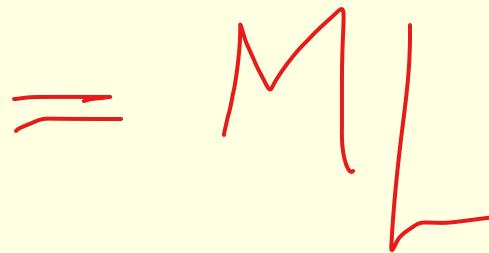


บทที่ 10

เรื่อง Artificial Neural Network
(โครงข่ายประสาทเทียม)



การเรียนรู้ Learning

F1 E2 C1 S5

1. Supervised Learning การเรียนแบบมีการสอน

เป็นการเรียนแบบที่มีการตรวจคำตอบเพื่อให้วงจรปั้ยปรับตัว ชุดข้อมูลที่ใช้สอนวงจรปั้ยจะมีคำตอบไว้โดยตรวจดูว่าวงจรปั้ยให้คำตอบที่ถูกหรือไม่ ถ้าตอบไม่ถูก วงจรปั้ยก็จะปรับตัวเองเพื่อให้ได้คำตอบที่ดีขึ้น (เปรียบเทียบกับคน เมื่อนักเรียนโดยมีครูผู้สอนค่อยแนะนำ)

2. Unsupervised Learning การเรียนแบบไม่มีการสอน

เป็นการเรียนแบบไม่มีผู้แนะนำ ไม่มีการตรวจคำตอบว่าถูกหรือผิด วงจรปั้ยจะจัดเรียงโครงสร้างด้วยตัวเองตามลักษณะของข้อมูล ผลลัพธ์ที่ได้ วงจรปั้ยจะสามารถจัดหมวดหมู่ของข้อมูลได้ (เปรียบเทียบกับคน เช่นการที่เราสามารถแยกแยะพันธุ์พืช พันธุ์สัตว์ตามลักษณะรูปร่างของมันได้เองโดยไม่มีครรสอน)

ความทัศน์รย์ของสมองมนุษย์

- สมองมนุษย์มีประสิทธิภาพและมั่นคงมาก ทุกวันมีเซลล์ประสาทในสมองตาย โดยที่ ไม่ส่งผลกระทบต่อประสิทธิภาพของสมองโดยรวม
- ระบบสมองของมนุษย์ดียุ่นมาก สามารถปรับตัวเข้ากับสิ่งแวดล้อมใหม่ โดยการเรียนรู้ (ผิดกับคอมพิวเตอร์ที่จะต้องโปรแกรมใหม่)
- สมองมนุษย์สามารถจัดการกับข้อมูลที่มีความ ไม่แน่นอน, มี สัญญาณรบกวน, และ ไม่สม่ำเสมอ ได้ดี
- สมองสามารถประมวลผลข้อมูลขนาดมหาศาล เช่นรูปภาพ ในลักษณะการประมวลผลแบบขนาน ได้ดี
- สมองมีขนาดเล็กและใช้พลังงานน้อย ✓
- โครงสร้างของสมองมนุษย์ได้วิวัฒนาการมาเป็นเวลาหลายล้านปี และ ได้รับพิสูจน์ จากธรรมชาติ ทราบว่า จะทั่งทุกวันนี้

โครงสร้างของสมองมนุษย์ VS คอมพิวเตอร์

สมอง



- สมองประกอบด้วย Neuron จำนวนประมาณ 100,000 ล้านเซลล์

- เซลล์ Neuron แต่ละเซลล์มีการทำงานที่ไม่ซับซ้อน โดยสัญญาณที่ส่งออกมาจากแต่ละเซลล์จะเป็นลูกคลื่นสัญญาณทางไฟฟ้า

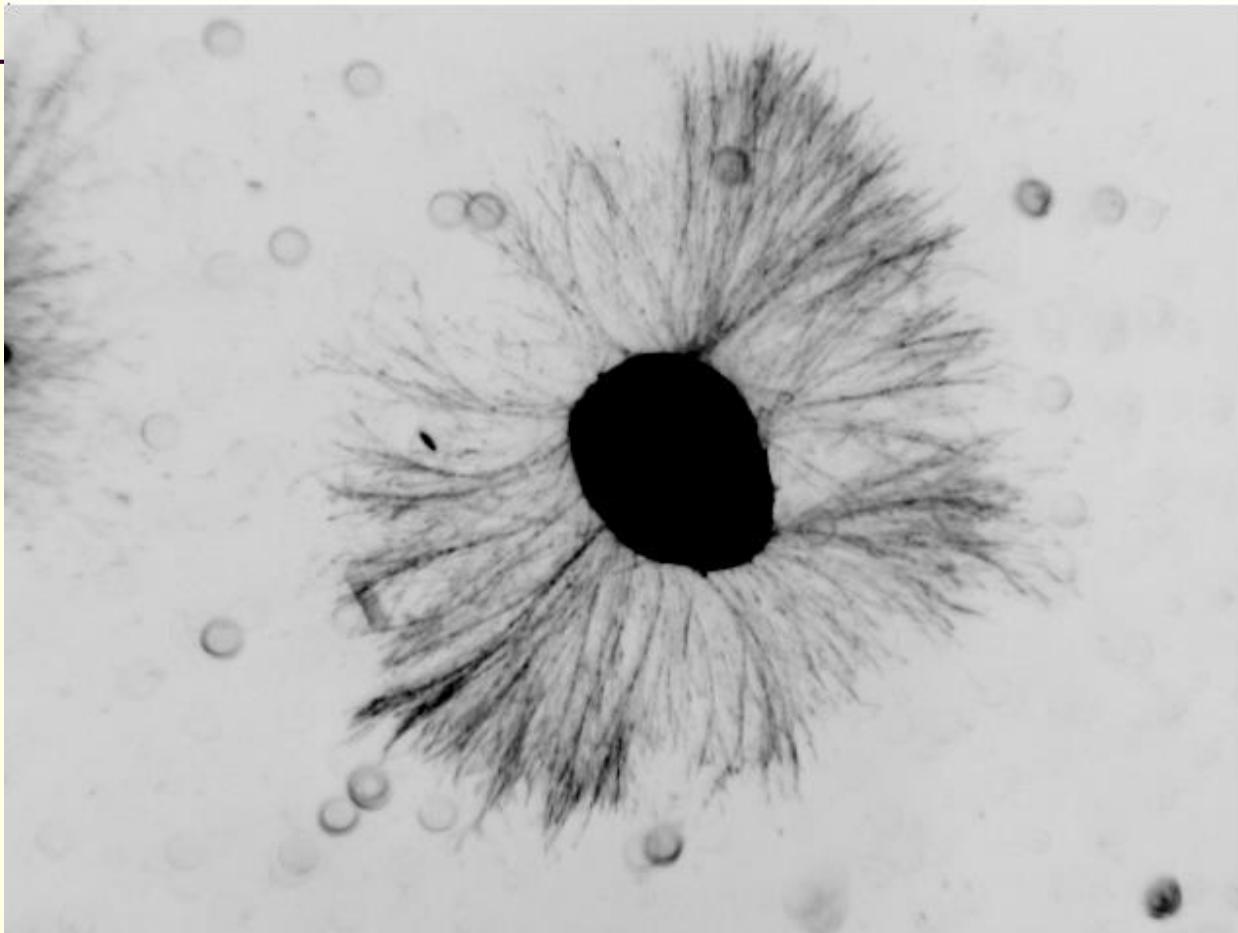
- ปกติ Neuron แต่ละเซลล์จะเชื่อมต่อกับ Neuron เซลล์อื่นประมาณ 10,000 เซลล์ บางเซลล์อาจมีจุดต่อมากกว่า 200,000 จุด

- สมองไม่มีความจำส่วนกลาง แต่สมองจะเรียนรู้และจดจำความจำระยะยาวโดยการปรับโครงสร้างของสมองเป็นหลัก (การจัดรูปเก็บกันสาขของเซลล์ประสาท)

- สมองจะเรียนรู้ได้ต้องมีการฝึกหลายครั้งจนเกิดความชำนาญ

Example of neurons

Y e Z J N



From http://www.ams.sunysb.edu/research/pinezich/neuron_reconstruction/

โครงสร้างของสมองมนุษย์ VS คอมพิวเตอร์

คอมพิวเตอร์

- หน่วยประมวลผลของคอมพิวเตอร์ (**CPU**) มีความซับซ้อนมาก มีความสามารถมาก ในเครื่องคอมพิวเตอร์แต่ละเครื่อง มีจำนวน **CPU** ไม่มาก และการเชื่อมต่อระหว่าง **CPU** ไม่ได้ซับซ้อน
- คอมพิวเตอร์มีหน่วยความจำส่วนกลาง ใช้เก็บโปรแกรมและข้อมูล
- คอมพิวเตอร์ถูกโปรแกรมในลักษณะเป็นชุดคำสั่งให้ปฏิบัติตามเป็นลำดับที่แน่นอน

อะไรคือวิชาของจรข่ายนิวรอตเทียม

ศาสตร์ว่าด้วยการคำนวณ โดยอาศัยวิจารข่ายที่ เลียนแบบการทำงานของระบบ
การทำงานของสมองของมนุษย์

วงจรข่ายนิวรอตเทียม กับ คอมพิวเตอร์ทั่วไป

- การโปรแกรมของคอมพิวเตอร์โดยทั่วไป ใช้ชุดคำสั่งเป็นลำดับขั้นตอน แต่วงจรข่ายนิวรอจะเรียนรู้โดยการฝึกฝนจาก ชุดข้อมูลสำหรับฝึกหัด (**Training set**)
- วงจรข่ายนิวรอจดจำได้โดยการปรับค่า **weight** ของ **connections** ที่ทำให้วงรมีข้อผิดพลาดจากการฝึกหัด (**training error**) ต่ำที่สุด
- การปรับ **weight** จะค่อยๆปรับทีละน้อยในการฝึกแต่ละครั้ง เมื่อฝึกบ่อยๆ ค่าความผิดพลาดก็จะลดลงเรื่อยๆ
- ปัจจุบันโปรแกรมวงจรข่ายนิวรอลมักจะใช้การจำลองบนคอมพิวเตอร์แทนส่วนที่เป็นวงจรเครือข่ายอัน слับซับซ้อน โดยใช้ซอฟต์แวร์เป็นหลัก ส่วนฮาร์ดแวร์ที่เลียนแบบวงจรข่ายนิวรอโดยตรงมีน้อยมาก เนื่องจากความยากลำบากในการสร้าง

วงจรข่ายนิวرونเทียมกับการใช้งาน



วงจรข่ายนิวرونเป็นเครื่องมือเอนกประสงค์ที่เหมาะสมจะใช้กับงาน:

1. งานการจดจำรูปแบบที่มีความไม่แน่นอน เช่น ลายมือ ลายเซ็นต์ ตัวอักษร รูปหน้า
2. งานการประมาณค่าฟังก์ชันหรือการประมาณความสัมพันธ์ (มี inputs และ outputs แต่ไม่ทราบว่า inputs กับ outputs มีความสัมพันธ์กันอย่างไร)
3. งานที่สิ่งแวดล้อมเปลี่ยนแปลงอยู่เสมอ (วงจรข่ายนิวรอสามารถปรับตัวเองได้)
4. งานจัดหมวดหมู่และแยกแยกสิ่งของ
5. งานทำนาย เช่น พยากรณ์อากาศ พยากรณ์หุ้น

ML

มหัศจรรย์

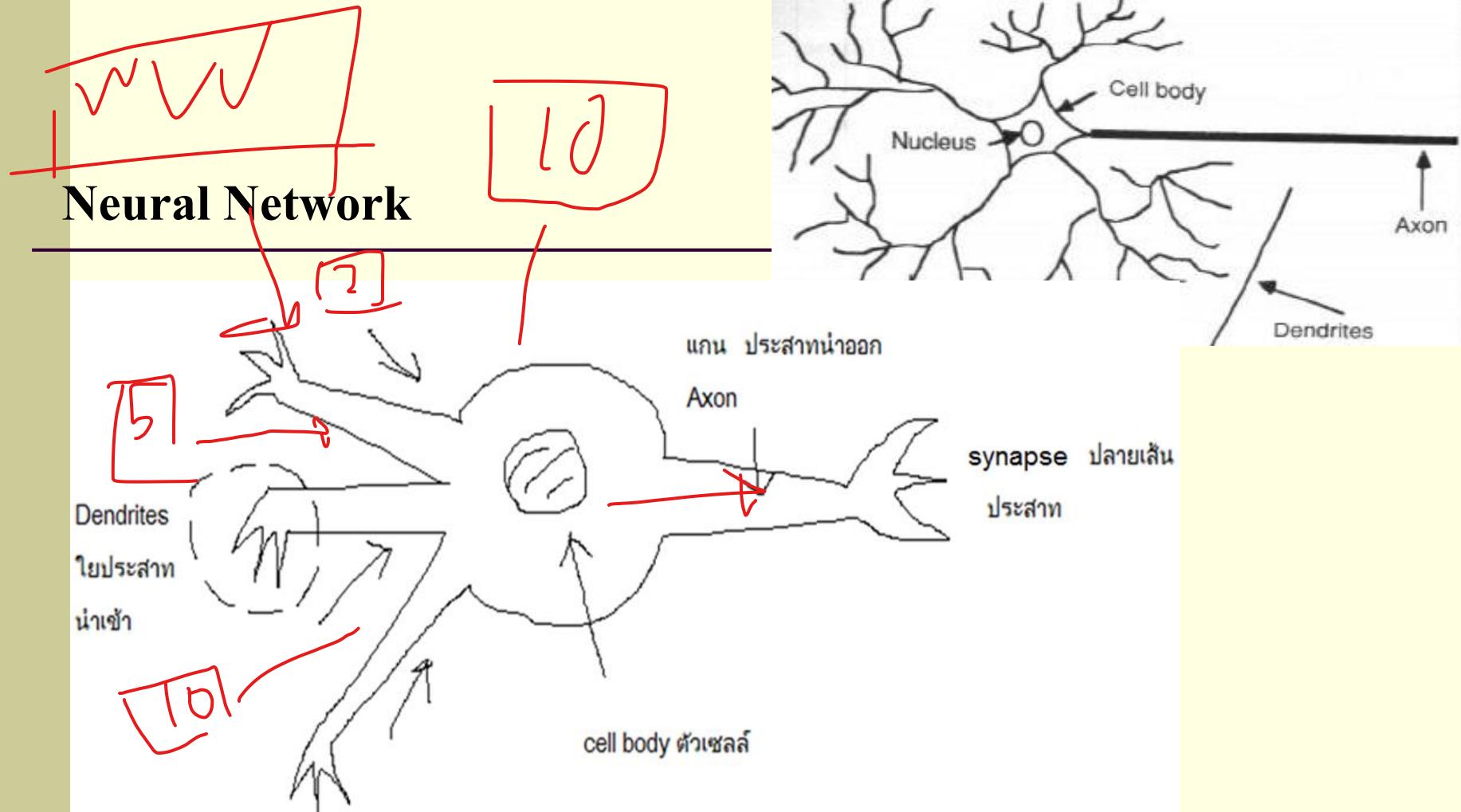
Artificial Neural Network (ANN)

เป็นกระบวนการทาง AI วิธีการหนึ่งที่สามารถนำไปใช้ในการตัดสินใจแทนมนุษย์ ซึ่งเป็นการ จำลองการเลียนแบบ กระบวนการคิดจากการประมวลผลของมนุษย์ โดยจำลองมาจากการกลุ่มเซลล์ประสาทของมนุษย์ ที่เชื่อมโยงกันเป็นระบบประสาทที่สามารถรับรู้หลาย ๆ สิ่งในเวลาเดียวกัน ด้วยการประมวลผลแบบขนาน (Parallel network) ทำให้ได้ระบบที่สามารถตัดสินใจได้ใกล้เคียงกับมนุษย์ บางครั้ง เรียกว่า โครงข่ายประสาทสมอง, โครงข่ายประสาบที่ym

ในสมองของคนจะประกอบด้วยเซลล์ประสาท (Neuron)

จำนวน 10^{11} ตัว ประมวลแสณล้านตัว

แนวคิดเริ่มต้นของเทคนิคนี้ได้มาจากการศึกษาข่ายงานไฟฟ้าชีวภาพ (bioelectric network) ในสมอง ซึ่งประกอบด้วย เซลล์ประสาท หรือ “นิรอน” (neurons) และ จุดประสานประสาท (synapses) แต่ละเซลล์ประสาทประกอบด้วยปลายในการรับกระแสประสาท เรียกว่า "เดนไ/drท์" (Dendrite) ซึ่งเป็น input และปลายในการส่งกระแสประสาทเรียกว่า "แอคชอน" (Axon) ซึ่งเป็นเหมือน output ของเซลล์ เซลล์เหล่านี้ทำงานด้วยปฏิกิริยาไฟฟ้าเคมี เมื่อมีการกระตุ้นด้วยสิ่งเร้าภายนอกหรือกระแสตุ้นด้วยเซลล์ด้วยกัน กระแสประสาทจะวิ่งผ่านเดนไ/drท์เข้าสู่นิวเคลียสซึ่งจะเป็นตัวตัดสินว่าต้องกระตุ้นเซลล์อื่น ๆ ต่อหรือไม่ ถ้ากระแสประสาทแรงพอ นิวเคลียสก็จะกระตุ้นเซลล์อื่น ๆ ต่อไปผ่านทางแอคชอนของมัน

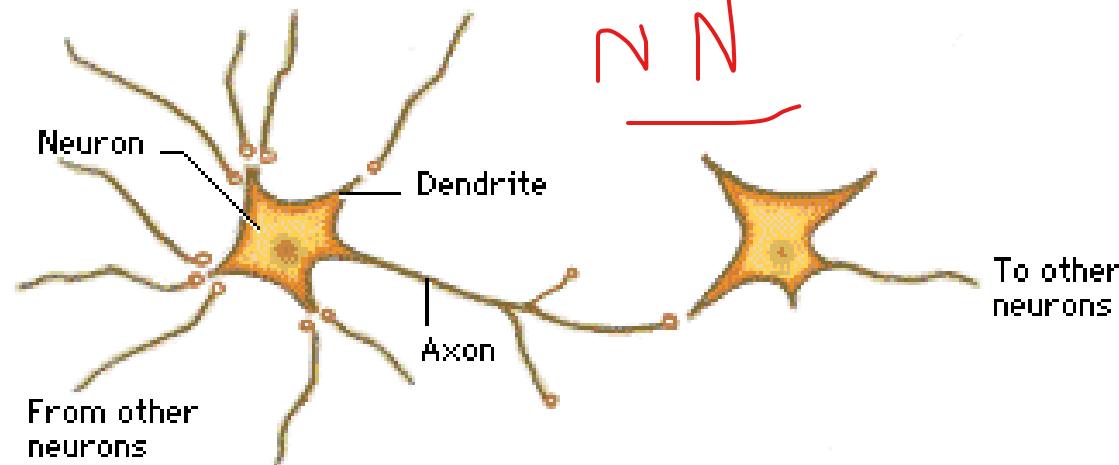


Dendrites ทำหน้าที่รับสัญญาณไฟฟ้าซึ่งสัญญาณ Cell ประสาท ใกล้เคียง Cell ประสาทด้วยหนึ่งจะเชื่อมต่อกับ Cell อีกๆ 10,000 ตัว เมื่อ สัญญาณ พ.พ ที่รับเข้ามาเกินค่าหนึ่ง Cell จะถูกกระตุ้นและส่งสัญญาณ ไปทางแกนประสาทน้ำออก

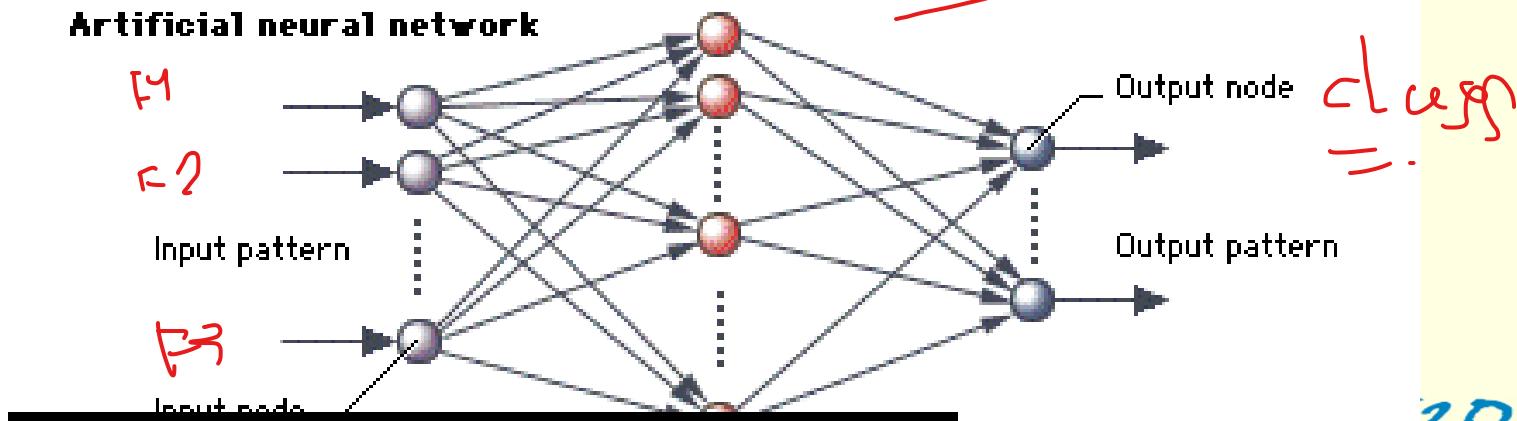
วงจรข่ายนิวรอนเทียม *Artificial Neural Networks*

เดียนแบบการทำงานของสมองมนุษย์ โดยใช้หน่วยประมวลผลง่ายๆ จำนวนมาก
ต่อกันเป็นโครงสร้างขึ้นมา

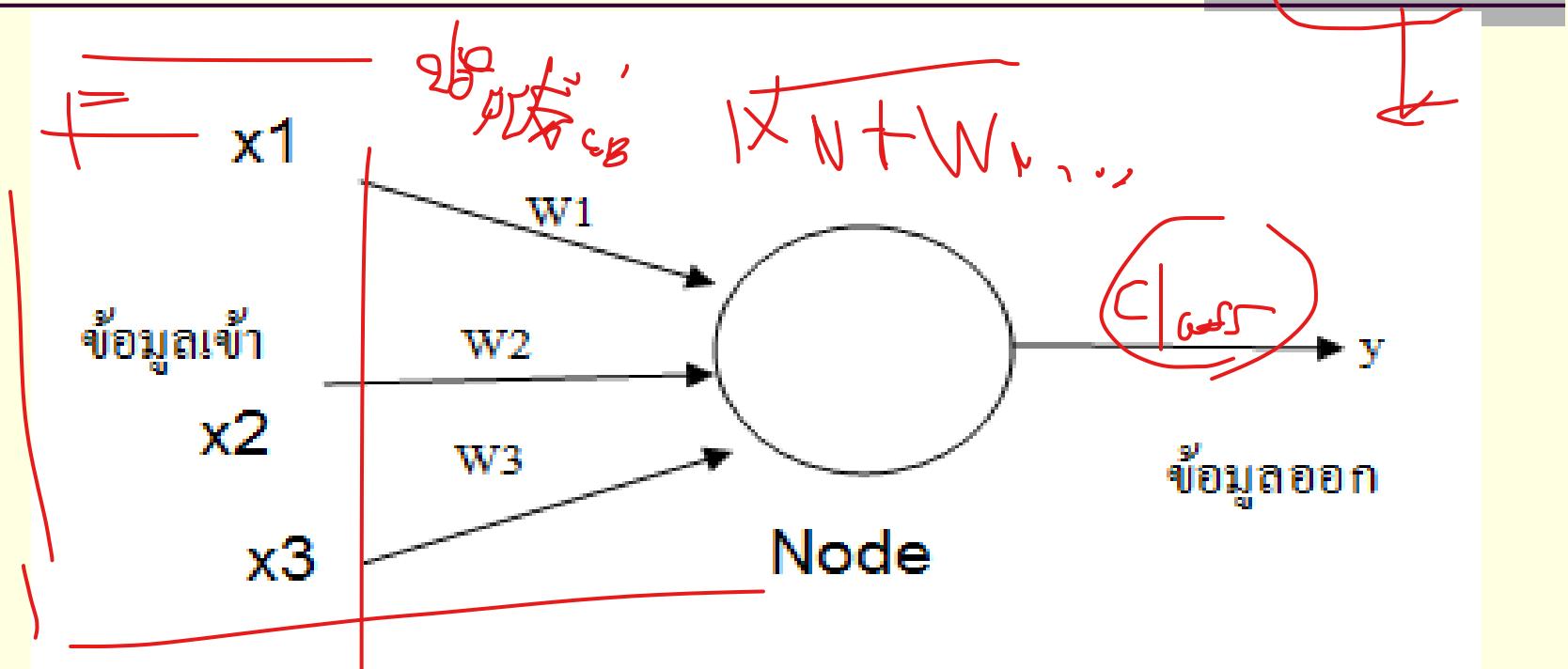
Neural connections in animals



Artificial neural network



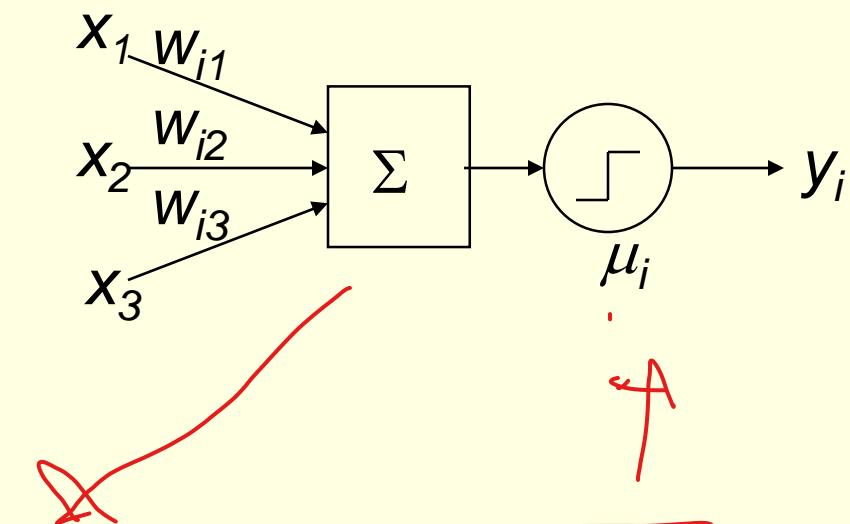
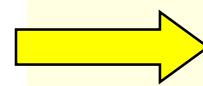
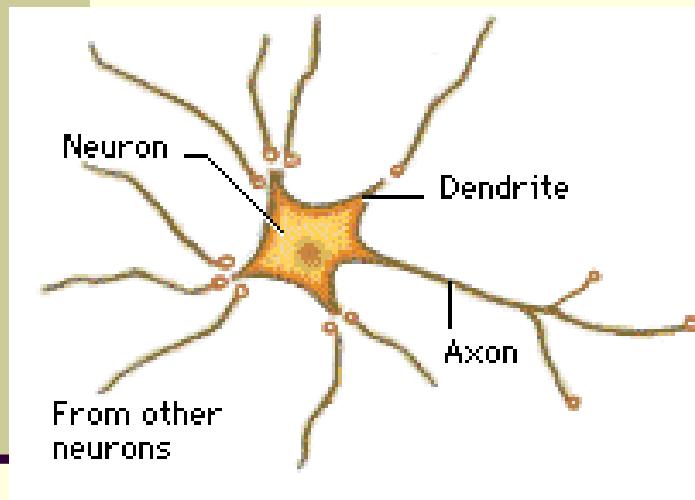
Artificial Neural Network จาก NN นำมายกต์เป็น ANN



การเลียนแบบกระบวนการคิดของธรรมชาติและนำมาใช้ เครื่องจักรมีจุดประสงค์เพื่อลดข้อจำกัดในการประมวลผลจาก วิธีการอื่นๆ และเพิ่มประสิทธิภาพการเรียนรู้ของคอมพิวเตอร์

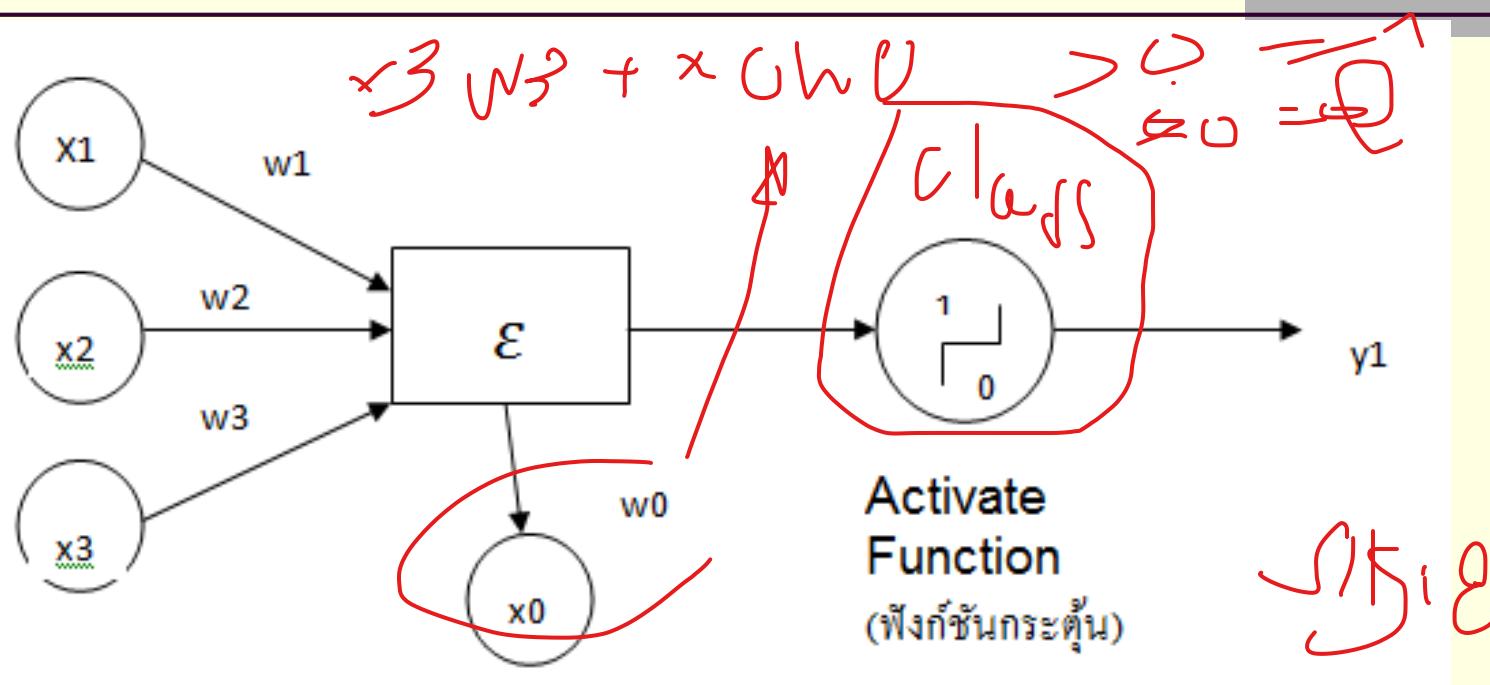
Supervised Learning: Perceptron Networks

Rosenblatt และเพื่อนร่วมงานเป็นผู้เริ่มแนวคิดของ Perceptron เมื่อปี 1962 โดยได้รับแรงบันดาลใจจากโมเดลการทำงานของเซลล์ประสาทของ McCulloch-Pitts



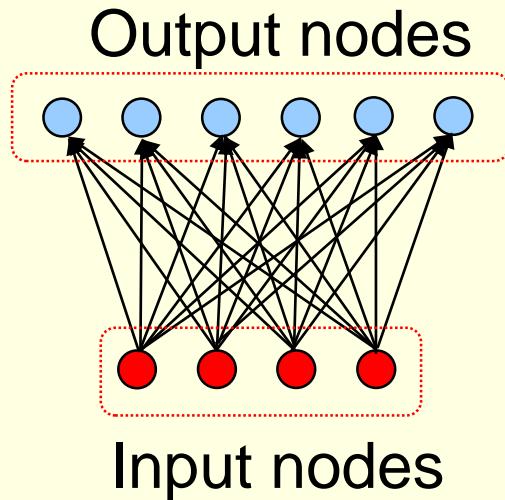
$$x_1 w_1 + x_2 w_2 + x_3 w_3 = \boxed{5}$$

Perceptron คือโครงข่ายประสาทเทียมแบบง่ายมีที่จำลองลักษณะของเซลล์ประสาทดังรูป

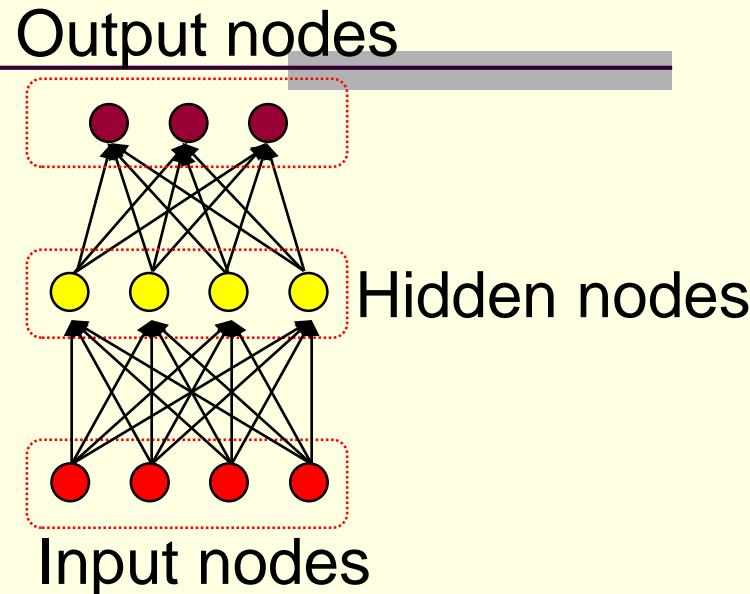


- I/P Vector เป็นจำนวนจริง แล้วคำนวณหาค่าผลรวมเชิงเส้น(linear combination) แบบถ่วงน้ำหนักของ I/P ($X_1, X_2, X_3, \dots, X_n$) โดยที่ $w_1, w_2, w_3, \dots, w_n$ เป็นค่าถ่วงน้ำหนัก
- Activation Function คือฟังก์ชันกระตุ้น ซึ่งจะเป็นการดูในส่วน O/P เช่นถ้าต้องให้ O/P เป็นใช่หรือไม่ใช่เราอาจจะใช้ Threshold

Perceptron Networks



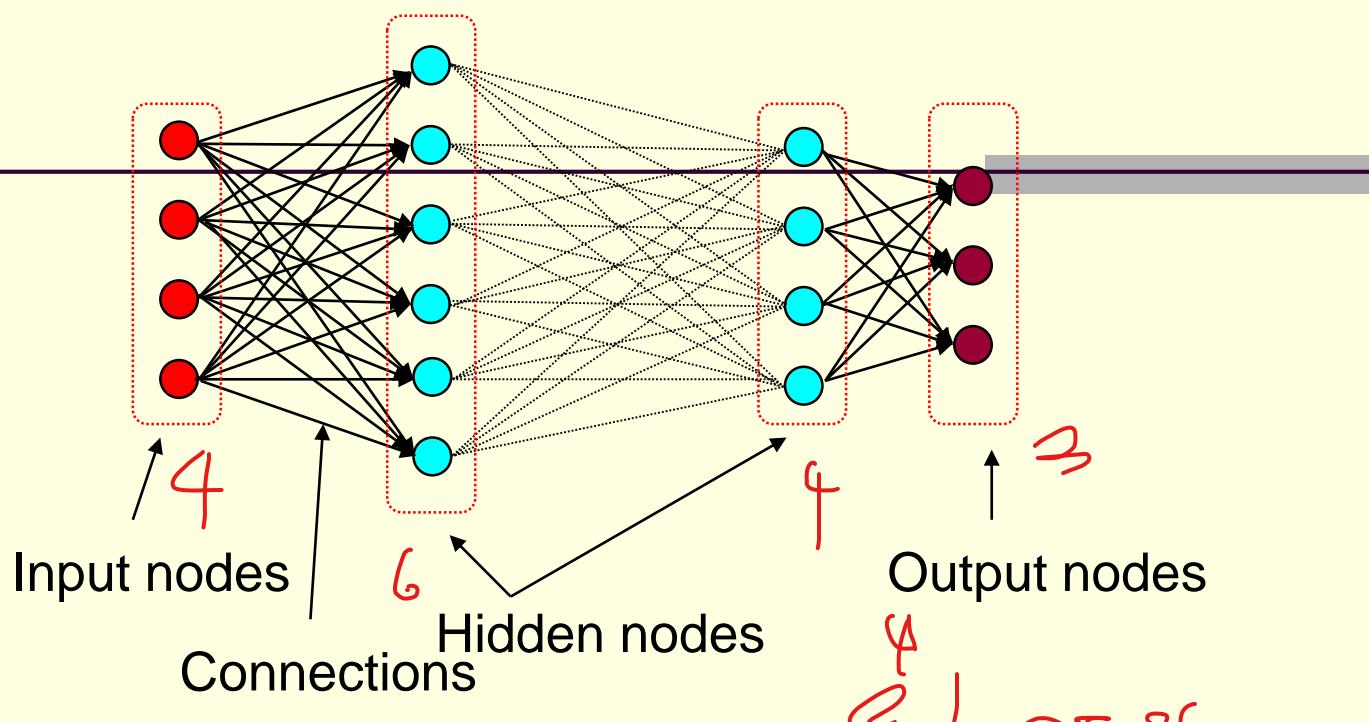
Single layer perceptron
network



Multilayer perceptron
network
(this case: 2 layers)

EF

ตัวอย่างโครงสร้างของเครือข่ายนิวรอติก



Output ของแต่ละโนนด

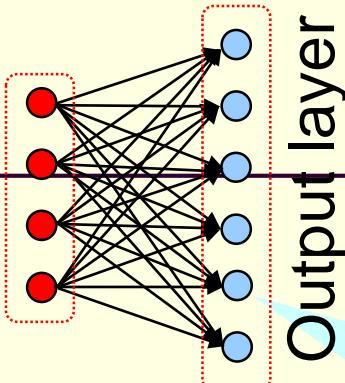
$$y_i = f(w_i^1 x_1 + w_i^2 x_2 + w_i^3 x_3 + \dots + w_i^m x_m)$$

$$= f(\sum_j w_i^j x_j)$$

X_i = input จากโนนดอื่นๆ W_i^j = น้ำหนัก (weight) ของแต่ละแขน (connection)

A Single Layer Perceptron Network

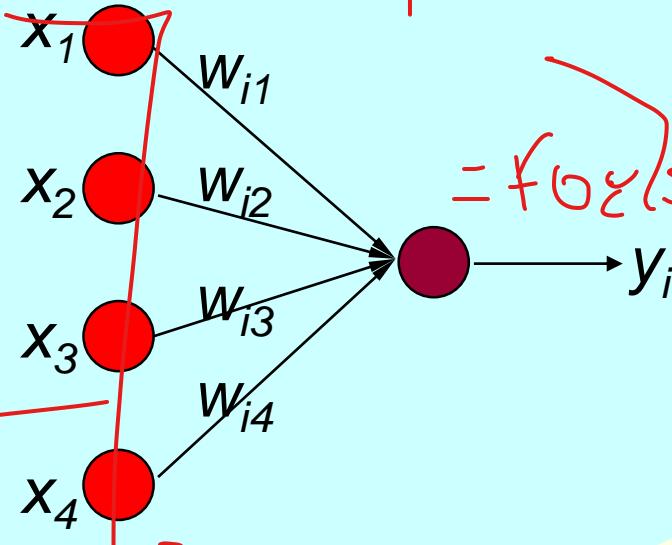
Input layer



For each node, the output is given by

T_{loss}

$$y_i = g\left(\sum_{j=1}^N w_{ij}x_j - \mu_i\right)$$



$= f_{loss}$

$- q$

w_{ij} = Connection weight of branch (i,j)

x_j = Input data from node j in the input layer

μ_i = Threshold value of node i in the output layer

g = **Activation function**

Single Layer Perceptron Networks (cont.)

1. จำนวน input nodes ขึ้นอยู่กับจำนวน components ของ input data

2. Activation function ขึ้นอยู่กับลักษณะ ข้อมูล ของ Output เช่น
ถ้า output ที่ต้องการเป็น “ใช่” หรือ “ไม่ใช่” เราจะต้องใช้ *Threshold function*

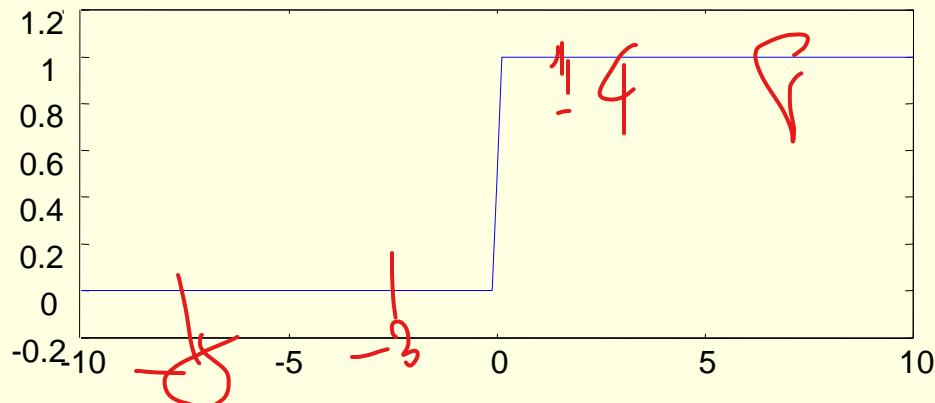
$$f(x) = \begin{cases} 1 & \text{if } x \geq T \\ 0 & \text{if } x < T \end{cases} \quad T = \text{Threshold level}$$

หรือถ้า output เป็นค่าตัวเลขที่ต้องเนื่อง เราต้องใช้ continuous function เช่น *Sigmoid function*

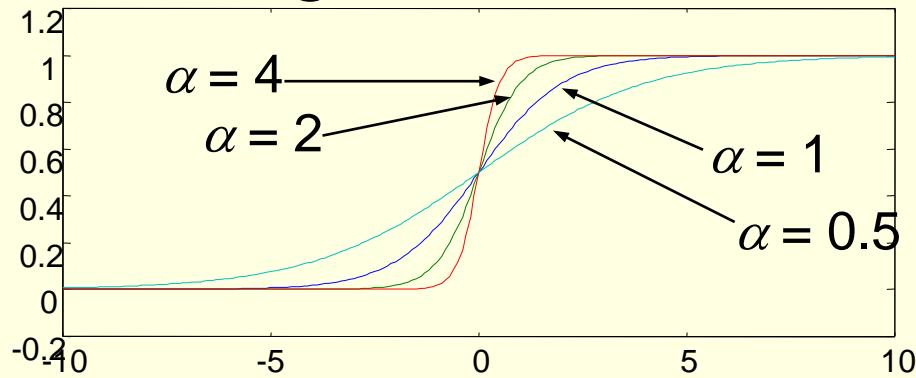
$$f(x) = \frac{1}{1 + e^{-\alpha x}}$$

Activation function

Threshold function ($T=0$)



Sigmoid function



การณ์ Threshold เป็น Binary Function



$$Y_i = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 + \dots + w_nx_n > \theta \\ 0 & \text{if } w_1x_1 + w_2x_2 + \dots + w_nx_n \leq \theta \end{cases}$$

Class

ที่รีθ

$$Y_i = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 + \dots + w_nx_n - \theta > 0 \\ \square & \\ 0 & \text{if } w_1x_1 + w_2x_2 + \dots + w_nx_n - \theta \leq 0 \end{cases}$$

- θ สามารถแทนเป็น w_0 จะได้ $-\theta = w_0$

$$Y_i = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 + \dots + w_nx_n + w_0x_0 > 0 \\ 0 & \text{if } w_1x_1 + w_2x_2 + \dots + w_nx_n + w_0x_0 \leq 0 \end{cases}$$

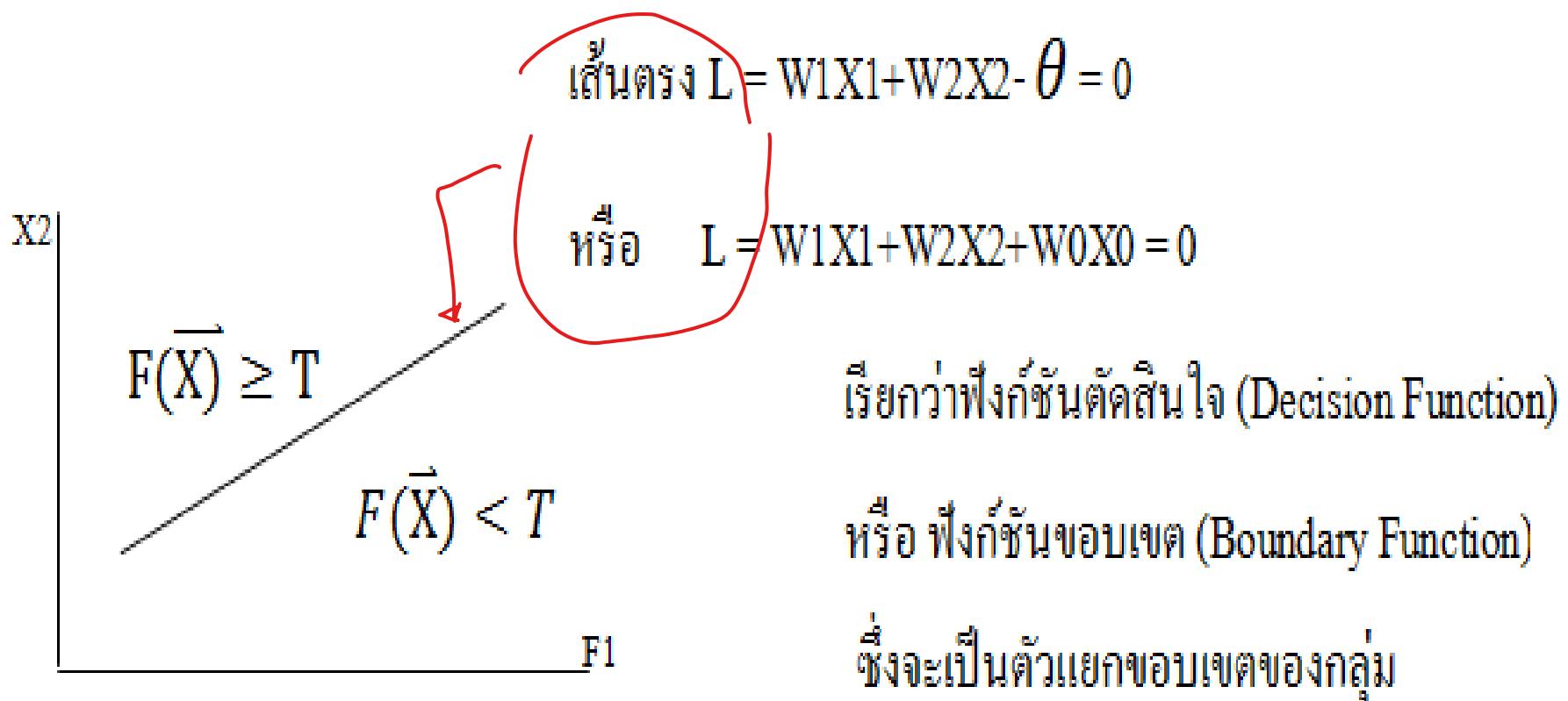
$$Y_i = \begin{cases} 1 & \text{if } F(\vec{X}) \geq T \\ 0 & \text{if } F(\vec{X}) < T \end{cases}$$

$$F(\vec{X}) = \sum_{i=0}^n w_i x_i - \theta$$

X_i คือ I/P vector

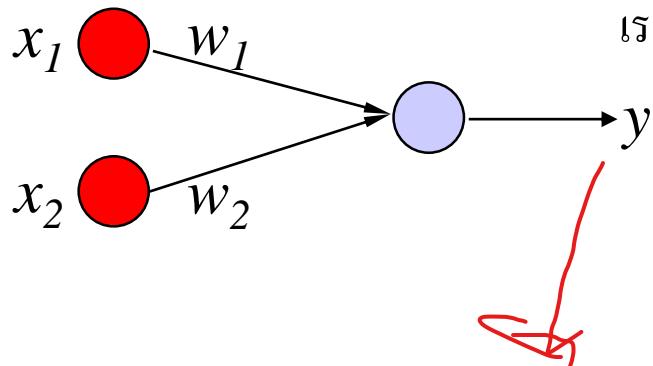
w_i คือ ค่าถ่วงน้ำหนัก

สมมุติที่ 2 Feature จะได้

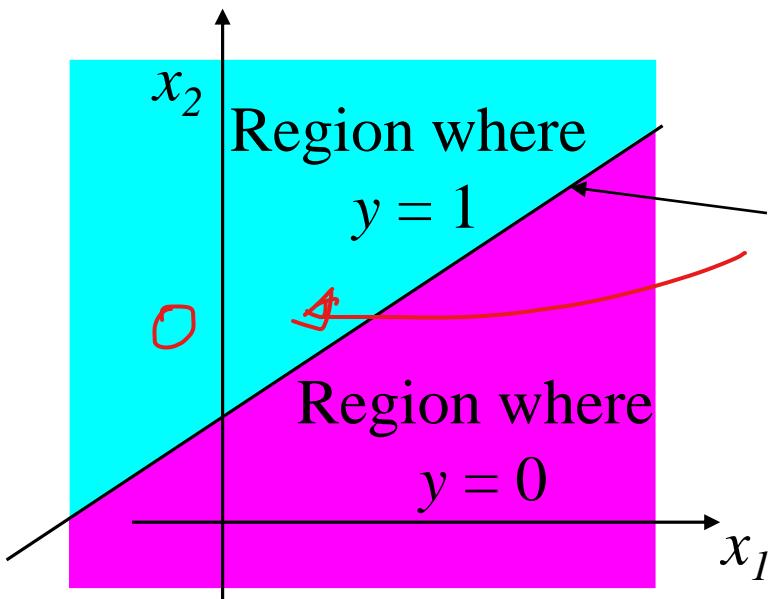


How a single layer perceptron works

สมมุติว่าเรามีวงจรข่าย perceptron ที่มี 2 input nodes และมี activation function เป็น threshold function เราจะได้ Binary output



$$y = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 - \theta \geq 0 \\ 0 & \text{if } w_1x_1 + w_2x_2 - \theta < 0 \end{cases}$$



เส้นตรง $L = w_1x_1 + w_2x_2 - \theta = 0$

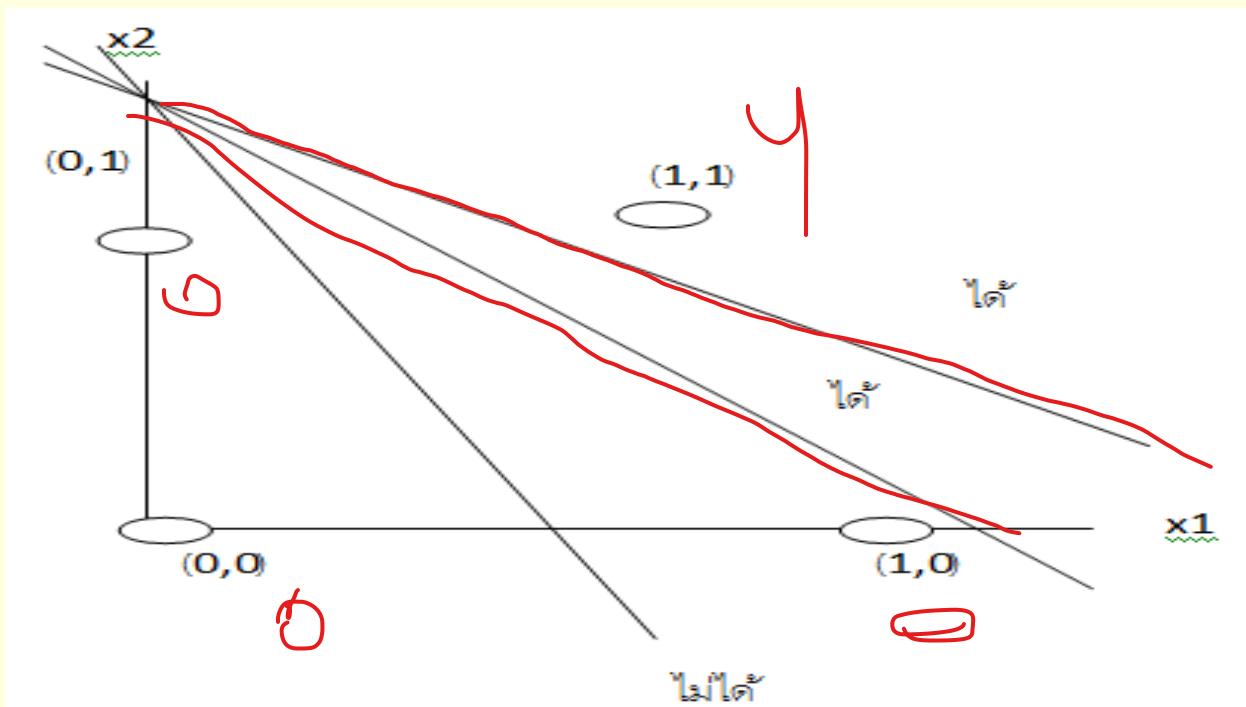
- ถ้า (x_1, x_2) อยู่เหนือเส้นตรง L จะได้ $y = 1$
- ถ้า (x_1, x_2) อยู่ใต้เส้นตรง L จะได้ $y = 0$

ดังนั้นเราเรียกเส้นตรง L นี้ว่า พิริยัณฑ์ตัดสิน

Decision function หรือพิริยัณฑ์ของเขต
Boundary Function

ตำแหน่งความชันจะขึ้นอยู่กับ Parameter W1,W2 และ W0 (ค่าถ่วงน้ำหนักนั้นเอง) โดยวิธีการของ AI จะทำการปรับค่า Weight parameter ให้สามารถแบ่งหรือจำแนกข้อมูล โดยพยายามมีค่า Error ที่น้อยที่สุด เช่น Function AND

| X1 | X2 | Y |
|----|----|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



จากตัวอย่าง จะเป็น Perceptron ทั้งเป็นเส้นตรง แต่ถ้าในกรณีที่ I/P>2 Perceptron จะเป็นรูปแบบตัดสินใจหลายมิติ (hyperplane decision surface)

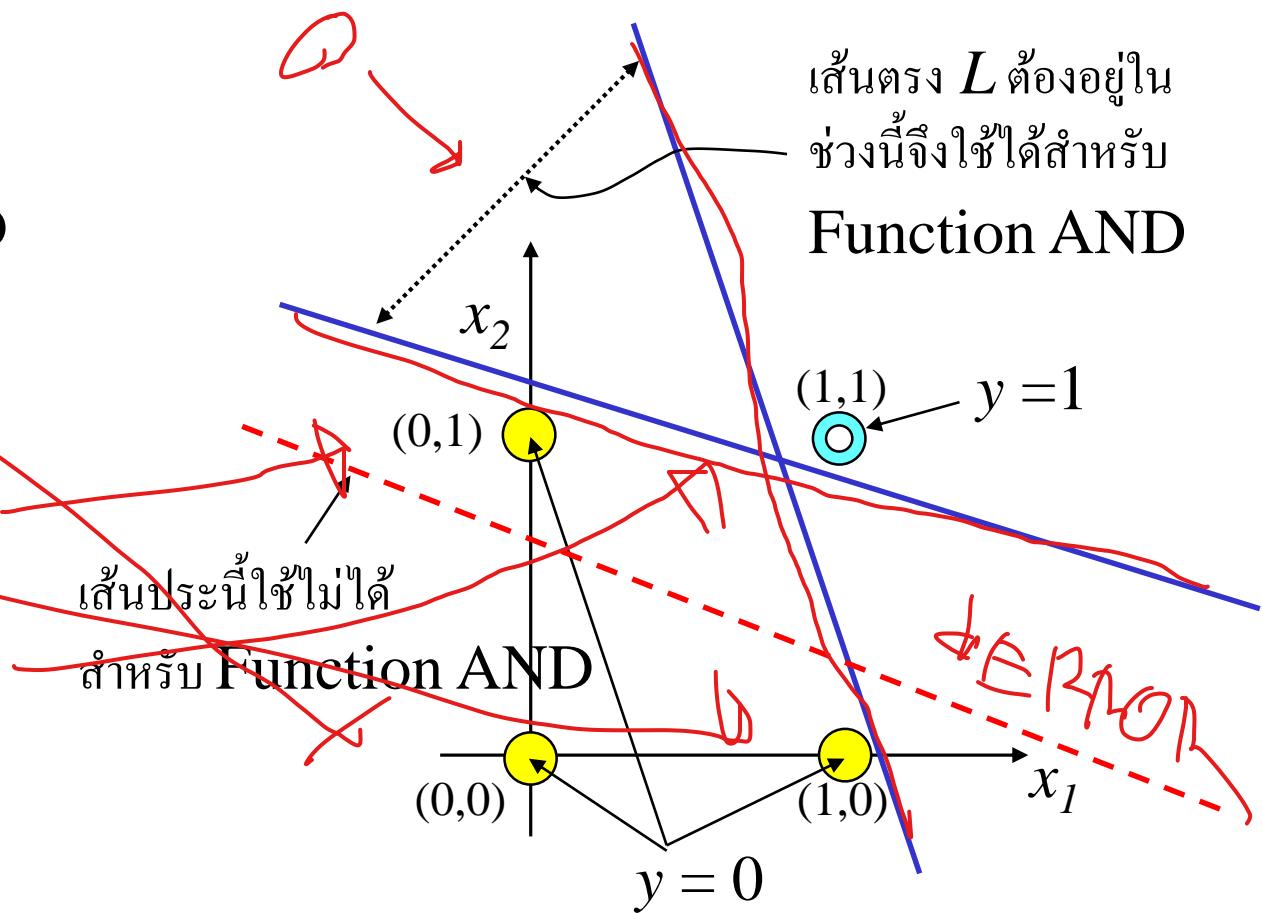
How a single layer perceptron works (cont.)

ความชันและตำแหน่งของเส้นตรง ขึ้นอยู่กับพารามิเตอร์ เราจะต้องปรับพารามิเตอร์เหล่านี้ให้ได้เส้นตรง L ที่ให้ผลลัพธ์ถูกต้อง

ตัวอย่าง

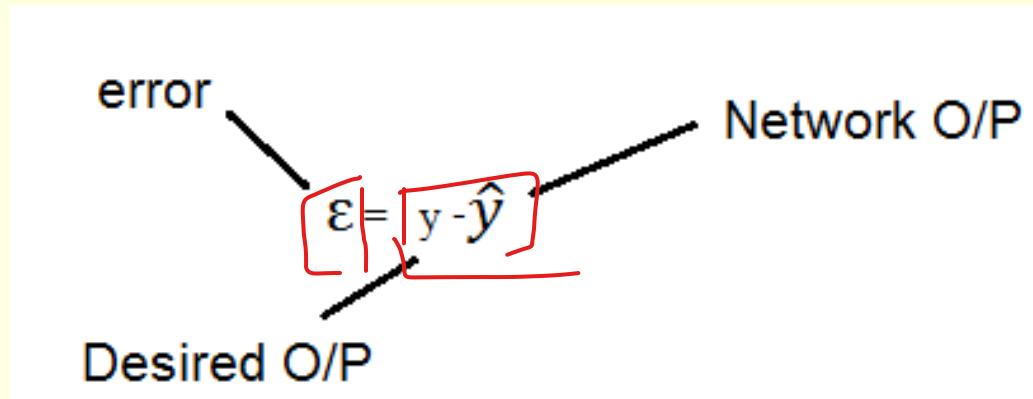
Function AND

| x_1 | x_2 | y |
|-------|-------|-----|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



หลักการปรับตัวของวงจรข่าย เป็นการปรับค่า Parameter ต่างๆ

ให้ไปในทิศทางที่จะลดค่าความผิดพลาดลงได้



ขั้นตอนการฝึกหัดเครือข่าย (Training the network)

- ป้อน I/P เข้า network input (x_1, x_2, \dots, x_n) และป้อน Desired o/p

(ข้อมูล o/p ที่รู้ว่ามันอยู่ class อะไร)

- คำนวณค่า network o/p

$$\hat{y} = F(w_1x_1 + w_2x_2 - \theta)$$

หรือ

$$\hat{y} = F(w_1x_1 + w_2x_2 + w_0x_0)$$

$$y = 1 C_2$$

$$y = Red$$

- คำนวณ หา error

$$\epsilon = |y - \hat{y}|$$

- ปรับ weight parameter ทุกค่า

$$w^{\text{new}} = w^{\text{old}} + \Delta w$$

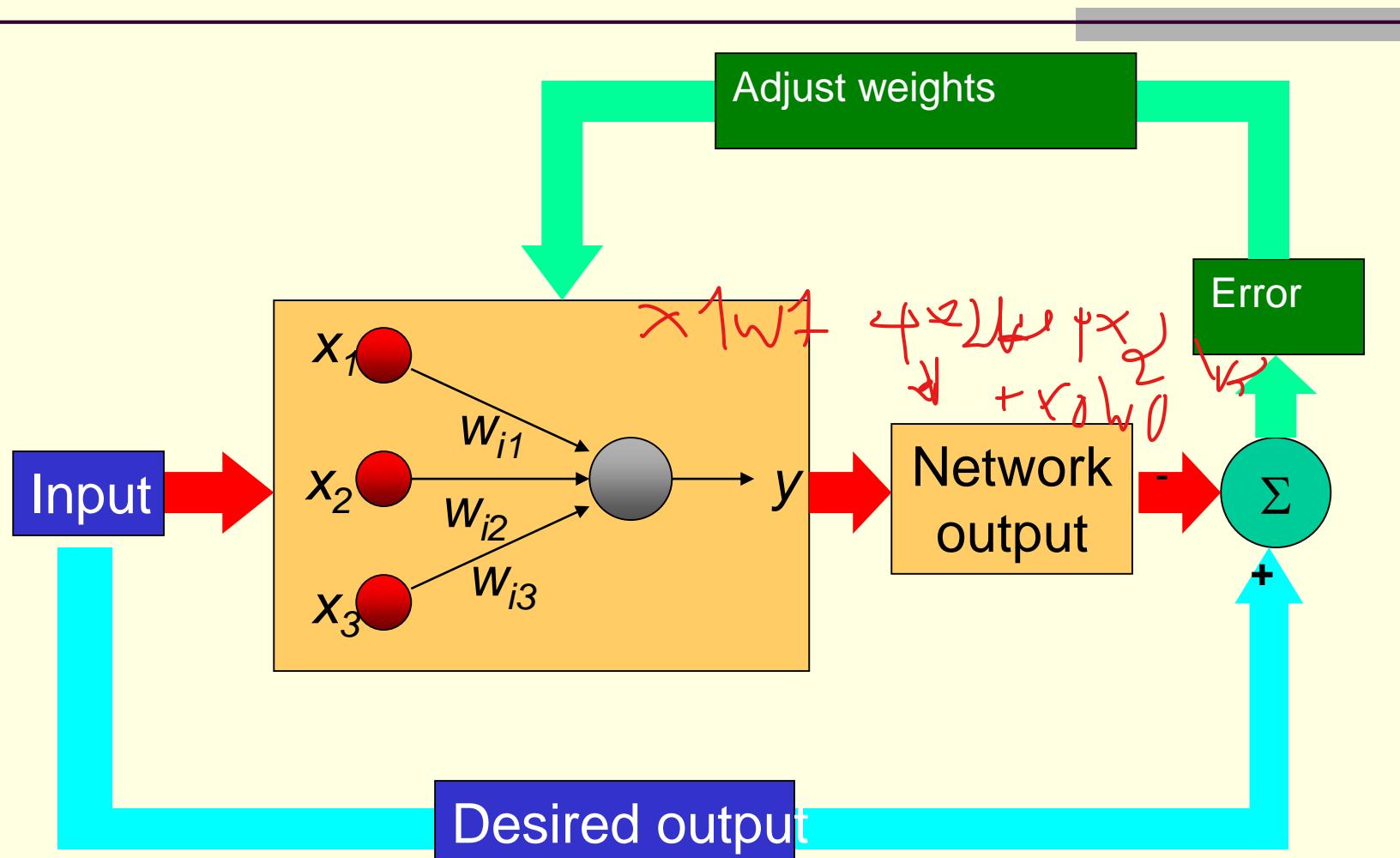
Δw อัตราการเปลี่ยนแปลงเฉลี่ย

$$\theta^{\text{new}} = \theta^{\text{old}} + \Delta \theta$$

⇒ 1 ← 1

- กลับไปทำข้อที่ 1 ใหม่ จนกว่า error จะค่อนขอนยอมรับได้ $|\epsilon| < t$

តម្រូវការ learning Algorithm : traning perceptron networks



ตัวอย่าง ลู่ทางการปรับ Weight

$$\Delta w_i = \alpha(y - \hat{y}) x_i$$

$$\Delta \theta = -\alpha(y - \hat{y})$$

α คือ Learning rate อัตราการเรียน

ตัวอย่าง จากข้อมูลดังต่อไปนี้ จงทำการจำแนกกลุ่มข้อมูลโดยใช้ ANN

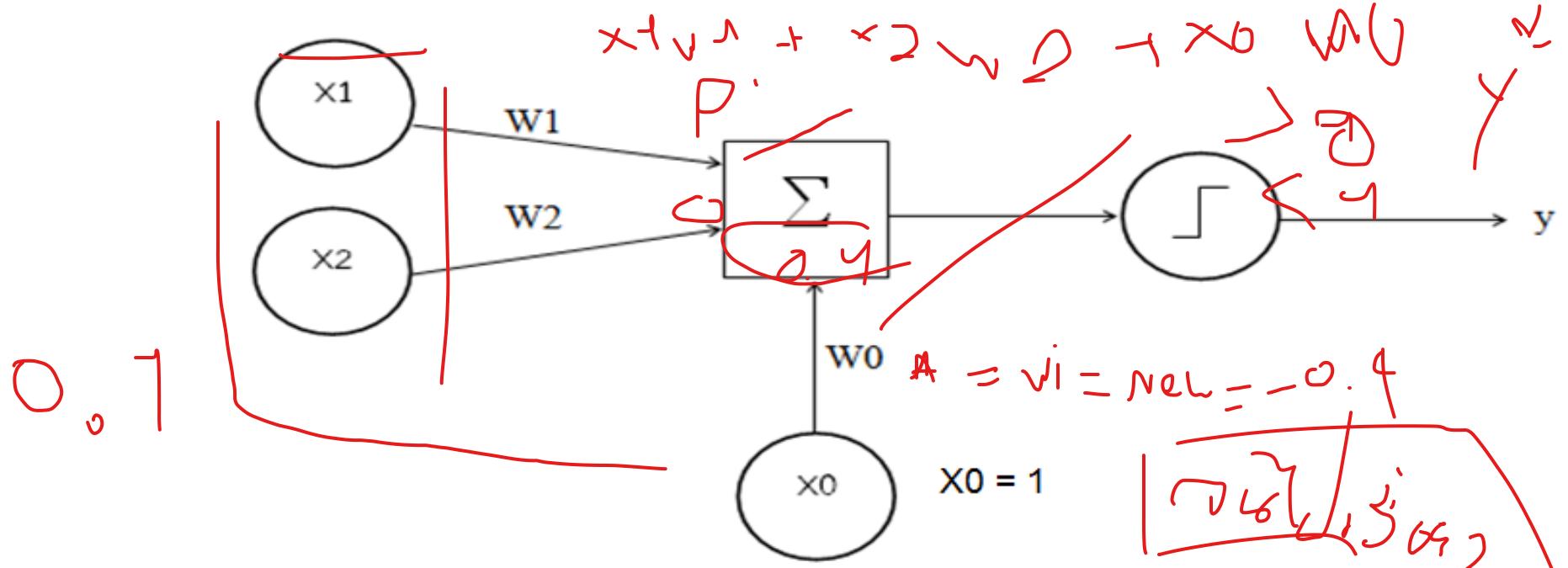
| X1 | X2 | Y | |
|----|----|---|---------|
| 0 | 0 | 0 | Class 0 |
| 0 | 1 | 0 | |
| 1 | 0 | 0 | |
| 1 | 1 | 1 | Class 1 |

$$0.1 - (0.1 + 0.1)$$

$$+ 0.1 (-0.4) \\ = -0.3$$

$$0.1 > = 1$$

ให้ $w_0 = 0.1, w_1 = 0.1, w_2 = 0.1 \alpha = 0.5$ จงทำการจำแนกโดยใช้ ANN



วิธีทำ จากสมการดังต่อไปนี้

$$\Delta w^1 = x \hat{y} - \hat{x}_0$$

$$\hat{y} = \begin{cases} 1 & \text{if } W_1X_1 + W_2X_2 + W_0X_0 \geq 0 \\ 0 & \text{if } W_1X_1 + W_2X_2 + W_0X_0 < 0 \end{cases}$$

$$\varepsilon = (y - \hat{y}) \rightarrow 0.6 \times 2 - 1.4 > 0$$

$$\Delta w_i = \alpha (y - \hat{y}) x_i$$

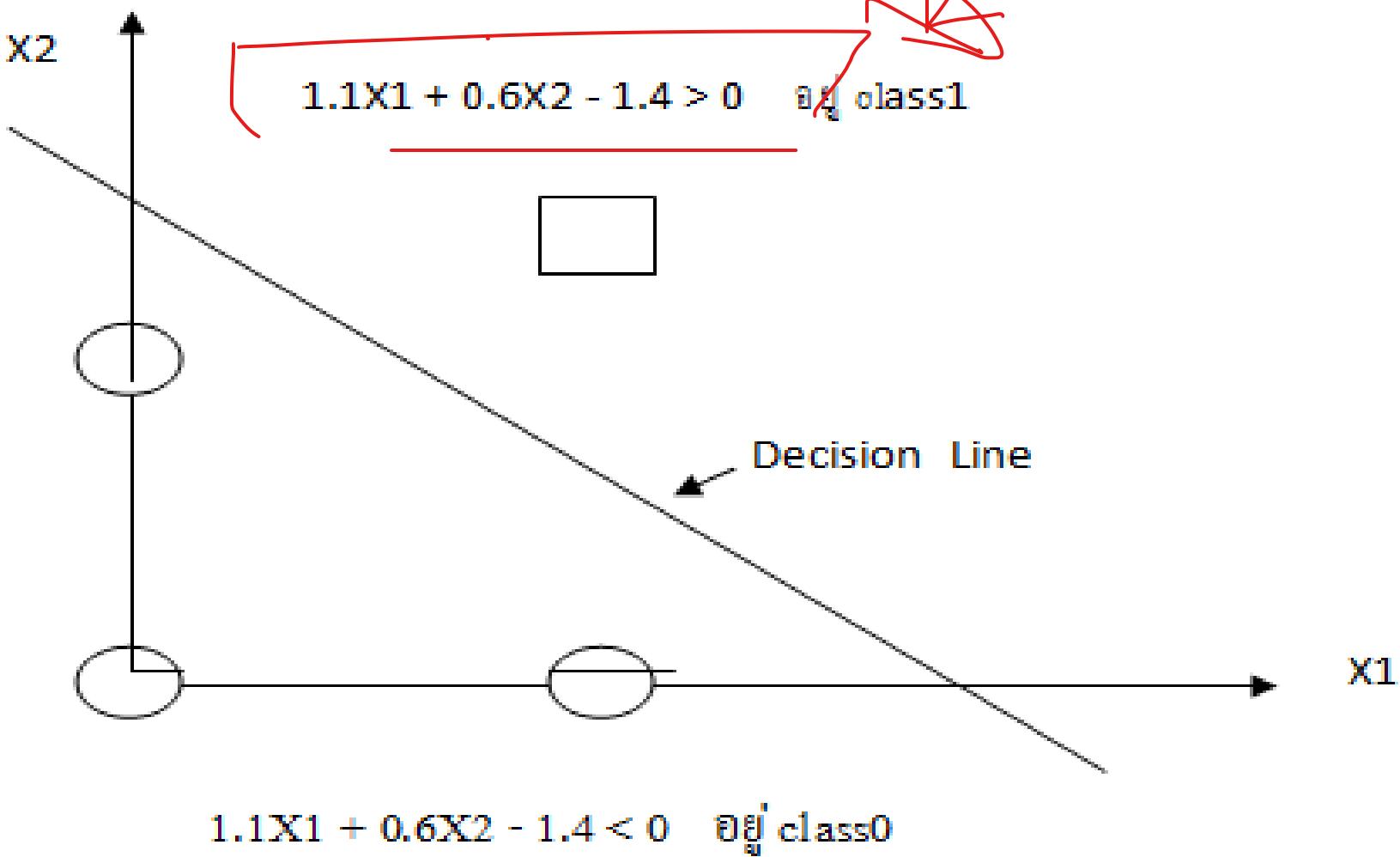
$$w_i^{\text{new}} = w_i^{\text{old}} + \Delta w_i$$

$$\Delta w_0 = \alpha (y - \hat{y}) x_0$$

$$w_0^{\text{new}} = w_0^{\text{old}} + \Delta w_0$$

| | X1 | X2 | y | Σ | \hat{y} | $y - \hat{y}$ | Δw_1 | w_1^{new} | Δw_2 | w_2^{new} | Δw_0 | w_0^{new} |
|-----|----|----|---|----------|-----------|---------------|--------------|-------------|--------------|-------------|--------------|-------------|
| 1 | 0 | 0 | 0 | 0.1 | 1 | -1 | 0 | 0.1 | 0 | 0.1 | -0.5 | -0.4 |
| | 0 | 1 | 0 | -0.3 | 0 | 0 | 0 | 0.1 | 0 | 0.1 | 0 | -0.4 |
| | 1 | 0 | 0 | -0.3 | 0 | 0 | 0 | 0.1 | 0 | 0.1 | 0 | -0.4 |
| | 1 | 1 | 1 | -0.2 | 0 | 1 | 0.5 | 0.6 | 0.5 | 0.6 | 0.5 | 0.1 |
| 2 | 0 | 0 | 0 | 0.1 | 1 | -1 | 0 | 0.6 | 0 | 0.6 | -0.5 | -0.4 |
| | 0 | 1 | 0 | 0.2 | 1 | -1 | 0 | 0.6 | -0.5 | 0.1 | -0.5 | -0.9 |
| | 1 | 0 | 0 | -0.3 | 0 | 0 | 0 | 0.6 | 0 | 0.1 | 0 | -0.9 |
| | 1 | 1 | 1 | -0.2 | 0 | 1 | 0.5 | 1.1 | 0.5 | 0.6 | 0.5 | -0.4 |
| 3-5 | ∫ | ∫ | ∫ | ∫ | ∫ | ∫ | ∫ | ∫ | ∫ | ∫ | ∫ | ∫ |
| 6 | 0 | 0 | 0 | -1.4 | 0 | 0 | 0 | 1.1 | 0 | 0.6 | 0 | -1.4 |
| | 0 | 1 | 0 | -0.8 | 0 | 0 | 0 | 1.1 | 0 | 0.6 | 0 | -1.4 |
| | 1 | 0 | 0 | -0.3 | 0 | 0 | 0 | 1.1 | 0 | 0.6 | 0 | -1.4 |
| | 1 | 1 | 1 | 0.3 | 1 | 0 | 0 | 1.1 | 0 | 0.6 | 0 | -1.4 |

จะได้สมการ สำหรับ Artificial Neural Network คือ



Classification Data

การใช้ a single layer perceptron network ในการแยกแยะ (classify) ข้อมูล

- เราเรียก input ที่มีหลายมิติว่า **input pattern** หรือ input vector
- Output จะต้องมีลักษณะเป็น **class**

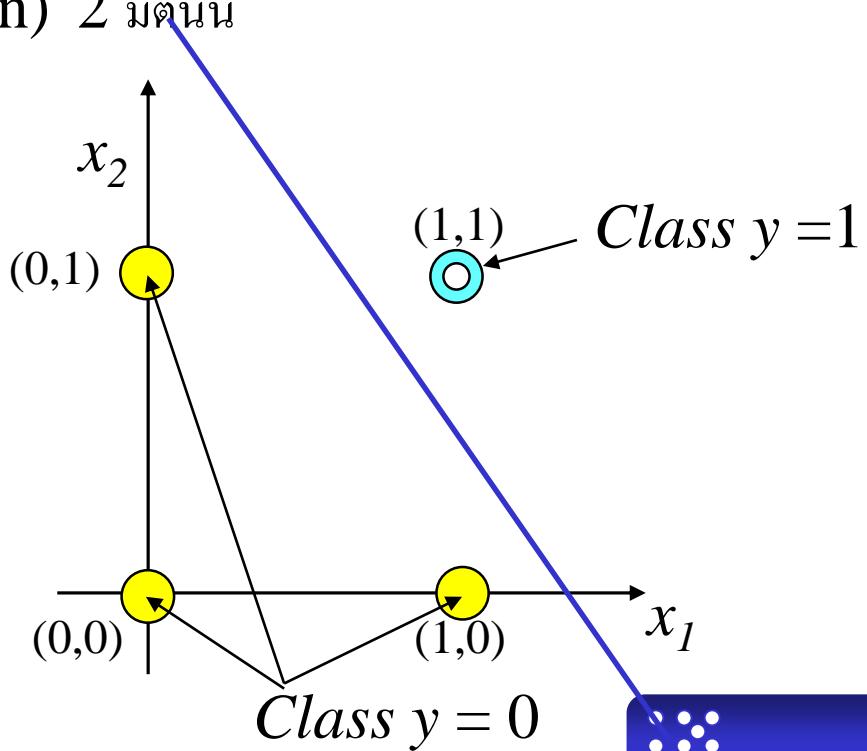
คุณสมบัติของข้อมูลที่สามารถจะใช้ a single layer perceptron network แยกแยะได้

- ใน **feature space** (input domain) 2 มิตินี้

class แต่ละ class จะต้องสามารถแยกจาก

class อื่นได้โดยใช้เส้นตรงสั้นเดียว

เป็นตัวแบ่ง



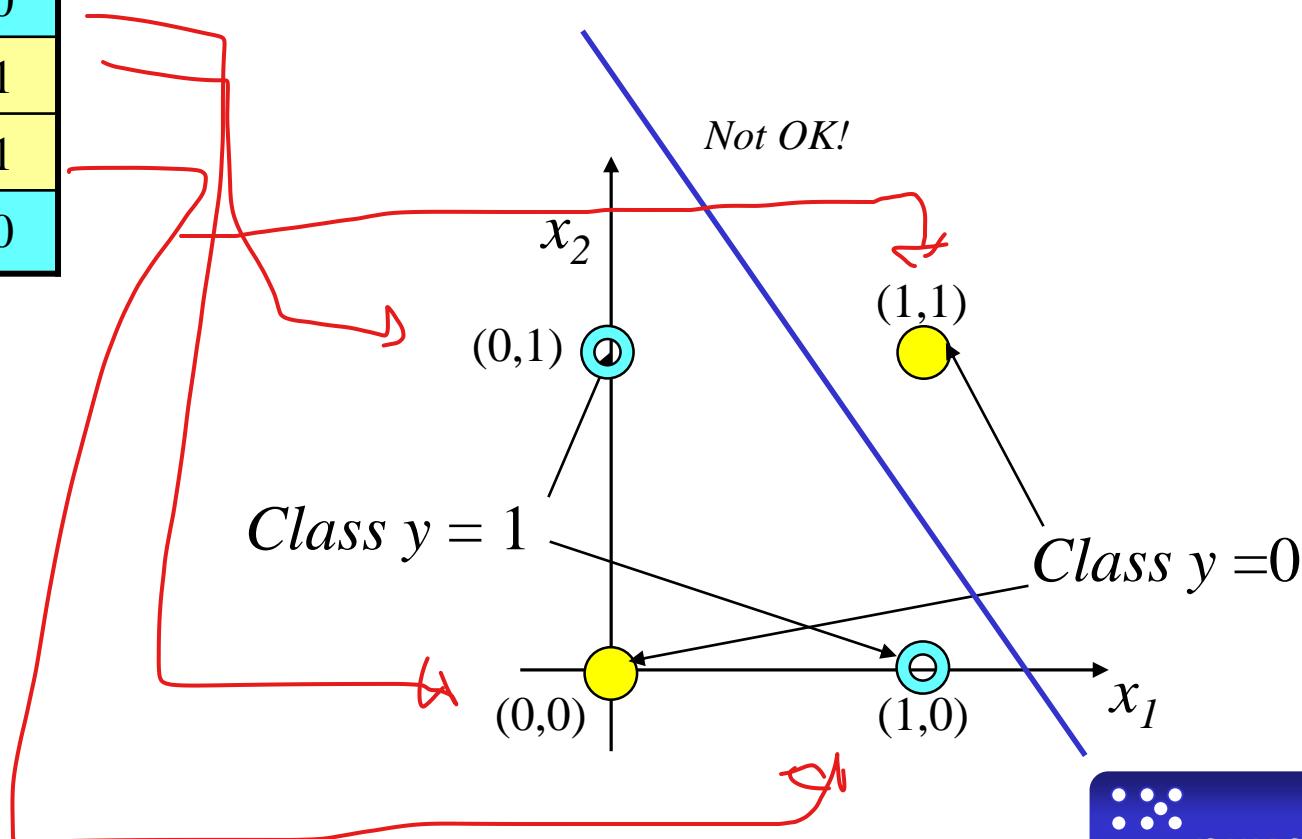
Classification Data (cont.)

ตัวอย่างที่ใช้ a single layer perceptron ไม่ได้

Function XOR

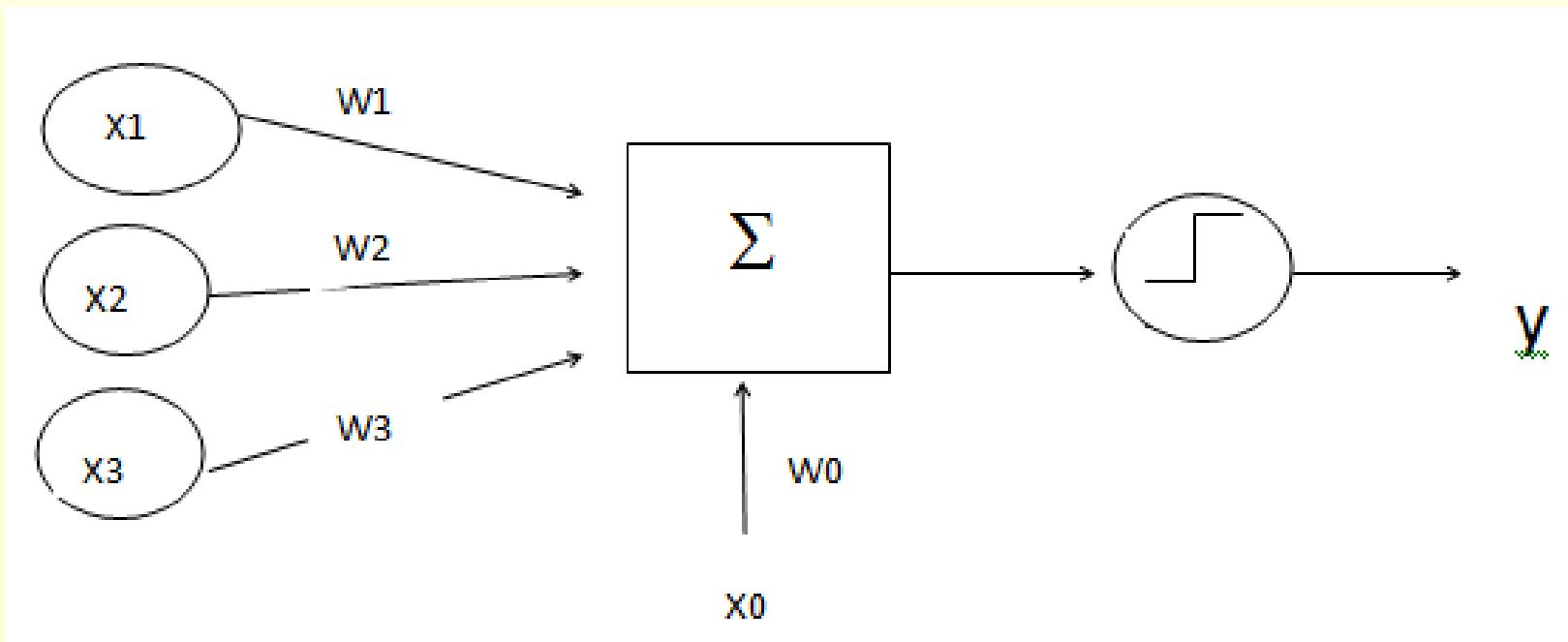
| x_1 | x_2 | y |
|-------|-------|-----|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Function XOR เราไม่สามารถใช้ เส้นตรงเดียวในการแบ่งแยก Class $y=0$ กับ Class $y=1$ ได้

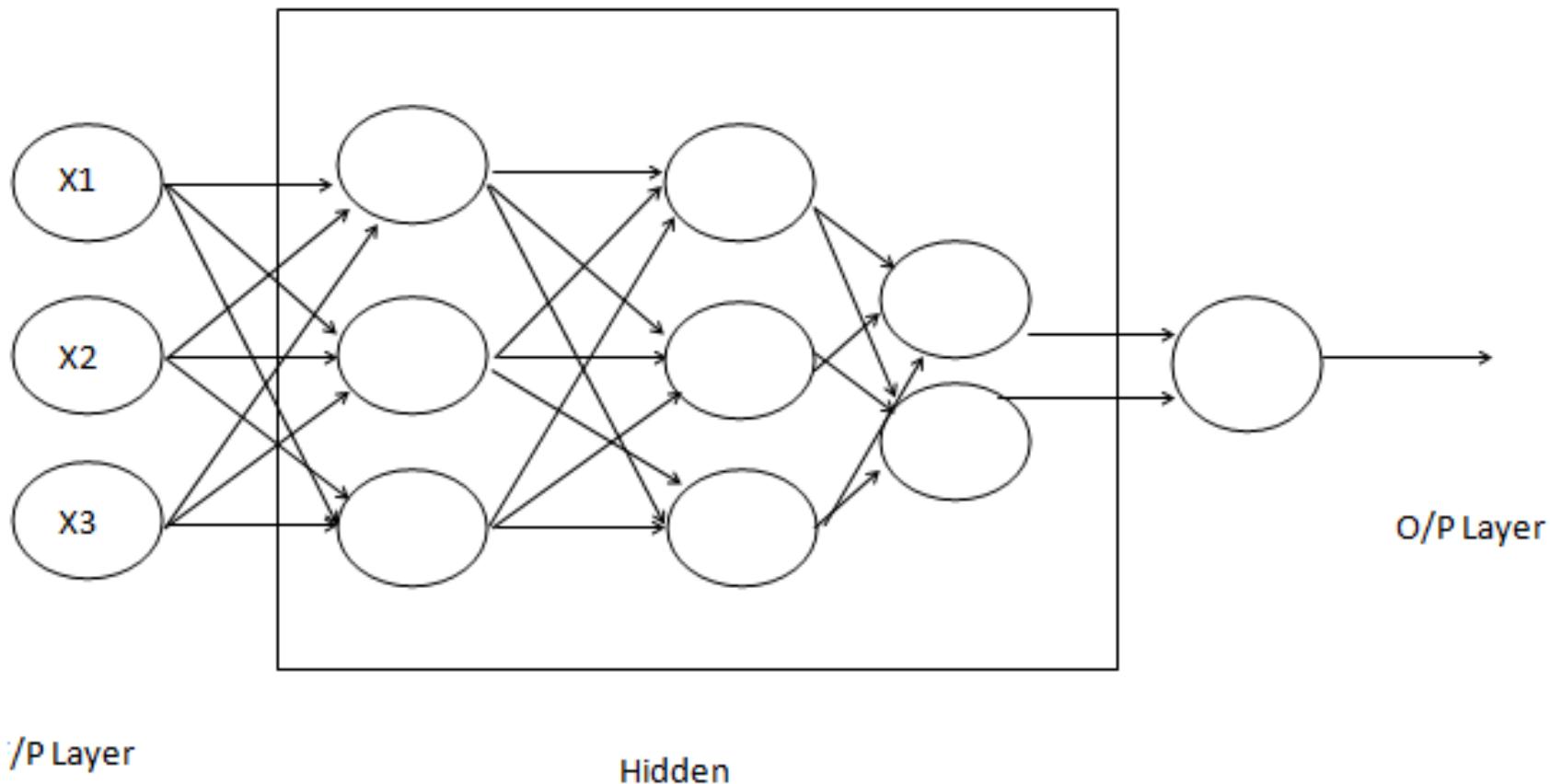


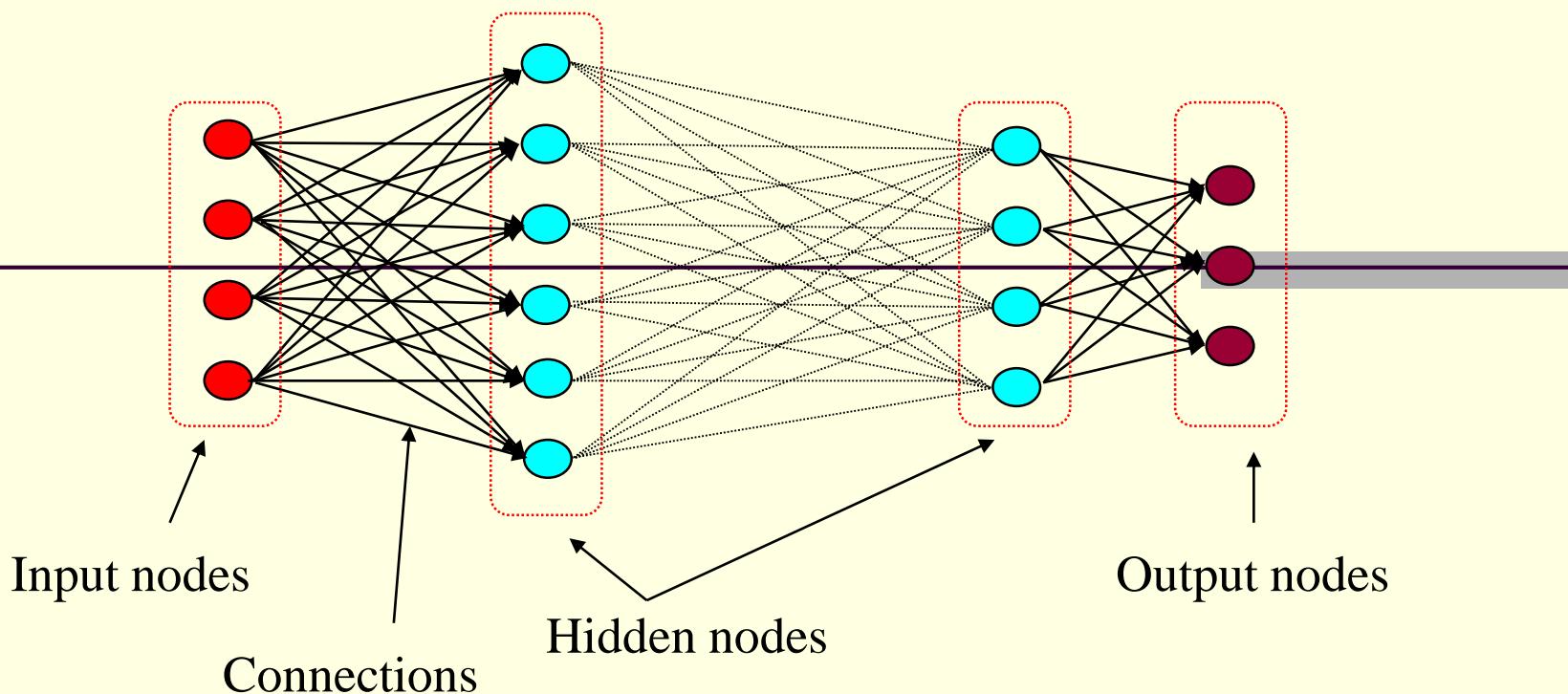
Perceptron แบ่งได้ 2 ชนิด

1 Single Layer Perceptron (SLP) คือ Perceptron แบบชั้นเดียว โดยผลลัพธ์จะคำนวณได้จากการรวมของข้อมูลเข้า และค่า俈หนัก (Weight Parameter) ของแต่ละจุดที่เชื่อมโยงกัน

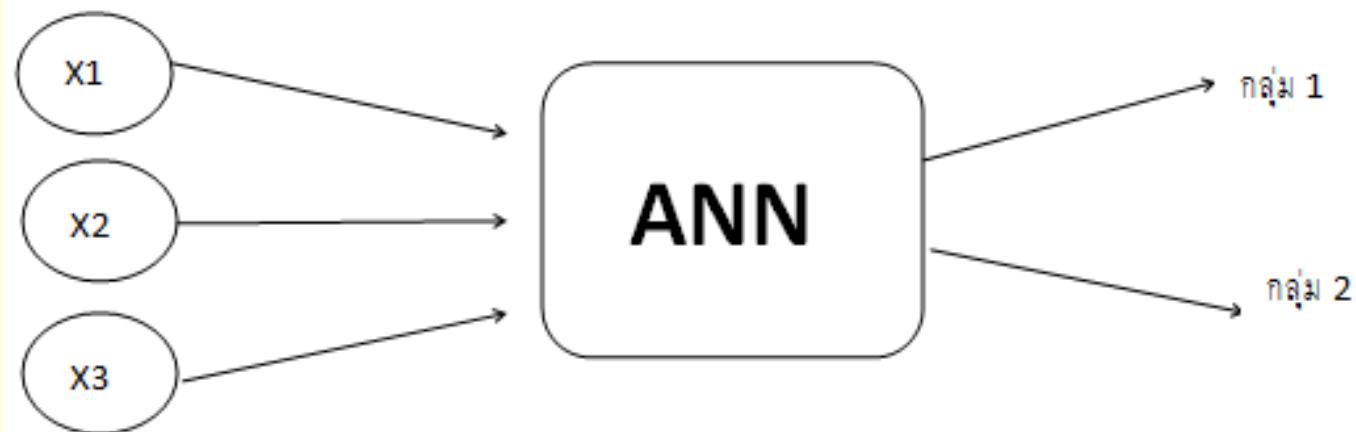


2 Multi-Layer Perceptron (MLP) คือ Perceptron ที่มีมากกว่า 1 ชั้น (Layer) โดยจะทำการรับข้อมูลทีละชั้นแล้วส่งไปชั้นถัดไป





สิ่งที่เราสนใจในระบบ AI ดังภาพ

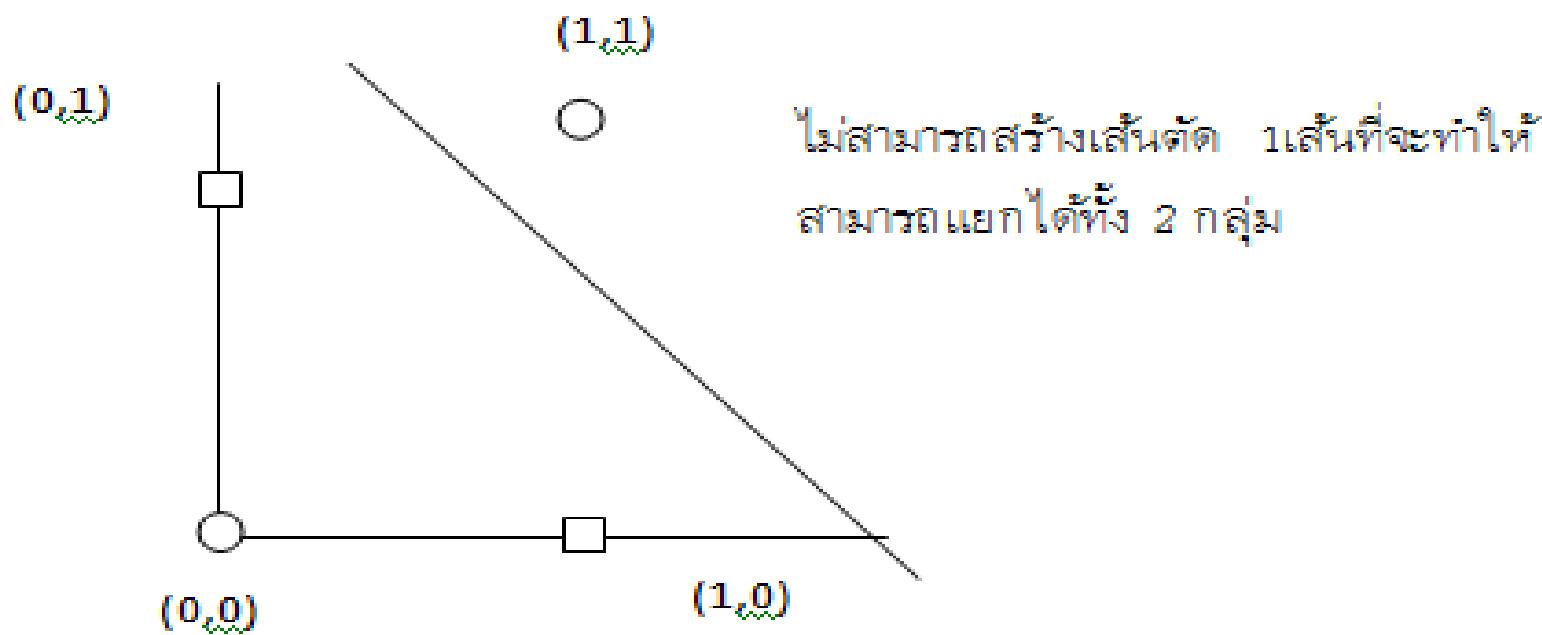


XOr เมื่อinputเป็น 0 ต่างกันเป็น 1

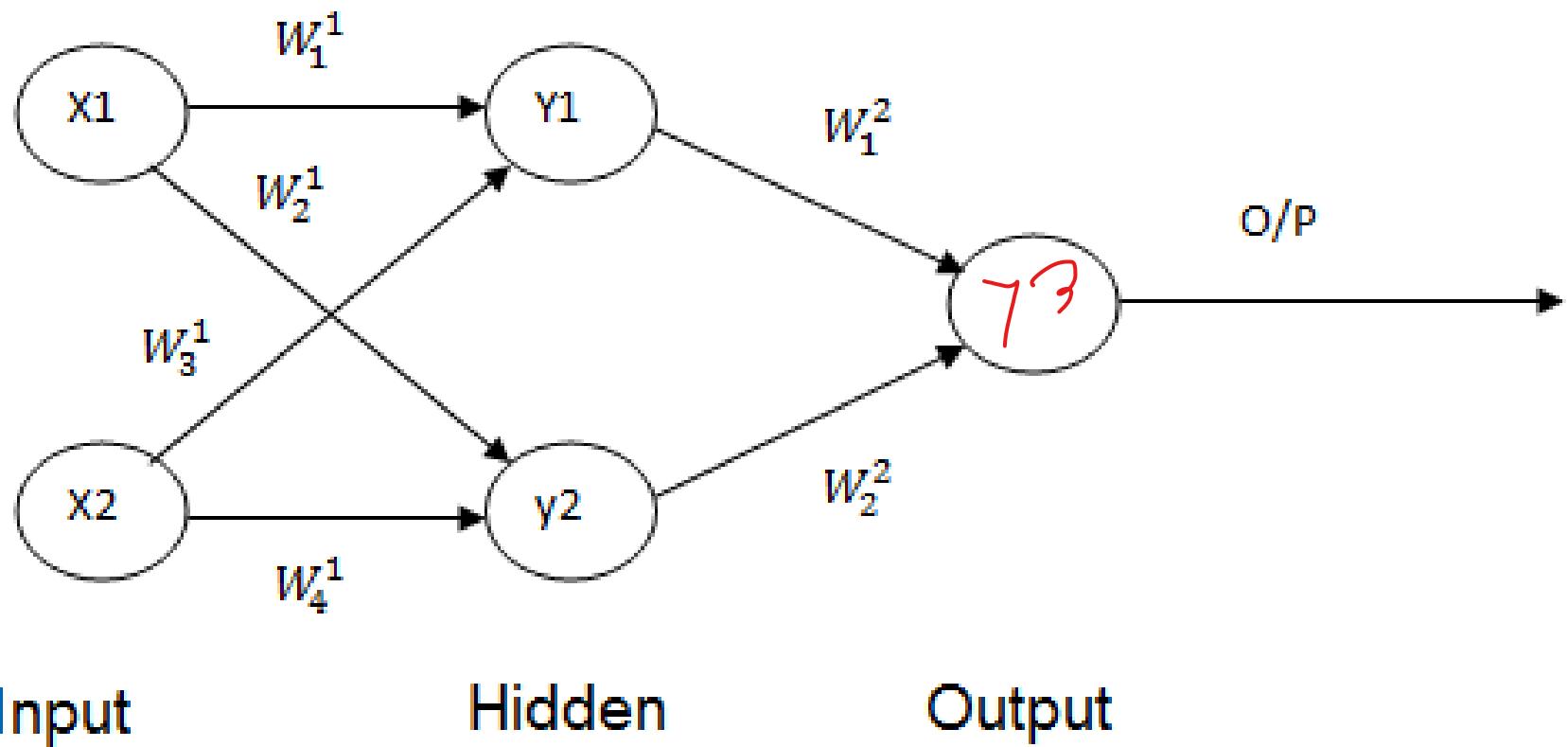
ตัวอย่าง Multi Layer Perceptron

| X1 | X2 | Y |
|----|----|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

ถ้าแบบนี้ใช้ Single Perceptron ไม่ได้ เพราะ



ดังนั้นจะใช้ MLP



Input

Hidden

Output

$$x_1 w_1 + x_2 w_2 + x_3 w_3 + w_0$$

Hidden Layer

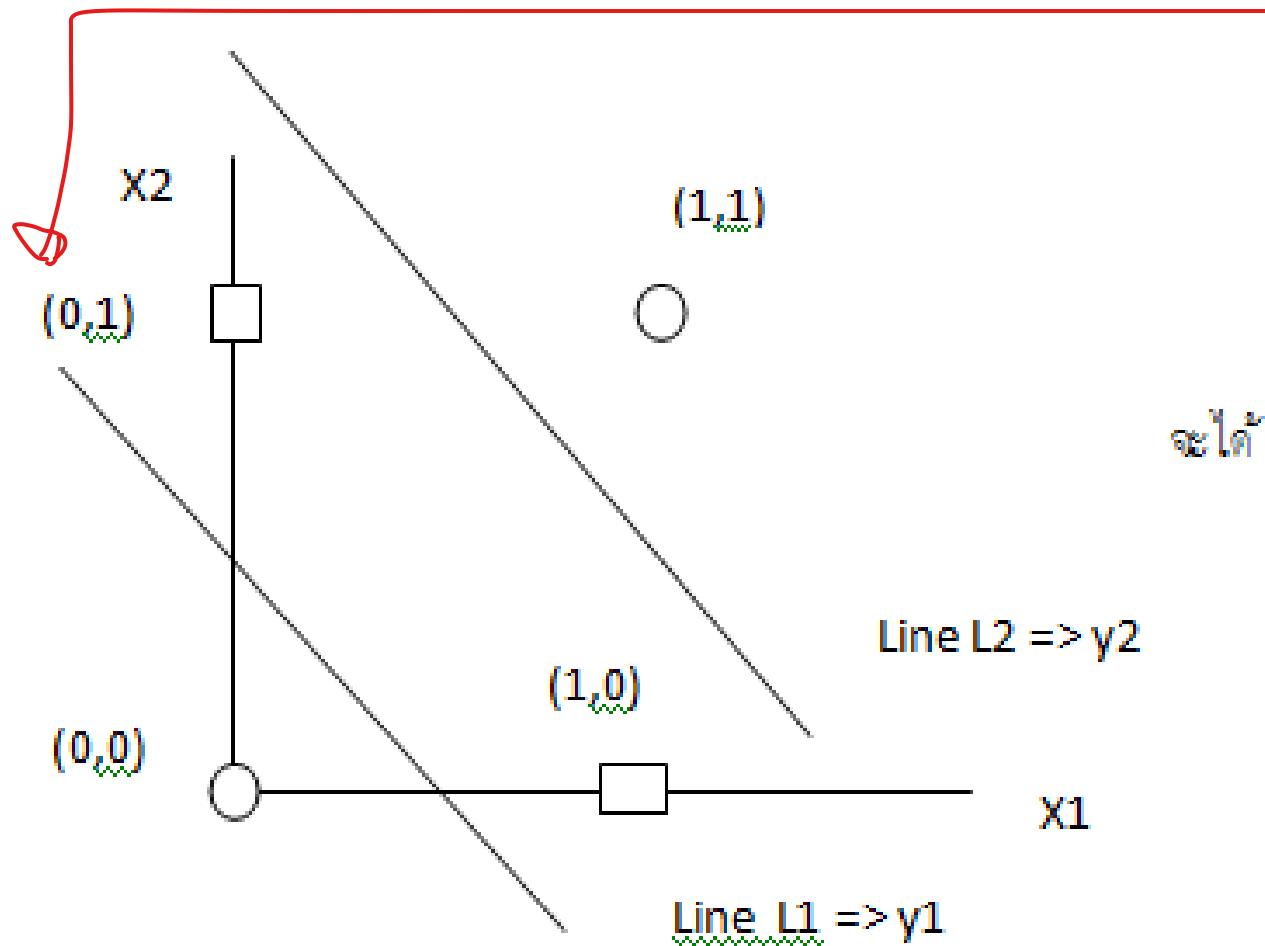
$$= a_1 \cdot c_1$$

$$y_1 = F(w_1^1 X_1 + w_3^1 X_2 + \theta_1^1) \quad \text{---} \quad y_1$$

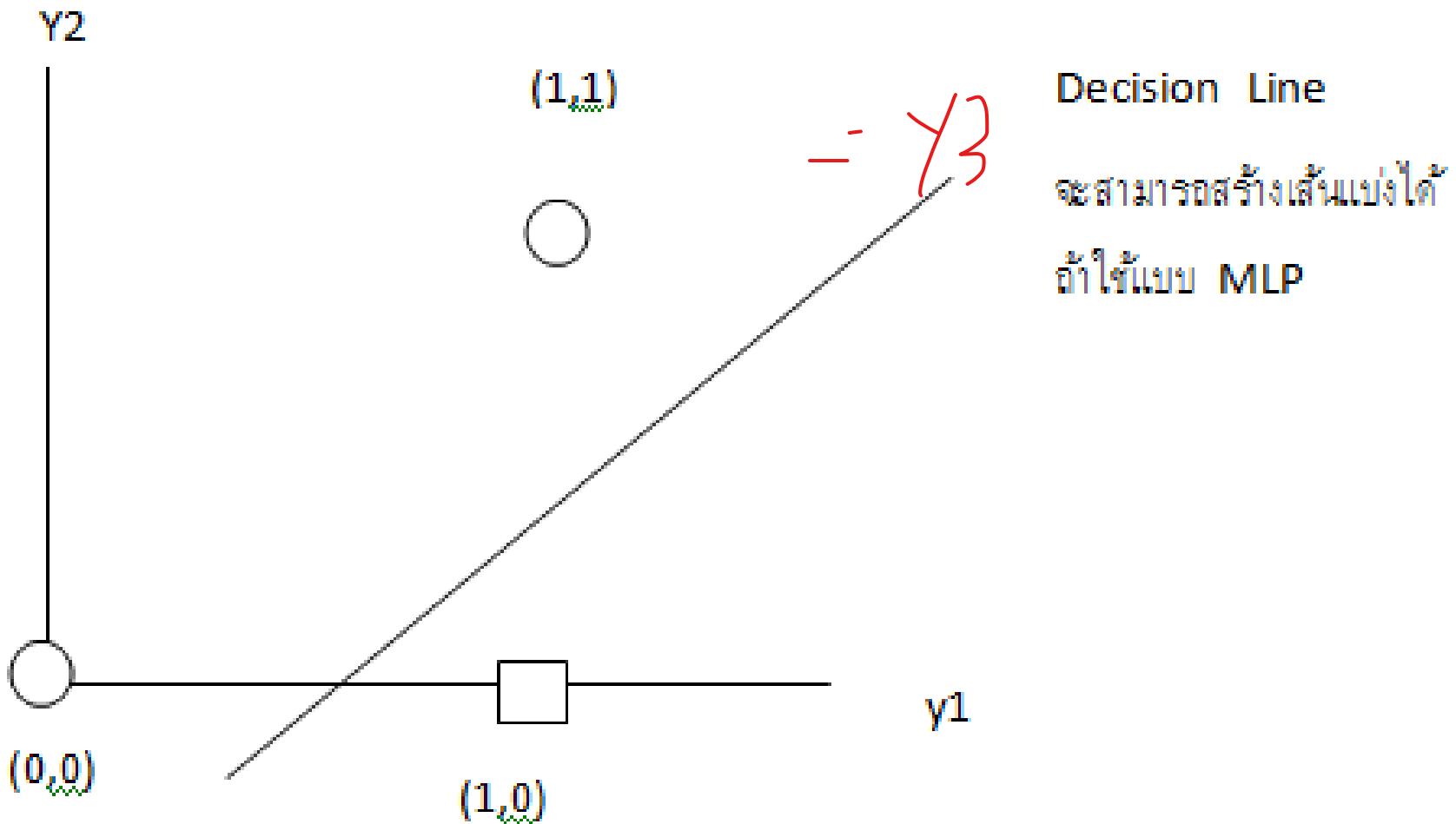
$$y_2 = F(w_2^1 X_1 + w_4^1 X_2 + \theta_2^1)$$

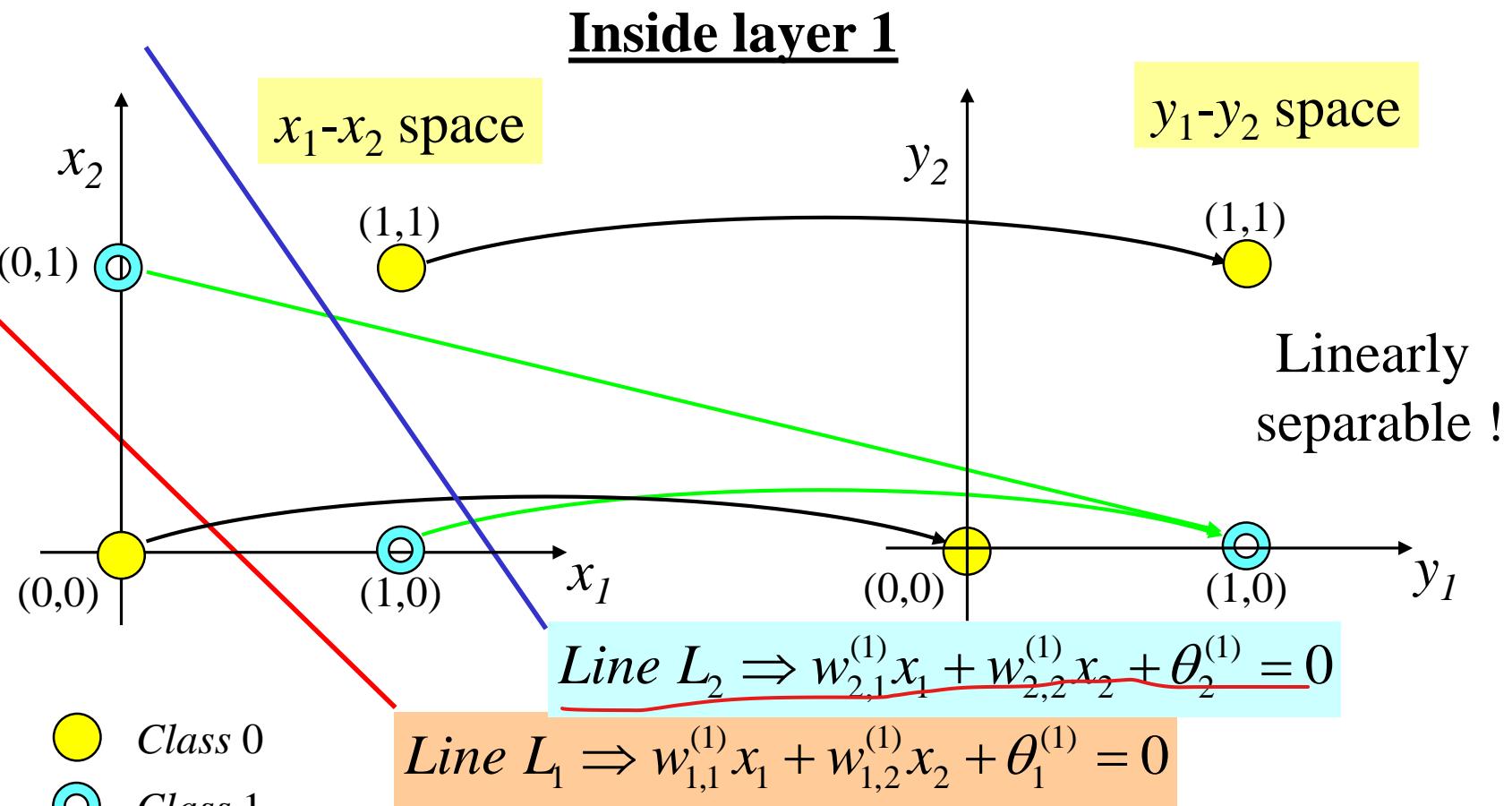
O/P Layer

$$o = F(w_1^2 y_1 + w_2^2 y_2 + \theta_1^2)$$



| x_1 | x_2 | y_1 | y_2 |
|-------|-------|-------|-------|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |





$$\cancel{w_1} \times \cancel{x_2} w_0 + x_3 + \cancel{w}$$



Multilayer Perceptron : How it works (cont.)

การทำงานของ hidden layers

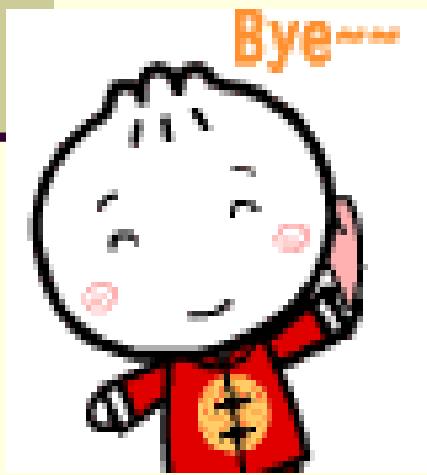
- พยายามจะแปลงข้อมูลที่เข้ามาที่ layer นั้นๆ ให้เป็น linearly separable ก่อนที่ข้อมูลจะถูกส่งไปถึง output layer
- ในขั้นตอนสุดท้ายที่ Hidden layer, ข้อมูลต้องเป็น linearly separable
- อาจจำต้องใช้ hidden layer มากกว่า 1 layer ในการแปลงข้อมูลให้อยู่ในรูป linearly separable
- โดยทั่วไป Activation function ของแต่ละ layer ไม่จำเป็นต้องเป็น Thresholding function และไม่จำเป็นต้องเป็น function เดียวกัน

NL- ๔๗๒๖๗๙

โครงข่ายประสาทเทียม (NN) คือ การสร้างคอมพิวเตอร์ที่จำลอง
การทำงานของสมองมนุษย์ คล้ายกับโครงข่ายประสาทของมนุษย์
เพื่อช่วยให้คอมพิวเตอร์สามารถเรียนรู้และตัดสินใจได้ ในการเรียนรู้ของ
เครือข่ายประสาท จะอาศัย Back-propagation Algorithm (วิธีการปรับ
ค่า Weight ให้เหมาะสมโดยจะส่งค่าผิดพลาดค่านวณย้อนกลับเพื่อให้ได้
Weight Parameter ที่ดี)

ในการเขียน การสร้างการเรียนรู้สำหรับ Neural Network เพื่อให้มี
มีความคิดเห็นมนุษย์ มีสองวิธี คือ Supervised Learning การเรียนรู้
แบบมีการสอน เปรียบเทียบกับคน เมื่อกับการสอนนักเรียน โดยมี
ครูผู้สอนอยแนะนำ และ Unsupervised Learning การเรียนรู้แบบไม่มี
การสอน เปรียบเทียบกับคน เช่น การที่เราสามารถแยกแยะพันธุ์พืช พันธุ์
สัตว์ตามลักษณะรูปร่างของมัน ได้เองโดยไม่มีการสอน

THE END



Lab

Lab: ให้เขียนโปรแกรมในการจำแนก หรือการวินิจฉัยโดยวิธี ANN จากข้อมูลใน Data Set และ ให้เปรียบเทียบกับวิธีที่เคยสอนทั้งหมด โดยแสดงค่า accuracy, precision, recall , F measure ของแต่ละวิธี (ทุกวิธีใช้วิธี Train 70, Test 30)

$$\boxed{70 / 30}$$

Matlab

% 1. โหลดข้อมูล Iris

load fisheriris

X = meas'; % เลือกข้อมูลแต่ละ colum นี้เป็นตัวแปรอิสระ

T = zeros(3, length(species)); % สร้างเมทริกซ์เป้าหมายเป็น [0 0 0] (setosa, versicolor, virginica)

T(1, strcmp(species,'setosa')) = 1;

T(2, strcmp(species,'versicolor')) = 1;

T(3, strcmp(species,'virginica')) = 1;

% 2. สร้างโครงข่ายประสาทเทียม

hiddenLayerSize = 10; % จำนวนเซลในชั้นซ่อน

net = patternnet(hiddenLayerSize);

% 3. แบ่งข้อมูลเป็นชุดฝึกฝนและชุดทดสอบ

```
net.divideParam.trainRatio = 0.7;
```

```
net.divideParam.valRatio = 0.15;
```

```
net.divideParam.testRatio = 0.15;
```

% 4. ฝึกโกรงข่ายประสาทเทียม

```
[net, tr] = train(net, X, T);
```

% 5. ทดสอบโกรงข่ายที่ฝึกแล้ว

```
y = net(X);
```

```
e = gsubtract(T,y);
```

```
performance = perform(net,T,y);
```

% 6. แสดงผลลัพธ์

figure;

plotconfusion(T, y); % แสดง Confusion Matrix

title('Confusion Matrix');

% 7. ระบุค่าความแม่นยำ

accuracy = 1 - performance;

fprintf('ความแม่นยำของโครงนี้: %f\n', accuracy);

- โหลดข้อมูล Iris และกำหนดตัวแปรอิสระ (X) และตัวแปรเป้าหมาย (T) โดย T คือเมทริกซ์เป้าหมายที่แทนคลาสของดอกไม้แต่ละตัว.
- สร้างโครงข่ายประสาทเทียมด้วย patternnet โดยกำหนดจำนวนเซลล์ในชั้นซ่อน (hidden layer).
- แบ่งข้อมูลเป็นชุดฝึกฝนและชุดทดสอบ โดยกำหนดอัตราส่วนการแบ่ง.
- ฝึกโครงข่ายประสาทเทียม โดยใช้ train และเก็บผลการฝึกไว้ในตัวแปร tr.
- ทดสอบโครงข่ายด้วยข้อมูล X และคำนวณค่าความแม่นยำและความผิดพลาด.
- แสดงผลลัพธ์ในรูปแบบ Confusion Matrix.
- ระบุค่าความแม่นยำของโครงข่าย
- hiddenLayerSize = 10 หมายความว่าให้ทุกชั้นซ่อนในโครงข่ายประสาทเทียมมีจำนวนโหนด (neurons) เท่ากับ 10 โหนดทุกชั้น เช่น ชั้นซ่อนที่ 1: 10 โหนด ชั้นซ่อนที่ 2: 10 โหนด ชั้นซ่อนที่ 3: 10 โหนด ... ชั้นซ่อนที่ n: 10 โหนด

Python

```
import numpy as np  
import tensorflow as tf  
from sklearn import datasets  
from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import OneHotEncoder
```

1. ໂກດຂໍອມລ Iris

```
iris = datasets.load_iris()
```

```
X = iris.data
```

```
y = iris.target
```

2. แบ่งข้อมูลเป็นชุดฝึกและชุดทดสอบ

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
random_state=42)
```

3. แปลงข้อมูลเป้าหมายให้อยู่ในรูปแบบ One-Hot Encoding

```
encoder = OneHotEncoder(sparse=False)
```

```
y_train_onehot = encoder.fit_transform(y_train.reshape(-1, 1))
```

```
y_test_onehot = encoder.transform(y_test.reshape(-1, 1))
```

4. สร้างโครงข่ายประสาทเทียม

```
model = tf.keras.Sequential([
```

```
    tf.keras.layers.Dense(10, input_dim=4, activation='relu'), # ชั้นซ่อน
```

```
    tf.keras.layers.Dense(3, activation='softmax') # ชั้นผลลัพธ์
```

```
])
```

5. คอมไพล์และฝึกโกรงป่าย

```
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

model.fit(X_train, y_train_onehot, epochs=100, batch_size=10, verbose=1)
```

6. ทดสอบโกรงป่าย

```
y_pred_onehot = model.predict(X_test)

y_pred = np.argmax(y_pred_onehot, axis=1)
```

7. คำนวณความแม่นยำ

```
accuracy = np.mean(y_pred == y_test)

print(f'ความแม่นยำของโกรงป่าย: {accuracy}')
```

- ใช้ไลบรารี TensorFlow และ Keras เพื่อสร้างโครงข่ายประสาทเทียมและฝึกโครงข่าย.
- ข้อมูล Iris จะโหลดจาก sklearn และแบ่งเป็นชุดฝึกและชุดทดสอบ.
- ใช้ One-Hot Encoding ในการแปลงข้อมูลเป้าหมาย.
- โครงข่ายประสาทเทียมมี 1 ชั้นซ่อน (hidden layer) ที่มี 10 โหนดและ 1 ชั้นผลลัพธ์ที่มี 3 โหนด (เนื่องจากมี 3 คลาสในข้อมูล Iris).
- เราใช้ softmax activation ในชั้นผลลัพธ์เพื่อทำนายคลาสที่เป็นไปได้.
- ในส่วนการฝึกโครงข่าย เราใช้ categorical cross-entropy เป็นฟังก์ชันสูญเสีย.
- ในส่วนการทดสอบโครงข่าย เราคำนวณความแม่นยำและแสดงผลลัพธ์.