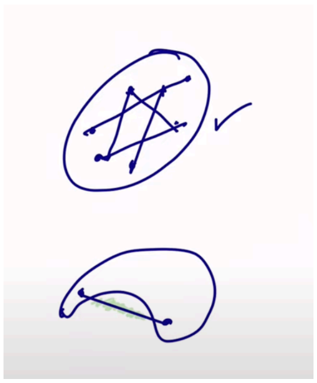
1. Linearly Separable

In classification problem using perceptron-based neural network, the data is needed to linearly separable

Linearly separable is when you can draw a smallest *convex* that could group all the same class of items in a way that each convex doesn't touch with each other

Convex is a shape a line can't go through the shape if you draw the line to any 2 point inside it.



2. Dichotomies and Capacity of a Perceptron

General Position

General position mean in *n* dimension space, every points in the system (*p* points) aren't placed in a way that there's a hyperplane of *n-1* dimension could overlapped with *n+1* or more points

EX. in 2D space, the system has general position properties if you can't draw a line (1D hyperplane) that goes through 3 or more points

Capacity: how good you can binary-separated the preset points in the system.

Ex: If there's 4 points in 2D space, there's maximum of 2^4 ways to binary-separated the points using 1D hyperplane, if you could only do 14, then the capacity is 14/16

If there's n+1 points or less in n dimension space, it's guarantee that the capacity will be 1 in any arrangement

P(p,n) is capacity

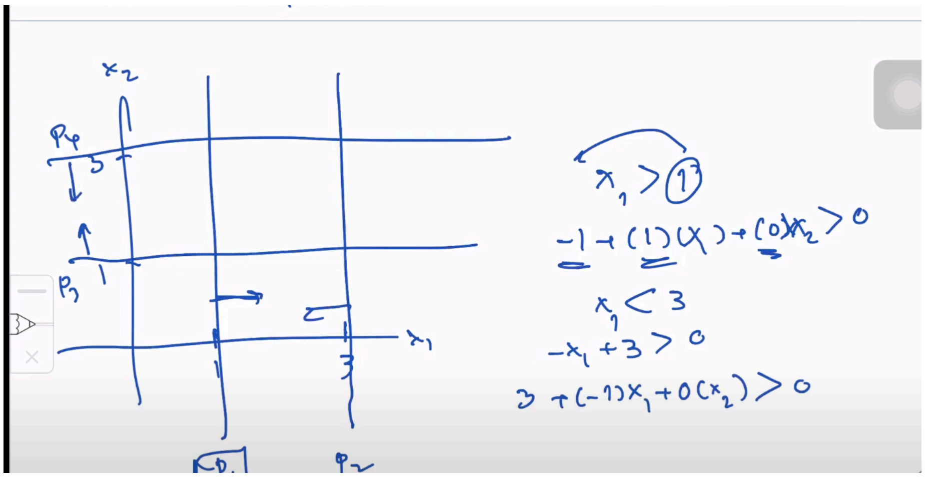
L(p,n) is something

$$P(p,n) = \frac{L(p,n)}{2^p} = \begin{cases} 2^{1-p} \sum_{i=0}^{n} \binom{p-1}{i} & \begin{matrix} n < p-1 \\ p > n+1 \end{matrix} \\ 1 & \underline{p \leq n+1} \end{cases}$$

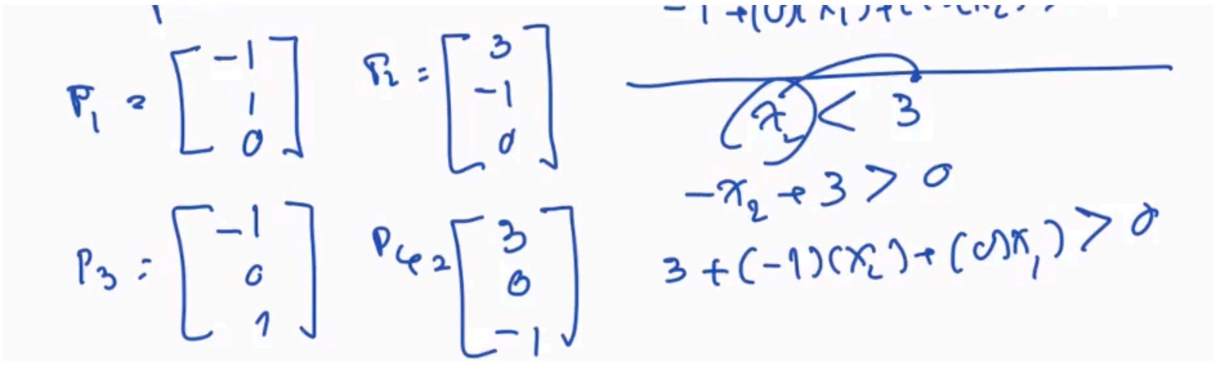
3. Multilayer Perceptron

If we use single (layer) perceptron to classified the points, it's can only be separate using simple hyperplane form. But if we want more complex shape, we could do logical operation (and, or, xor) using another layer of perceptron.

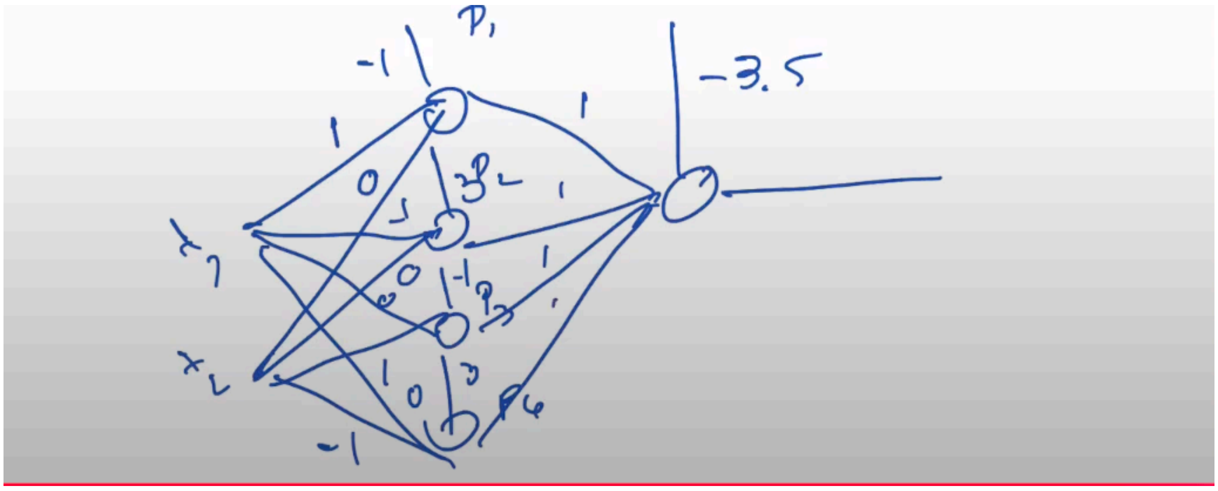
EX. If we want the middle rectangle to be positive zone



We could use 4 straight line that point toward the middle. Where each weight would look like


$P_1 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$
$P_2 = \begin{bmatrix} 3 \\ -1 \\ 0 \end{bmatrix}$
$P_3 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$
$P_4 = \begin{bmatrix} 3 \\ 0 \\ -1 \end{bmatrix}$
$x_2 < 3$
$-x_2 + 3 > 0$
$3 + (-1)(x_2) + (0)x_1 > 0$

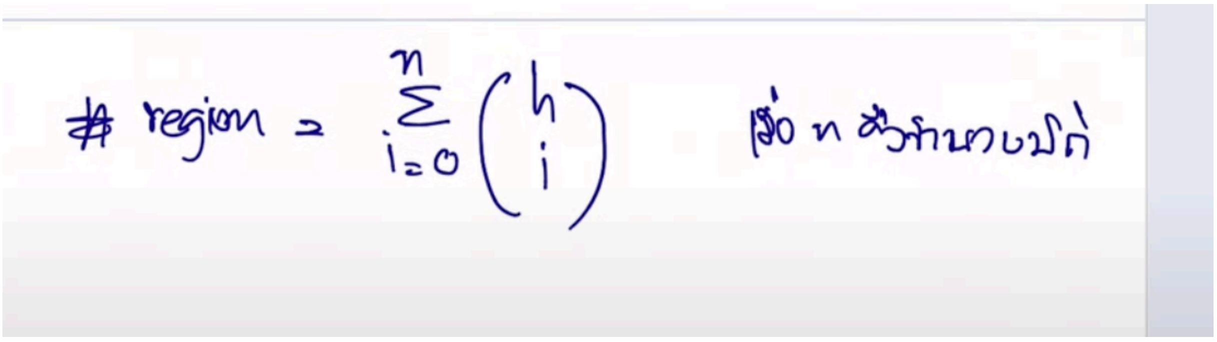
Then we can use each result as a input to another perceptron (next layer) that essentially perform logical AND ops (Something like `-3.5 + x1 + x2 + x3 > 0` -> `[-3.5, 1, 1, 1]`)



4. Number of Regions Separated by Nodes in a Hidden Layer

In *n* dimension space, we could use *n-1* dimension hyperplane (called hidden layer since it's somewhat originated from [[3. Multilayer Perceptron]]) to separate the space into multiple regions.

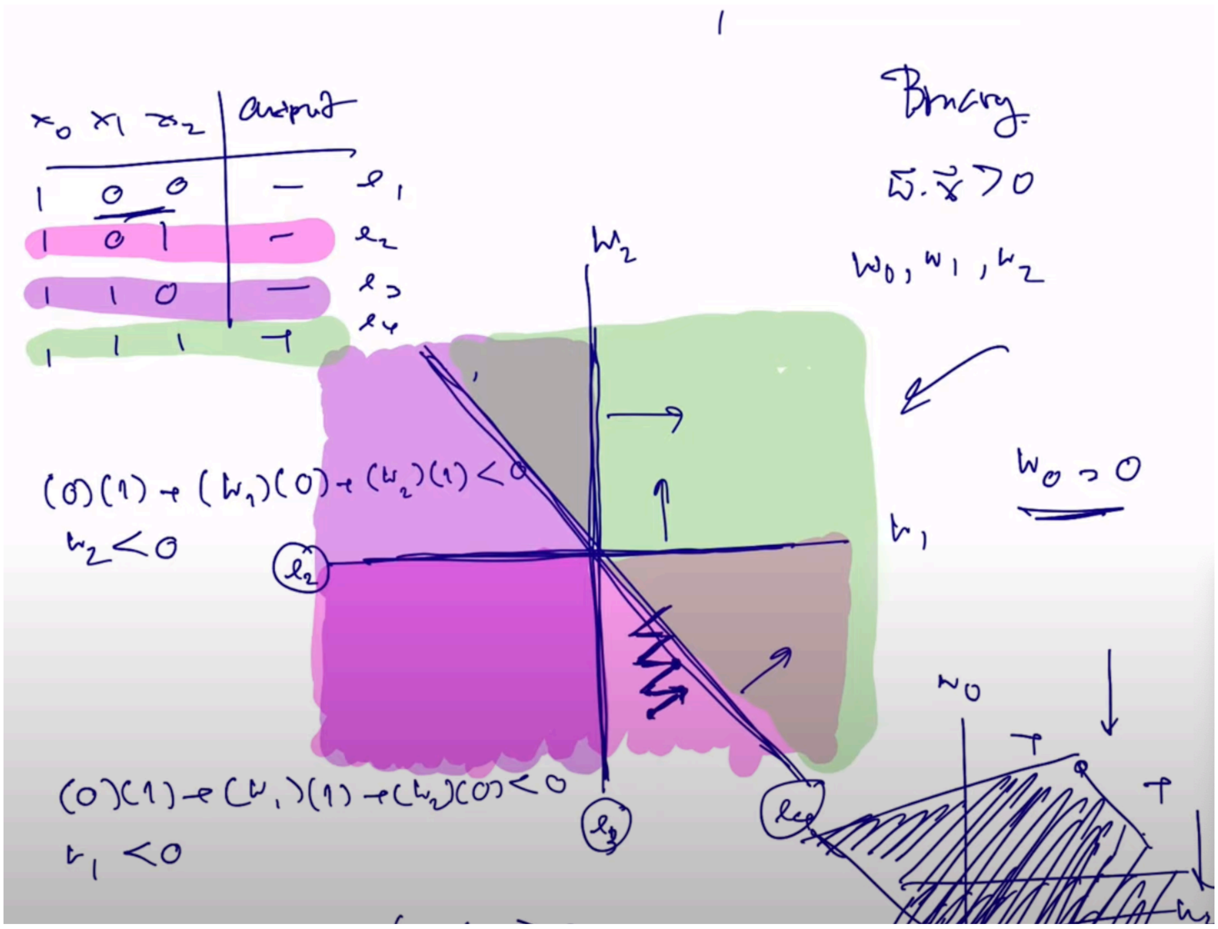
If we has *h* hidden layers, we could calculate the maximum number of region that could be separated with

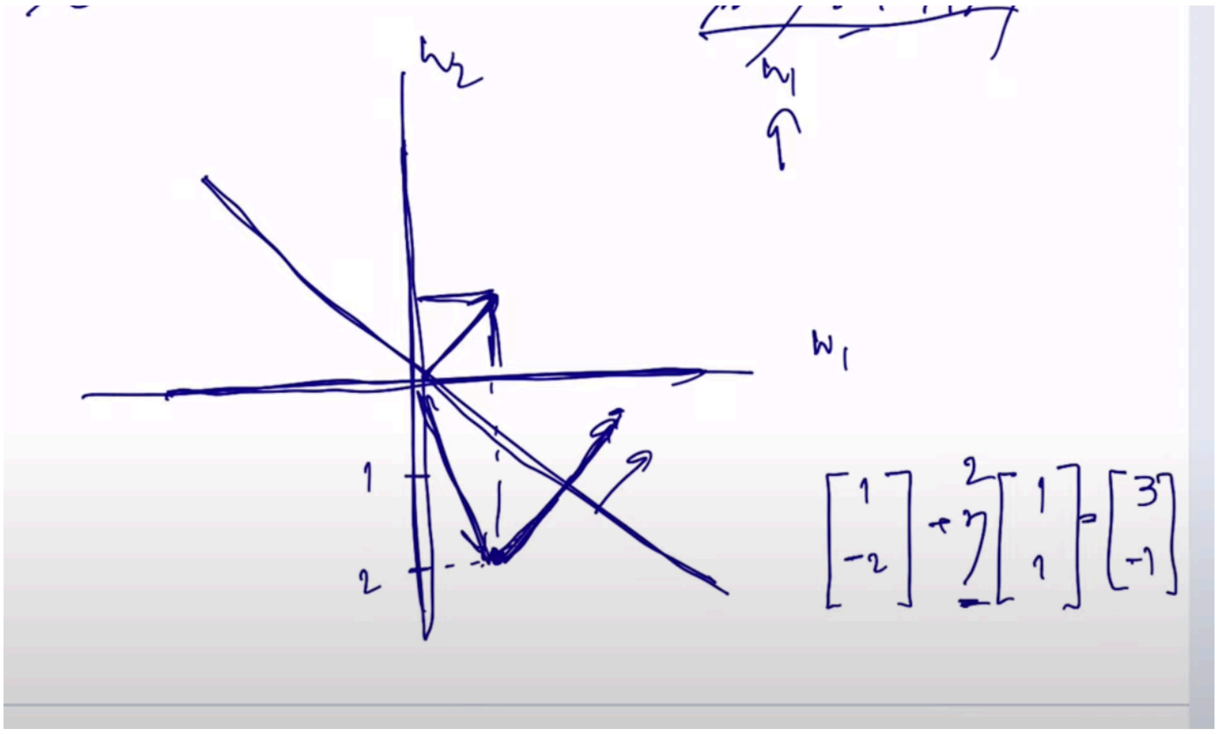


$$\# \text{region} = \sum_{i=0}^{n} \binom{h}{i}$$

เมื่อ n คือจำนวนมิติ

5. TLN (Binary Neural) Training





w_2
w_1
w_1
1
2
$\begin{bmatrix}1\\-2\end{bmatrix} + \overset{2}{\eta}\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}3\\-1\end{bmatrix}$

$$\overrightarrow{W_{new}} = \overrightarrow{W_{old}} + \eta \cdot \overrightarrow{input}$$

- η = Learning Rate
- Sometime input will get normalized by ||input||

Rule
- If actual is + but output ($\vec{w} \cdot \vec{x}$ <= 0), then $\vec{W} = \vec{W} + \eta{*}\vec{x}$
- If actual is - but output > 0, then $\vec{W} = \vec{W} - \eta{*}\vec{x}$

6. Training Linear Node & Sigmoid

Training

Error: $e_i = \frac{1}{2}(t_i - o_i)^2$ where
- t_i = Expected answer
- $o_i = f(\vec{w} \cdot \vec{x})$: result with f is signal function

$$\overrightarrow{W_{new}} = \overrightarrow{W_{old}} + \Delta\vec{W}$$

$$\Delta\vec{W} = -\nabla e_i = -\frac{\partial e_i}{\vec{W}}$$

Ex 1, if signal function is linear ($f(x) = x$)

$$\overrightarrow{W_{new}} = \overrightarrow{W_{old}} + (t - o)\vec{x}$$

Ex 2, if signal function is sigmoid ($f(x) = \frac{1}{1+e^{-x}}$)

$$\overrightarrow{W_{new}} = \overrightarrow{W_{old}} + \eta(t_i - o_i)(o_i)(1 - o_i)\vec{x}$$

Weiner Solution

In Linear system, we can use Weiner Solution that's faster to training multiple data (using matrix-based mult)

Given that E

$$X$$

= Expected valued of vector (average value)

$$P = E[t_k * x_k^T]$$

(P is vector of expected value for each x)

$$R = E[x_k * x_k^T]$$

EX

8. จากตารางด้านล่างนี้

4.3

Input		Output
X1	X2	O
2	5	18
-1	4	7
3	-1	9
2	1	10

$$\begin{bmatrix}1\\2\\5\end{bmatrix} \cdot \begin{bmatrix}1 & 2 & 5\end{bmatrix} \quad \begin{matrix}1 & 2 & 5\\2 & 4 & 10\\5 & 10 & 25\end{matrix}$$

ให้หาค่า P และ R เพื่อหา weight ด้วยวิธี Weiner Solution (ให้หาค่า P และ R โดยไม่ต้องหาค่า W)

$$P = \text{expected} \begin{bmatrix}1 & 2 & 5\\1 & -1 & 4\\1 & 3 & -1\\1 & 2 & 1\end{bmatrix} \& \begin{bmatrix}18\\7\\9\\10\end{bmatrix} \rightarrow \begin{bmatrix}18 & 2\cdot18 & 5\cdot18\\7 & -7 & 4\cdot7\\9 & 3\cdot9 & -9\\10 & 2\cdot10 & 10\end{bmatrix} \text{ SUM}$$

$$\rightarrow \left[11 \quad 19 \quad 29.75\right]^{4}$$

$$R = \begin{bmatrix}1 & 2 & 5\\2 & 4 & 10\\5 & 10 & 25\end{bmatrix} + \begin{bmatrix}1 & -1 & 4\\-1 & 1 & -4\\4 & -4 & 16\end{bmatrix} + \begin{bmatrix}1 & 3 & -1\\3 & 9 & -3\\-1 & -3 & 1\end{bmatrix} + \begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix} \Big/ 4$$