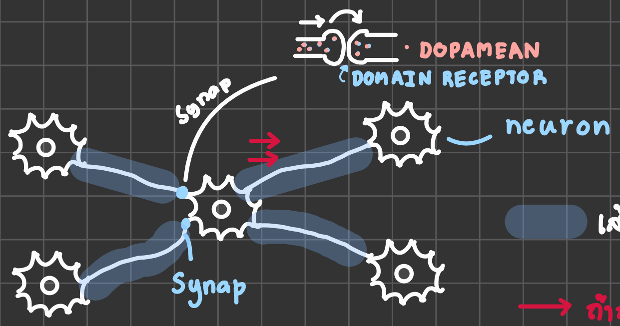
Artificial Intelligence
A Modern Approach

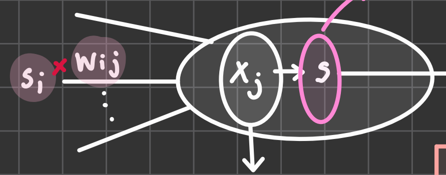
Artificial Neurons



เส้นแต่ละเส้นมีความสำคัญต่างกัน !

→ ถ้าการรวมกันของ signal มากพอจะส่ง activate (fire !)

→ เป็นตัว perform ว่า signal มากพอไหม ? (> 0)

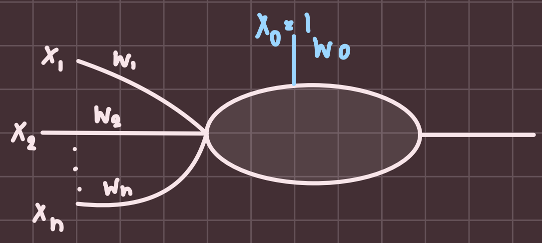


→ s_j (สัญญาณขาออกของ node j ใดๆ)

$$x_j = \boxed{\sum_{i=1}^{n} w_{ij} s_i} + \boxed{\theta_j}$$

w_{ij} (weight i & j) : weight ที่เชื่อมโยงระหว่าง neuron i และ j

● = Input signal



$$w_1 x_1 + w_2 x_2 + \dots + w_n x_n > b$$

$$-b + w_1 x_1 + w_2 x_2 + \dots + w_n x_n > 0$$

$$\boxed{-b} + \boxed{\sum_{i=1}^{n} w_i x_i} > 0$$

bias

สมมติ $w_0 = -b$: $w_0 x_0 + \sum_{i=1}^{n} w_i x_i > 0$
($x_0 = 1$)

$$\downarrow$$

$$\sum_{i=0}^{n} w_i x_i > 0$$

Signal function : — binary , linear , sigmoidal , Gassion , Relu

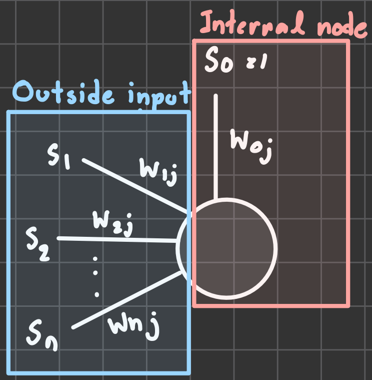
Neuron Equation

Binary Threshold Signal Function

$$S(x_j) = \begin{cases} 1 & x_j \geq 0 \\ 0 & x_j < 0 \end{cases} \qquad \Big| \qquad S(x_j) = \begin{cases} 1 & x_j \geq 0 \\ -1 & x_j < 0 \end{cases}$$

$x_j \downarrow \sum_{i=0} w_{ij} s_i$

binary — bipolar



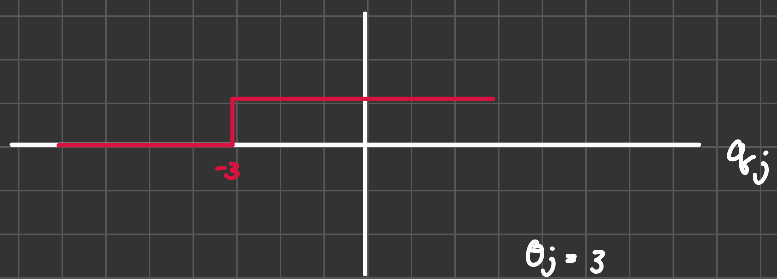
$$q_j = \sum_{i=1}^{n} w_{ij} s_i$$

$$x_j = q_j + \theta_j$$

$\downarrow$

+3 มาเติม

$$x_j = \underbrace{\sum_{i=1}^{n} w_{ij} s_i}_{\text{outside input}} + \underbrace{3}_{\text{internal node}} \quad (\theta_j = 3)$$



$\theta_j = 3$

เราใช้ trian NN เป็นรอบๆ (discrete time)

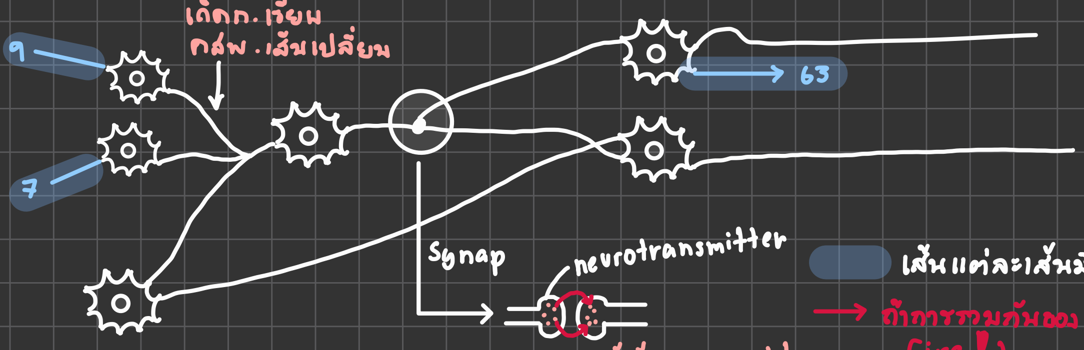
Threshold Logic Neuron (TLN) in Discrete time

$$S(x_j^{\text{รอบ } k+1}) = \begin{cases} 1 & x_j^{k+1} > 0 \\ S(x_j^{\text{รอบ } k}) & x_j^{k+1} = 0 \\ 0 & x_j^{k+1} < 0 \end{cases}$$

(เอา signal เดิมมาออก)

extended เป็น bipolar ได้

13 Jan 2024 . Artificial Neuron

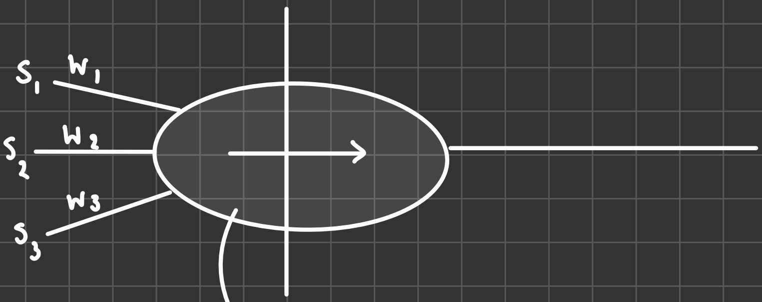


เส้นแต่ละเส้นมีความสำคัญต่างกัน!

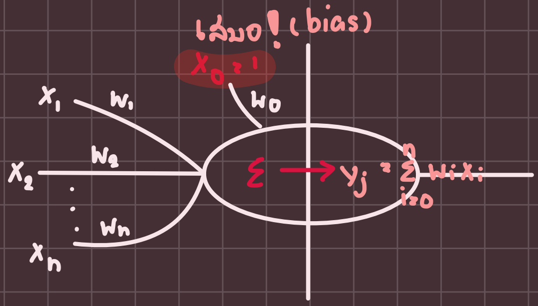
→ ถ้าการรวมกันของ signal มากพอจะส่ง activate (fire!)

Elasticity , Plasticity

- Elasticity: พึ่งเรียน แล้วลืม (กลับจุดเดิม)
- Plasticity: เกิดการทำซ้ำ



$y_j = w_1 s_1 + w_2 s_2 + \dots + w_n s_n$



$w_1x_1 + w_2x_2 + \dots + w_nx_n > b$

$-b + w_1x_1 + w_2x_2 + \dots + w_nx_n > 0$

$w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n > 0$

? >, ≥ depends on user

$$S(n_j) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i > b \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Output } S(y_j) = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} n_i x_i \boxed{\geq} 0 \\ 0 & \text{otherwise} \end{cases}$$

Signal Function

เราใช้ trian NN เป็นรอบๆ (discrete time)

Threshold Logic Neuron (TLN) in Discrete time

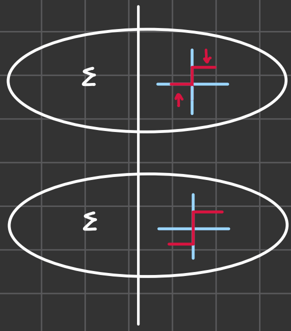
$$S(x_j \text{ รอบ } k+1) = \begin{cases} 1 & x_j\ k+1 > 0 \\ S(x_j \text{ รอบ } k) & x_j\ k+1 = 0 \quad \text{(เอา signal เดิมมาออก)} \\ 0 & x_j\ k+1 < 0 \end{cases}$$

extended เป็น bipolar ได้

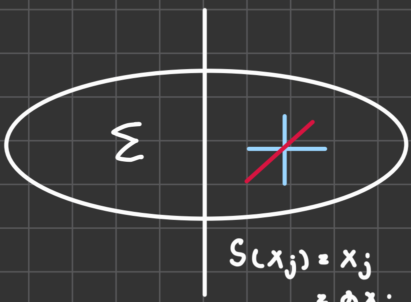
TLN perceptron (neuron) → { 0, 1 } binary

$\begin{cases} 1 \\ -1 \end{cases}$

→ { 1, -1 } bipolar



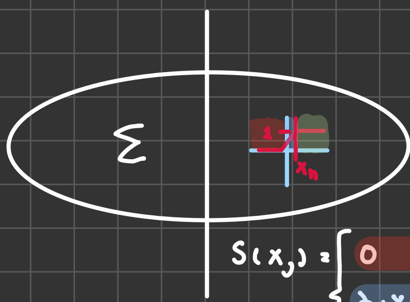
Linear Signal



$S(x_j) = x_j$
$= a x_j$

shift x-axis
$a(x-c)+b$
shift y-axis
$ax+b$

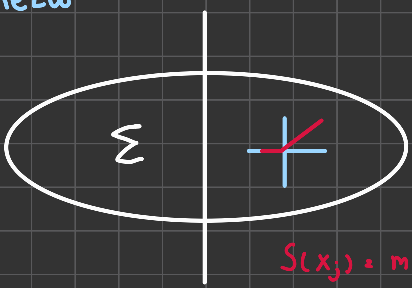
Linear Threshold Signal



$$S(x_j) = \begin{cases} 0 & x_j \le 0 \\ a_j x_j & 0 < x_j < x_m \\ 1 & x_j \ge x_m \end{cases}$$

$S(x_j) = \max(0, \min(a_j x_j, 1))$

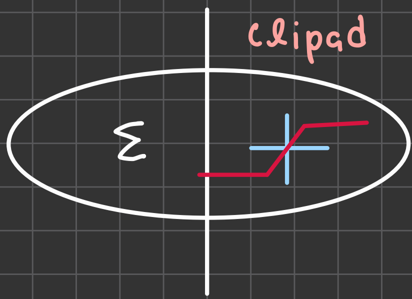
ReLu



shift x-axis
$\max(0-c)$
shift y-axis
$\max(0-c)+b$

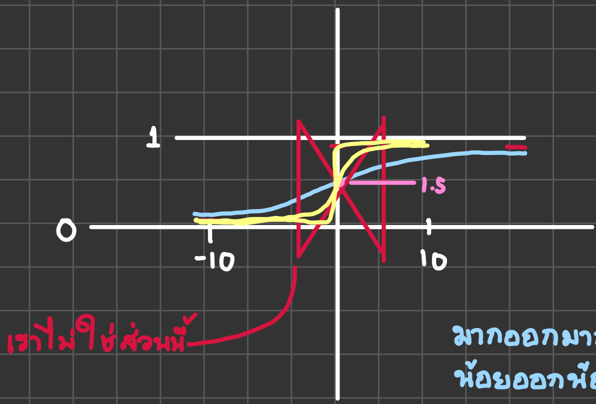
$$S(x_j) = \max(0, x_j) = \begin{cases} 0 & \text{if } x \le 0 \\ x & \text{if } x > 0 \end{cases}$$

Clipped



Sigmoid [Perceptron]

$$\Sigma(y_j) = \frac{1}{1+e^{-y_j}} = \frac{1}{1+e^{-\lambda_j y_j}}$$

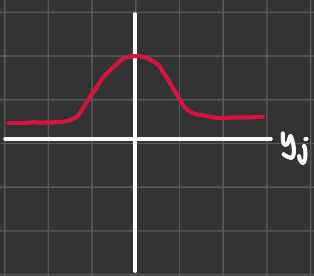


- ยิ่ง λ มีค่ามากกราฟก็จะชัน (เส้นเหลือง)
  ยิ่งชันมากยิ่งเข้าใกล้พฤติกรรม binary
  ∴ monotonic
  continuous
  bounded

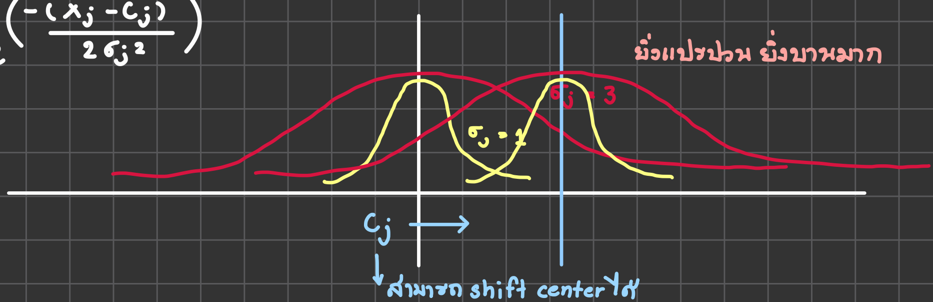
มากออกมาก
น้อยออกน้อย

Gousian Signal Function

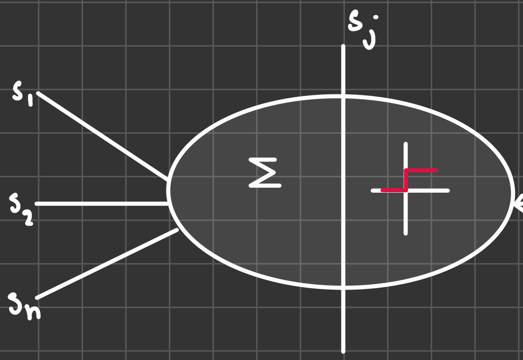
(non-monotonic) อยากได้ output ที่ใกล้ center มากสุด
ใกล้center ออก max, 1 (ตรงกับ ก.ต้องการเท่านั้นออก max)



$$S(x_j) = e^{\left(\frac{-(x_j - c_j)^2}{2\sigma_j^2}\right)}$$



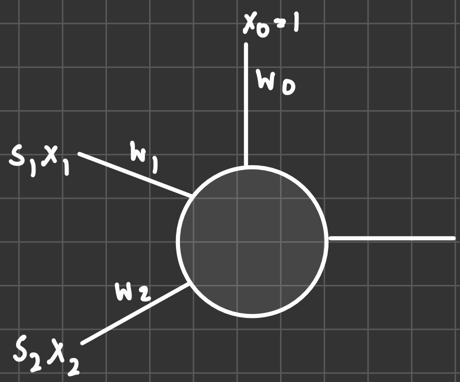
20 Jan 2025 : TLN Architecture



$$\text{output} = \begin{cases} 1 & \text{if } y_j > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{classification}$$

output $\in \{0,1\} \rightarrow$ binary

$$y_j = \sum_{i=0} w_i s_i$$



$w_1 x_1 + w_2 x_2 > b$ (มากกว่า treshow ค่าหนึ่งจึงจะ activate)

$-b + w_1 x_1 + w_2 x_2 > 0$

$w_0 + w_1 x_1 + w_2 x_2 > 0$

เราเริ่มเอา neural ก้อนนี้ไป classify logic And

AND TABLE : + , T = 1 , | − , F = 0

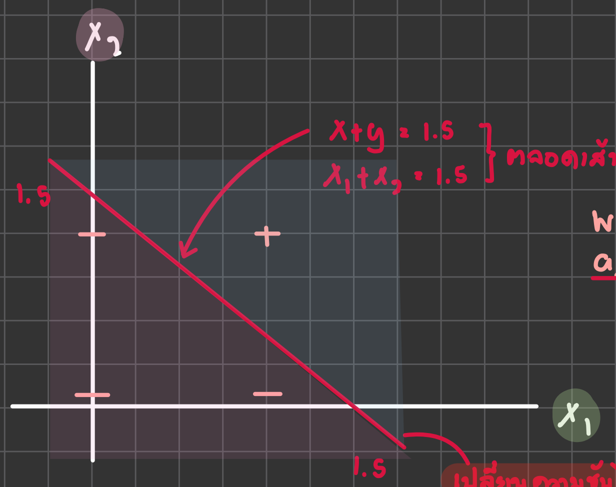
	x_1	x_2	target (output)		
x_1	0	0	−	$\rightarrow w_0 + w_1(0) + w_2(0) < 0$	$w_0 < 0$
x_2	0	1	−	$\rightarrow w_0 + w_1(0) + w_2(1) < 0$	$w_0 + w_2 < 0$
x_3	1	0	−	$\rightarrow w_0 + w_1(1) + w_2(0) < 0$	$w_0 + w_1 < 0$
x_4	1	1	+	$\rightarrow w_0 + w_1(1) + w_2(1) > 0$	$w_0 + w_1 + w_2 > 0$

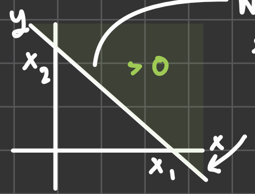
target = − (false)

target = + (true)

หา set ของ $w_0 w_1 w_2$ = $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$ $\vec{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $\vec{w} \cdot \vec{x}$

หน้าที่หลักคือการหาเส้นนี้





$w_0 + w_1x_1 + w_2x_2 = 0$

$x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_0}{w_2}$

$ax + by + c = 0$

$y = -\frac{a}{b}x - \frac{c}{b}$

$w_0 + w_1x + w_2y > 0$

$a_x + by + c > 0$ (สมการเชิงเส้น)

ถ้าอยากรู้ว่า > 0 ตรงไหนก็หาเส้นตรง

ความชัน

จุดตัดแกน.

เปลี่ยนความชันได้ (มี infinite เส้น); เส้นนี้ suppose มาเพื่อความง่าย.

จากสมการ $w_0 + w_1x + w_2y > 0$ จะได้ว่า

$x_1 + x_2 > 1.5$

$-1.5 + x_1 + x_2 > 0$

$-1.5 + (1)x_1 + (1)x_2 > 0$

$\hat{w} = \begin{bmatrix} -1.5 \\ 1 \\ 1 \end{bmatrix}$

$x_1 + x_2 < 1.5$

$1.5 - x_1 - x_2 > 0$

$1.5 + (-1)x_1 + (-1)x_2 > 0$

$\hat{w} = \begin{bmatrix} 1.5 \\ -1 \\ -1 \end{bmatrix}$

---

	x_1	x_2	target (output)
$\bar{x}_1$	0	0	−
$\bar{x}_2$	0	1	−
$\bar{x}_3$	1	0	−
$\bar{x}_4$	1	1	+

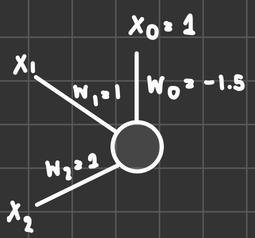
เลือกใช้สมการ $-1.5 + x_1 + x_2 > 0$

$-1.5 + 0 + 0 = -1.5 > 0$; FALSE (−)

$-1.5 + 0 + 1 = -0.5 > 0$; FALSE (−)

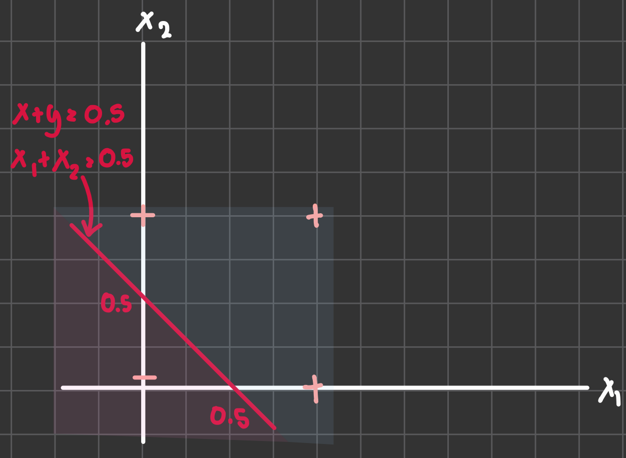
$-1.5 + 1 + 0 = -0.5 > 0$; FALSE (−)

$-1.5 + 1 + 1 = 0.5 > 0$; TRUE (+)



OR TABLE : +,T = 1 , | -,F = 0

	x_1	x_2	target (output)
x^1	0	0	−
x^2	0	1	+
x^3	1	0	+
x^4	1	1	+



จะได้ $x_1+x_2 > 0.5$

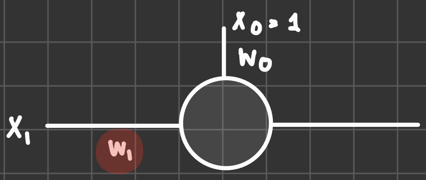
$-0.5+x_1+x_2 > 0$ ✓

$\vec{w} = \begin{bmatrix} -0.5 \\ 1 \\ 1 \end{bmatrix}$

$x_1 + x_2 < 0.5$

$0.5 - x_1 - x_2 > 0$

$\vec{w} = \begin{bmatrix} 0.5 \\ -1 \\ -1 \end{bmatrix}$



เข้า 0 อยากให้มากกว่า 1
เข้า 1 อยากให้น้อยกว่า 0

x_1	target
0	+
1	−

จาก $w_0 + w_1x_1 > 0$ จะได้ว่า

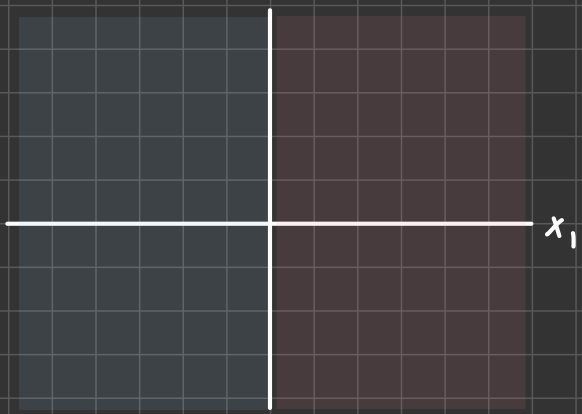
$w_0 + w_1(0) > 0$

$w_0 + w_1(1) \leq 0$

$\rightarrow < 0$

; $w_0 > 0$

$w_0 + w_1 \leq 0$



$x_1 = 0$

(เข้าลบออกบวก)

$x_1 < 0$

$w_0 + w_1x_1 < 0$

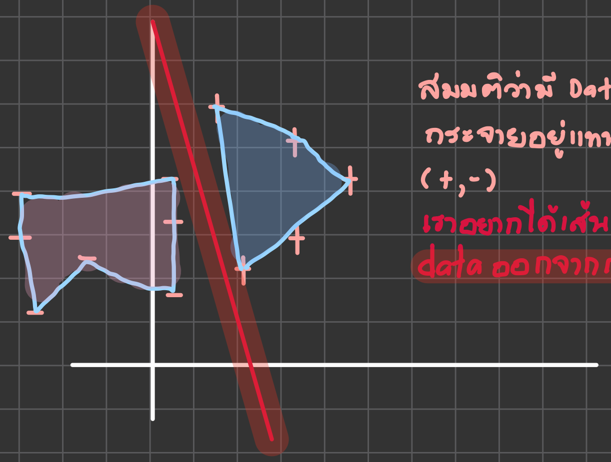
$-\cancel{w_0} - w_1x_1 > 0$

$x_1 > 0$

$w_0 + w_1x_1 > 0$

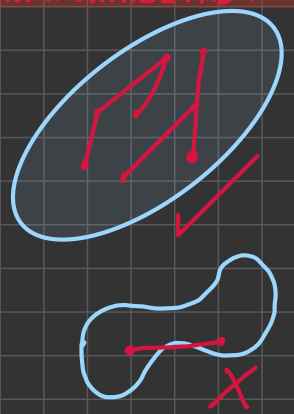
$\vec{w} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$

Linearly Seperable : สามารถจำแนกได้โดยเส้นตรง ลงไป 1 เส้น



สมมติว่ามี Data
กระจายอยู่แทนด้วย
(+,-)
เราอยากได้เส้น split
data ออกจากกัน

สร้างพท.รอบกลุ่มตย. ที่เล็กที่สุด (intersection)
ที่สามารถครอบ AREA ได้ Convex house

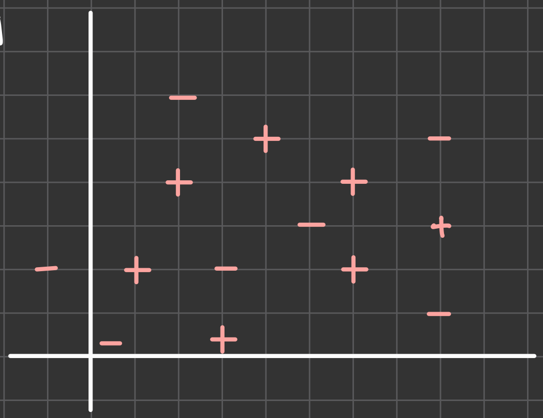


โดยที่เส้นที่เชื่อมต้อง
ไม่ออกจากพท.
และ convex house
ของทั้ง 2 classes
จะต้องไม่ทับกัน
(intersect แล้ว
เป็น empty area)

Non-Linearly Separable

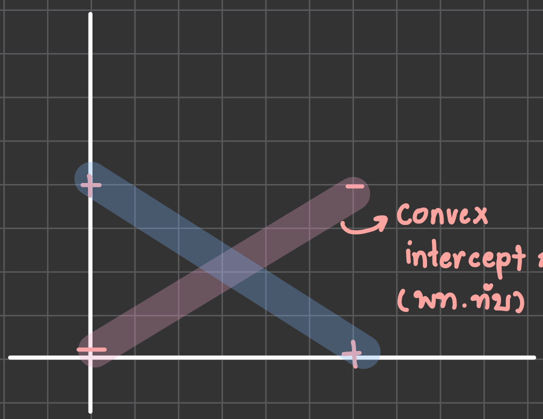
Linearly Non-Separable
(ยึดคำนี้)

TLN



มีจุดกระจายตัวใน space
จำนวน P จุด กระจายใน n มิติ
← 14 จุด กระจายใน 2 มิติ
อยากรู้ว่าโอกาสเป็น linearly separable = ?

แล้วถ้าแยกไม่ได้โดยเส้นตรง (Linearly Non-Separable)

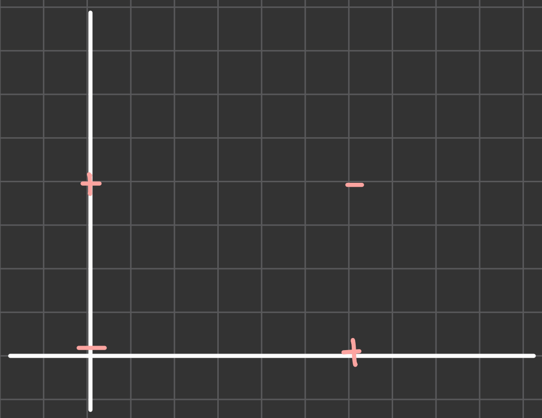


convex
intercept กัน
(พท.ทับ)

XOR TABLE (แยกไม่ได้โดย linearly)

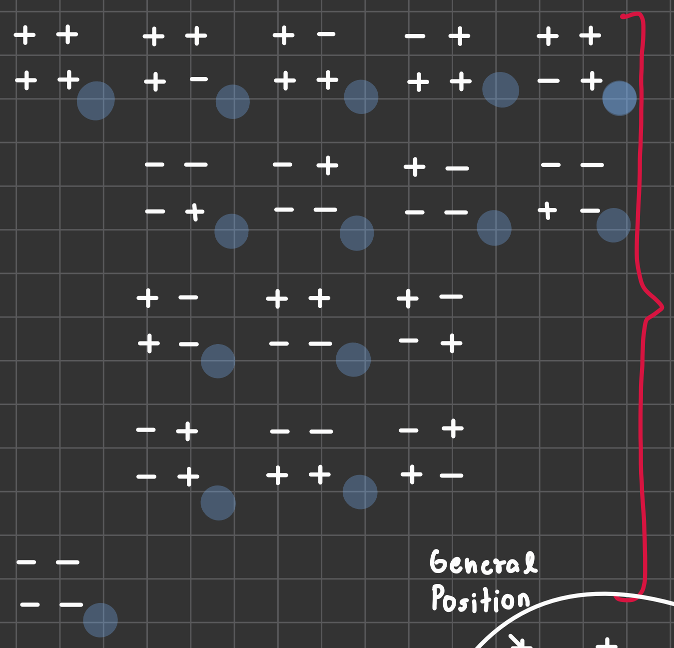
x_1	x_2	target
0	0	−
0	1	+
1	0	+
1	1	−

(เพศเดียวกันไม่สามารถมีลูกได้)



มีจุด 4 จุดใน space 2 มิติ

Assign ค่าได้ 16 แบบ

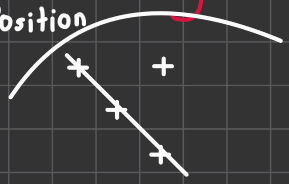


16 แบบ

$P = \frac{14}{16}$

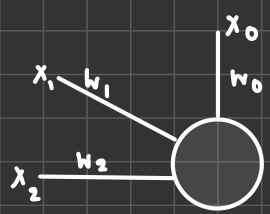
ค่าความจุของ perceptron!

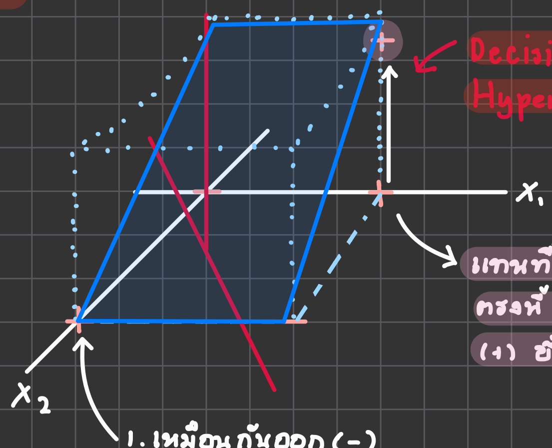
General Position



Linearly Seperable

ถ้าแยกไม่ได้ทำอย่างไร ?





Decision Surface

Hyperplane

แทนที่จุด (+) จะอยู่ตรงนี้ เราจะ shift up (+) ขึ้นไปแทน

1. เหมือนกันออก (-)

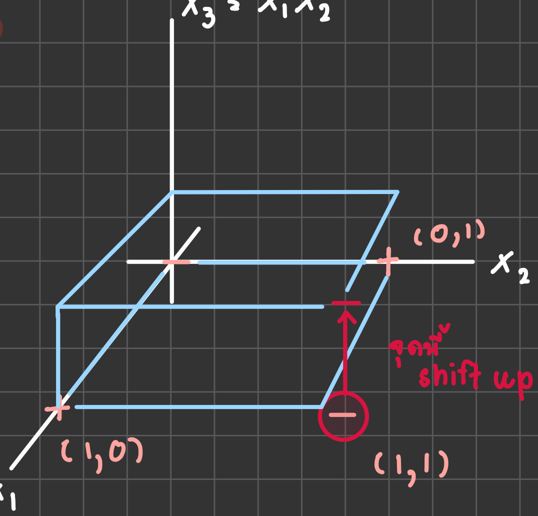
เพิ่ม Dimension โดยการจับ $x_1 \cdot x_2$ [+ = 1 , - = 0]

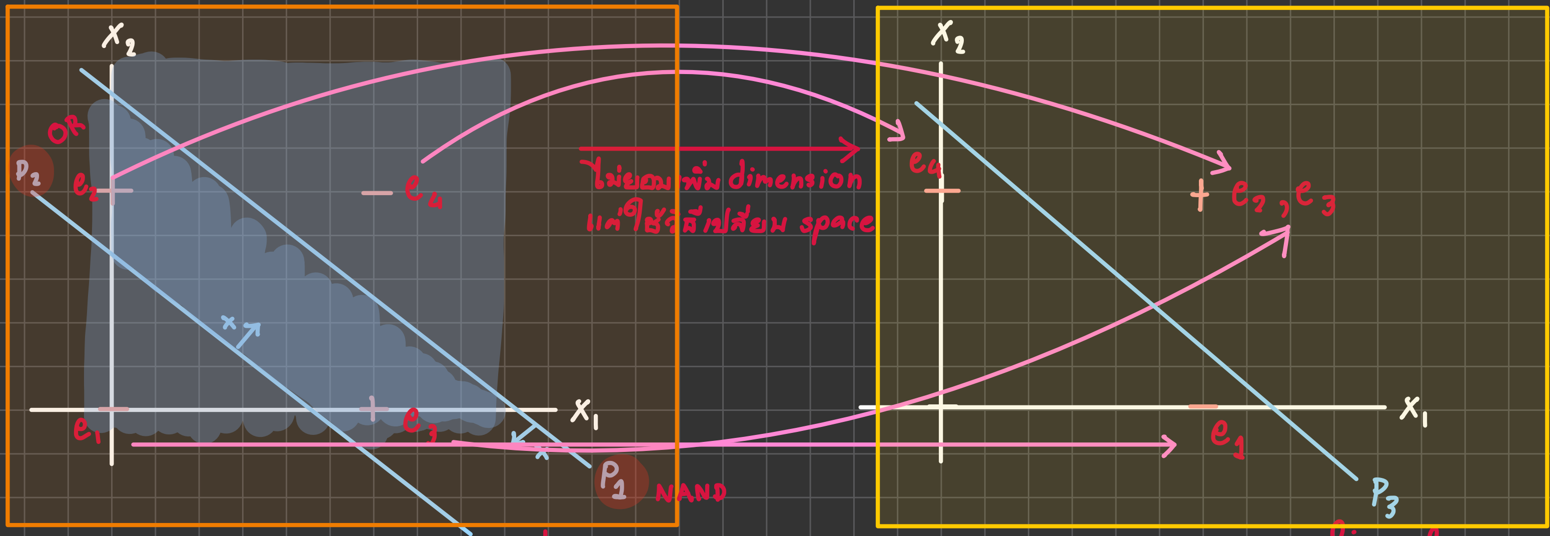
x_1	x_2	(2) $x_1 \cdot x_2$	target	
0	0	0	0	−
0	1	0	1	+
1	0	0	1	+
1	1	1	0	−

$w_0 + w_1x_1 + w_2x_2 + w_3(x_1 \cdot x_2) > 0$

(จะหา decision surface)

$x_3 = x_1x_2$

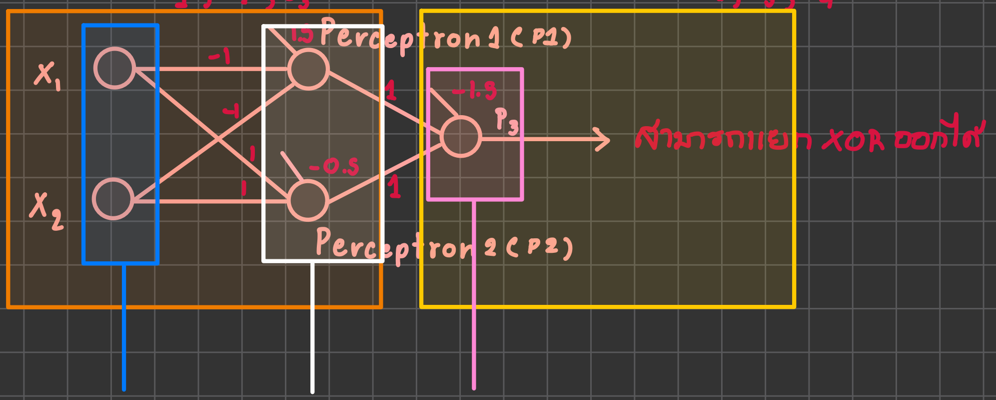




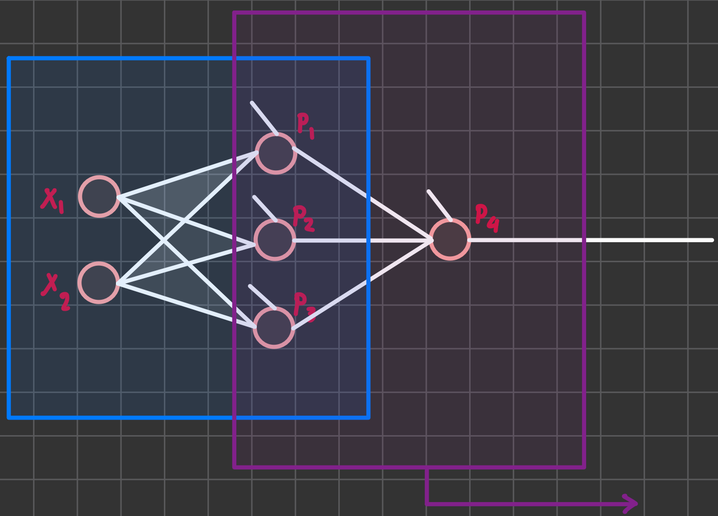
linearly Separable

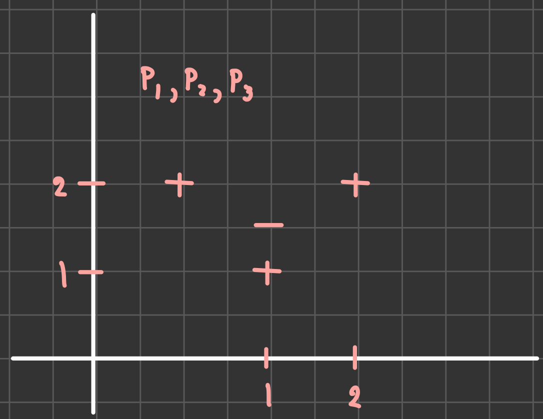
	x_1	x_2	P_1	P_2	target
e_1	0	0	1	0	0
e_2	0	1	1	1	1
e_3	1	0	1	1	1
e_4	1	1	0	1	0

ใช้ P1 จำแนก e_1, e_2, e_3 เป็นบวก ใช้ P2 จำแนก e_2, e_3, e_4 เป็นบวก



INPUT LAYER + HIDDEN LAYER + OUTPUT LAYER = MLP Multi Layer Perceptron





$-$: (1,1)

$+$: (0,2), (1,0.5), (2,2)

27 Jan : Dichotomise and Capacity of a Perceptron & Number of Regions Seperated by Perceptrons

อยากรู้ว่าจุด p จุดใน n มิติ (gen position)
แยกได้โดยเส้นตรงกี่เส้น

Session overview (คลิปแรก)

Defination ของ
General Position

ex. กระจายตัวยังไงใน 2D (จุด 3 จุดไม่อยู่บนเส้นตรงเดียวกัน)
3D (จุด 4 จุดไม่อยู่ในระนาบเดียวกัน

จุด 4 จุดใน 2 มิติ

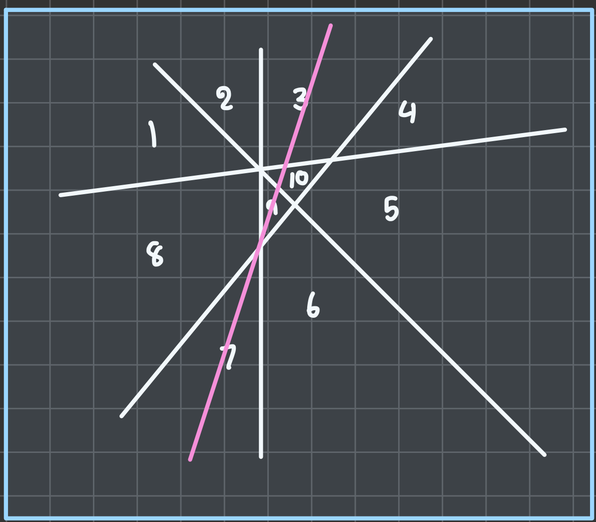
มีการแบ่ง (Dichotomines) $2^4 = 16$ แบบ

14 แบบ (linearly separable)

$\therefore$ capacity $= \frac{14}{16}$

จุด 4 จุดใน 3 มิติ $\therefore$ capacity $= \frac{16}{16}$

Session overview (คลิปสอง)

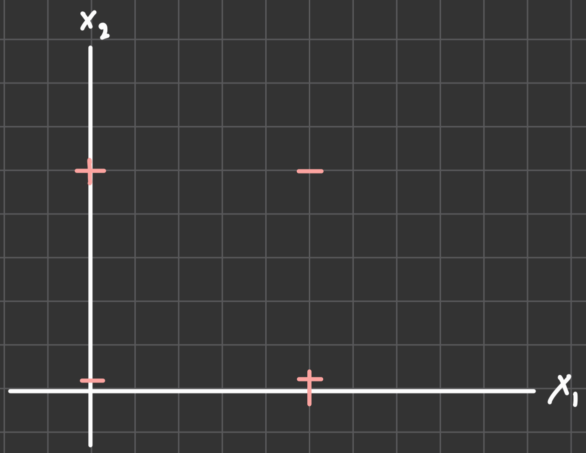


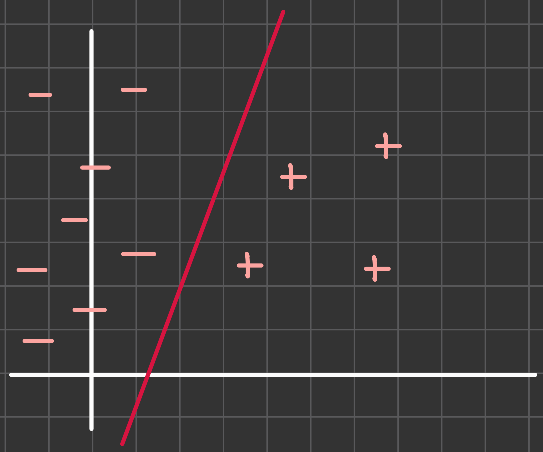
มีกี่ Region และถ้ามีเส้นใหม่มา อยากรู้ว่า
เราจะหา Region ยังไง
(จะเหลือ fomular ที่อยากให้เข้าใจ
การ prove และ used)

XOR (ไม่สามารถแบ่งได้ด้วย linearly Separable)

x_0	x_1	x_2	target (output)
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

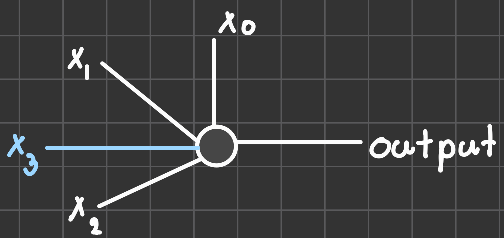
● = เหมือนกันออก 0





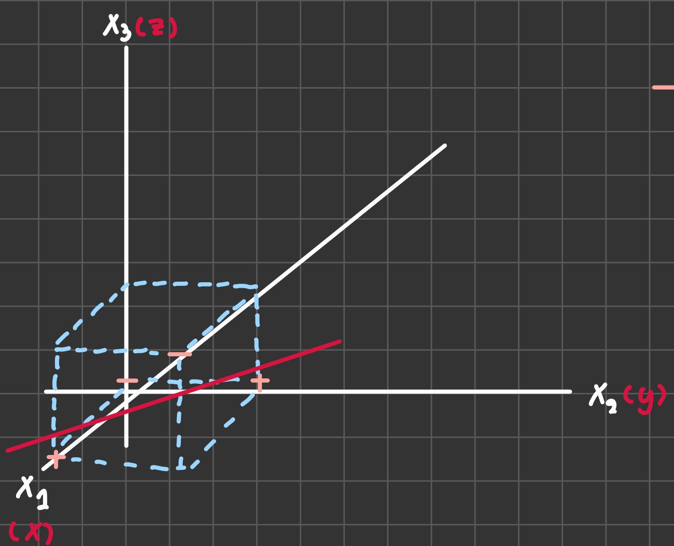
แล้วจะมีกี่ case ที่แยก sample ได้

3D



$$w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 > 0$$

x y z (มองเป็น x-y-z)

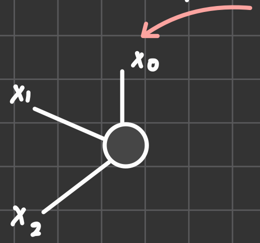


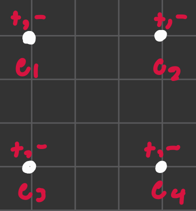
สมมติให้ x_3 เกิดจากการนำ $x_1 \cdot x_2$ จะได้

x x_1	y x_2	z x_3	target (output)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

∴ การเพิ่มมิติทำให้เราแยกตัวอย่างได้ดีขึ้น

เลยมี concept ว่า perceptron มี potential ในการแยก sample ได้ดีขนาดไหน



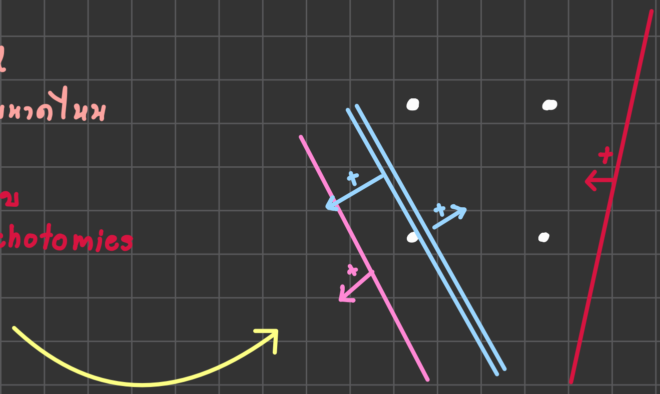


2^4 แบบ

16 dichotomies

มีจุดกระจายตัวใน general position.

จุด p จุดใน n มิติ จะอยู่ใน gen pos ก็ต่อเมื่อไม่มีจุดจำนวน n+1 จุด วางใน hyperplane : n-1 มิติ

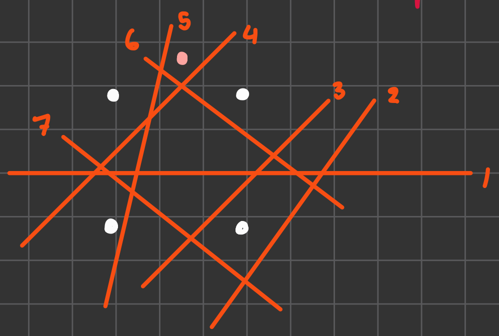


เราอยากรู้ว่ามีจุดจำนวนเท่าไหร่ โดยวางเส้นตรงลงไปใน space ที่จุดกระจายตัว

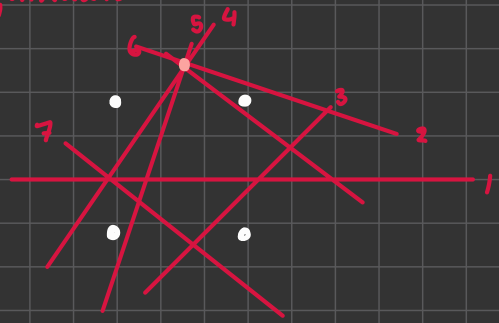
7 เส้น (14 แบบจาก 16 แบบที่สามารถแบ่งได้ด้วยเส้นตรงได้)

Suppose : datermine $L(p,n)$ • = จุดที่เพิ่มใหม่

↳ p+1 (เพราะมี 1 จุดเพิ่มมา)



Before



After.

L_p = จำนวนการแบ่งที่ลากผ่านจุดใหม่ได้

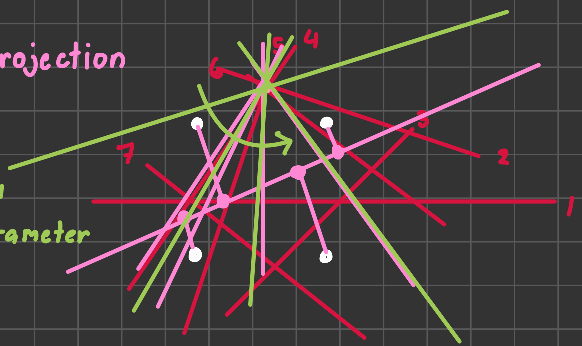
$L(p+1,n) = L(p,n) - L_p + 2L_p$ (3 จุด left / right)

จำนวนการแบ่งเดิมที่ลากผ่านจุดใหม่ไม่ได้

$L(p+1,n) = L(p,n) + L_p$ (หน้าที่คือหา L_p)

1. — : ลาก projection ⊾

2. การปรับมุม ได้แค่ 1 parameter



$L_p = L(p,n-1)$

$L(p+1,n) = L(p,n) + L(p,n-1)$

$L(1,0) = 2$

$L(1,1) = 2$

$L(1,2) = 2$

$\therefore L(1,n) = 2 \quad n \geq 0$

$L(4,3) = L(3,3) + L(3,2)$

จุด 4 จุดใน space 3 มิติ $= p+1$; $p = 3$

$L(3,3) = L(2,3) + L(2,2)$

$L(3,2) = L(2,2) + L(2,1)$

$L(4,3) = L(2,3) + L(2,2) + L(2,2) + L(2,1)$

$= L(2,3) + 2L(2,2) + L(2,1)$

$L(2,3) = L(1,3) + L(1,2)$

$L(2,2) = L(1,2) + L(1,1)$

$L(2,1) = L(1,1) + L(1,0)$

$L(4,3) = L(1,3) + L(1,2) + 2L(1,2) + 2L(1,1) + L(1,1) + L(1\text{-}0)$

$= 1 \cdot L(\overset{2}{\cancel{1}},3) + 3L(\overset{2}{\cancel{1}},2) + 3L(\overset{2}{\cancel{1}},1) + 1 \cdot L(\overset{2}{\cancel{1}},0)$ — สัมประสิทธิ์ ; เป็น 2 ตาม theory

$2 \quad 6 \quad 6 \quad 2$ = 16 แบบ

$\binom{3}{3} \quad \binom{3}{2} \quad \binom{3}{1} \quad \binom{3}{0}$

1 3 3 3

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

$$L(p,n) = \sum_{i=0}^{p-1} \binom{p-1}{i} \underbrace{L(1,n-i)}_{2\text{ (ตาม theory)}}$$

$$L(p,n) = 2\sum_{i=0}^{p-1} \binom{p-1}{i}$$; บางที n น้อย / หมดก่อน

* $\boxed{= 2\sum_{i=0}^{\min(n,p-1)} \binom{p-1}{i}}$

อยากได้ 4 จุดใน 2 มิติ จาก $L(p+1,n) = L(p,n) + L(p,n-1)$

$L(4,2) = 2\sum_{i=0}^{\min(2,3)} \binom{3}{i}$

$= 2\left[\binom{3}{0} + \binom{3}{1} + \binom{3}{2}\right]$

$= 2[1 + 3 + 3]$

$= 2(7) = 14$ แบบ

อยากได้ 5 จุดใน 2 มิติ จาก $L(p+1,n) = L(p,n) + L(p,n-1)$

$L(5,2) = 2\sum_{i=0}^{\min(2,4)} \binom{4}{i}$

$= 2\left[\binom{4}{0} + \binom{4}{1} + \binom{4}{2}\right]$

$= 2[1 + 4 + 6] = 2(11)$

$= 22$ แบบ

โดยสามารถหา prob ได้จาก $P(p,n) = \dfrac{L(p,n)}{2^p} = \begin{cases} 2^{1-p}\sum_{i=0}^{n}\binom{p-1}{i} & ; \begin{matrix} n < p-1 \\ p > n+1 \end{matrix} \\ 1 & ; p \leq n+1 \text{ (แยกได้ง่าย)} \end{cases}$

①

$$\frac{2\sum_{i=0}^{p-1}\binom{p-1}{i}}{2^p}$$

$$\boxed{\frac{2\left[\binom{p-1}{0}+\binom{p-1}{1}+\binom{p-1}{2}+\ldots+\binom{p-1}{p-1}\right]}{2^p}} \rightarrow \frac{(2^{p-1})(2)}{2^p} = 1$$

② ③ ④

For instance

$$\binom{4}{0}+\binom{4}{1}+\binom{4}{2}+\binom{4}{3}+\binom{4}{4}$$

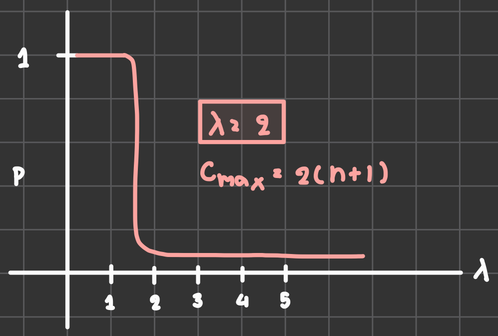
$$1 + 4 + 6 + 4 + 1 = 16 \rightarrow 4^2$$

$$2^4$$

$$\binom{3}{0}+\binom{3}{1}+\binom{3}{2}+\binom{3}{3}$$

$$1 + 3 + 3 + 1 = 8 \rightarrow 2^3$$

* ให้จุดใดๆ $P = \lambda(n+1)$

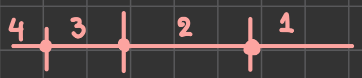


ถ้าใน Hidden Layer มี node จำนวนมาก จะหา region ที่เกิดขึ้นจาก Hidden Nodes ได้อย่างไร ?

(Induction proof)

(1 เส้น อยากรู้ว่าตัดกี่เส้น)

พิจารณาจากกรณี 1 มิติ



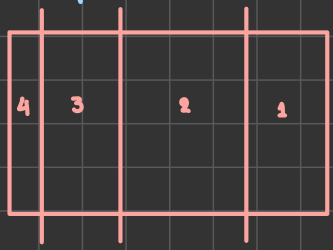
∴ n+1

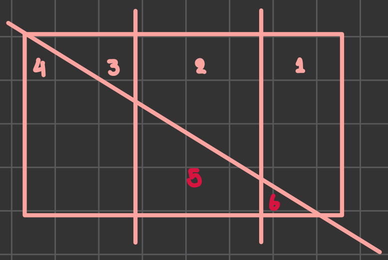
3+1 = 4

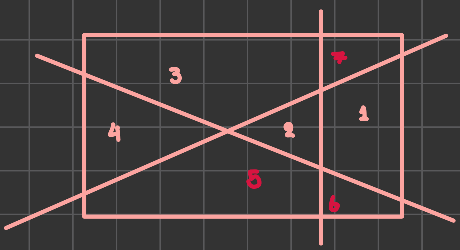
$$\binom{3}{1} + \binom{3}{0}$$

3 1

Suppose ยืดออกมาเป็น 2 มิติ เส้น ——— สามารถ บิดเพื่อเกิด region ใหม่ได้



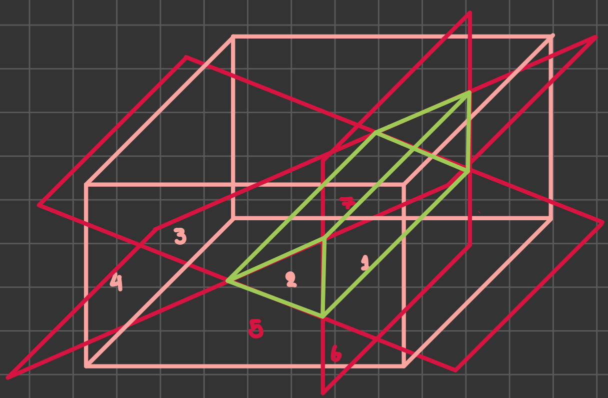


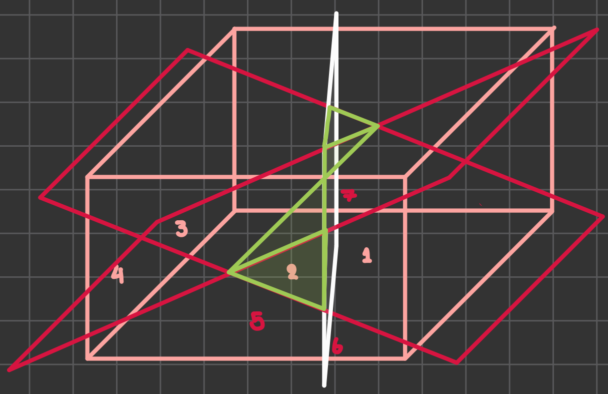


3 + 3+1 = 7

$$\binom{3}{2} + \binom{3}{1} + \binom{3}{0}$$

Suppose เพิ่มมาอีก 1 มิติ

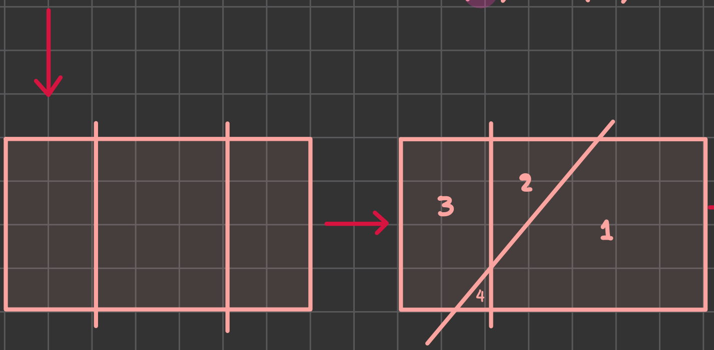




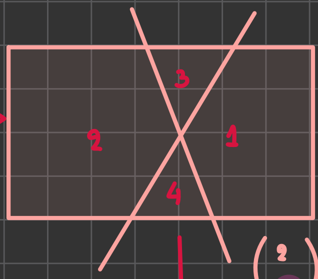
1 + 3 + 3+1 = 8

$$\binom{3}{3} + \binom{3}{2} + \binom{3}{1} + \binom{3}{0}$$

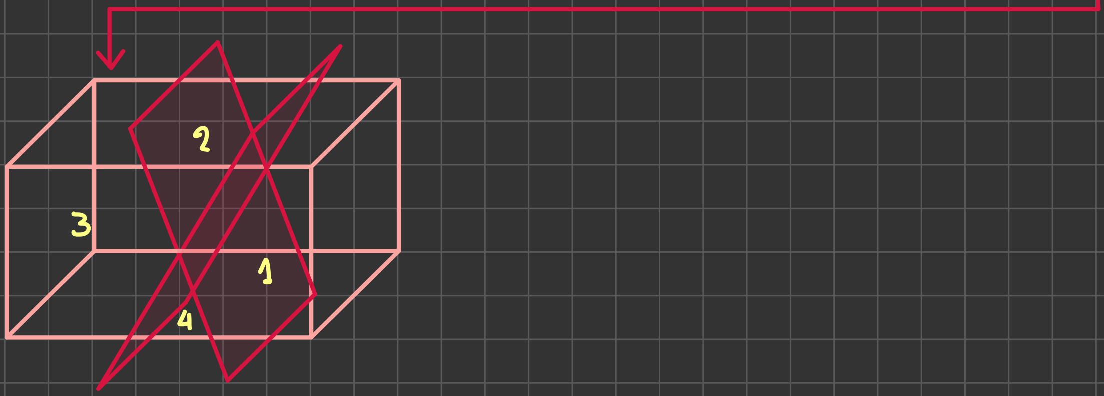
$2+1 \quad \binom{2}{1}+\binom{2}{0}$



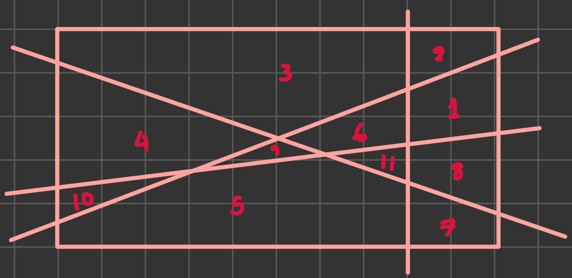
$1+2+1$



$\binom{2}{2}+\binom{2}{1}+\binom{2}{0}$



$4+1 = 5$



$6+4+1 \qquad \binom{4}{2}+\binom{4}{1}+\binom{4}{0}$

$$\boxed{\#\ \text{region} = \sum_{i=0}^{n}\binom{n}{i} \quad \text{เมื่อ } n \text{ คือจำนวนมิติ}}$$

20

3 Feb: TLN Training

ทบทวน !

General Position: จุด n+1 จุดจะต้องไม่วางตัวใน space (n-1) มิติ

$L(p,n) = 2\sum_{i=0}^{\min(n,p-1)} \binom{p-1}{i}$: จำนวน Dichotomies ที่แบ่งโดยเส้นตรงหนึ่งเส้นได้

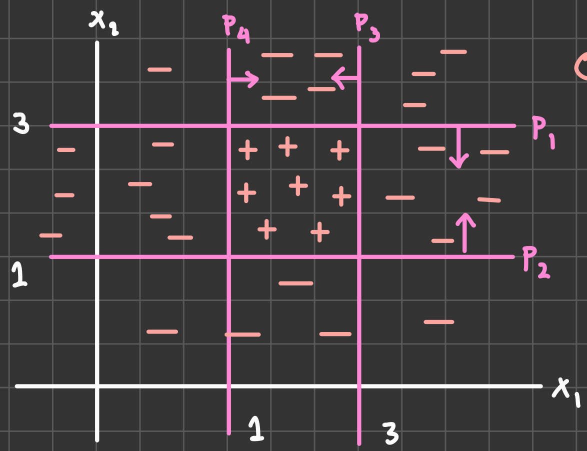
↓

จุด p จุด ใน n มิติ

ex. หาจุด p จุด ใน space 2 มิติ

Region = $\sum_{i=0}^{n}\binom{h}{i}$: เมื่อ n คือจำนวนมิติและ h คือจำนวน hyperplane

MLP : ถ้าเกิดว่ามี + กลาง - ล้อมรอบ อยากรู้ว่าโครงสร้างของ network ที่จำแนกชุดนี้ออกมาได้เป็นอะไร ?



Q: จงเขียน poceptron structure weight ที่ classifier sample (+,-) โดยใช้ TLN (ในคลิป MLP Specific Area)

; ต้องการเส้นตัด 4 เส้น ($P_1 - P_4$) และสมการ NN

$W_0 + W_1X_1 + W_2X_2 > 0$

(P_1) $x_2 < 3$

$-x_2 + 3 > 0$ (ย้าย x_2 มาให้ตรงตามสมการ nn) ; $w_2 = -1$ (ดูเลขสัมประสิทธิ์หน้า x_2)
ส่วน $w_1 = 0$ (เพราะไม่มี x_1)

(P_2) $x_2 > 1$

$-1 + x_2 > 0$

(P_3) $x_1 < 3$

$-x_1 + 3 > 0$

(P_4) $x_1 > 1$

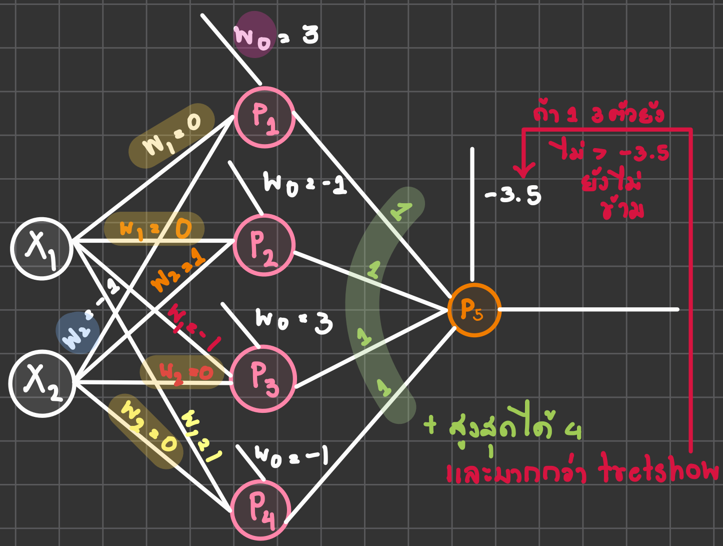
$-1 + x_1 > 0$

↓

ได้ weight ละ

P_1	P_2	P_3	P_4	
T	T	T	T	
1	1	1	1	→ 1

AND 4 INPUT



Wiegth = 0 ตัดทิ้งได้

Train TLN

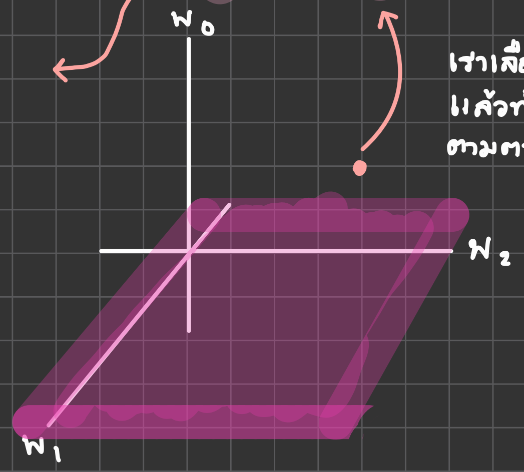
AND TABLE

x_1	x_2	target
0	0	−
0	1	−
1	0	−
1	1	+

สมการที่อยากได้

$w_0 + w_1x_1 + w_2x_2 > 0$

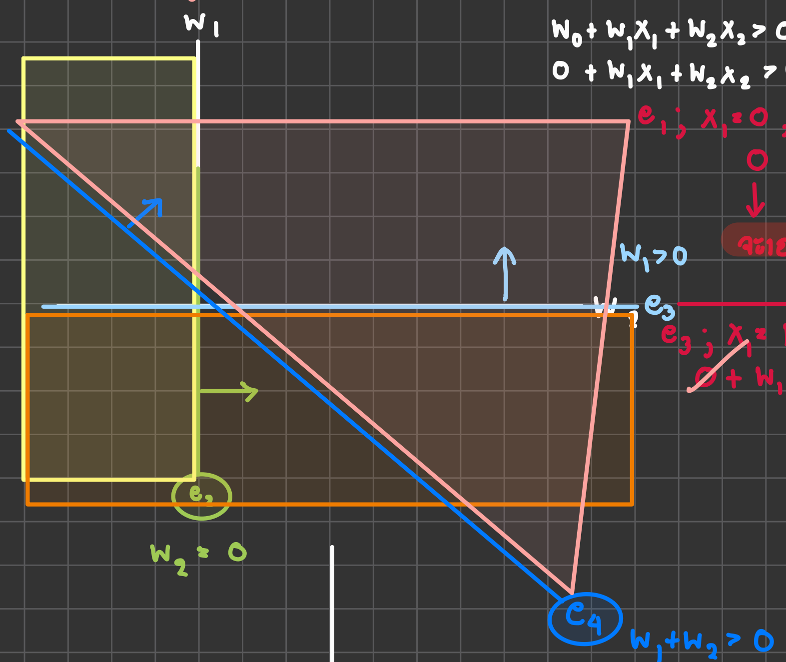
เราเลือกมาจุดนึงใน 3D space
แล้วทำให้ทุก w classifier
ตามตารางทั้งหมด



ถ้าให้ $w_0 = 0$

$w_0 + w_1x_1 + w_2x_2 > 0$

$0 + w_1x_1 + w_2x_2 > 0$



e_1; $x_1 = 0$, $x_2 = 0$

0 > 0 FALSE ทั้งแถบ

จะเอา w_1, w_2 ตรงไหนก็ได้

e_2; $x_1 = 0$, $x_2 = 1$

$\cancel{0} + \cancel{w_1(0)} + w_2(1) > 0$

$\boxed{w_2 > 0}$

e_3; $x_1 = 1$, $x_2 = 0$

$\cancel{0} + w_1(1) + \cancel{w_2(0)} > 0$

$\boxed{w_1 > 0}$

e_4; $x_1 = 1$, $x_2 = 1$

$0 + w_1(1) + w_2(1) > 0$

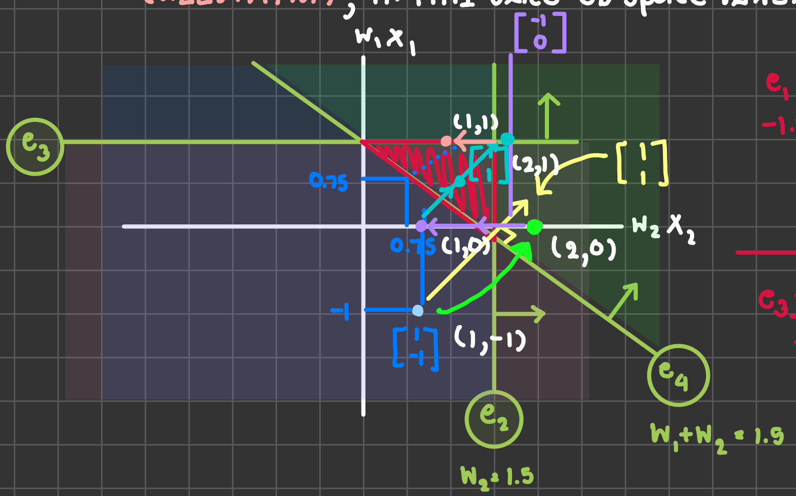
$\boxed{w_1 + w_2 > 0}$

ไอเดีย ! : อยากได้ region ร่วมกันและการปรับ weight

∴ ไม่มี region ร่วมกัน

$w_0 = -1.5$

(ลบอะไรก็ได้) ; ทำ MRI slice 3D space เป็นขนมปังแผ่น ๆ แล้วหยิบแผ่น $w_0 = -1.5$



e_1; $x_1 = 0$, $x_2 = 0$

$-1.5 + w_1(0) + w_2(0) > 0$

$-1.5 > 0$

ALL FALSE !

e_2; $x_1 = 0$, $x_2 = 1$

$-1.5 + w_1(0) + w_2(1) > 0$

$-1.5 + w_2 > 0$

e_3; $x_1 = 1$, $x_2 = 0$

$-1.5 + w_1(1) + w_2(0) > 0$

$-1.5 + w_1 > 0$

e_4; $x_1 = 1$, $x_2 = 1$

$-1.5 + w_1(1) + w_2(1) > 0$

$-1.5 + w_1 + w_2 > 0$

$w_1 + w_2 > 1.5$

∴ หยิบจุดใดๆใน $\triangle$ จะต้องจำแนก sample ได้ถูกต้องทั้งหมด !

สมมติตอนแรกสุ่มจุดสีฟ้า (•) มา classifier e_1, e_2, e_3 ถูก

↓

How to ขยับ จุดสีฟ้า (•) มาอยู่เหนือเส้นได้

ANS + vector $\begin{bmatrix}1\\1\end{bmatrix}$ จากตำแหน่งเดิม $\begin{bmatrix}1\\-1\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}2\\0\end{bmatrix}$

↓

ตก e_1, e_3, e_4

อยากให้วกกลับมา $\begin{bmatrix}-1\\0\end{bmatrix} + \begin{bmatrix}2\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \longrightarrow$ + vector $\begin{bmatrix}1\\1\end{bmatrix}$

ตก e_1, e_2, e_3

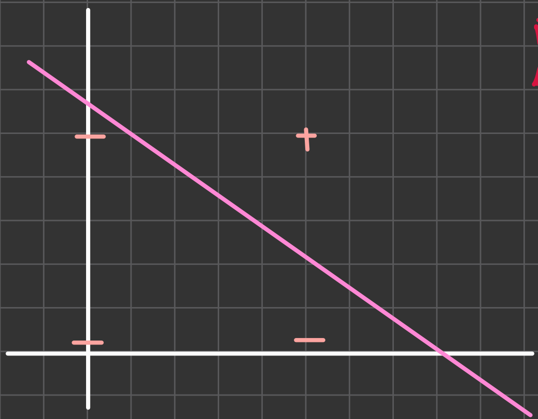
จะเห็นว่าเวลา + vector อาจเลย ทำให้มี 2 กรณี :

$\vec{w} = \vec{w} + \Delta w$

$\Delta w = \begin{cases} ๑\ \vec{x} & \text{ถ้าเป็น ตย บวก และ } \vec{w}\cdot\vec{x} < 0 \\ ๒\ -\vec{x} & \text{ถ้าเป็น ตย.ลบ และ } \vec{w}\cdot\vec{x} > 0 \end{cases}$

- คูณด้วย $\underbrace{\text{learning rate}}_{\text{ค่าคงที่น้อยๆ.}}$ เพื่อไม่ให้ปรับค่าไปไกลเหมือนตัวอย่าง

10 Feb : Signal Function

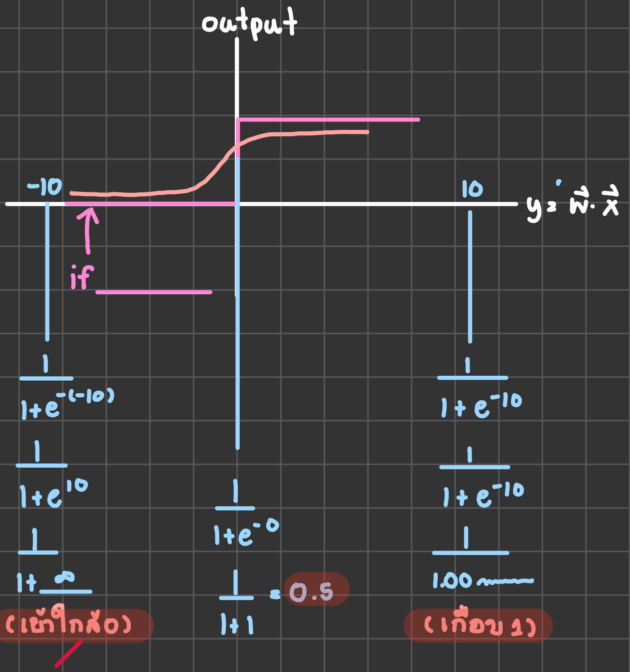


$\vec{W} = \vec{W} + \Delta W$

$$\Delta\vec{W} = \begin{cases} \eta\vec{x} & \text{if } \vec{W}\cdot\vec{x} < 0 \text{ and positive} \\ -\eta\vec{x} & \text{if } \vec{W}\cdot\vec{x} > 0 \text{ and negative} \end{cases}$$

ปรับจนไปลงตัวบางจุด

Signal Function TLN



มากกว่า 0 ออก 1

น้อยกว่า 0 ออก 0

↓

ถ้าคำนวณเยอะๆจะเหนื่อยเลยขอ

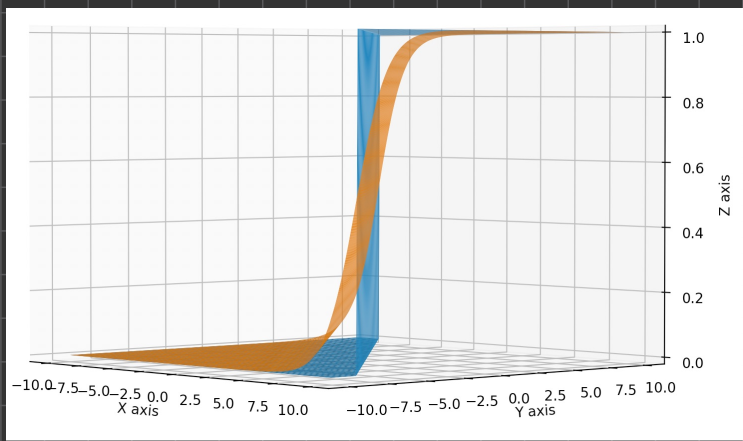
ใช้ Sigmoid $\sigma(y) = \frac{1}{1+e^{-y}}$

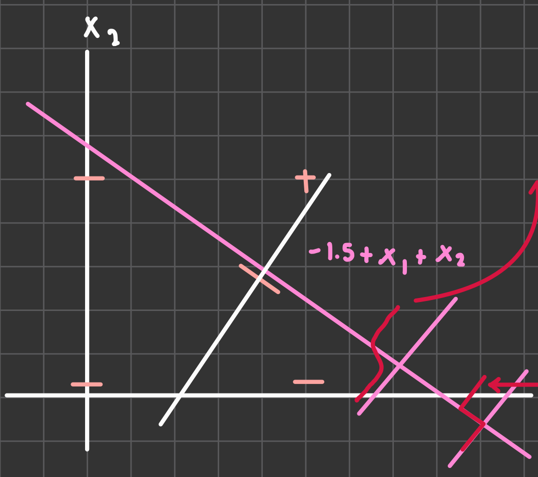
$\sigma(\vec{W}\cdot\vec{x}) = \frac{1}{1+e^{-\vec{W}\cdot\vec{x}}}$

↓

ทำหน้าที่แบบเดียวกัน แต่ sigmoid

ไม่ต้อง if !





error $\in \mathbb{R}$

$\in [0,1]$

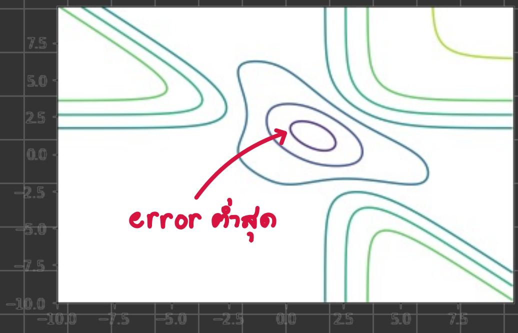
error $\in \{0,1\}$

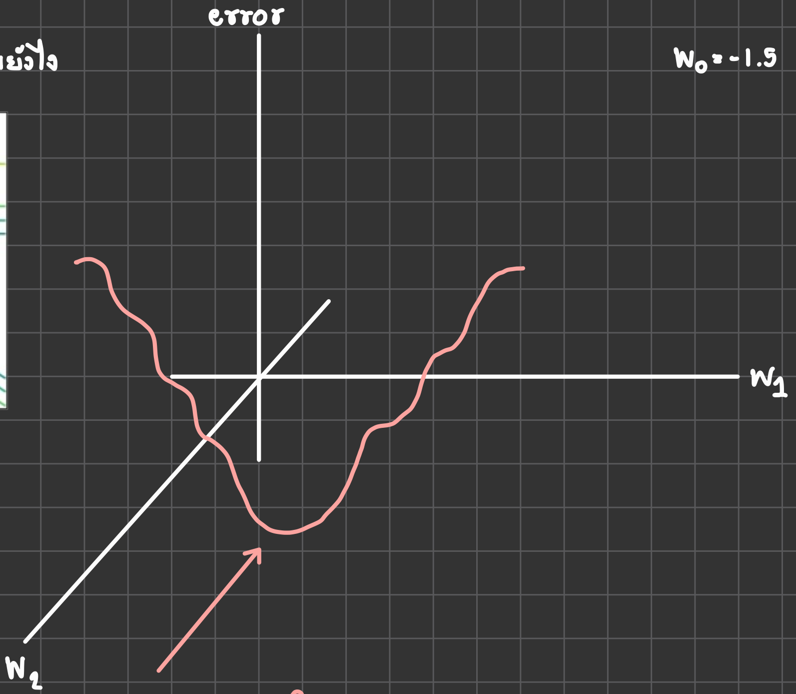
error = $(\text{target} - \text{output})^2$

สมมติว่ามองจาก view นี้เลย

ถ้าเกิดใช้ Sigmoid error surface จะเป็นยังไง

$W_0 = -1.5$





$e = \frac{1}{2}(\text{target}-\text{output})^2$

ถ้า error อยู่ใน form นี้ ชอบมี 1/2 ไว้หักหัว

$\nabla e = \frac{\partial}{\partial w} \cdot \frac{1}{2}(\text{target}-\text{output})^2$

$= \frac{1}{2}\frac{\partial}{\partial w}(\text{target}-\text{output})^2$

$= \cancel{\frac{1}{2}}(\cancel{2})(t-o)\frac{\partial}{\partial w}(t-o)$

target = constant

$= (t-o)\frac{\partial}{\partial w}(\cancel{t}-o)$

$= (t-o)\frac{\partial}{\partial w}(-o)$

↳ output = sigmoid

$= -(t-o)\frac{\partial}{\partial w}\left(\frac{1}{1+e^{-w\cdot x}}\right)$

$\partial(\vec{w}\cdot\vec{x})$ และถ้า diff เทียบตัวมันเองจะได้

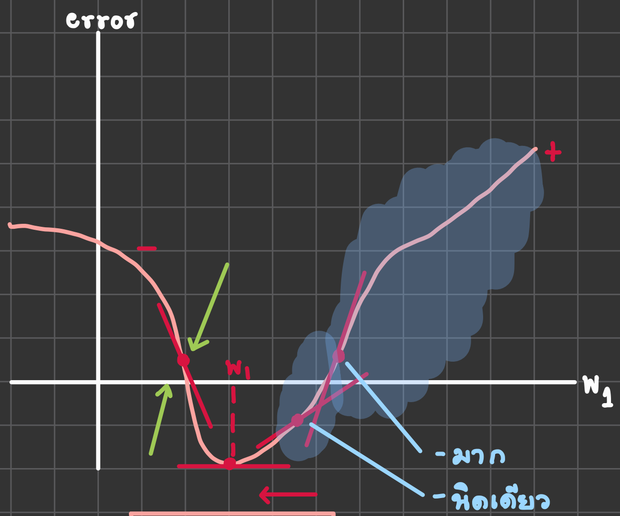
$\frac{\partial \sigma(\vec{w}\cdot\vec{x})}{\partial \vec{w}\cdot\vec{x}} = \sigma(\vec{w}\cdot\vec{x})(1-\sigma(\vec{w}\cdot\vec{x}))$

$\frac{\partial}{\partial u}\frac{1}{1+e^{u}} = \frac{\partial}{\partial u}(1+e^{-u})^{-1}$

$= -\frac{1}{(1+e^{-u})^2}\frac{\partial}{\partial u}e^{-u}$

$= \boxed{+\frac{e^{-u}}{(1+e^{-u})^2}}$

มองจาก w_2 สนใจเฉพาะ w_1



ความชันเป็น (+) มาก ๆ ลด w_1 ให้ลด

$w_1 = w_1 \boxed{-\eta\,\text{ความชัน}}$

$w_1 = w_1 - \eta\,$ความชัน (ความชันติดลบ ; ลบความชันทำให้ค่าของ w_1 เพิ่มขึ้น)

$-\eta\cdot$ ความชันตอนปรับจะได้ขยับน้อยลง

∴ ไม่ว่ามองจาก w ไหนก็เป็นแบบนี้หมด การปรับ weight ปรับเป็น weight ของ vector ได้เลย !

$\vec{w} = \vec{w} - \eta\nabla e$ (ความชัน, gradient ของ error surface)

$\vec{w} = \vec{w} - \eta(t-o)(-o)(1-o)\vec{x} \rightarrow \vec{w} = \vec{w} + \eta(t-o)(o)(1-o)\vec{x}$

chaining

$\frac{\partial\sigma(\vec{w}\cdot\vec{x})}{\partial w} = \frac{\partial\sigma(\vec{w}\cdot\vec{x})}{\partial\vec{w}\cdot\vec{x}} \cdot \frac{\partial\vec{w}\cdot\vec{x}}{\partial\vec{w}}$

$= \sigma(\vec{w}\cdot\vec{x})(1-\sigma(\vec{w}\cdot\vec{x}))\cdot\vec{x}$

$= -O(1-O)\vec{X}$ (เป็นความชัน)

เปลี่ยนเป็น output

$\left(\frac{1}{1+e^{-u}}\right)\left(1-\frac{1}{1+e^{-u}}\right)$

$\left(\frac{1}{1+e^{-u}}\right)\left(\frac{1+e^{-u}-1}{1+e^{-u}}\right) = \frac{e^{-u}}{(1+e^{-u})^2}$

ความชันของ Sigmoid Function (จากคลิป)

$$\sigma(y) = \frac{1}{1+e^{-y}} \qquad y = w_i x_i = \vec{w}\cdot\vec{x}$$

เราจะหา diff sigmoid เทียบกับ y

$$\frac{d}{dy}\sigma(y) = \sigma(y)(1-\sigma(y))$$; ความชันเทียบ y

หา error

$$error = \frac{1}{2}(t-o)^2$$

diff error เทียบ w

$$\frac{de}{d\vec{w}} = \frac{1}{2}\frac{d}{d\vec{w}}(t-o)^2$$

$$= \frac{1}{\cancel{2}}\cancel{2}(t-o)\frac{d}{d\vec{w}}(t-o)$$

ค่าคงที่

$$= (t-o)\left(-\frac{do}{d\vec{w}}\right)$$

$$= -(t-o)\frac{d}{d\vec{w}}\sigma(\vec{w}\cdot\vec{x})$$

$$= -(t-o)\sigma(\vec{w}\cdot\vec{x})(1-\sigma(\vec{w}\cdot\vec{x}))\frac{d}{d\vec{w}}\vec{w}\cdot\vec{x}$$

$$= -(t-o)\sigma(\vec{w}\cdot\vec{x})(1-\sigma(\vec{w}\cdot\vec{x}))\vec{x}$$

$$= -(t-o)(o)(1-o)\vec{x}$$

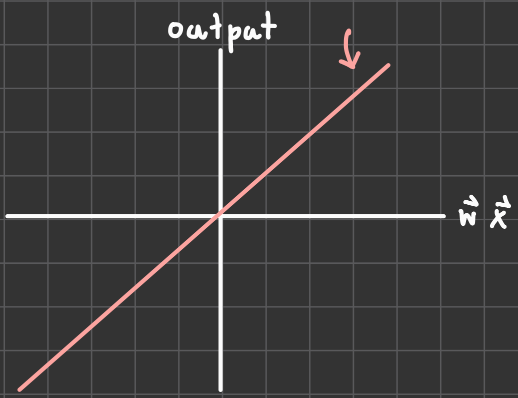
$$\vec{w} \longleftarrow \vec{w} + \Delta\vec{w}$$

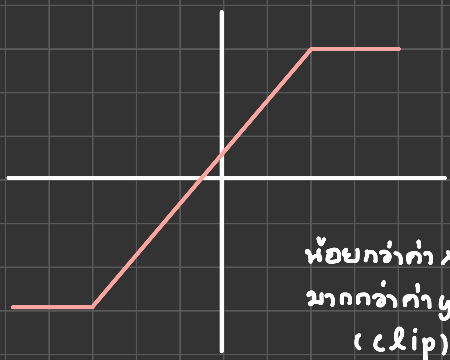
$$\Delta\vec{w} = \eta(t-o)(o)(1-o)\vec{x}$$

Sigmoid node . $\vec{w} = \vec{w} + \Delta\vec{w}$

$$\Delta\vec{w} = \eta\underbrace{(t-o)}_{\text{เกิดจากการคำนวณ error}}\underbrace{(o)(1-o)\vec{x}}_{\text{เกิดจากความชัน}}$$

Linear node





น้อยกว่าค่า x = 0
มากกว่าค่า y = 1
(clip)

$$\begin{aligned}
\text{output} &= \vec{w}\cdot\vec{x} \\
\text{error} &= \frac{1}{2}\times(t-o)^2 \\
\vec{w} &= \vec{w}\cdot\eta\nabla e \\
\nabla e &= \frac{\partial}{\partial\vec{w}}\left(\frac{1}{2}\right)(t-o)^2 \\
&= \frac{1}{\cancel{2}}(\cancel{2})(t-o)\frac{\partial}{\partial\vec{w}}(\overset{0}{\cancel{t}}-o) \\
&= -(t-o)\frac{\partial o}{\partial\vec{w}} \\
&= -(t-o)\cancel{\frac{\partial\vec{w}}{\partial\vec{w}}}\cdot\vec{x} \\
&= -(t-o)\vec{x}
\end{aligned}$$

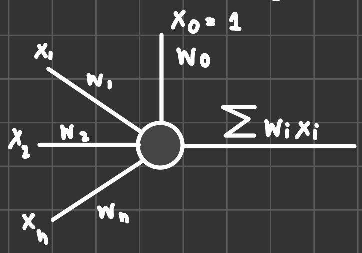
$\vec{w} = \vec{w} + \Delta\vec{w}$

$\Delta\vec{w} = \eta(t-o)\vec{x}$

ข้อสอบ: ให้ activation func มา อยากดูสมการในการปรับ weigth

17 Feb : Weiner Solution / Gaussian

Linear node : $S(y) = y$



$\vec{w} \leftarrow \vec{w} + \Delta w$ ใหม่ เดิม

$\Delta w = \eta (t - o) x \longrightarrow$ train ทีละตัว ในหนส. d_k (target ตัว k ใดๆ)

t_k

$$\varepsilon_k = \frac{1}{2}(t_k - \vec{x}_k^T \vec{w}_k)^2 = \frac{1}{2} e_k^2$$

$$= \frac{1}{2}(t_k^2 - 2t_k \vec{x}_k^T \vec{w}_k + \underbrace{\vec{w}_k^T \vec{x}_k \vec{x}_k^T \vec{w}_k}_{\text{metrix}^2})$$

transpot เพราะอยากให้เป็นสเกล่าร์

expected val

$$\varepsilon = E[\varepsilon_k] = \frac{1}{2} E[t_k^2] - E[t_k \vec{x}_k^T] \vec{w}_k + \frac{1}{2} \vec{w}_k^T E[\vec{x}_k \vec{x}_k^T] \vec{w}_k$$

สเกล่าร์ แทนด้วย P แทนด้วย R column vector

$x_1, x_2 = 2 \times 1$

$$\varepsilon = \frac{1}{2} E[t_k^2] - P\vec{w}_k + \frac{1}{2} \vec{w}^T R w_k$$

หา error surface diff เทียบ w (หา gradient)

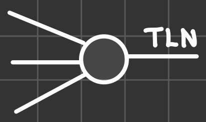
$$\frac{\partial \varepsilon}{\partial \vec{w}} = -P + RW = 0$$

$$W = PR^{-1}$$

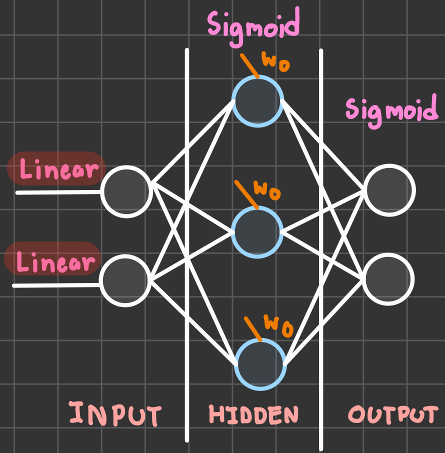
24 Mar: Backpropagation Neural Network

ข้อเสีย train network/neural (TLN) เวลาปรับ w ต้อง if ไม่ใช้ TLN ทำ BNN

$\therefore$ Main node = Sigmoid ที่ให้ผลลัพธ์เป็น
พฤติกรรมแบบเดียวกับ TLN

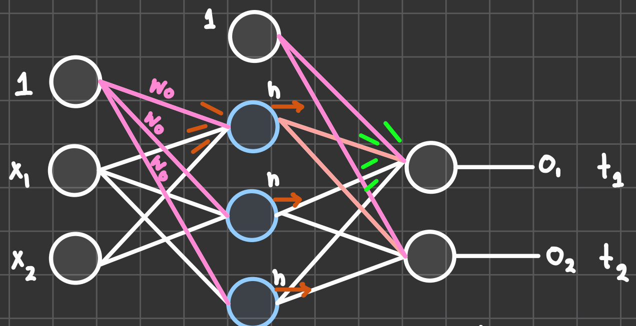


if x positive $\vec{w}\cdot\vec{x} < 0$
if x negative



เข้าอะไรออกแบบนั้น
เป็น signal input แบบนึง

แนว QUIZ BNN !!
1 ถามค.เข้าใจในการปรับ w



$y = \vec{w}\cdot\vec{x}$ $\quad z = \vec{w}\cdot S(y)$
$S(y) = \sigma(\vec{y})$ $\quad \vec{h}$

— h effect ต่อ output ทุก node !

error แต่ละ node สามารถปรับ weight independently

$e = \frac{1}{2}(t-o)^2$

o : output ที่ออกจาก output layer
h : output ที่ออกจาก hidden layer

Constant

$\frac{\partial e}{\partial w_{ho}} = \left(\frac{1}{\cancel{2}}\right)(\cancel{2})(t-o)\frac{\partial}{\partial w_{ho}}(\cancel{t}-o)$

W จาก h → O

$= -(t-o)\frac{\partial}{\partial w}\sigma(w_{ho}\cdot\vec{h})$

$= -(t-o)\boxed{\frac{\partial\,\sigma(\vec{w}_{ho}\cdot\vec{h})}{\partial w_{ho}\cdot\vec{h}}}\;\frac{\partial\,\vec{w}_{ho}\cdot\vec{h}}{\partial w_{ho}}$

Output

Chaining Diff
$\sigma(y)(1-\sigma(y))$

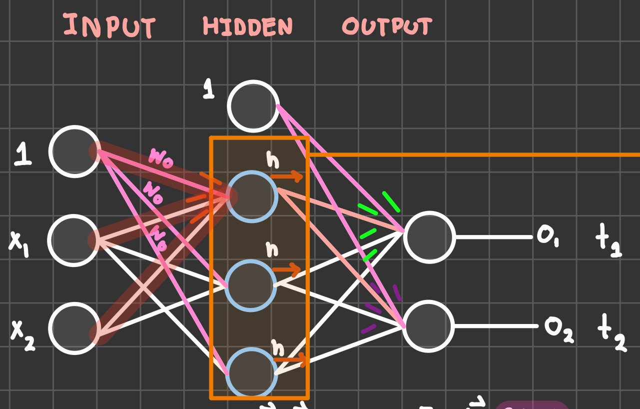
$= -(t-o)(o)(1-o)\vec{h}$

$\vec{w}_{ho} = \vec{w}_{ho} - \eta(-(t-o)(o)(1-o)\vec{h})$

$= \vec{w}_{ho} + \eta\underbrace{(t-o)(o)(1-o)}_{\delta_o}\vec{h}$

δ_o → delta output

$= \vec{w}_{ho} + \eta\delta_o\vec{h}$



$y = \vec{w} \cdot \vec{x}$

$S(y) = \sigma(\vec{y})$

$z = \vec{w} \cdot S(y)$, $S(y) = \vec{h}$

พออยู่ในชั้น hidden ไม่มี target !

– hidden effect ต่อ output ทุก node !
การเปลี่ยนแปลง weight ก่อให้เกิดผล
ต่อ output (O_1, O_2)
∴ error จะเป็น summation ของ output
ทุก node

Weight (input → hidden)

error ที่เกิดจาก w_{ih} $= \sum_o \text{error}_o$

$$= \sum_o \frac{1}{2}(t-o)^2$$

$$\frac{\partial \text{error}}{\partial w_{ih}} = \frac{\partial}{\partial w_{ih}} \sum_o \frac{1}{2}(t-o)^2$$

$$= \frac{1}{2}\sum_o \frac{\partial}{\partial w_{ih}}(t-o)^2$$

$$= \frac{1}{\cancel{2}}(\cancel{2})\sum_o (t-o)\frac{\partial}{\partial w_{ih}}(\cancel{t}-o)$$

constant

$$= \sum_o -(t-o)\frac{\partial o}{\partial w_{ih}}$$

$$= -\sum_o (t-o)\frac{\partial}{\partial w_{ih}}\sigma(w_{ho}\cdot\vec{h})$$

$$= -\sum_o (t-o)\boxed{\frac{\partial\,\sigma(w_{ho}\cdot\vec{h})}{\partial \vec{w}_{ho}\cdot\vec{h}}}\frac{\partial \vec{w}_{ho}\cdot\vec{h}}{\partial w_{ih}}$$

$\sigma(\vec{w}_{ho}\cdot\vec{h})(1-\sigma(\vec{w}_{ho}\cdot\vec{h}))$

output

constant

$$= -\sum_o (t-o)\boxed{(o)(1-o)}\frac{\partial \vec{w}_{ho}\cdot\sigma(\vec{w}_{ih}\cdot\vec{x})}{\partial \vec{w}_{ih}}$$

$$= -\sum_o \delta_o \vec{w}_{ho}\frac{\partial\sigma(\vec{w}_{ih}\cdot\vec{x})}{\partial \vec{w}_{ih}}$$

$$= -\sum_o \delta_o \vec{w}_{ho}\frac{\partial\sigma(\vec{w}_{ih}\cdot\vec{x})}{\partial \vec{w}_{ih}\cdot\vec{x}}\frac{\partial \vec{w}_{ih}\cdot\vec{x}}{\vec{w}_{ih}}$$

output ที่ออกจาก h

$$= -\sum_o \delta_o \vec{w}_{ho}\boxed{h(1-h)\vec{x}}$$

→ Sum กับ output node
แต่เกิดจาก hidden ∴ ไม่จำเป็นจะต้องอยู่ใน Σ

$$= -h(1-h)\vec{x}\sum_o \delta_o \vec{w}_{ho}$$

$\vec{W}_{ih} = \vec{W}_{ih} + \boxed{\eta\left[h(1-h)\sum_{o}\delta_o \vec{W}_{ho}\right]\vec{x}}$ δh (delta ของ node ในชั้น h)

$\vec{W}_{ih} = \vec{W}_{ih} + \eta\delta_h\vec{x}$ (input → hidden)

$\vec{W}_{ho} = \vec{W}_{ho} + \eta\delta_o\vec{h}$ (input → output)

$\vec{W}_i = \vec{W}_i + \eta\delta_i\vec{x}_i$ input ที่วิ่งเข้า node นั้น

Backpropagation Algorithm

Initialize all weights to small random numbers.
Until satisfied, Do

- For each training example, Do
  1. Input the training example to the network and compute the network outputs
  2. For each output unit k

$$\boxed{\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)}$$

  3. For each hidden unit h

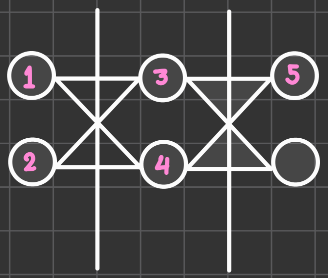
$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

  4. Update each network weight $w_{i,j}$

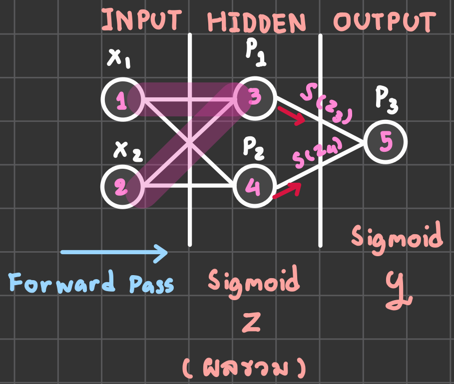
$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

where

$$\Delta w_{i,j} = \eta\delta_j x_{i,j}$$



Backpropagation Neural Network (from clip)



XOR

Forward Pass

Hidden Layer

$$Z_h = \sum_{i=0}^{n} w_{ih} S(x_i)$$

w_{ih} (weight จาก input → hidden)
n : จำนวน node ในชั้น input

$$Z_3 = \sum_{i=0}^{2} w_{i3} S(x_3)$$

$$= w_{03}(1) + w_{13} S(x_1) + w_{23} S(x_2) \quad [\, \vec{w} \cdot \vec{x} \,]$$

$$S(z_h) = \frac{1}{1+e^{-zh}}$$

Output Layer

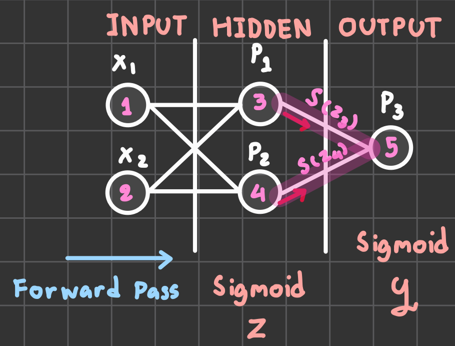
q : จำนวน node ในชั้น hidden

$$y_j = \sum_{h=0}^{q} w_{hj} S(z_h)$$

$$y_5 = \sum_{h=0}^{2} w_{h5} S(z_h)$$

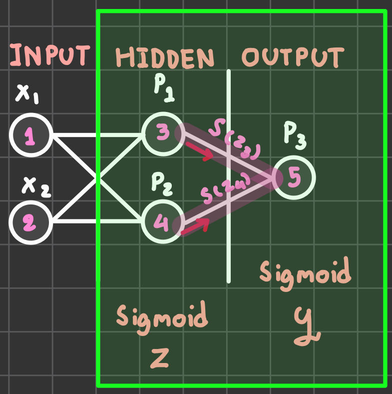
$$y_5 = w_{05}(1) + \underset{(w_{35})}{w_{15}} S(z_1) + \underset{(w_{45})}{w_{25}} S(z_2) \quad [\, \vec{w} \cdot \vec{x} \,]$$

$$S(y_j) = \frac{1}{1+e^{-y_j}}$$



เปลี่ยนตามเลข node

Backward Pass



$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial S(y_j)} \frac{\partial S(y_j)}{\partial y(j)} \frac{\partial y(j)}{\partial w_{hj}} \qquad \boxed{E = \frac{1}{2}(t - S(y_j))^2}$$

w_{hj} = Weight จาก hidden → output

$$\frac{\partial E}{\partial S(y_j)} = \frac{\partial}{\partial S(y_j)} \frac{1}{2}(t - S(y_j))^2$$

$$= \frac{1}{\cancel{2}}(\cancel{2})(t - S(y_j)) \frac{\partial(\overset{\text{Constant}}{\cancel{t}} - S(y_j))}{\partial S(y_j)}$$

$$= (t - S(y_j))(-1)$$

$$= -(t - S(y_j))$$

$$\frac{\partial S(y_j)}{\partial y(j)} = S(y_j)(1 - S(y_j)) \quad \text{———} \quad \frac{d(\sigma(y))}{dy} = \sigma(y)(1 - \sigma(y))$$

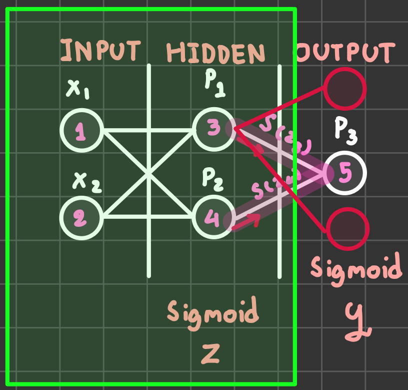
$$\frac{\partial y(j)}{\partial w_{hj}} = \frac{\partial}{\partial w_{hj}} \sum w_{hj} S(z_h) = S(z_h)$$

$$\frac{\partial E}{\partial w_{ho}} = \underbrace{\underbrace{-(t - S(y_j))}_{c}\ S(y)(1 - S(y))}_{\delta_j}\ S(z_h)$$

δ_j (delta node ในชั้น output / node$_j$)

$$= -\delta_j S(z_h)$$

Input Layer (ไม่มี error ที่ชัดเจนเหมือน output)



z_h 1 ตัวส่งผลกระทบถึง node ทั้งหมดใน output layer

$$\frac{\partial E}{\partial w_{ih}} = \frac{\partial E}{\partial \delta(z_h)} \frac{\partial \delta(z_h)}{\partial z_h} \frac{\partial z_h}{\partial w_{ih}}$$

$$\frac{\partial E}{\partial \delta(z_h)} = \sum_{j=1}^{p} \left(\frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \delta(z_h)} \right)$$

p: เป็นจำนวน node ใน output ทั้งหมด

$$\frac{\partial \delta(z_h)}{\partial z_h} = \delta(z_h)(1-\delta(z_h))$$

$$\frac{\partial z_h}{\partial w_{ih}} = \frac{\partial}{\partial w_{ih}} w_{ih}\,\delta(x_i) = \delta(x_i)$$

$$\frac{\partial E}{\partial w_{ih}} = \sum_{j=1}^{p} \left(\boxed{\frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \delta(z_h)}} \right) \delta(z_h)(1-\delta(z_h))\,\delta(x_i)$$

$$= \sum_{j=1}^{p} \left(\frac{\partial E}{\partial \delta(y_j)} \frac{\partial \delta(y_j)}{\partial y_j} \frac{\partial y_j}{\partial \delta(z_h)} \right) \delta(z_h)(1-\delta(z_h))\,\delta(x_i)$$

$$\frac{\partial E}{\partial \delta(y_j)} = -e_j \qquad \frac{\partial S(y_j)}{\partial y_j} = S(y_j)(1-S(y_j)) \qquad \frac{\partial y_j}{\partial \delta(z_h)} = \partial w_{hj}\,\delta(z_h) = w_{hj}$$

$$= \sum_{j=1}^{p} (\underbrace{-e_j S(y_j)(1-S(y_j))}_{\delta_j} w_{hj})\,\delta(z_h)(1-\delta(z_h))\,\delta(x_i)$$

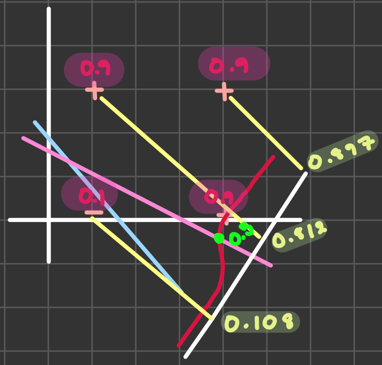
$$= \underbrace{\underbrace{\sum_{j=1}^{p} (\delta_j w_{hj})}_{e_h} \delta(z_h)(1-\delta(z_h))}_{\delta_h} \delta(x_i)$$

$$\delta_h = e_h\,\delta(z_h)(1-\delta(z_h))$$

$$\frac{\partial E}{\partial w_{ih}} = -\delta_h x_i$$

31 Mar: Support Vector Machine (SVM)

Brief concept



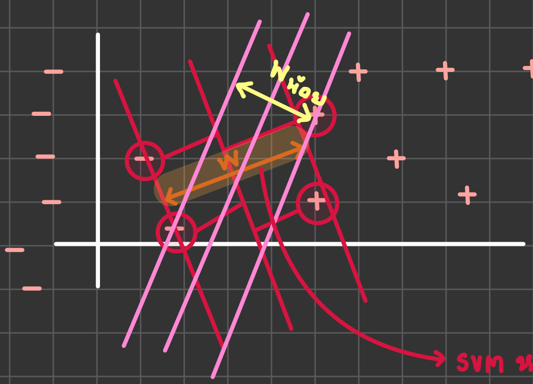
ในบางทีอาจไม่ได้วางตัวตาม 0.9(+), 0.1(-1) เป๊ะ (มี error อยู่)

↓

∴ จะมีการขยับเส้นสีชมพูให้มี error ต่ำสุด
เป็นเหตุผลว่า sigmoid วางในตำแหน่งเดิมทุกครั้ง โดยประมาณ

TLN
แบ่ง sample ได้ปั๊บหยุด
(ไม่ปรับ weight)

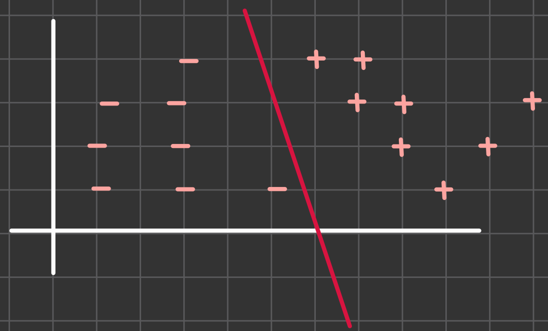
สมมติว่ามี sample (+,-) วางกระจายตัว



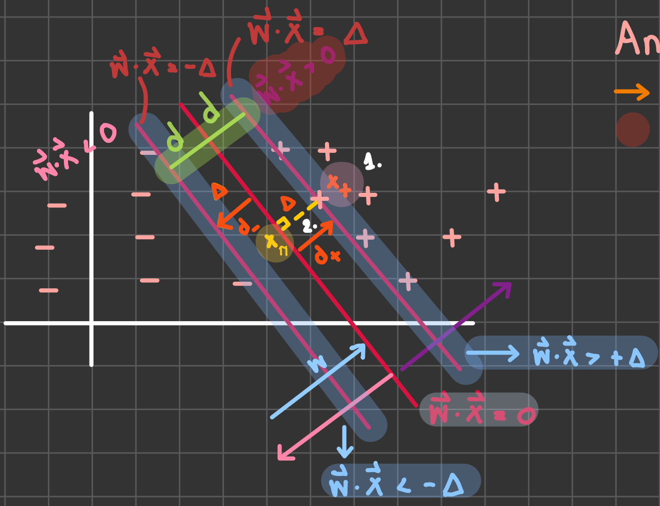
∴ เส้นแบ่งที่ดีที่สุด
คือเส้นที่วางแล้วได้ W ที่กว้าง ที่สุด

→ SVM หา $W = W_0X_0 + W_1X_1 + W_2X_2 + \dots + W_nX_n$ (2-dimension)

From clip



คำถาม: เส้นตรงใดควรจะแยกตัวอย่างได้ดีที่สุด?



Answer

→ แตะ sample (+,-) ได้มากที่สุด เส้นกลางวางตัวตรงกลางพอดี

อยากให้ไม่แบ่งด้วยเส้นกลางแต่อยากให้แบ่งด้วยเส้น

1. sample (+) ที่แตะเส้น $\vec{w}\cdot\vec{x} > \Delta$ แทนด้วย x_+
2. ลาก projection ⊥ ได้ x_π

$d_+ = \lVert \vec{x}_+ - \vec{x}_\pi \rVert$ $w\cdot x_+ = \Delta$ $w\cdot x_+ = 1$ —① $w\cdot x_\pi = 0$ —②	①-②: $\vec{w}\cdot(\vec{x}_+ - \vec{x}_\pi) = 1$ —③ vector 2 ตัว dot กันได้ $\lVert\vec{w}\rVert \lVert \vec{x}_+ - \vec{x}_\pi \rVert = 1$ $\lVert \vec{x}_+ - \vec{x}_\pi \rVert = \frac{1}{\lVert\vec{w}\rVert}$

margin (d) $= \frac{2}{\lVert\vec{w}\rVert}$

อยากทำให้สมการ $\vec{w}\cdot\vec{x} = \Delta$ ง่ายขึ้น !

$\vec{w}\cdot\vec{x} > \Delta \quad d = 1$
$\vec{w}\cdot\vec{x} < -\Delta \quad d = -1$

→ $d(\vec{w}\cdot\vec{x}) > \Delta$
$d(\vec{w}\cdot\vec{x}) > \Delta$

เอา $d\cdot(\vec{w}\cdot\vec{x})$

ตัวระบุ class

ในกรณีที่ sample (+)

ในกรณีที่ sample (−)

→ $d(\vec{w}\cdot\vec{x}) > \Delta$
$\frac{d(\vec{w}\cdot\vec{x})}{\Delta} > 1$

เอา Δ ไปหาร ตรงไหนก็ได้ constant

→ $d(\vec{w}\cdot\vec{x}) > 1$

maximize margin (ทำให้ถนน [margin] มี size มากสุด)

maximize $\frac{2}{\lVert\vec{w}\rVert}$ ⇒ minimize $\lVert\vec{w}\rVert$!! ด้วยวิธี Lagrange

optional

minimize $\frac{1}{2}\lVert w\rVert^2$, constraint $d_i(\vec{w}\cdot\vec{x}_i) \geq 1$
$d_i(\vec{w}\cdot\vec{x}_i) \geq 0$

$$L_p(w,\lambda) = \frac{1}{2}\lVert\vec{w}\rVert^2 - \sum_{i=1}^{\alpha} \lambda_i \left[d_i(\vec{w}\cdot\vec{x}) - 1\right]$$

$\vec{w}\cdot\vec{x}$ → $\vec{w}\cdot\vec{x} + w_0$

$$\frac{\partial L_p(w,\lambda)}{\partial w} = \vec{w} - \sum_{i=1}^{\alpha} \lambda_i (d_i\cdot\vec{x}_i)$$

$$= \vec{w} - \sum_{i=1}^{\alpha} \lambda_i (d_i\cdot\vec{x}_i) = 0 \quad \text{(ความชัน = 0)}$$

$$\vec{w} = \sum_{i=1}^{\alpha} \lambda_i d_i\cdot\vec{x}_i$$

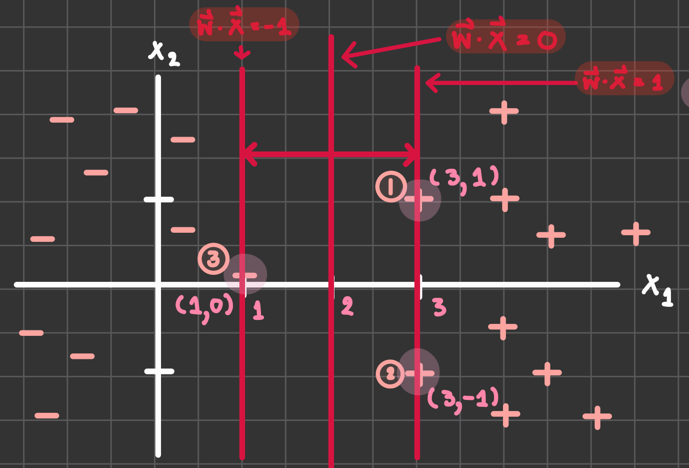
$$\boxed{\vec{w} = \sum_{i=1}^{\alpha} \lambda_i d_i\cdot\vec{x}_i}$$

$d_i(\vec{w}\cdot\vec{x} + w_0) - 1$

$\frac{\partial}{\partial w_0}$

$$\frac{\partial L_p(w,w_0,\lambda)}{\partial w_0} = \sum_{i=1}^{\alpha} \lambda_i d_i = 0$$

$$\boxed{\sum_{i=1}^{\alpha} \lambda_i d_i = 0}$$



ตัวอย่างบนเส้น Support Vector

$$\text{ตัวอย่างบวก} = \left\{ \begin{array}{c} 1 \\ \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} \end{array}, \begin{array}{c} 2 \\ \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix} \end{array} \right\} \qquad \text{ตัวอย่างลบ} = \left\{ \begin{array}{c} 3 \\ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \end{array} \right\}$$

อย่าลืม x_0

$$\vec{w} = \sum_{i=1}^{3} \lambda_i d_i \vec{x}_i$$

$$\vec{w} = \lambda_1 (1) \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} + \lambda_2 (1) \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix} + \lambda_3 (-1) \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

1 สำหรับตย.บวก
-1 สำหรับตย.ลบ

$$\vec{w} = \begin{bmatrix} \lambda_1 \\ 3\lambda_1 \\ \lambda_1 \end{bmatrix} + \begin{bmatrix} \lambda_2 \\ 3\lambda_2 \\ -\lambda_2 \end{bmatrix} + \begin{bmatrix} -\lambda_3 \\ -\lambda_3 \\ 0 \end{bmatrix}$$

$$\vec{w} = \begin{bmatrix} \lambda_1 + \lambda_2 - \lambda_3 \\ 3\lambda_1 + 3\lambda_2 - \lambda_3 \\ \lambda_1 - \lambda_2 \end{bmatrix}$$

$$\vec{w} \cdot \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} = 1 \quad \Bigg| \quad \vec{w} \cdot \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix} = 1 \quad \Bigg| \quad \vec{w} \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = -1$$

1.

$$\begin{bmatrix} \lambda_1 + \lambda_2 - \lambda_3 \\ 3\lambda_1 + 3\lambda_2 - \lambda_3 \\ \lambda_1 - \lambda_2 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} = 1$$

$$\begin{bmatrix} \lambda_1 + \lambda_2 - \lambda_3 \\ 3\lambda_1 + 3\lambda_2 - \lambda_3 \\ \lambda_1 - \lambda_2 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix} = 1$$

$$\begin{bmatrix} \lambda_1 + \lambda_2 - \lambda_3 \\ 3\lambda_1 + 3\lambda_2 - \lambda_3 \\ \lambda_1 - \lambda_2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = -1$$

1. $\lambda_1+\lambda_2+\lambda_3+9\lambda_1+9\lambda_2-3\lambda_3+\lambda_1-\lambda_2 = 1 \longrightarrow 11\lambda_1+9\lambda_2-4\lambda_3 = 1$

2. $\lambda_1+\lambda_2-\lambda_3+9\lambda_1+9\lambda_2-3\lambda_3-\lambda_1+\lambda_2 = 1 \longrightarrow 9\lambda_1+11\lambda_2-4\lambda_3 = 1$

3. $\lambda_1+\lambda_2-\lambda_3+3\lambda_1+3\lambda_2-\lambda_3 = -1 \longrightarrow 4\lambda_1+4\lambda_2-2\lambda_3 = -1$

1. $- 2 \times$ 3. $\quad 3\lambda_1+\lambda_2 = 3$ —— 4

2. $- 2 \times$ 3. $\quad \lambda_1+3\lambda_2 = 3$ —— 5

4 $- 3 \times$ 5 $\quad -8\lambda_2 = -6$

$$\lambda_2 = \frac{3}{4} = 0.75$$

แทนลง 4 จะได้

$$3\lambda_1+\frac{3}{4} = 3$$

$$3\lambda_1 = 2.25$$

$$\lambda_1 = 0.75$$

แทนลง 1 จะได้

$$11(0.75)+9(0.75)-4\lambda_3 = 1$$

$$15-4\lambda_3 = 1$$

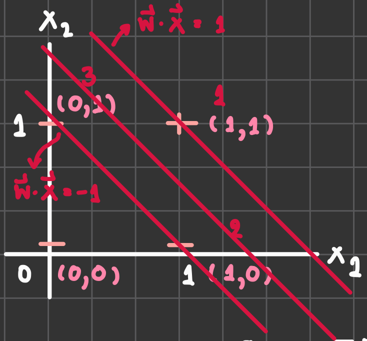
$$4\lambda_3 = -14$$

$$\lambda_3 = 3.5$$

$$\vec{w} = \begin{bmatrix} \lambda_1+\lambda_2-\lambda_3 \\ -3\lambda_1+3\lambda_2-\lambda_3 \\ \lambda_1-\lambda_2 \end{bmatrix} = \begin{bmatrix} 0.75+0.75-3.5 \\ 3(0.75)+3(0.75)-3.5 \\ 0.75-0.75 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

$$w_0+w_1x_1+w_2x_2 \rightarrow \boxed{\begin{aligned} -2+x_1 &= 0 \\ x_1 &= 2 \end{aligned}}$$

ตัวอย่าง



AND TABLE ใช้ SVM ได้ $\vec{w}$ = ?

ตัวอย่างบวก = $\left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$ ตัวอย่างลบ = $\left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right\}$

จาก $\vec{w} = \sum_{i=1}^{4} \lambda_i d_i \vec{x}_i$

$$\vec{w} = \lambda_1(1)\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \lambda_2(-1)\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \lambda_3(-1)\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\vec{w} = \begin{bmatrix} \lambda_1 \\ \lambda_1 \\ \lambda_1 \end{bmatrix} + \begin{bmatrix} -\lambda_2 \\ -\lambda_2 \\ 0 \end{bmatrix} + \begin{bmatrix} -\lambda_3 \\ 0 \\ -\lambda_3 \end{bmatrix}$$

$$\vec{w} = \begin{bmatrix} \lambda_1 - \lambda_2 - \lambda_3 \\ \lambda_1 - \lambda_2 \\ \lambda_1 - \lambda_3 \end{bmatrix}$$

$$\vec{w} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 1 \quad \Big| \quad \vec{w} \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = -1 \quad \Big| \quad \vec{w} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = -1 \quad \Big| \quad \vec{w} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = -1$$

$$\begin{bmatrix} \lambda_1 - \lambda_2 - \lambda_3 \\ \lambda_1 - \lambda_2 \\ \lambda_1 - \lambda_3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \lambda_1 - \lambda_2 - \lambda_3 + \lambda_1 - \lambda_2 + \lambda_1 - \lambda_3$$
$$= 3\lambda_1 - 2\lambda_2 - 2\lambda_3 = 1 \quad \text{①}$$

$$\begin{bmatrix} \lambda_1 - \lambda_2 - \lambda_3 \\ \lambda_1 - \lambda_2 \\ \lambda_1 - \lambda_3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \lambda_1 - \lambda_2 - \lambda_3 + \lambda_1 - \lambda_2$$
$$= 2\lambda_1 - 2\lambda_2 - \lambda_3 = -1 \quad \text{②}$$

$$\begin{bmatrix} \lambda_1 - \lambda_2 - \lambda_3 \\ \lambda_1 - \lambda_2 \\ \lambda_1 - \lambda_3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \lambda_1 - \lambda_2 - \lambda_3 + \lambda_1 - \lambda_3$$
$$= 2\lambda_1 - \lambda_2 - 2\lambda_3 = -1 \quad \text{③}$$

② − ③ $\quad -\lambda_2 + \lambda_3 = 0$

$\lambda_2 = \lambda_3$

แทนใน ① จะได้

$3\lambda_1 - 4\lambda_2 = 1$ ④

แทนใน ② จะได้

$2\lambda_1 - 3\lambda_2 = -1$ ⑤

(4) $\times$ 2 $\quad 6\lambda_1 - 8\lambda_2 = 2$ ——— (6)

(5) $\times$ 3 $\quad 6\lambda_1 - 9\lambda_2 = 3$ ——— (7)

(6) $-$ (7) $\quad \lambda_2 = 5$

$\lambda_3 = 5$

แทนใน (4) ได้ $\lambda_1 = 7$

$$\vec{w} = \begin{bmatrix} \lambda_1 - \lambda_2 - \lambda_3 \\ \lambda_1 - \lambda_2 \\ \lambda_1 - \lambda_3 \end{bmatrix} = \begin{bmatrix} 7-5-5 \\ 7-5 \\ 7-5 \end{bmatrix} = \begin{array}{c} \begin{bmatrix} -3 \\ 2 \\ 2 \end{bmatrix} \end{array} \begin{array}{c} w_0 \\ w_1 \\ w_2 \end{array}$$

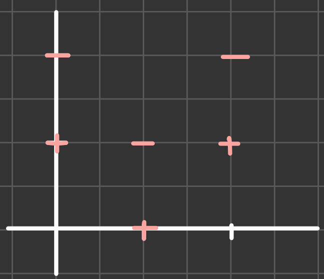
$\therefore -3 + 2x_1 + 2x_2 = 0$

$\boxed{-1.5 + x_1 + x_2 = 0}$

$$\vec{w} = \begin{bmatrix} -3 \\ 2 \\ 2 \end{bmatrix} \begin{bmatrix} -1.5 \\ 1 \\ 1 \end{bmatrix}$$

Kernal

บ่อยครั้งที่เราไม่สามารถแยก sample ได้ เราก็จะใช้ idea ของการแปลงให้อยู่ใน space หึ่ง



$\Phi(x)$ map(x)

$$\Phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

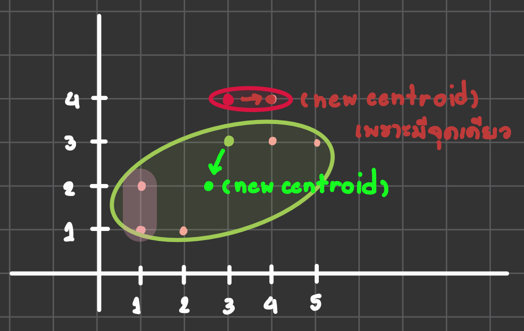
$$+ = \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right\}$$

$$- = \left\{ \begin{bmatrix} 0 \\ 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right\}$$

plot ทำให้จุดกระจายตัว ทำให้เส้นตัดได้ !

7 Apr: Self Organizing Feature Map (SOM)

K-Means Part



K-Mean : mean ของ k ตัวโดยที่ต้องกำหนดก่อนว่ามีกี่ cluster (ระบุค่า k ก่อน) สมมติให้ k = 2 เราต้องระบุ centroid (จุดกลางของ cluster เริ่มต้น) • (3,4) • (3,3)

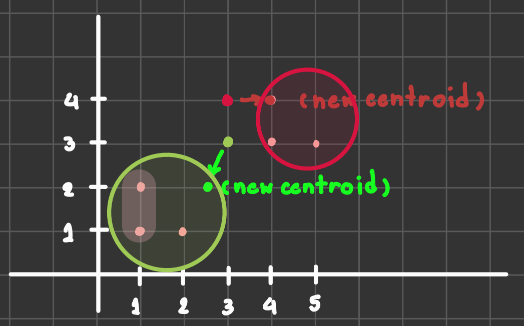
1. cluster โดยการวัดระยะ
   จุด (4,4) ใกล้จุด • (1) มากกว่า • ($\sqrt{2}$)
   ∴ ให้จุด (4,4) อยู่กลุ่มสีแดง •

2. วิธีสร้าง centroid อันใหม่คือหา mean x ในสีเขียว, สีแดง

$$\left(\frac{1+1+2+4+5}{5}, \frac{1+1+2+3+3}{5}\right)$$

● 1;2 ตัว

(2.6, 2)



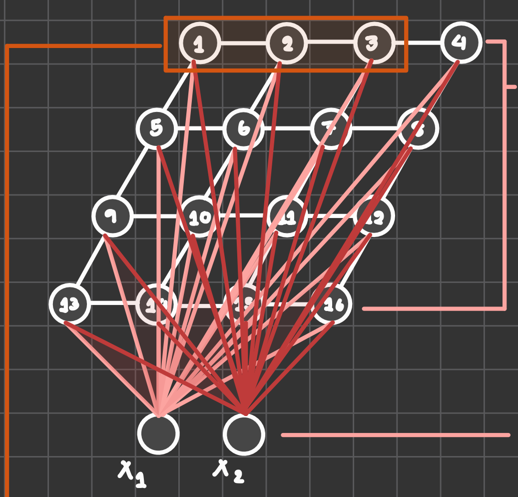
centroid ใหม่จะเป็น

$$\left(\frac{1+1+2}{3}, \frac{1+1+2}{3}\right) = (1.33, 1.33)$$

$$\left(\frac{4+4+5}{3}, \frac{3+3+4}{3}\right) = (4.33, 3.33)$$

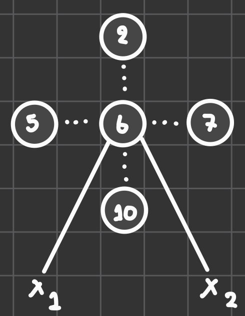
∴ สมาชิกไม่เปลี่ยน, centroid ไม่เปลี่ยน เลิกทำ K-Mean!

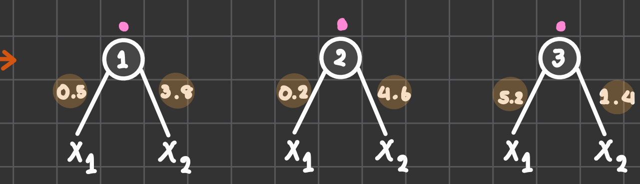
Structure of SOM



Kohonen Layer
16 nodes มีความสัมพันธ์แบบ neighbourhood
(1 เป็นเพื่อนบ้านกับ 2, 5)
neighbourhood จะมีพฤติกรรมการปรับ weight เหมือนกัน

Input Layer (ไม่มี x_0!)
x_1, x_2 เชื่อมไปที่ Kohonen ทุกอัน

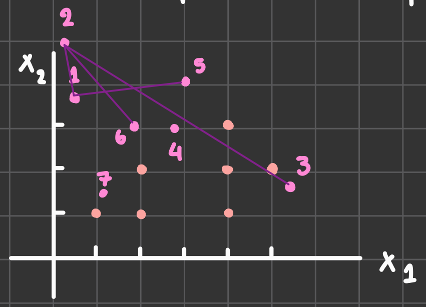




● Weight

แล้วโครงสร้างนี้แทนอะไร?

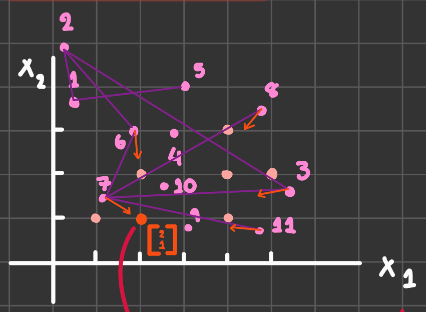
สมมติมี sample กระจายใน space



จะมี node (·) 16 points และจะมี neighbourhood (—)

SOM Training Concept

เอา structure มา classify sample โดยการหยิบ sample มาทีละตัว (·) แล้วดูว่า node ไหนใกล้สุด (winner node)



- ใกล้ node 7 สุด โดยที่การปรับ weight นั้น

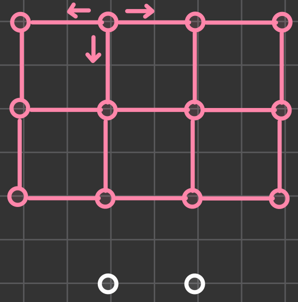
∴ ปรับ winner node ให้ใกล้ input node (·)
และปรับ neighbourhood ให้ใกล้ input node ด้วย

$\vec{W}_w = \vec{W} + \eta(\vec{X} - \vec{W}_w)$ | $\vec{W}_n = \vec{W}_n + 0.7\eta(\vec{X} - \vec{W}_n)$

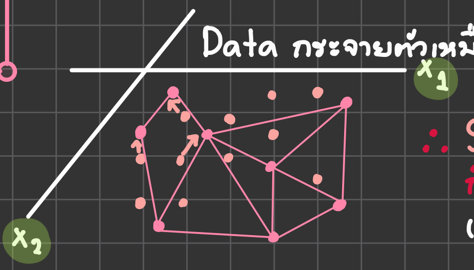
(ปรับแบบไม่เท่ากัน)

21 Apr: SOM, CNN, LSTM

SOM

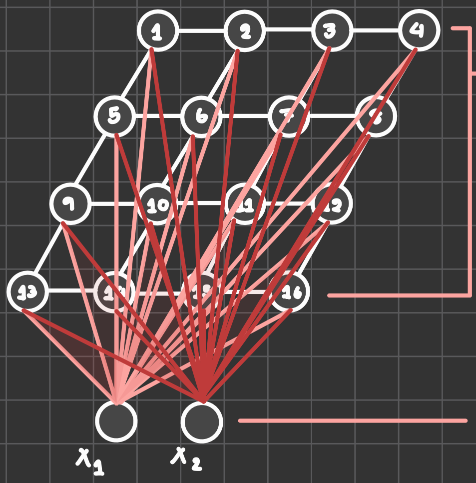


Kohonen Layer : ความพยามยามกระจายตัวครอบคลุม Sample
ไอเดีย: ต้องกำหนดขนาดตุ่มของแห (กำหนดเผื่อไว้ได้)

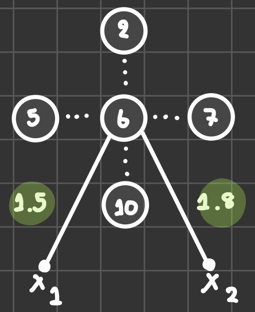


Data กระจายตัวเหมือนตอนเป็น K- Mean

∴ Sample(·) อยู่ใกล้ตุ่ม(·) ไหน ก็เป็นสมาชิก cluster กลุ่มนั้น ๆ
(บางตุ่มก็อาจไม่มี สมาชิก)



Kohonen Layer
มีความสัมพันธ์แบบ neighbourhood ชัดเจน



Input Layer (ไม่มี x_0 !)
x_1, x_2 เชื่อมไปที่ Kohonen ทุกอัน

∴ สามารถใช้ node(6) และ weight (x_1, x_2) ระบุ position ของ node ที่ลง space ของ x,y ได้ !
(เหมือนค่าใน centroid)

ไอเดียการใช้ SOM จัดกลุ่ม

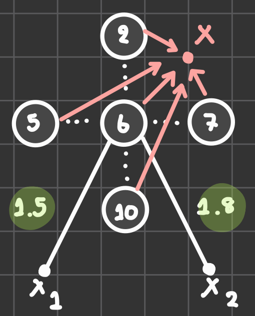
1. Random pick a data point
2. find the winning node in Kohenen layer
3. adjust weight of the wining node towards the datapoint in 1

   $\vec{w} = \vec{w} + \eta(\vec{x} - \vec{w})$ —— ปรับ weight พุ่งเข้าหา X

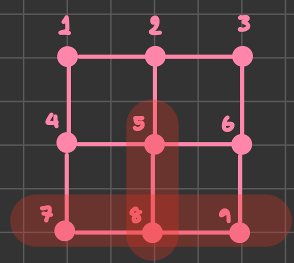
4. adjust weight of neigbours of the winner node

   $\vec{w}_n = \vec{w}_n + \alpha\eta(\vec{x} - \vec{w})$ —— adjust ด้วย co-efficence (α alpha)

สมมติ node 6 ชนะ !



สมมติ Sample



สุ่มได้(5,3) และ wining node คือ node 8 ปรับ weight (5,7,9)

node 8

$$\begin{bmatrix} 4 \\ 2 \end{bmatrix} + \eta\left(\begin{bmatrix} 5 \\ 3 \end{bmatrix} - \begin{bmatrix} 4 \\ 2 \end{bmatrix}\right), \quad \eta = 0.1$$

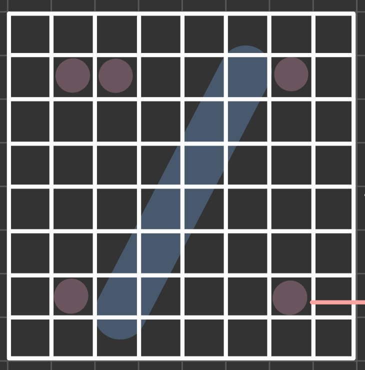
$$\begin{bmatrix} 4 \\ 2 \end{bmatrix} + 0.1\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4.1 \\ 2.1 \end{bmatrix}$$

node 9

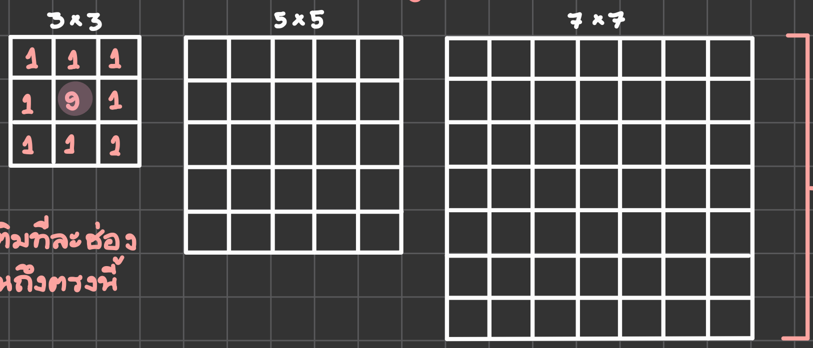
$$\begin{bmatrix} 4 \\ 4 \end{bmatrix} + \eta\left(\begin{bmatrix} 5 \\ 3 \end{bmatrix} - \begin{bmatrix} 4 \\ 4 \end{bmatrix}\right)$$

$$\begin{bmatrix} 4 \\ 4 \end{bmatrix} + 0.1\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 4.1 \\ 3.9 \end{bmatrix}$$

CNN : Feed forward



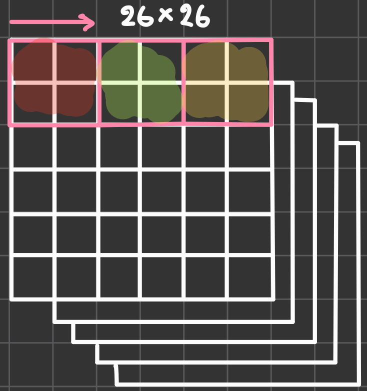
เอา mask ลาก : มีตัวเลข weight ประจำช่อง !



เติมที่ละช่อง จนถึงตรงนี้

1. การทำ convolute โดยเอา kernel ไปวางทาบข้อมูลแล้วเอา weight คูณกับค่าด้านในข้อมูล

$28 \times 28 = 784$



2. เอา 5 kernel มาทาบ (random weight)

มี data ทั้งหมด

$26 \times 26 \times 5 = 3380$

(เยอะเกินไป)

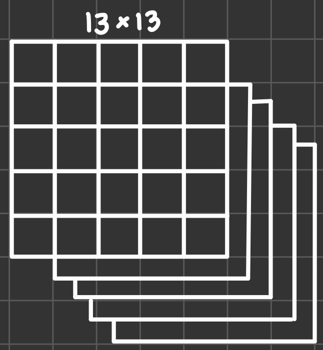
3. การลดขนาดโดยการทำ pooling

- max pooling
- sum

(ตอนขยับ strike จะไม่ทับของเดิม)

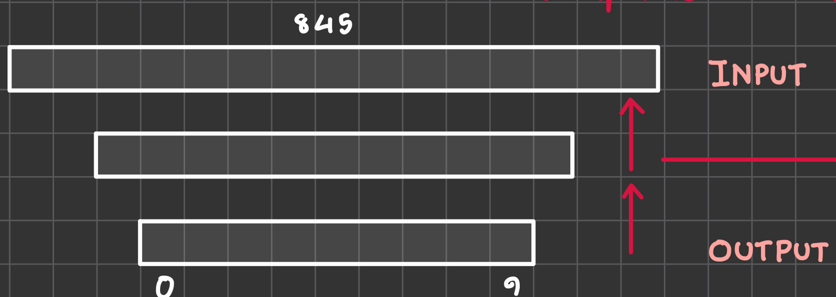
ในช่อง 4×4 ใคร max ส่งไปต่อ

$13 \times 13 \times 5 = 845$



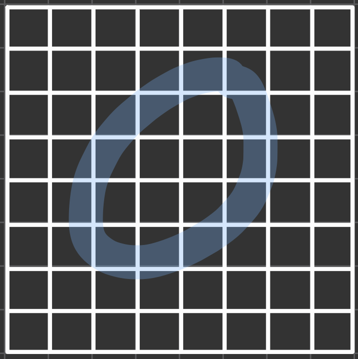
4 ทำ convolute, pooling ต่อ

5. จังหวะสดท้ายจะมีการแพร่ให้แบน ไปเป็น input node ในชั้น backpropagation

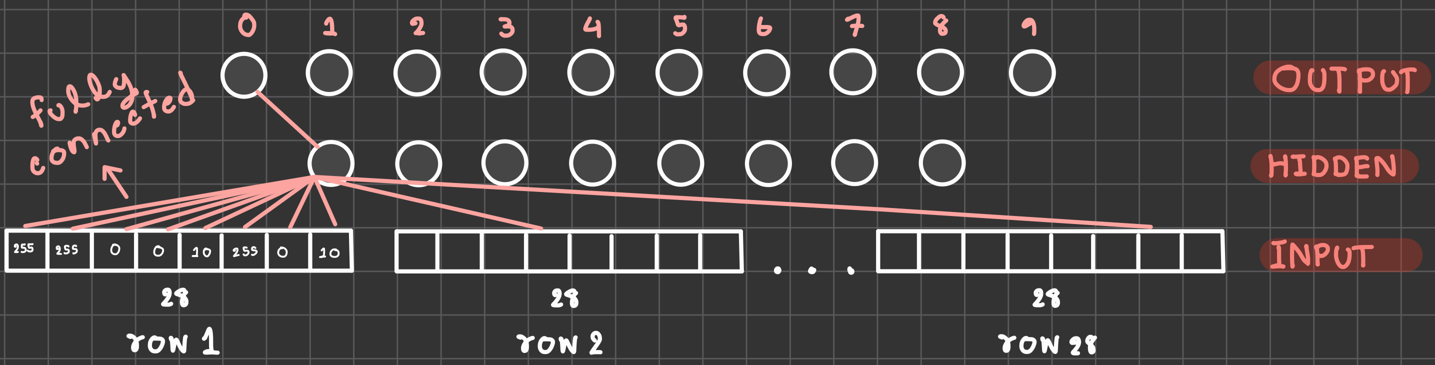


สร้าง delta แล้วปรับย้อน weight ใน kernel จะถูกปรับด้วย

- sum : delta กระจายทุก node

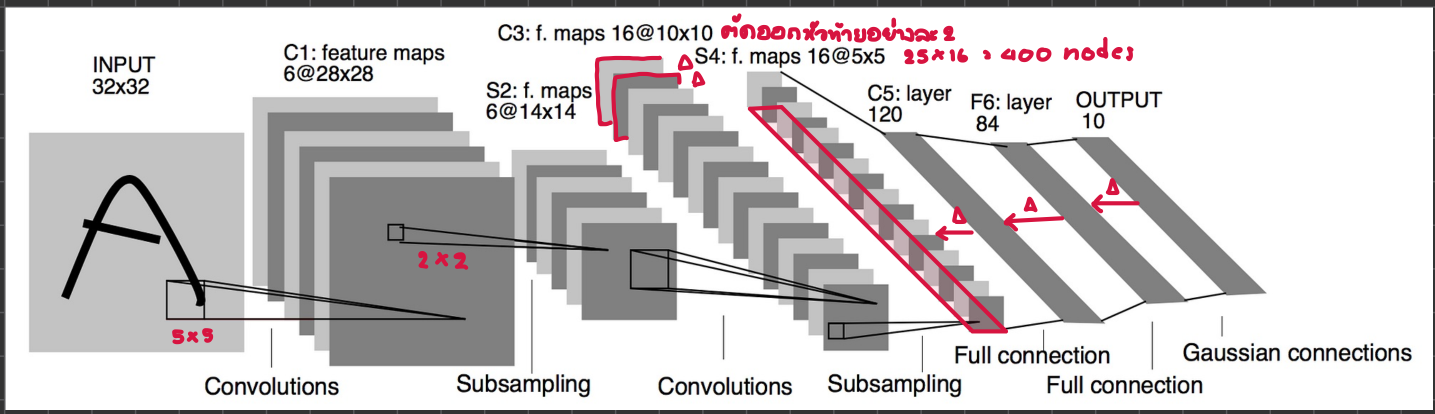


MNIST

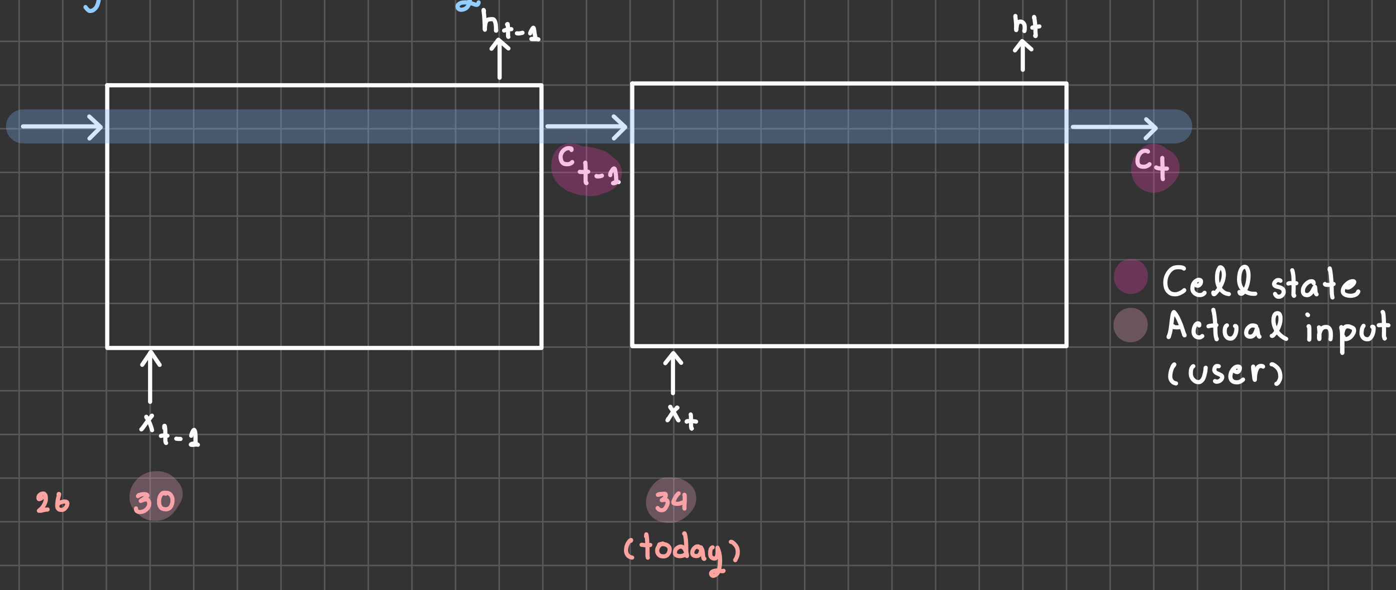


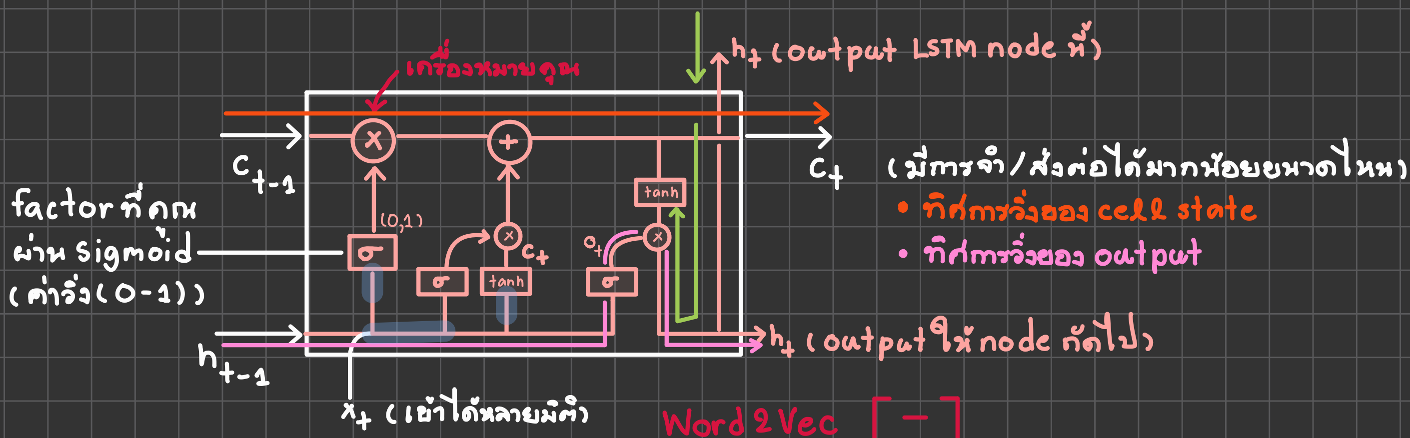
LENET

if 255 = black and 0 = white



Long Short-Term Memory (LSTM)





$[h_{t-1}, x_t]$
เอาเวกเตอร์มาต่อกัน

Word 2 Vec
Single layer perceptron