

2.2 Artificial Neural Networks [2]

เป็นแขนงหนึ่งของ (Computational Intelligence) ซึ่งได้รับแรงบรรดาลใจมาจากการทำงานของสมองของมนุษย์ ซึ่งประกอบด้วยหน่วยเล็กๆ เรียกว่า cell Neuron แต่ละ Neuron ก็จะเชื่อมต่อโยงกันด้วยเส้นประสาทเรียกว่า ไซแนปส์ (Synapse) เพื่อส่งสัญญาณไฟฟ้า ที่เกิดจากสิ่งเร้าต่างๆ ว่าจะตอบสนองต่อสิ่งเร้านั้นอย่างไร โดยแต่ละ Neuron จะได้รับ Input หลาย ๆ ส่วนจากกิ่งก้านสาขาของ Dendrite และนำมาประมวลผล ออกมายield 1 Output ออกไปที่ Axon เพื่อส่งต่อไปให้ Dendrite ของ Neuron อื่น ๆ ใช้เป็น Input ต่อไป

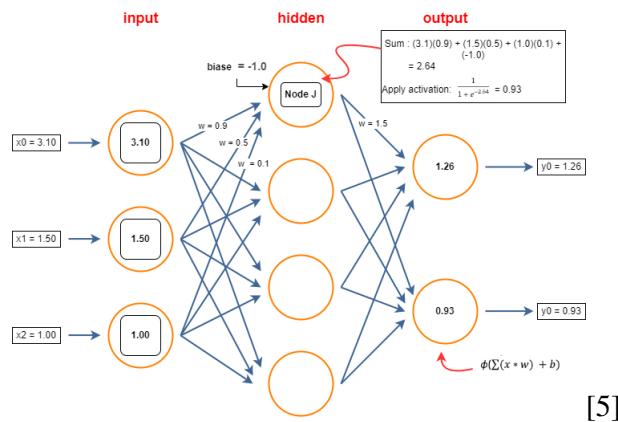
เมื่อมนุษย์เติบโตขึ้น หาก Neuron ไหนตอบสนองต่อสิ่งเร้าประเภทไหนได้ดี ก็จะสามารถส่งสัญญาณไฟฟ้าที่มีความแรงมากกว่า Neuron อื่นๆ เมื่อ Neuron หลายๆ อันต่อกันหลายๆ Layer ก็จะกลายเป็น Neural network ของมนุษย์

โดยในทาง Computational Intelligence จะใช้ node เป็นตัวแทนของ Neuron โดยจะเรียกเป็นชั้นๆ (Layer) โดยสัญญาณที่ส่งออกจากแต่ละ node จะมี weight ที่กำหนดความแรงของสัญญาณนั้นๆ เมื่อมีหลายๆ node และต่อกันหลายๆ layers ก็จะกลายเป็น neural network ขนาดใหญ่ เรียกว่า deep learning ซึ่งเลียนแบบการการทำงานของสมองมนุษย์ ทำให้ computers สามารถ process ข้อมูลในลักษณะเดียวกับที่ สมองของมนุษย์ทำการประมวลผลข้อมูล

2.2.1 Multilayer Perceptron [3]

เป็นส่วนพื้นฐานของ neural network เป็นการ ที่ node ในแต่ละ layer เชื่อมต่อกับ ทุก node ของ layer+1 (fully connected) โดยทุกเส้นการเชื่อมต่อของ $node_i$ กับ $node_j$ จะมี weight w_{ji} ซึ่งเป็นความแรงของสัญญาณอยู่

โดยจะมี Input layer สำหรับข้อมูล (สิ่งเร้า) hidden layer ในการตัดสินใจ และ output layer ในการเลือกการกระทำกับสิ่งเร้านั้นๆ



รูปที่ 2.2: Multilayer perceptron

2.2.2 Multilayer perceptron - Feed forward [4]

โดยในแต่ละ node ที่ไม่ใช่ Input layer จะรับค่าผลรวมจาก node ก่อนหน้า เป็นผลรวมจากทุก Input (ทุก Dendrite) ของ node นั้นๆ

โดยสมการผลรวม Input ของ $node_j$

$$node_j = v_j = \sum (\forall w_{j_i}) + biase \quad (2.1)$$

Activation function [6] คือ พังก์ชันที่รับ ผลรวมจากทุก Input (v_j) และคำนวณว่าจะส่งต่อเป็น Output เท่าไร ซึ่งมี function มากมายที่นิยมใช้ทำ Activation function ตัวอย่างเช่น

- Linear activation: $\phi(\mathbf{v}) = a + \mathbf{v}'\mathbf{b}$,
- ReLU activation: $\phi(\mathbf{v}) = \max(0, a + \mathbf{v}'\mathbf{b})$,
- Heaviside activation: $\phi(\mathbf{v}) = 1_{a+\mathbf{v}'\mathbf{b}>0}$,
- Logistic activation: $\phi(\mathbf{v}) = (1 + \exp(-a - \mathbf{v}'\mathbf{b}))^{-1}$.

โดยในทุกๆ $node_j$ จะมี Activation function สำหรับคำนวณค่า output ที่จะส่งต่อไปยัง layer ถัดไป

$$outputnode_j = Y_j = y(v_i) \quad (2.2)$$

ตัวอย่าง ReLU Activation function ค่าที่ส่งออกไปยัง layer ถัดไปจะเป็น

$$y(v_i) = \max(0, v_i) \quad (2.3)$$

2.2.3 Classification

จาก Input layer ข้อมูลจะถูกส่งต่อไปยัง Layer ถัดๆไป เรียกว่า Feed forward และเมื่อถึง Layer สุดท้ายของ Neural Network เรียกว่า Output layer โดยส่วนใหญ่ใน output layer นี้จะมีจำนวนของ node เท่ากับจำนวนของ class ของข้อมูลที่จะทำการ classification โดยแต่ละ node ใน Output layer จะเป็นตัวแทนของ class ซึ่ง หาก node ได้ให้ค่า Output node เ酵ะที่สุด Input data ก็ถูก classify เป็น class ของ node นั้นๆ

2.2.4 Training

คือการฝึกสอนโดยใช้ dataset เปลี่ยน weight ในแต่ละเส้นของ MLP เพื่อให้ตอบสนองต่อ Input ให้ใกล้เคียงกับสิ่งที่ควรจะเป็นมากขึ้น โดยหนึ่งในวิธีการเปลี่ยน weight คือ backpropagation

โดยค่า error ที่ได้มาจากการ output layer กับ ค่าที่ควรได้จาก Input ที่ได้เข้าไป เพื่อเปลี่ยน weight ให้ค่า error มีค่าน้อยลงกว่าเดิม

$$outputnode_j = Y_j = y(v_i) \quad (2.4)$$

สามารถคำนวณหาค่า error ในแต่ละ node ของ output layer ได้โดย

$$e_j(n) = d_j(n) - y_j(n) \quad (2.5)$$

where $d_j(n)$ is the desired target value , and $y_j(n)$ value produced by the perceptron at node $_j$

$$\mathcal{E}(n) = \frac{1}{2} \sum_{\text{output node } j} e_j^2(n). \quad (2.6)$$

และใช้ gradient descent ในการเปลี่ยน weight ในแต่ละเส้น

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n) \quad (2.7)$$