บทที่ 3

แบ็กพรอเพเกชัน

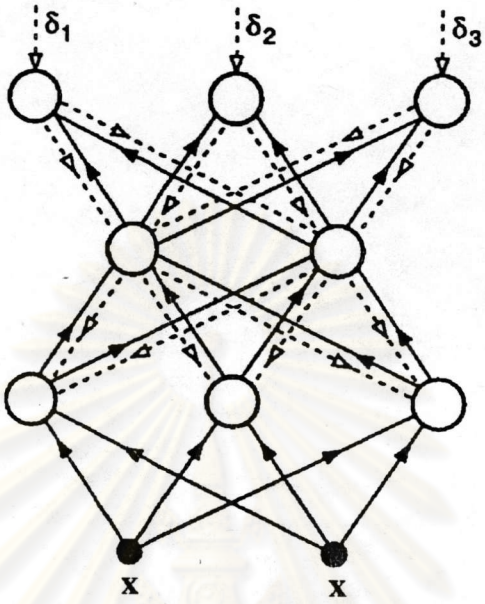
ประมาณปี 1970 เพอเซปทรอน (Perceptron) ซึ่งเป็นนิวรอลเนตเวิร์กประเภทข่ายงานชั้นเดียว (Single Layer Neural Network) ถูกโจมตีว่าไม่สามารถแก้ปัญหา XOR ได้ ทำให้ความสนใจในวิทยาการนิวรอลเนตเวิร์กได้ลดน้อยลงไป จนมีการเสนอวิธีการใหม่ที่สามารถแก้ปัญหานี้คือ เปลี่ยนแปลงโครงสร้างจากข่ายงานชั้นเดียวเป็นข่ายงานหลายชั้น (Multi Layer Neural Network) ทำให้นิวรอลเนตเวิร์กได้รับความสนใจมากยิ่งขึ้น จนกระทั่งถึงปัจจุบันได้มีการพัฒนานิวรอลเนตเวิร์กแตกแขนงออกไปหลากหลายรูปแบบ เพื่อใช้แก้ปัญหาในลักษณะต่างๆ

ความหมายของแบ็กพรอเพเกชัน (Backpropagation)

แบ็กพรอเพเกชันหรือบางครั้งเรียกว่าเพอร์เซบทรอนหลายชั้น (Multi Layer Perceptron) เนื่องจากมีความคล้ายคลึงกับเพอร์เซบทรอนมาก เรียกอย่างย่อว่าบีพี (BP) เป็นรูปแบบหนึ่งของนิวรอลเนตเวิร์กที่ได้รับความนิยมมาก เนื่องจากสามารถนำมาใช้แก้ปัญหาได้หลายด้าน เป็นการเรียนรู้แบบมีครู (Supervised Training) นั่นคือการมีชุดการสอนที่ส่งเข้าสู่ระบบ และมีสถาปัตยกรรมแบบข่ายงานหลายชั้น มีลักษณะการติดต่อระหว่างนิวรอนแบบไปในชั้นถัดไป (Feedforward) แต่ค่าความผิดพลาดจะถูกนำกลับมาปรับปรุงค่าน้ำหนักใหม่ในระหว่างการสอน (Back Propagate Error)

จากรูป 3.1 แสดง BP ในรูปข่ายงานสามชั้น (Three-layer Network) โดยมีการส่งผ่านข้อมูลไปข้างหน้า แสดงด้วยเส้นทึบ และส่งกลับค่าความผิดพลาด (δ) แสดงด้วยเส้นประ

การประเมินผลความผิดพลาดจากการคำนวณ สำหรับแบ็กพรอเพเกชันแล้วนิยมใช้หลักการหาค่าความผิดพลาดกำลังสองต่ำสุด (Least Mean Squared Error) หรือเรียกอย่างย่อว่า (LMS) เมื่อค่าความผิดพลาดยังไม่ถึงจุดต่ำสุดก็จะทำการปรับปรุงค่าน้ำหนักใหม่แล้วผ่านการคำนวณหาผลลัพธ์ใหม่ในรอบการสอนถัดไป จนกระทั่งความผิดพลาดนั้นอยู่ในเกณฑ์ยอมรับได้



รูป 3.1 แสดงแบ็กพรอเพเกชันแบบสามชั้นมีการส่งค่าความผิดพลาดกลับ

<u>สถาปัตยกรรมของแบ็กพรอเพเกชัน</u>

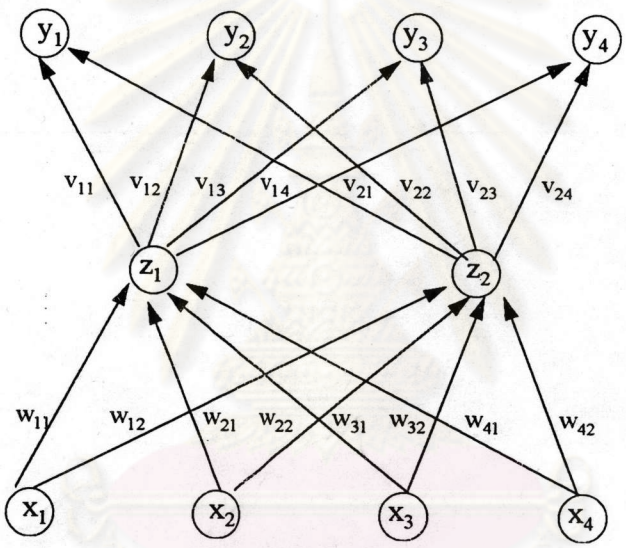
ประกอบด้วยโครงสร้างเป็นชั้น คือ ชั้นนำเข้าข้อมูล, ชั้นแอบแฝง, และชั้นผลลัพธ์ ซึ่งชั้นแอบแฝงอาจจะมีมากกว่า 1 ชั้นก็ได้ ภายในชั้นแต่ละชั้น จะประกอบด้วยนิวรอน (neuron) หรืออาจเรียกว่าโหนด (node) จำนวนหนึ่ง และมีการติดต่อกับโหนดในชั้นถัดไปด้วยค่าน้ำหนักชุดหนึ่ง

กรณีมี 1 ชั้นแอบแฝง จะใช้ชุดน้ำหนัก 2 ชุดคือ ชุดของค่าน้ำหนักระหว่างชั้นนำเข้าข้อมูลและชั้นแอบแฝง และชุดของค่าน้ำหนักระหว่างชั้นแอบแฝงกับชั้นผลลัพท์ และในกรณีมีชั้นแอบแฝงมากกว่า 1 ชั้นก็จะมีชุดค่าน้ำหนักระหว่างชั้นแอบแฝงกับชั้นแอบแฝงอื่นด้วยกันเอง ซึ่งจะกำหนดค่าเริ่มต้นของค่าน้ำหนักแบบสุ่ม

การสอนให้ระบบเรียนรู้ต้องมีชุดของข้อมูลนำเข้าและค่าเป้าหมายที่ต้องการ แล้วส่งค่าข้อมูลนำเข้าสู่กระบวนการเรียนรู้และปรับปรุงค่าน้ำหนัก ตามรูป 3.2 แสดงถึงข่ายงานแบ็กพรอเพเกชันแบบสองชั้น และเป็นข่ายงานที่มีลักษณะการติดต่อของนิวรอนแบบติดต่อถึงกันหมด (Fully

Connected Net) ประกอบด้วยชุดน้ำหนัก 2 ชุด คือ V และ W ส่วนข้อมูลนำเข้าคือ X ผลลัพธ์ในการคำนวณจากข้อมูลนำเข้า X กับค่าน้ำหนัก V จะส่งสัญญาณไปยังชั้นถัดไปให้กับ Z และผลลัพธ์ในการคำนวณจากข้อมูลแอบแฝง Z กับค่าน้ำหนัก W จะส่งสัญญาณไปยังชั้นถัดไปให้กับ Y ซึ่งเป็นชั้นผลลัพธ์

แต่ละโหนดผลลัพธ์ Y_k ในชั้นผลลัพธ์ จะถูกเปรียบเทียบกับค่าเป้าหมายเพื่อหาความผิดพลาด ถ้าค่าความผิดพลาดยังสูงหรือยอมรับไม่ได้ ก็จะทำการส่งกลับค่าความผิดพลาดกลับไปยังชั้นก่อนหน้าเพื่อให้แต่ละชั้นทำการปรับปรุงค่าน้ำหนักใหม่



รูป 3.2 แบ็กพรอเพเกชันที่มี 1 ชั้นแอบแฝง

<u>ขั้นตอนวิธีของการเรียนรู้ (Learning Algorithm)</u>

อาจกล่าวได้ว่า แบ็กพรอเพเกชัน จะประกอบด้วย การนำชุดข้อมูลที่จะใช้สอนเข้าสู่ระบบ และทำการคำนวณหาผลลัพธ์ การวัดค่าความผิดพลาดในการสอนรวมทั้งการส่งค่าความผิดพลาดกลับไปยังชั้นก่อนหน้าเพื่อทำการปรับปรุงค่าน้ำหนัก

ขั้นตอนการสอนของแบ็กพรอเพเกชันมีลักษณะนำชุดข้อมูลที่ต้องการสอนเข้าสู่ระบบ ซึ่งประกอบด้วยชุดข้อมูลตัวอย่างที่ต้องการสอนหรือหน่วยนำเข้าข้อมูล (X_i) กับค่าเป้าหมายที่ต้องการ

จากรูป 3.2 แต่ละหน่วยนำเข้าข้อมูล (X_i) จะส่งสัญญาณหรือข้อมูลไปยังหน่วยแอบแฝง Z_0 ถึง Z_p แต่ละหน่วยแอบแฝง จะทำการคำนวณหาผลลัพธ์และผ่านแอคติเวชั่นฟังก์ชันเพื่อแปลงค่าผลลัพธ์ให้มีความหมาย และส่งสัญญาณไปยัง Y_1 ถึง Y_m แต่ละหน่วยผลลัพธ์ จะทำการคำนวณหาผลลัพธ์และผ่านแอคติเวชั่นฟังก์ชันเช่นกัน และผลลัพธ์ที่ได้จะนำมาเปรียบเทียบกับค่าเป้าหมายเพื่อหาค่าความผิดพลาด แล้วส่งค่าความผิดพลาดนั้นไปยังชั้นก่อนหน้าหรือชั้นแอบแฝงเพื่อทำการปรับปรุงค่าน้ำหนักระหว่างชั้นแอบแฝงและชั้นผลลัพธ์ และจะหาค่าความผิดพลาดที่ชั้นแอบแฝงเพื่อส่งกลับไปทำการปรับปรุงค่าน้ำหนักระหว่างชั้นนำเข้าข้อมูลและชั้นแอบแฝง ที่ต้องมีการคำนวณค่าความผิดพลาดที่ชั้นแอบแฝงด้วย เพราะการคำนวณในชั้นแอบแฝงอาจมีผลต่อความผิดพลาดในชั้นผลลัพธ์

จากขั้นตอนการสอนดังกล่าวเมื่อปรับปรุงค่าน้ำหนักเรียบร้อยแล้ว ข้อมูลนำเข้าชุดถัดไปจะถูกนำมาเข้าสู่ขั้นตอนการสอน ซึ่งค่าหน่วยนำเข้าข้อมูล (X_i) ก็จะเปลี่ยนไปและค่าเป้าหมายก็จะเปลี่ยนไปตามชุดข้อมูลนำเข้าด้วย จนกระทั่งในที่สุดทุกๆชุดข้อมูลนำเข้าถูกสอนหมด ก็จะนับเป็น 1 รอบการสอน (Epoch or Cycle) และการปรับปรุงค่าน้ำหนักจะกระทำกับทุกๆ ชุดข้อมูลการสอน

ปกติแล้วแบ็กพรอเพเกชันจะต้องการการสอนหลายๆรอบจนกว่าระบบจะเกิดการเรียนรู้ ซึ่งการเรียนรู้จะวัดได้จากค่าความผิดพลาดในชั้นผลลัพธ์ ถ้าค่าความผิดพลาดน้อยเพียงพอก็จะหยุดการสอน

ในขั้นตอนการเรียนรู้สำหรับแบ็กพรอเพเกชันจะประกอบด้วย 2 หลักการสำคัญคือ การกำหนดฟังก์ชันแอคติเวชัน และการกำหนดวิธีการปรับปรุงค่าน้ำหนัก

<u>แอคติเวชั่นฟังก์ชัน (Activation Function)</u>

แอคติเวชั่นฟังก์ชันสำหรับแบ็กพรอเพเกชันจะมีลักษณะสำคัญหลายประการ คือ จะต้องเป็นแบบต่อเนื่อง และมีลักษณะการหาค่าอนุพันธ์ และต้องการสูตรการคำนวณที่ง่ายมีประสิทธิภาพ และสามารถหาค่าผลลัพธ์ที่อยู่ในขอบเขตได้

ลักษณะของแอคติเวชั่นฟังก์ชันสำหรับแบ็กพรอเพเกชัน จะให้ผลลัพธ์ที่สามารถสร้างกราฟออกเป็นรูปตัว S ได้ ซึ่งประกอบด้วยฟังก์ชันโลจิสติค (Logistic Function) หรือฟังก์ชันไบ นารี (Binary Sigmoid) ที่ให้ผลลัพธ์อยู่ในช่วง (0,1) และ ฟังก์ชั่นไฮเปอร์บอลิคแทนเจนท์ (Hyperbolic Tangent Function) หรือฟังก์ชันไบโพลา (Bipolar Sigmoid) ที่ให้ผลลัพธ์อยู่ในช่วง (-1,1) หรืออาจเป็นค่าอื่นเช่น (-0.5,0.5) แต่มีลักษณะที่เป็นช่วงของค่าบวกและลบ

จากการคำนวณหาผลลัพธ์ของนิวรอนด้วยสูตร ตาม (1) คือผลบวกของผลคูณข้อมูลนำเข้ากับค่าน้ำหนัก อาจทำให้ผลลัพธ์นั้นกระจัดกระจายเนื่องจากการกำหนดค่าน้ำหนักเริ่มต้นเป็นค่าสุ่ม ดังนั้นจำเป็นต้องผ่านกระบวนการวัดค่าเพื่อให้ค่านั้นมีความหมายขึ้น โดยผ่านเข้าสู่ฟังก์ชันซิกมอยด์เพื่อจำกัดขอบเขตผลลัพธ์ ดังเช่นเมื่อใช้ฟังก์ชันโลจิสติคจะใช้สูตรตาม (2) และเมื่อใช้ฟังก์ชันฟังก์ชั่นไฮเปอร์บอลิคแทนเจนท์จะใช้สูตรตาม (3) เพื่อหาผลลัพธ์แล้วส่งผลลัพธ์ไปเป็นข้อมูลนำเข้าในชั้นถัดไป

$$net = \sum w_i i_i \qquad \ldots (1)$$

$$F(x) = 1 / (1+ \exp(-x)) \qquad \ldots (2)$$

$$F(x) = \tanh(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x}) \qquad \ldots (3)$$

<u>การหาค่าความผิดพลาด</u>

แบ็กพรอเพเกชันดั้งเดิม จะมีการประมาณการค่าความผิดพลาดโดยใช้วิธีคำนวณหาค่าเฉลี่ยของค่าความผิดพลาดกำลังสอง (Mean Squared Error) หรือ MSE ซึ่งที่จริงแล้วการใช้วิธีคำนวณแบบนี้ไม่ได้จำเป็นสำหรับแบ็กพรอเพเกชัน เราอาจกำหนดวิธีการหรือฟังก์ชันอื่นๆในการประมาณการค่าความผิดพลาดได้ แต่อย่างไรก็ตามวิธีประมาณการค่าความผิดพลาดก็นิยมที่จะใช้วิธีที่สามารถช่วยลดความแตกต่างระหว่างค่าเป้าหมายกับค่าผลลัพธ์ได้

สำหรับแบ็กพรอเพเกชัน การที่จะตรวจสอบว่าระบบเรียนรู้แล้วและสมควรจะหยุดสอนเมื่อใดนั้น วิธีที่นิยมใช้ที่สุดเพราะเข้าใจง่ายคือจะพิจารณาจากค่าเฉลี่ยความผิดพลาดกำลังสองที่น้อยที่สุด (Least Mean Squared Error) ว่าน้อยจนยอมรับได้แล้วหรือไม่มีทางหาค่าที่น้อยกว่านี้ได้อีก

จากหลักการหาค่าเฉลี่ยความผิดพลาดกำลังสอง ตามที่รูเมลฮาร์ท และ แมคคลีแลน (Rumelhart and McClelland) ได้กำหนดความหมายของค่าความผิดพลาดไว้ โดยหาผลรวมของผลต่างของค่าเป้าหมายกับค่าผลลัพธ์ทุกๆหน่วยในชั้นผลลัพธ์ สำหรับทุกๆชุดข้อมูลการสอน (Training Pattern) ตาม (4) แล้วหาค่ารากของกำลังสองของผลรวมดังกล่าว ซึ่งจะเป็นการประมาณค่าความผิดพลาดทั้งหมด แล้วหารด้วยจำนวนชุดที่ใช้สอนทั้งหมด ซึ่งจะได้เป็นค่าเฉลี่ยของค่าความผิดพลาดจากการสอน

$$E_p = \sum_{i=1}^{n} (t_{pi} - o_{pi})^2 \qquad ...(4)$$

เมื่อ p เป็นดัชนีมีค่าอยู่ในขอบเขตจำนวนชุดของข้อมูลนำเข้า

i มีค่าอยู่ในขอบเขตจำนวนของผลลัพธ์

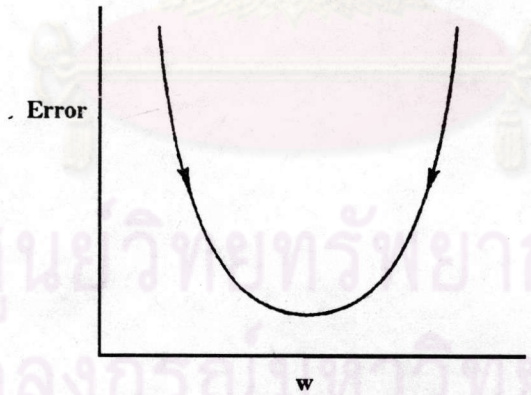
E_p แทนค่าความผิดพลาดของชุดการสอนที่ p

t_{pi} เป็นค่าเป้าหมาย (target) สำหรับผลลัพธ์ที่ i ในชุดการสอนที่ p

o_{pi} เป็นผลลัพธ์ที่ได้ สำหรับลำดับที่ i ในชุดการสอนที่ p

<u>การปรับปรุงค่าน้ำหนัก</u>

ขั้นตอนวิธีการเรียนรู้ในแบ็กพรอเพเกชันถูกกำหนดขึ้นหลากหลายรูปแบบ แต่ที่ได้รับการนิยมมากที่สุดเป็นกฎเดลต้า (Generalized Delta Rule) โดยรูเมลฮาร์ทและสมาชิกของกลุ่มพีดีพี (Parallel Distribute Processing) ได้ร่วมกันวิจัยขึ้น เป็นขั้นตอนวิธีการเรียนรู้ที่นำไปใช้ได้กับข่ายงานทุกรูปแบบรวมทั้งข่ายงานหลายชั้นด้วย โดยไม่จำกัดว่าจะใช้แอคติเวชันแบบใด แต่ต้องเป็นการเรียนรู้แบบมีครู



รูปที่ 3.3 แสดงความสัมพันธ์ระหว่างค่าน้ำหนักกันค่าความผิดพลาด

การปรับปรุงค่าน้ำหนักจะใช้หลักการกราเดียนเดสเซนท์ (Gradient Descent) ซึ่งเป็นกระบวนการหาชุดของค่าน้ำหนัก ที่ทำให้ค่าเฉลี่ยของค่าความผิดพลาดกำลังสองมีค่าน้อยที่สุด นั่นคือเมื่อข้อมูลของแต่ละชุดที่เข้าไปสู่ขั้นตอนการสอนด้วยค่าน้ำหนักชุดหนึ่ง เมื่อผลลัพธ์ไม่ตรง

กับค่าเป้าหมายและยอมรับค่าผิดพลาดนั้นไม่ได้ จะทำการปรับปรุงค่าน้ำหนักใหม่โดยที่จะต้องปรับปรุงค่าน้ำหนักทุกชุด เพราะมีส่วนที่ทำให้เกิดความผิดพลาดในชั้นผลลัพธ์ จนกระทั่งการสอนในรอบถัดไปทำให้ค่าความผิดพลาดลดลงไปยังจุดต่ำสุดของแต่ละชุดข้อมูล ตามรูป 3.3

และการปรับปรุงค่าน้ำหนักตามแนวความคิดกราเดียน จะได้ตามสูตร (8) คือจะทำการเปลี่ยนแปลงค่าน้ำหนักตามสัดส่วนของค่าลบของอนุพันธ์ของค่าความผิดพลาด

$$\Delta w_{ij} = -k(\frac{\partial E_p}{\partial w_{ij}}) \qquad \text{... (8)}$$

เมื่อ k คือ ค่าคงที่ หรือจะทำให้อยู่ในรูปที่ง่ายขึ้นดังนี้

$$\Delta w_{ij} = \varepsilon \delta_{pi} i_{pj}$$

เมื่อ ε = อัตราการเรียนรู้

$\delta_{pi} = t_{pi} - o_{pi}$

δ_{pi} เป็นความแตกต่างระหว่างค่าเป้าหมายสำหรับโหนดที่ i ของชุดการสอนที่ pกับค่าผลลัพธ์จริงจากการคำนวณ

โดยมีการกำหนดค่าตัวแปรขึ้น คือค่าอัตราการเรียนรู้ (Learning Rate) ε พบว่าเมื่อค่าอัตราการเรียนรู้มีค่าน้อยการเรียนรู้ช้ามาก แต่ถ้ากำหนดค่าสูงเกินไปการเรียนรู้ไวแต่อาจหาคำตอบให้กับปัญหาไม่ได้

นอกจากค่าอัตราการเรียนรู้ เป็นตัวแปรสำคัญในการปรับปรุงค่าน้ำหนักแล้ว ยังมีค่าโมเมนตัม (Momentum) เป็นตัวแปรอีกส่วนหนึ่งที่จะทำการปรับปรุงค่าน้ำหนักตามสัดส่วนของการปรับปรุงค่าน้ำหนักในรอบที่ผ่านมา

เมื่อแต่ละค่าน้ำหนักถูกปรับปรุงจนได้ค่าผิดพลาดน้อยที่สุดแล้ว ก็จะสามารถใช้ค่าน้ำหนักชุดนี้แก้ปัญหาได้ ถ้าข่ายงานไม่สามารถแก้ปัญหาได้อย่างถูกต้องแสดงว่าชุดของค่าน้ำหนักนี้ยังส่งผลให้มีข้อผิดพลาดอยู่ ถึงแม้ว่าผลการคำนวณหาค่าความผิดพลาดเฉลี่ยจะมีค่าน้อยที่สุดแล้ว แต่มันอาจเป็นจุดต่ำสุดท้องถิ่น (Local Minimum) ก็ได้

การเลือกวิธีการกำหนดค่าเริ่มต้นให้กับค่าน้ำหนัก จะมีผลต่อระบบในการหาค่าความผิดพลาด เพราะอาจทำให้ค่าความผิดพลาดนั้นเป็นค่าความผิดพลาดต่ำสุดโดยรวม (Global Minimum of Error) หรือเป็นค่าความผิดพลาดต่ำสุดท้องถิ่น (Local Minimum of Error)

สิ่งสำคัญคือค่าผิดพลาดนั้นควรเป็นค่าความผิดพลาดต่ำสุดโดยรวม เพราะจะสรุปได้ว่าระบบเกิดการเรียนรู้จริง นั่นคือต้องพยายามหลีกเลี่ยงการใช้ชุดค่าน้ำหนักที่สุ่มขึ้นมา แล้วทำให้การผ่านแอคติเวชันได้ผลลัพธ์ออกมาเป็น 0 และชุดค่าน้ำหนักนั้นต้องไม่ใหญ่เกินไปหรือเล็กเกินไป และถ้าชุดค่าน้ำหนักเล็กเกินไปเมื่อผ่านแอคติเวชันก็อาจจะส่งผลให้ผลลัพธ์นั้นเข้าใกล้ 0 ด้วย ซึ่งจะทำให้การเรียนรู้ช้า

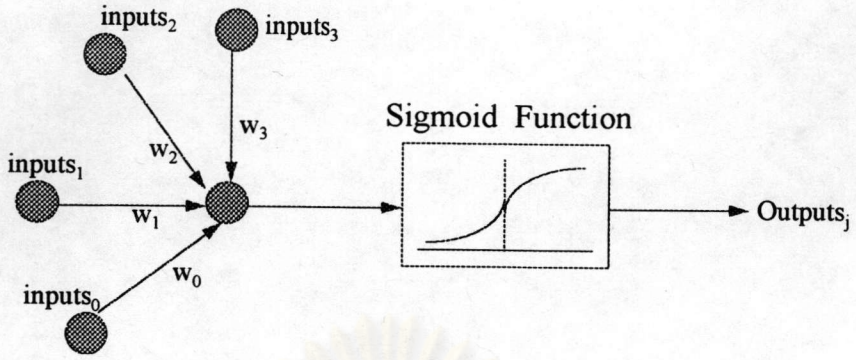
<u>ขั้นตอนการเรียนรู้และสูตรคำนวณ</u>

ข้อมูลเริ่มต้นที่ระบบต้องเตรียมก่อนจะสอนประกอบด้วย

1. ชุดข้อมูลนำเข้าที่ใช้สอนแบ่งเป็น 2 ส่วนคือ ชุดค่าข้อมูลตัวอย่างและชุดค่าข้อมูลเป้าหมายที่ต้องการจากการสอน
2. การกำหนดค่าพารามิเตอร์ต่างๆ เช่น ค่าอัตราการเรียนรู้ ค่าโมเมนตัม ค่าความผิดพลาดที่สามารถยอมรับได้ และจำนวนรอบการสอน
3. การกำหนดค่าน้ำหนัก ซึ่งในครั้งแรกที่ทำการสอนจะกำหนดให้ทำการสุ่มค่าขึ้นมา

การสอนให้เกิดการเรียนรู้ของการนิวรอลเนตเวิร์กแบบเรียนรู้โดยมีครู เช่น แบ็กพรอเพเกชัน จะมีลักษณะของให้ตัวอย่างของข้อมูลหรือชุดการสอน (Training Pattern) จะประกอบด้วยคู่ของค่าข้อมูลนำเข้าและค่าเป้าหมายที่ถูกต้องไว้ เมื่อเราส่ง 1 ชุดการสอนไปยังข่ายงานเพื่อประมวลผล หลังจากการคำนวณจะมีการเปรียบเทียบผลลัพธ์ที่ได้กับค่าเป้าหมาย เมื่อยังมีค่าผิดพลาดที่ยอมรับไม่ได้อยู่จะทำการปรับปรุงค่าน้ำหนักใหม่ เพื่อที่จะหาชุดของค่าน้ำหนักที่ทำให้ได้ผลลัพธ์ใกล้เคียงค่าเป้าหมายที่สุด

ขั้นตอนการทำงานเริ่มจากส่งค่าข้อมูลนำเข้าจากชั้นนำเข้าข้อมูลไปยังชั้นถัดไป ซึ่งคือชั้นแอบแฝง ที่ชั้นนี้จะมีการคำนวณค่าผลรวมของชุดข้อมูลนำเข้าที่ติดต่อมายังโหนดนั้นในชั้นแอบแฝงคูณกับค่าน้ำหนักของแต่ละข้อมูลนำเข้า ($\sum X_i W_i$)จากนั้นผลลัพธ์ที่ได้จะผ่านฟังก์ชันแอคติเวชัน หรือในที่นี้คือ ฟังก์ชันซิกมอยด์ และผลลัพธ์สุดท้ายจะเป็นค่าผลลัพธ์ของโหนดหนึ่งในชั้นแอบแฝง (Y_j)และเป็นข้อมูลนำเข้าในชั้นถัดไป ตามรูป 3.4



รูปที่ 3.4 แสดงการคำนวณที่โหนดผลลัพธ์

$$Y_j = F(x) = F(\sum_{i=1}^{n} x_i w_1[i][j]) \quad \text{... (1)}$$

Y_j คือ ค่าผลลัพธ์ของโหนดที่ j ที่จะเป็นข้อมูลนำเข้าของชั้นต่อไป

F(x) คือ ฟังก์ชันโลจิสติค ซึ่งเป็นประเภทหนึ่งของฟังก์ชันซิกมอยด์

$F(x) = 1 / (1 + \exp(-x))$

$w_1[i][j]$ คือ ค่าน้ำหนักชุดที่1 ของโหนดที่ i ในชั้นนำเข้าข้อมูลที่ส่งไปยังโหนดที่ j ในชั้นแอบแฝง ซึ่งค่าเริ่มต้นจะเป็นค่าสุ่มเล็กๆ เช่น $\pm$ 0.3

x_i คือ ข้อมูลนำเข้าที่ i

n คือ จำนวนข้อมูลนำเข้าของชั้นแอบแฝง

ค่าเริ่มต้นของค่าน้ำหนักในรอบแรกของการสอนจะเป็นจะเป็นค่าสุ่ม และใช้คำนวณในรอบแรกเท่านั้นเพราะในรอบถัดไปจะมีการปรับปรุงค่าน้ำหนักใหม่ และค่าผลลัพธ์ Y_j ของโหนดที่ j ในชั้นแอบแฝง จะเป็นข้อมูลนำเข้าของชั้นถัดไป ในที่นี้คือชั้นผลลัพธ์เนื่องจากเป็นข่ายงานที่มี 1 ชั้นแอบแฝง

ขั้นตอนที่กล่าวมาข้างต้นนี้จะต้องคำนวณหาค่าผลลัพธ์ของทุกๆโหนดในชั้นแอบแฝงเพื่อเป็นข้อมูลนำเข้าในชั้นผลลัพธ์ก่อน หลังจากนั้นจึงจะใช้วิธีการคำนวณแบบเดียวกันในชั้นผลลัพธ์ดังนี้

$$Z_k = F(x) = F(\sum_{j=1}^{m} Y_j w_2[j][k]) \quad \text{... (2)}$$

Z_k คือ ค่าผลลัพธ์ของโหนดที่ k ในชั้นผลลัพธ์ที่ใช้เปรียบเทียบกับค่าเป้าหมาย

Y_j คือ ข้อมูลนำเข้าของชั้นผลลัพธ์ ซึ่งคือค่าผลลัพธ์ของโหนดที่ j ของชั้นแอบแฝง

F(x) คือ ฟังก์ชันโลจิสติค ซึ่งเป็นประเภทหนึ่งของฟังก์ชันซิกมอยด์

$F(x) = 1 / (1 + \exp(-x))$

$w_2[j][k]$ คือ ค่าน้ำหนักชุดที่2 ของโหนดที่ j ในชั้นแอบแฝงที่ส่งไปยังโหนดที่ k ในชั้นผลลัพธ์ ซึ่งค่าเริ่มต้นจะเป็นค่าสุ่มเล็กๆ เช่นกัน

m คือ จำนวนข้อมูลนำเข้าของชั้นผลลัพธ์

หลังจากนั้นจะเปรียบเทียบผลลัพธ์จากการสอนกับค่าเป้าหมาย เพื่อหาค่าความผิดพลาดของแต่ละโหนดในชั้นผลลัพธ์ และรวมทั้งหาค่าความผิดพลาดของแต่ละโหนดในชั้นแอบแฝงด้วย เพื่อที่จะปรับปรุงชุดค่าน้ำหนักสำหรับใช้ในการคำนวณในรอบต่อไป โดยมีขั้นตอนดังนี้

1. หาค่าความผิดพลาดของผลลัพธ์ของทุกโหนดที่ k ในชั้นผลลัพธ์

$$\delta_k = (T_k - Z_k) Z_k (1 - Z_k) \quad \text{... (3)}$$

δ_k คือ ค่าความผิดพลาดของโหนดที่ k ในชั้นผลลัพธ์

T_k คือ ค่าเป้าหมาย (target value)

Z_k คือ ค่าผลลัพธ์ของโหนดที่ k ในชั้นผลลัพธ์ที่ใช้เปรียบเทียบกับค่าเป้าหมาย

2. หาค่าความผิดพลาดของโหนดทุกโหนดในชั้นแอบแฝงที่ทำให้เกิดความผิดพลาดในชั้นผลลัพธ์ ของทุกโหนดที่ j ในชั้นแอบแฝง

$$E_j = Y_j (1 - Y_j) \sum_{j=1}^{m} w_2[j][k] \delta_k \quad \text{...(4)}$$

E_j คือ ค่าความผิดพลาดของโหนดที่ j ในชั้นแอบแฝง

δ_k คือ ค่าความผิดพลาดของโหนดที่ k ในชั้นผลลัพธ์

Y_j คือ ค่าผลลัพธ์ของโหนดที่ j ในชั้นแอบแฝง หรือค่าข้อมูลนำเข้าในชั้นผลลัพธ์นั่นเอง

m คือ จำนวนโหนดของชั้นผลลัพธ์

$w_2[j][k]$ คือ ค่าน้ำหนักจากโหนดที่ j ในชั้นแอบแฝงไปยังโหนดที่ k ในชั้นผลลัพธ์

3. การปรับปรุงค่าน้ำหนักในชั้นแอบแฝง จากค่าผิดพลาดของชั้นผลลัพธ์ ซึ่งจะปรับปรุงค่าน้ำหนักจากโหนดที่ j ในชั้นแอบแฝงไปยังทุกโหนดในชั้นผลลัพธ์

$$\Delta w_2[j][k] = \varepsilon \delta_k Y_j + \alpha \Delta w_2[j][k]_{LastCycle} \quad \text{... (5)}$$

$\Delta w_2[j][k]$ คือ ค่าที่ใช้ปรับปรุงค่าน้ำหนักจากโหนดที่ j ในชั้นแอบแฝงไปยังโหนดที่ k ในชั้นผลลัพธ์

$\Delta w_2[j][k]_{LastCycle}$ คือ ค่าที่ใช้ปรับปรุงค่าน้ำหนักจากโหนดที่ j ในชั้นแอบแฝงไปยังโหนดที่ k ในชั้นผลลัพธ์ ในรอบการสอนที่ผ่านมา

δ_k คือ ค่าความผิดพลาดของโหนดที่ k ในชั้นผลลัพธ์

Y_j คือ ข้อมูลนำเข้าของชั้นผลลัพธ์ ซึ่งคือค่าผลลัพธ์ของโหนดที่ j ของชั้นแอบแฝง

ε คือ ค่าอัตราการเรียนรู้ (learning rate) ของชั้นผลลัพธ์

α คือ ค่าโมเมนตัม

ค่าน้ำหนักใหม่ คือ $w_2[j][k] + \Delta w_2[j][k]$

4. การปรับปรุงค่าน้ำหนักในชั้นนำเข้าข้อมูล จากค่าผิดพลาดของชั้นผลลัพธ์ ซึ่งจะปรับปรุงค่าน้ำหนักจากโหนดที่ i ในชั้นนำเข้าข้อมูล ไปยังทุกโหนดในชั้นแอบแฝง

$$\Delta w_1[i][j] = \lambda E_j x_i + \alpha \Delta w_1[i][j]_{LastCycle} \quad \text{... (6)}$$

$w_1[i][j]$ คือ ค่าน้ำหนักจากโหนดที่ i ในชั้นนำเข้าข้อมูลไปยังโหนดที่ j ชั้นแอบแฝง

$w_1[i][j]_{LastCycle}$ คือ ค่าน้ำหนักจากโหนด i ในชั้นนำเข้าข้อมูลไปยังโหนดที่ j ชั้นแอบแฝง ในรอบการสอนที่ผ่านมา

E_j คือ ค่าความผิดพลาดของโหนดที่ j ในชั้นแอบแฝง

x_i คือ ข้อมูลนำเข้าของโหนดที่i ของชั้นแอบแฝง

λ คือ ค่าอัตราการเรียนรู้ (learning rate) ของชั้นแอบแฝง

ค่าน้ำหนักใหม่ คือ $w_1[i][j] + \Delta w_1[i][j]$

จาก สูตร (6) และ (7) นี้ จะได้ว่ากระบวนการเรียนรู้จะมีการปรับปรุงค่าน้ำหนัก ตามสัดส่วนของค่าความผิดพลาด และค่าอัตราการเรียนรู้ (Learning rate) ε และจะระบบจะเรียนรู้ช้าเมื่อค่าอัตราการเรียนรู้น้อย แต่เมื่อค่าอัตราการเรียนรู้ (ε) มีค่ามาก ๆ จะมีการปรับปรุงค่าน้ำหนักมากด้วย ทำให้เกิดการแกว่งของค่าความผิดพลาด วิธีแก้ปัญหาดังกล่าวจะเพิ่มโมเมนตัม (Momentum) α เข้าไปในการคำนวณ นั่นคือส่วนหนึ่งของการปรับปรุงค่าน้ำหนัก จะขึ้นกับค่าโมเมนตัมของการปรับปรุงค่าน้ำหนักในรอบที่ผ่านมา หรือจะได้ว่า

$$\Delta w_2[j][k](n+1) = \varepsilon \delta_k Y_j + \alpha \Delta w_2[j][k](n) \qquad \text{... (7)}$$

$$\Delta w_1[i][j](n+1) = \lambda E_j x_i + \alpha \Delta w_1[i][j](n) \qquad \text{... (8)}$$

(n) เป็น ดัชนี ระบุ presentation number

α เป็น ค่าโมเมนตัมมีค่าคงที่ กำหนดผลกระทบของการปรับปรุงค่าน้ำหนักที่ผ่านมา จะมีค่าระหว่าง 0 ถึง 1

ระบบจะเรียนรู้ได้เร็วขึ้น เมื่อกำหนดค่าโมเมนตัม α และค่าอัตราการเรียนรู้ใหญ่ ๆ เพราะจะทำการปรับปรุงค่าน้ำหนักมากกว่าการกำหนดให้มีค่าเล็ก ดังนั้นมีการทดสอบค่าต่างๆที่เหมาะสมของ momentum ค่าหนึ่งคือ 0.9 แต่อย่างไรก็ตามการกำหนดค่านี้ขึ้นกับแต่ละปัญหาด้วย

จำนวนชุดข้อมูลการสอน

มีการพิสูจน์แล้วว่า ในการสอนให้ระบบสามารถเรียนรู้ในงานแบ่งแยกประเภท (Pattern Classification) ความถูกต้องจากการทดสอบระบบที่เรียนรู้แล้วจะคิดเป็นเปอร์เซนต์ของจำนวนชุดการสอน และการทดสอบพบว่าระบบจะให้คำตอบถูกต้องเมื่อข้อมูลที่ใช้ทดสอบอยู่ในกลุ่มของตัวอย่างที่ใช้สอน

โดยมีความสัมพันธ์ระหว่างจำนวนชุดการสอนกับจำนวนค่าน้ำหนักกับความถูกต้องในการทดสอบดังนี้

$$P = W/e$$

เมื่อ P คือ จำนวนชุดการสอน

W คือ จำนวนค่าน้ำหนัก

1- e คือ ความต้องการความถูกต้องในการทดสอบระบบที่เรียนรู้แล้ว

ถ้าจำนวนค่าน้ำหนัก 80 ค่า ถ้าต้องการความถูกต้อง 90 % คือ e มีค่า 0.1 และจะต้องมีจำนวนชุดการสอน 800 ชุด

รอบการสอน (Cycle or Epoch)

ปัญหาอย่างหนึ่งของการสอนคือควรจะกำหนดให้มีการสอนกี่รอบ เนื่องจากการสอนมากเกินจนเกินไปไม่ได้ทำให้ระบบจะเรียนรู้ได้มากขึ้น เราจึงจำเป็นต้องประมาณการจำนวนน้อยที่สุดของการสอนที่จำเป็นสำหรับการแก้ปัญหาหนึ่งๆที่ทำให้ระบบเกิดการเรียนรู้ และสามารถหยุดการสอนได้ พบว่าระบบควรหยุดการสอนเมื่อค่าเฉลี่ยของค่าความผิดพลาดแต่ละชุดข้อมูลถึงจุดต่ำสุด

เมื่อระบบอยู่ในขั้นตอนเรียนรู้ แต่ละชุดการสอนจะมีการคำนวณค่าเฉลี่ยของค่าความผิดพลาด และค่านี้จะมีค่าลดลงเมื่อระบบเริ่มที่จะเรียนรู้ เมื่อใดก็ตามที่ค่านี้เริ่มสูงขึ้นแสดงว่าการเรียนรู้นั้นระบบเริ่มสูญเสียการเรียนรู้แล้ว

ในทางปฏิบัติเราต้องลองผิดลองถูก ด้วยวิธีนี้ทุกอย่างขึ้นอยู่กับประสบการณ์ของผู้ใช้ คืออาจกำหนดรอบการสอนให้มีค่ามากๆ และอาจจะหยุดการสอนเมื่อเห็นว่าค่าเฉลี่ยของค่าความผิดพลาดอยู่ในขอบเขตที่ยอมรับได้ ดังนั้นหลังการสอนจำเป็นต้องมีการตรวจสอบความสามารถของระบบ

การแปลงข้อมูลนำเข้า

สิ่งสำคัญอย่างหนึ่งการสอนคือการแปลงข้อมูลนำเข้าจากปัญหาที่ต้องการแก้ให้มีประสิทธิผล นั่นคือข้อมูลนำเข้าควรจะมีรูปแบบใดถึงจะทำให้ระบบเรียนรู้ได้รวดเร็ว และสามารถแก้ปัญหาได้

พบว่าเมื่อ ผลลัพธ์จากการผ่านแอคติเวชันแล้วมีค่าใกล้ 0 มักจะเรียนรู้ช้า หรืออาจจะไม่เรียนรู้เลย ดังนั้นการปรับปรุงการเรียนรู้ให้ดีขึ้นข้อมูลนำเข้าไม่ควรอยู่ในรูปไบนารี คือ ค่า 0 และ 1 เท่านั้น นั่นคือควรจะเป็นค่าที่เป็นค่าที่เข้าใกล้ 0 และ 1 และมีทั้งค่าลบ และค่าบวก

จากการทดสอบการแก้ปัญหา XOR ที่มีโครงสร้าง 2-4-1 คือ จำนวนโหนดข้อมูลนำเข้ามีขนาด 2 และจำนวนโหนดข้อมูลแอบแฝงมีขนาด 4 และจำนวนโหนดข้อมูลผลลัพธ์มีขนาด 1 นั่นคือต้องผลลัพธ์เพียงถูกหรือผิดเท่านั้น โดยกำหนด ค่าน้ำหนักเริ่มต้นอยู่ในขอบเขต -0.5 ถึง 0.5 และเมื่อกำหนดข้อมูลนำเข้าแบบไบนารีก็จะใช้แอคติเวชันแบบไบนารี และเมื่อข้อมูลนำเข้าแบบไบโพลาก็จะใช้แอคติเวชันแบบไบโพลา

เมื่อแทนค่าข้อมูลนำเข้าแบบไบนารีพบว่าต้องสอนถึง 3000 รอบถึงจะเรียนรู้ และในการแก้ปัญหาเดียวกันเมื่อแทนค่าแบบไบโพลาระบบเรียนรู้เร็วขึ้นสอนเพียง 390 รอบ ระบบก็เรียนรู้ได้แล้ว