฟกเตอร์ δ ถูกพิจารณาทั้งหมด ค่าน้ำหนักสำหรับทุกชั้นถูกปรับพร้อมกันการปรับค่าน้ำหนัก w_{jk} (จากหน่วยแฝง ZZ_j ไปยังหน่วยเอาต์พุต Y_k) อาศัยแฟกเตอร์ δ_j และค่าแอคติเวชั่น ของหน่วย Z_h การปรับน้ำหนัก v_{hj} (จากหน่วยแฝง Z_h ไปยัง หน่วยแฝง ZZ_j)อาศัยค่าแฟกเตอร์ δ_j และแอคติเวชั่นของหน่วย Z_h การปรับน้ำหนัก u_{ih} (จากหน่วยอินพุต X_i ไปยัง หน่วยแฝง Z_h) อาศัย δ_h และแอคติเวชั่นของหน่วยอินพุต

ขั้นตอนของการแพร่ย้อนกลับมาตรฐานสำหรับเนตที่แบบชั้นแฝงสองชั้นสรุปได้ดังนี้

Feedforward:

แต่ละหน่วยอินพุต (X_i, $i=1, \ldots, n$) :

กระจายสัญญานอินพุตไปยังหน่วยแฝง

แต่ละหน่วยแฝง (Z_h, $h=1, \ldots, q$) :

คำนวณสัญญานอินพุต

$$z\_in_h = u_{0h} + \sum_{i=1}^{n} x_i u_{ih}$$

นำแอคติเวชั่นฟังก์ชั่นมาใช้เพื่อคำนวณสัญญานเอาต์พุต

$$z_h = f(z\_in_h)$$

และส่งสัญญานไปยังหน่วยทั้งหมดในชั้นแฝงที่สอง

แต่ละหน่วยแฝง (ZZ_j ,$j=1, \ldots, p$)

คำนวณสัญญานอินพุต

$$zz\_in_j = v_{0j} + \sum_{h=1}^{n} z_h v_{hj}$$

นำแอคติเวชั่นฟังก์ชั่นมาใช้เพื่อคำนวณสัญญานเอาต์พุต

$$zz_j = f(zz\_in_j)$$

และส่งสัญญานเอาต์พุตไปยังหน่วยเอาต์พุต

แต่ละหน่วยหน่วยเอาต์พุต (Y_k ,$k=1, \ldots, m$)

รวมสัญญานอินพุตที่ถูกถ่วงน้ำหนัก

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

นำแอคติเวชั่นฟังก์ชั่นมาใช้เพื่อคำนวณสัญญานเอาต์พุต

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

Backpropagation of error:

แต่ละหน่วยเอาต์พุต ($Y_k, k=1, \ldots, m$) :

คำนวณค่าความผิดพลาด

$$e_k = (t_k - y_k)$$

สำหรับรูปแบบการฝึกคูณด้วยค่าอนุพันธ์ของฟังก์ชั่นแอกติเวชั่น(ถูกอธิบายในเทอมของ y_k)

$$\delta_k = e_k f'(y\_in_k)$$

คำนวณเทอร์มการปรับแก้น้ำหนัก(เพื่อใช้ปรับปรุง w_{jk} ภายหลัง)

$$\Delta w_{jk} = \eta \delta_k zz_j$$

คำนวณเทอร์มการปรับแก้ไบแอส(เพื่อใช้ปรับปรุง w_{0k} ภายหลัง)

$$\Delta w_{0k} = \eta \delta_k$$

และส่ง δ_k ไปยังหน่วยแฝง ($ZZ_j, j=1, \ldots, p$)

แต่ละหน่วยแฝง ($ZZ_j, j=1, \ldots, p$) :

รวมค่าอินพุตที่ถ่วงน้ำหนักจากหน่วยในชั้นข้างบนที่จะได้

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$$

คูณด้วยค่าอนุพันธ์ของฟังก์ชั่นแอคติเวชั่น(อธิบายในเทอม zz_j)ที่จะได้

$$\delta_j = \delta\_in_j f'(zz\_in_j)$$

คำนวณเทอร์มการปรับแก้น้ำหนัก(เพื่อใช้ปรับปรุง v_{ij} ภายหลัง)

$$\Delta v_{ij} = \eta \delta_j x_i$$

คำนวณเทอร์มการปรับแก้ไบแอส(เพื่อใช้ปรับปรุง v_{0k} ภายหลัง)

$$\Delta v_{0j} = \eta \delta_j$$

และส่ง δ_j ไปยังหน่วยแฝง ($Z_h, h=1, \ldots, q$)

แต่ละหน่วยแฝง ($Z_h\ h=1, \ldots, q$) :

รวมค่าอินพุตที่ถ่วงน้ำหนักจากหน่วยในชั้นข้างบนที่จะได้

$$\delta\_in_h = \sum_{j=1}^{p} \delta_j v_{hj}$$

คูณด้วยค่าอนุพันธ์ของฟังก์ชั่นแอคติเวชั่น(อธิบายในเทอม z_h)ที่จะได้

$$\delta_h = \delta\_in_h f'(z\_in_h)$$

คำนวณเทอร์มการปรับแก้น้ำหนัก(เพื่อใช้ปรับปรุง u_{ih} ภายหลัง)

$$\Delta u_{ih} = \eta \delta_h x_i$$

คำนวณเทอร์มการปรับแก้ไบแอส(เพื่อใช้ปรับปรุง u_{0h} ภายหลัง)

$$\Delta u_{oh} = \eta \delta_h$$

Update weights and biases :

สำหรับแต่ละหน่วยเอาต์พุต ($j=0, \ldots, p$; $k=1, \ldots, m$) :

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

สำหรับแต่ละหน่วยอินพุต ZZ_j ($h=0, \ldots, q$; $j=1, \ldots, p$) :

$$v_{hj}(\text{new}) = v_{hj}(\text{old}) + \Delta v_{hj}$$

สำหรับแต่ละหน่วยอินพุต Z_h($i=0, \ldots, n$; $h=1, \ldots, p$) :

$$u_{ih}(\text{new}) = u_{ih}(\text{old}) + \Delta u_{ih}$$

2.19 การใช้ค่าโมเมนตัม

นอกจากค่า η แล้วยังมีค่าโมเมนตัม ใช้สัญญลักษณ์ α (alpha)ช่วยให้การเรียนรู้เร็วขึ้น ค่าโมเมนตัมคิดค้นโดย Rumelhart Hinton และ William (1986) โดยสร้างเทคนิคที่ช่วยให้การฝึกเนตเวิร์กของการเรียนรู้แบบย้อนกลับเร็วขึ้น โดยการใส่ค่าโมเมนตัมเข้าไปทำให้เนตเวิร์กมีความคงตัวมากขึ้นและได้สมการเป็นดังนี้

$$w_{jk}(t+1) = w_{jk}(t) + \eta \delta_k z_j + \alpha[w_{jk}(t) - w_{jk}(t-1)]$$

$$w_{jk}(new) = w_{jk}(old) + \eta \delta_k z_j + \alpha[\Delta w_{jk}(old)]$$

หรือ

$$\Delta w_{jk}(t+1) = \eta \delta_k z_j + \alpha[\Delta w_{jk}(t)]$$

$$\Delta w_{jk}(new) = \eta \delta_k z_j + \alpha[\Delta w_{jk}(old)]$$

และ

$$v_{ij}(t+1) = v_{ij}(t) + \eta \delta_j x_i + \alpha[v_{ij}(t) - v_{ij}(t-1)]$$

$$v_{ij}(new) = v_{ij}(old) + \eta \delta_j x_i + \alpha[\Delta v_{ij}(old)]$$

หรือ

$$\Delta v_{ij}(t+1) = \eta \delta_j x_i + \alpha[\Delta v_{ij}(t)]$$

$$\Delta v_{ij}(new) = \eta \delta_j x_i + \alpha[\Delta v_{ij}(old)]$$

ค่า η ถ้ากำหนดมากเกินไปจะทำให้เกิดความไม่แน่นอนขึ้นในเนตเวิร์ก และทำให้การเรียนรู้ในเนตเวิร์กไม่ดีเท่าที่ควร หรือ ถ้ากำหนดน้อยเกินไปจะการเรียนรู้ช้ามาก นอกจากค่า η ค่าโมเมนตัม α ช่วยให้การเรียนรู้เร็วขึ้นและทำให้มีความคงตัวมากขึ้น

ค่า α ที่เหมาะสมอยู่ในช่วง 0.00 - 1.0 (Phillip D. Wasserman,1989)

และมีค่าเท่ากับ 0.9 (John Herz,Anders Krogh,Richard G. Palmer,1991)

ค่า η ที่เหมาะสมอยู่ในช่วง 0.00 - 1.0 (Phillip D. Wasserman,1989)

และ 0.25 - 0.75 (Judith E.Dayhoff,1990)