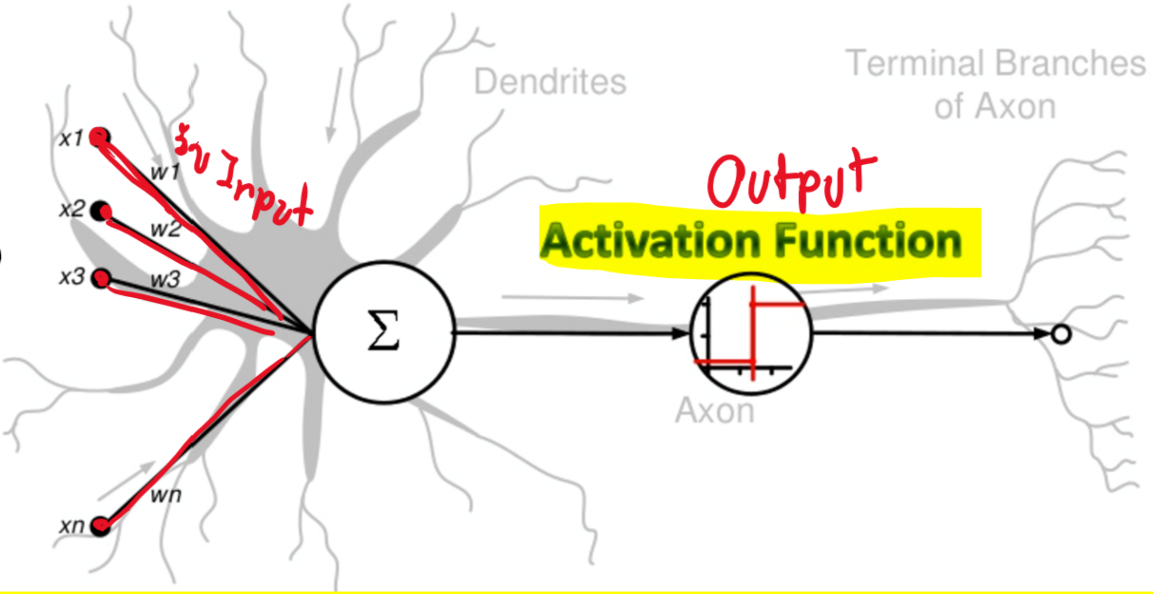
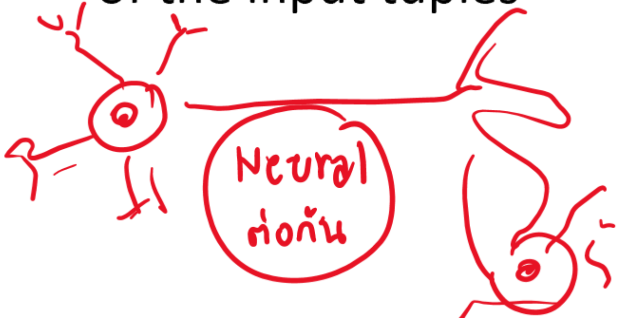
Artificial +
Neural Network for Classification

เลียนแบบเซลล์ประสาทของคน

- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
  - During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples

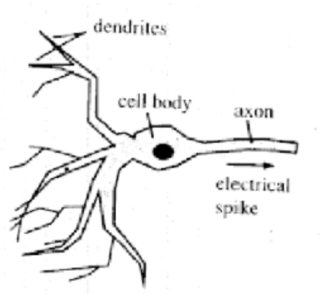


Artificial Neural Networks as an analogy of Biological Neural Networks



6.7 ข่ายงานประสาทเทียม

ข่ายงานประสาทเทียม (Artificial Neural Network) เป็นการจำลองการทำงานบางส่วนของสมองมนุษย์ เซลล์ประสาท (neuron) ในสมองของคนเราประกอบด้วยนิวเคลียส (nucleus) ตัวเซลล์ (cell body) ใยประสาทนำเข้า (dendrite) แกนประสาทนำออก (axon) แสดงใน รูปที่ 6–34



รูปที่ 6–34 เซลล์ประสาท

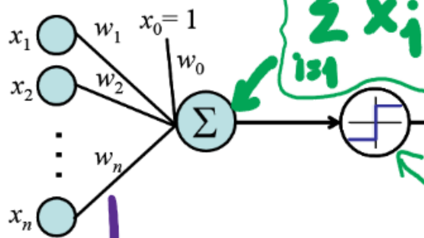
เดนไดรท์ทำหน้าที่รับสัญญาณไฟฟ้าเคมีซึ่งส่งมาจากเซลล์ประสาทใกล้เคียง เซลล์ประสาทตัวหนึ่งๆ จะเชื่อมต่อกับเซลล์ตัวอื่นๆ ประมาณ 10,000 ตัว เมื่อสัญญาณไฟฟ้าเคมีที่รับเข้ามาเกินค่าค่าหนึ่ง เซลล์จะถูกกระตุ้นและส่งสัญญาณไปทางแกนประสาทนำออกไปยังเซลล์อื่นๆ ต่อไป ประมาณกันว่าสมองของคนเรามีเซลล์ประสาทอยู่ทั้งสิ้นประมาณ 10^{11} ตัว

6.7.1 เพอร์เซปตรอน

เพอร์เซปตรอน (perceptron) เป็นข่ายงานประสาทเทียมแบบง่ายมีหน่วยเดียวที่จำลองลักษณะของเซลล์ประสาทดังรูปที่ 6–35

$\sum_{i=1}^{n} x_i w_i$

features



Y

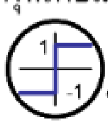
N

Activate Function

รูปที่ 6–35 เปอร์เซปตรอน

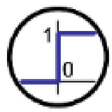
weights (สร้างโมเดล) → ยิ่ง weight มาก features มีความสัมพันธ์มาก

เพอร์เซปตรอนรับอินพุตเป็นเวกเตอร์จำนวนจริงแล้วคำนวณหาผลรวมเชิงเส้น (linear combination) แบบถ่วงน้ำหนักของอินพุต ($x_1, x_2, \ldots, x_n$) โดยที่ค่า $w_1, w_2, \ldots, w_n$ ในรูปเป็นค่าน้ำหนักของอินพุตและให้เอาต์พุต (o) เป็น 1 ถ้าผลรวมที่ได้มีค่าเกินค่าขีดแบ่ง (θ) และเป็น -1 ถ้าไม่เกิน ส่วน w_0 ในรูปเป็นค่าลบของค่าขีดแบ่งดังจะได้อธิบายต่อไป และ x_0 เป็นอินพุตเทียมกำหนดให้มีค่าเป็น 1 เสมอ



ฟังก์ชันกระตุ้น

ในรูปแสดง*ฟังก์ชันกระตุ้น (activation function)* ชนิดที่เรียกว่า*ฟังก์ชันสองขั้ว (bipolar function)* ซึ่งแสดงผลของเอาต์พุตเป็น 1 กับ -1 ฟังก์ชันกระตุ้นอื่นๆ ที่นิยมใช้ก็อย่างเช่น *ฟังก์ชันไบนารี (binary function)* ซึ่งแสดงผลของเอาต์พุตเป็น 1 กับ 0 และเขียนแทนด้วยรูป



เราสามารถแสดงเอาต์พุต (o) ในรูปของฟังก์ชันของอินพุต ($x_1, x_2, \ldots, x_n$) ได้ดังนี้

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & if \;\; w_1x_1 + w_2x_2 + \cdots + w_nx_n > \theta \\ -1 & if \;\; w_1x_1 + w_2x_2 + \cdots + w_nx_n < \theta \end{cases} \qquad (6.7)$$

เอาต์พุตเป็นฟังก์ชันของอินพุตในรูปของผลรวมเชิงเส้นแบบถ่วงน้ำหนัก น้ำหนักจะเป็นตัวกำหนดว่าในจำนวนอินพุตนั้น อินพุต (x_i) ตัวใดมีความสำคัญต่อการกำหนดค่าเอาต์พุต ตัวที่มีความสำคัญมากจะมีค่าสัมบูรณ์ของน้ำหนักมาก ส่วนตัวที่มีความสำคัญน้อยจะมีค่าใกล้ศูนย์ ในกรณีที่ผลรวมเท่ากับค่าขีดแบ่งค่าเอาต์พุตไม่นิยาม (จะเป็น 1 หรือ -1 ก็ได้)

จากฟังก์ชันในสูตรที่ (6.7) เราจัดรูปใหม่โดยย้าย θ ไปรวมกับผลรวมเชิงเส้นแล้วแทน $-\theta$ ด้วย w_0 เราจะได้ฟังก์ชันของเอาต์พุตดังด้านล่างนี้

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & if \;\; w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 & if \;\; w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n < 0 \end{cases} \qquad (6.8)$$

กำหนดให้ $g(\vec{x}) = \sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{x}$ โดยที่ $\vec{x}$ แทนเวกเตอร์อินพุต เราสามารถเขียนฟังก์ชันของเอาต์พุตได้ใหม่ดังนี้

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & if \;\; g(\vec{x}) > 0 \\ -1 & if \;\; g(\vec{x}) < 0 \end{cases} \qquad (6.9)$$

สมมติว่าเรามีอินพุตสองตัวคือ x_1 และ x_2 ซึ่งแสดงค่าส่วนสูงและน้ำหนักของเด็กนักเรียนประถมและหลังจากที่แพทย์ตรวจร่างกายของเด็กโดยละเอียดแล้วได้จำแนกนักเรียน

ตารางที่ 6–17 อัลกอริทึมกฎการเรียนรู้เพอร์เซปตรอน

Algorithm: Perceptron-Learning-Rule

1. Initialize weights w_i of the perceptron. → สำหรับ Data แต่ละตัว
2. UNTIL the termination condition is met DO
   - 2.1 FOR EACH training example DO
     - Input the example and compute the output. → $f(\Sigma wx)$
     - Change the weights if the output from the perceptron is not equal to the target output using the following rule.

weightใหม่ weightเก่า

$$w_i \leftarrow w_i + \Delta w_i \rightarrow \text{penulty}$$

$$\Delta w_i \leftarrow \underset{0.5}{\alpha}(\overset{0}{t} - \overset{1}{o}) x_i$$ Input แต่ละตัว

where t, o and α are the target output, the output from the perceptron and the learning rate, respectively.

การปรับน้ำหนักตามกฎการเรียนรู้เพอร์เซปตรอนโดยใช้อัตราการเรียนรู้ที่มีค่าน้อยเพียงพอ จะได้ระนาบหลายมิติที่จะลู่เข้าสู่ระนาบหนึ่งที่สามารถแบ่งข้อมูลออกเป็นสองส่วน (ในกรณีที่ข้อมูลสามารถแบ่งได้) เพื่ออธิบายผลที่เกิดจากการปรับค่าน้ำหนัก เราจะลองพิจารณาพฤติกรรมของกฎการเรียนรู้นี้ดูว่าทำไมการปรับน้ำหนักเช่นนี้จึงลู่เข้าสู่ระนาบที่แบ่งข้อมูลได้อย่างถูกต้อง

- พิจารณากรณีแรกที่เพอร์เซปตรอนแยกตัวอย่างสอนตัวหนึ่งที่รับเข้ามาได้ถูกต้อง กรณีนี้จะพบว่า (t-o) จะมีค่าเป็น 0 ดังนั้น Δw_i ไม่เปลี่ยนแปลงเพราะ $\Delta w_i = \alpha(\text{t-o})x_i$
- พิจารณาในกรณีที่เพอร์เซปตรอนให้เอาต์พุตเป็น –1 แต่เอาต์พุตเป้าหมายหรือค่าที่แท้จริงเท่ากับ 1 ในกรณีนี้หมายความว่าค่าที่เราต้องการคือ 1 แต่ค่าน้ำหนักไม่เหมาะสม ดังนั้นเพื่อที่จะทำให้เพอร์เซปตรอนให้เอาต์พุตเป็น 1 น้ำหนักต้องถูกปรับให้สามารถเพิ่มค่าของ $\vec{w} \cdot \vec{x}$ ในกรณีนี้หมายความว่าผลรวมเชิงเส้นน้อยเกินไปและน้อยกว่า 0 จึงได้เอาต์พุตเป็น -1 ดังนั้นสิ่งที่เราต้องการคือการเพิ่มค่าผลรวมเชิงเส้นเพราะถ้าเราเพิ่มค่าได้เรื่อยๆ จนมากกว่า 0 เพอร์เซปตรอนจะให้เอาต์พุตเป็น 1 ซึ่งตรงกับที่เราต้องการ พิจารณาดูดังต่อไปนี้ว่าการปรับค่าโดยกฎเรียนรู้ทำให้ผลรวมเชิงเส้นเพิ่มขึ้นได้อย่างไร กรณีนี้เราจะได้ว่า (t-o) เท่ากับ (1-(-1)) มีค่าเป็น 2 และลองพิจารณาค่าของอินพุต x_i แยกกรณีดังนี้

    - ถ้า $x_i > 0$ จะได้ว่า Δw_i มากกว่า 0 เพราะว่า $\Delta w_i \leftarrow \alpha(t-o)x_i$ และ α มากกว่า 0, (t-o) = 2 และ $x_i > 0$ จากสมการการปรับน้ำหนัก $w_i \leftarrow w_i + \Delta w_i$ เมื่อ Δw_i มากกว่า 0 จะทำให้ w_i มีค่าเพิ่มขึ้นและ $\sum w_i x_i$ ก็จะมีค่าเพิ่มขึ้น เมื่อผลรวมมีค่ามากขึ้นแสดงว่าการปรับไปในทิศทางที่ถูกต้องคือเมื่อปรับไปจนกระทั่งได้ผลรวมมากกว่า 0 จะทำให้เพอร์เซปตรอนเอาต์พุตได้ถูกต้องยิ่งขึ้น
    - ถ้า $x_i < 0$ เราจะได้ว่า $\alpha(t-o)x_i$ จะมีค่าน้อยกว่า 0 แสดงว่า w_i ตัวที่คูณกับ x_i ที่น้อยกว่า 0 จะลดลงทำให้ $\sum w_i x_i$ เพิ่มขึ้นเหมือนเดิม เพราะ x_i เป็นค่าลบและ w_i มีค่าลดลง ในที่สุดก็จะทำให้เพอร์เซปตรอนให้เอาต์พุตได้ถูกต้องยิ่งขึ้น
- ในกรณีที่เพอร์เซปตรอนให้เอาต์พุตเป็น 1 แต่เอาต์พุตเป้าหมายหรือค่าที่แท้จริงเท่ากับ -1 จะได้ว่า w_i ของ x_i ที่เป็นค่าบวกจะลดลง ส่วน w_i ของ x_i ที่เป็นค่าลบจะเพิ่มขึ้นและทำให้การปรับเป็นไปในทิศทางที่ถูกต้องเช่นเดียวกับในกรณีแรก

6.7.2 ตัวอย่างการเรียนฟังก์ชัน AND และ XOR ด้วยกฎเรียนรู้เพอร์เซปตรอน

พิจารณาตัวอย่างการเรียนรู้ของเพอร์เซปตรอนโดยจะให้เรียนรู้ฟังก์ชัน 2 ฟังก์ชัน ฟังก์ชันแรกคือฟังก์ชัน AND แสดงในตารางที่ 6–18 ในกรณีนี้เราใช้ฟังก์ชันไบนารีเป็นฟังก์ชันกระตุ้น

X Y

T∧T ≡ T

T∧F ≡ F

ตารางที่ 6–18 ฟังก์ชัน AND(x1,x2)

x_1	x_2	เอาต์พุตเป้าหมาย
0	0	0
0	1	0
1	0	0
1	1	1

ฟังก์ชัน AND ตามตารางด้านบนนี้จะให้ค่าที่เป็นจริงก็ต่อเมื่อ x1 และ x2 เป็นจริงทั้งคู่ (ดูที่สดมภ์เอาต์พุตเป้าหมาย) ผลการใช้กฎการเรียนรู้เพอร์เซปตรอนกับฟังก์ชัน AND แสดงในตารางที่ 6–19

ตารางที่ 6–19 ผลการเรียนรู้ฟังก์ชัน AND โดยกฎการเรียนรู้เพอร์เซปตรอน

1 epoch = 1 รอบ

1 รอบ

Perceptron Learning Example - Function AND											
					$\sum_{i=0}(x_iw_i)$			$f(\sum_{i=0}(x_iw_i))$			
		Bias Input x0=+1				y	Alpha =	0.5	learning rate		
Input	Input				Net Sum	Target	Actual	Alpha*	Weight Values		
x1	x2	1.0*w0	x1*w1	x2*w2	Input	Output	Output	Error	w0	w1	w2
									0.1	0.1	0.1
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	0.10	0.10
0	1	-0.40	0.00	0.10	-0.30	0	0	0.00	-0.40	0.10	0.10
1	0	-0.40	0.10	0.00	-0.30	0	0	0.00	-0.40	0.10	0.10
1	1	-0.40	0.10	0.10	-0.20	1	0	0.50	0.10	0.60	0.60
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	0.60	0.60
0	1	-0.40	0.00	0.60	0.20	0	1	-0.50	-0.90	0.60	0.10
1	0	-0.90	0.60	0.00	-0.30	0	0	0.00	-0.90	0.60	0.10
1	1	-0.90	0.60	0.10	-0.20	1	0	0.50	-0.40	1.10	0.60
0	0	-0.40	0.00	0.00	-0.40	0	0	0.00	-0.40	1.10	0.60
0	1	-0.40	0.00	0.60	0.20	0	1	-0.50	-0.90	1.10	0.10
1	0	-0.90	1.10	0.00	0.20	0	1	-0.50	-1.40	0.60	0.10
1	1	-1.40	0.60	0.10	-0.70	1	0	0.50	-0.90	1.10	0.60
0	0	-0.90	0.00	0.00	-0.90	0	0	0.00	-0.90	1.10	0.60
0	1	-0.90	0.00	0.60	-0.30	0	0	0.00	-0.90	1.10	0.60
1	0	-0.90	1.10	0.00	0.20	0	1	-0.50	-1.40	0.60	0.60
1	1	-1.40	0.60	0.60	-0.20	1	0	0.50	-0.90	1.10	1.10
0	0	-0.90	0.00	0.00	-0.90	0	0	0.00	-0.90	1.10	1.10
0	1	-0.90	0.00	1.10	0.20	0	1	-0.50	-1.40	1.10	0.60
1	0	-1.40	1.10	0.00	-0.30	0	0	0.00	-1.40	1.10	0.60
1	1	-1.40	1.10	0.60	0.30	1	1	0.00	-1.40	1.10	0.60
0	0	-1.40	0.00	0.00	-1.40	0	0	0.00	-1.40	1.10	0.60
0	1	-1.40	0.00	0.60	-0.80	0	0	0.00	-1.40	1.10	0.60
1	0	-1.40	1.10	0.00	-0.30	0	0	0.00	-1.40	1.10	0.60
1	1	-1.40	1.10	0.60	0.30	1	1	0.00	-1.40	1.10	0.60

ขั้นตอนแรกเริ่มจากการสุ่มค่า w_0 จนถึง w_2 ในที่นี้กำหนดให้เป็น 0.1 ทั้งสามตัว จากนั้นก็เริ่มป้อนตัวอย่างเข้าไป (ทีละแถว) ตัวอย่างแรกได้ผลรวมเชิงเส้น (Net Sum) เป็น 0.10 ซึ่งมากกว่า 0 ดังนั้นเปอร์เซปตรอนจะให้เอาต์พุตจริง (Actual Output) ออกมาเป็น 1 ซึ่งผิดเพราะเอาต์พุตเป้าหมาย (Target Output) จะต้องได้เป็น 0 ทำให้อัตราการเรียนรู้คูณค่าผิดพลาด (Alpha x Error) ได้ –0.50 หลังจากนี้ก็นำไปปรับน้ำหนักตาม $w_i \leftarrow w_i + \Delta w_i$ และ $\Delta w_i \leftarrow \alpha(t-o)x_i$ ดังนั้นจะได้เป็น $w_0 \leftarrow w_0 + \alpha(t-o)x_0 = w_0 + 0.50(-1) \times 1 = 0.10 + (-0.5) = -0.4$ ต่อไปก็ปรับค่า w_1 ในทำนองเดียวกัน $w_1 \leftarrow w_1 + \alpha(t-o)x_1 = w_1 + 0.50(-1) \times 0$ ดังนั้น w_1 จะเท่ากับ 0.10 คือไม่เปลี่ยนแปลง เช่นเดียวกับ w_2 ที่ไม่เปลี่ยนแปลง จะเห็นได้ว่าแม้มีค่าผิดพลาดแต่ไม่มีการปรับค่า w_1 และ w_2 เนื่องจากอินพุตที่ใส่เข้าไปเป็น 0 ทำ

Classifier Evaluation Metrics: Confusion Matrix

- **Confusion Matrix:**

ทาย Class

ทายว่า Positive ถูก Precision Recall

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

ทายว่า Positive ผิด

ทายว่า Negative ผิด

ทายว่า Negative ถูก

- In a confusion matrix w. *m* classes, $CM_{i,j}$ indicates # of tuples in class *i* that were labeled by the classifier as class *j*
  - May have extra rows/columns to provide totals
- **Example of Confusion Matrix:**

test_pos test_Neg

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes (Positive)	**6954**	**46**	7000
buy_computer = no (Negative)	**412**	**2588**	3000
Total	7366	2634	10000

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier accuracy,** or recognition rate
  - Percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All
- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class imbalance problem**
  - One class may be *rare*
    - E.g., fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - Measures handle the class imbalance problem
    - **Sensitivity** (recall): True positive recognition rate
      - **Sensitivity = TP/P**
    - **Specificity**: True negative recognition rate
      - **Specificity = TN/N**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: Exactness: what % of tuples that the classifier labeled as positive are actually positive?

$$P = \text{Precision} = \frac{TP}{TP + FP}$$

→ ตัวที่ Model ทายว่าเป็น Positive ถูกต้องมากแค่ไหน?

- **Recall:** Completeness: what % of positive tuples did the classifier label as positive?

$$R = \text{Recall} = \frac{TP}{TP + FN}$$

→ โมเดลเราหา ตัวที่เป็น positive จริงๆ ถูกต้อง มากน้อยแค่ไหน

  - Range: [0, 1]
- The "inverse" relationship between precision & recall
- ***F* measure (or *F*-score):** harmonic mean of precision and recall
  - In general, it is the weighted measure of precision & recall

$$F_\beta = \frac{1}{\alpha \cdot \frac{1}{P} + (1 - \alpha) \cdot \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Assigning β times as much weight to recall as to precision)

  - ***F1-measure (balanced F-measure)***
    - That is, when β = 1,

$$F_1 = \frac{2PR}{P + R}$$

P - Precision

R - Recall

$\boxed{\text{F สูงยิ่งดี}}$