[機器與深度學習基礎知識初探] DNN

黃志勝

義隆電子人工智慧研發部

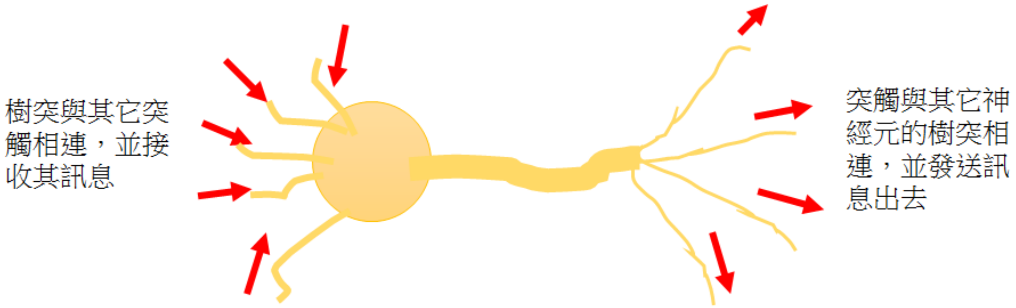
國立陽明交通大學AI學院合聘助理教授

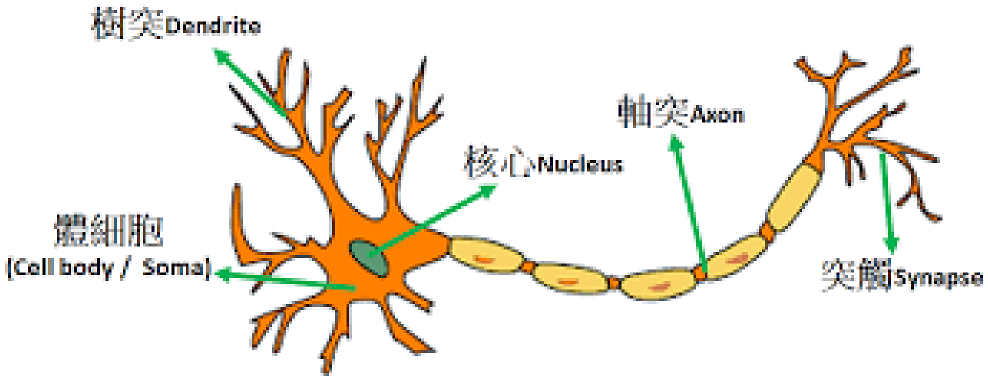
Outline

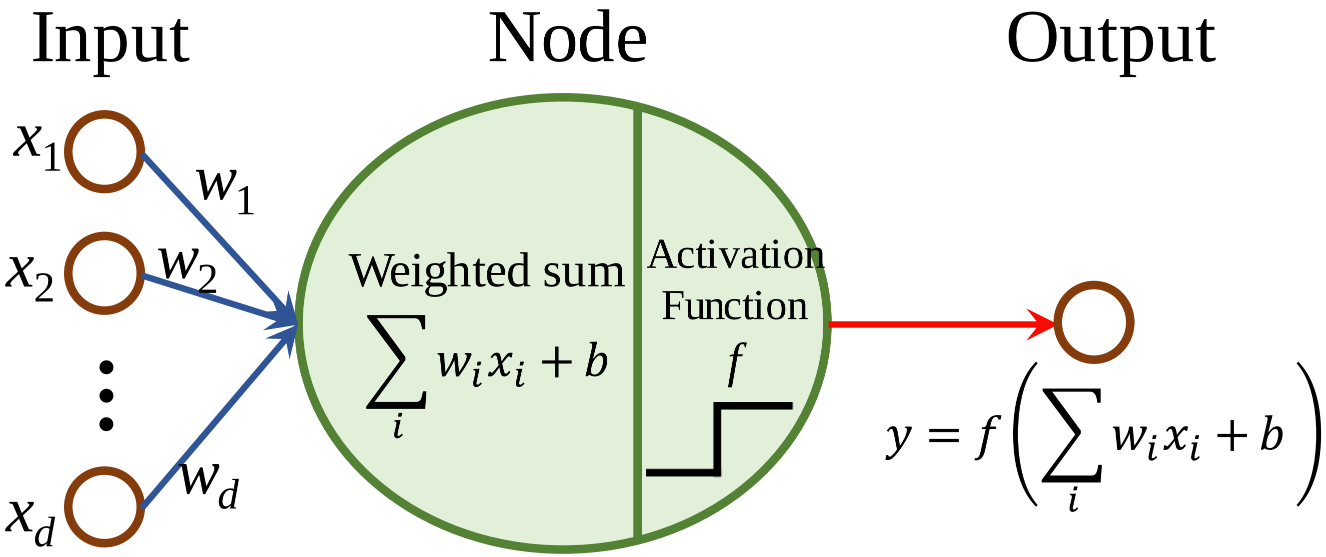
- 1.類神經網路(Neural Network, NN)
- 2.感知機(Perception)
- 3. Multi-layer perception (MLP)
- 4. How NN work?

類神經網路

基本上神經網路是基於感知機(Perceptron)神經網路開始，主要是希望用數學模型去模擬神經細胞的運作模式。







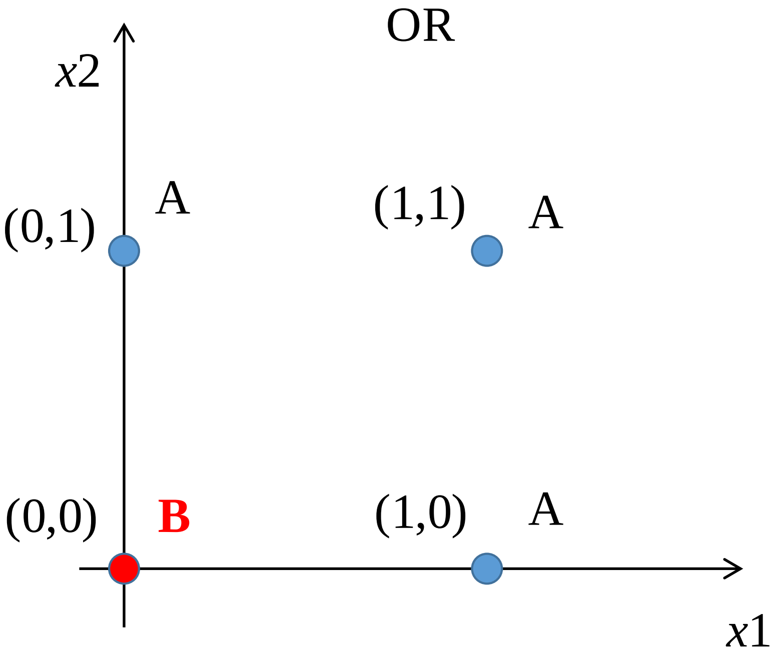
權重(w_i): Dendrite

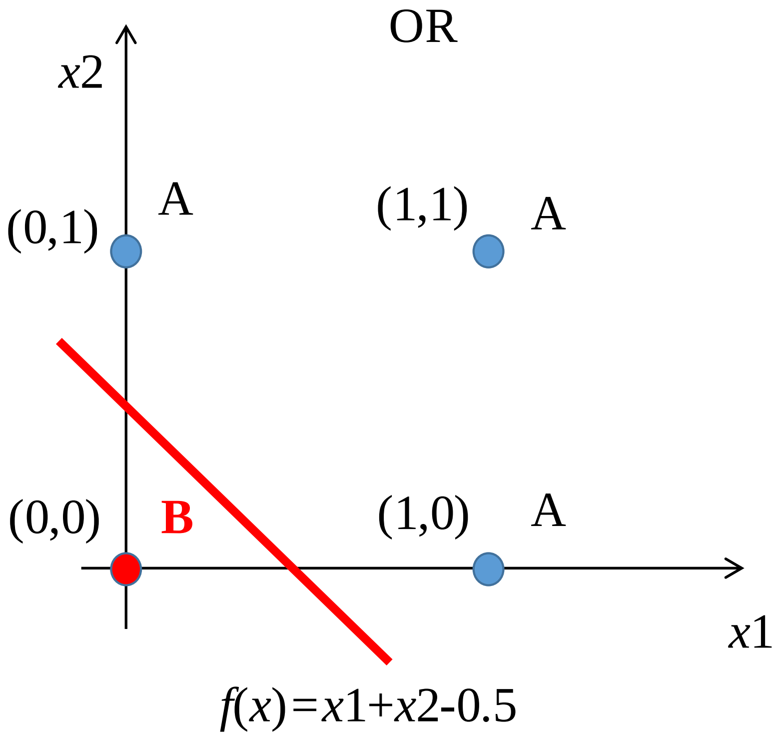
Input(x_i) and output (y) node: Synapse

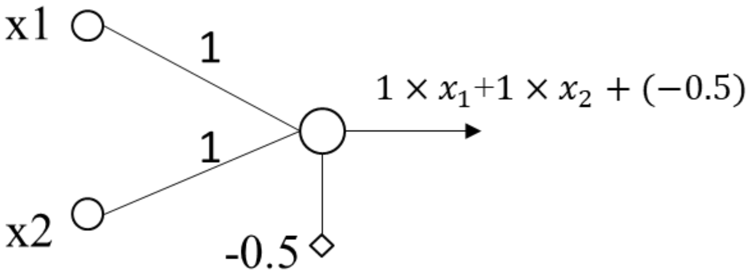
Node: Cell body

Output: Axon

NN for classification problem

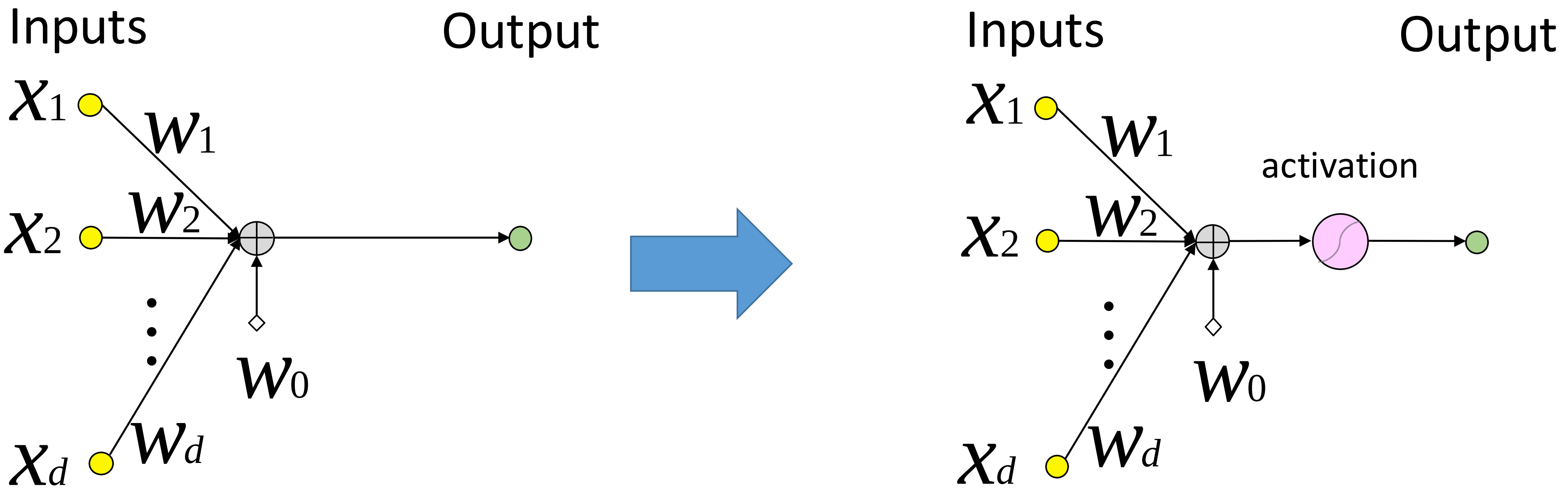




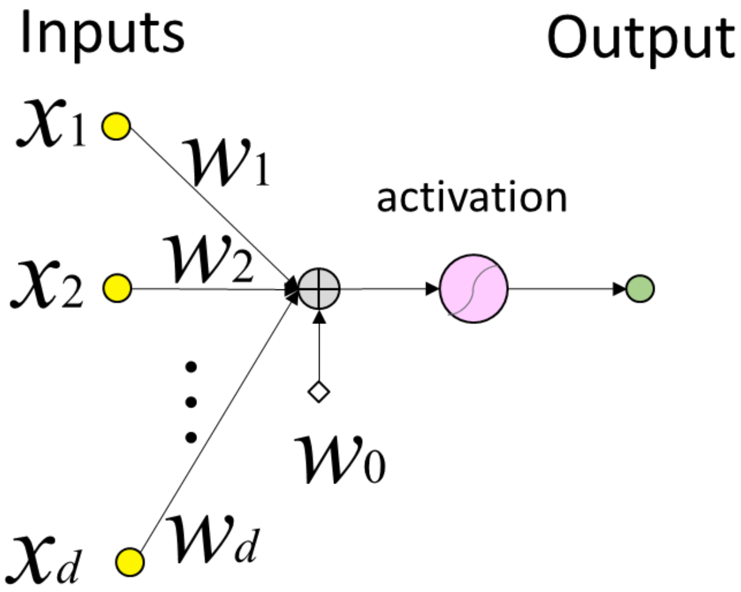


Perception

Perception can learn the nonlinear representation by activation function.



Perception



$$y = f(w_{10} + w_{11}x_{1+}w_{12}x_2 + \cdots + w_{1d}x_d) = f(\boldsymbol{W}^T\boldsymbol{x})$$

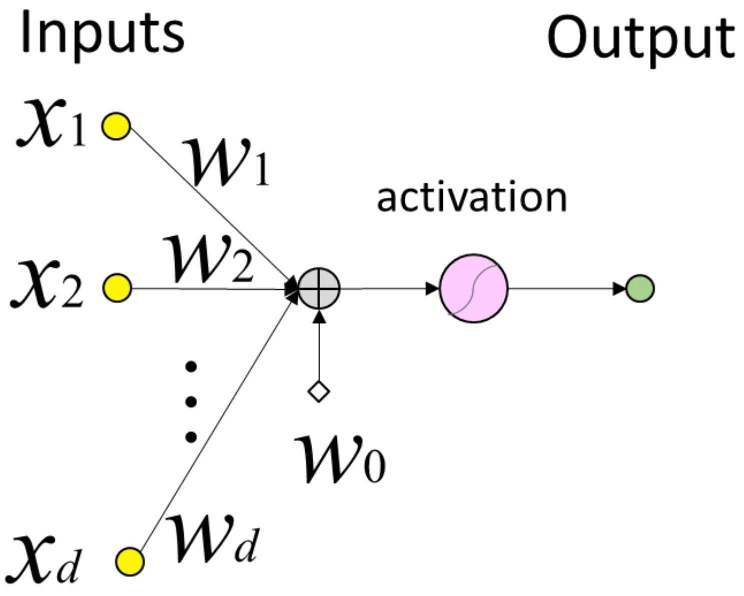
Classification: $f = \begin{cases} 1 & \boldsymbol{W}^T\boldsymbol{x} \geq 0 \\ 0 & O.W. \end{cases}$

Regression: $f(\boldsymbol{W}^T\boldsymbol{x})$

$$\boldsymbol{W} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$

神經元與回歸

神經元



=

回歸

$$f(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

$$y = \sigma\left(f(\boldsymbol{x})\right) = \frac{1}{1+e^{-f(\boldsymbol{x})}} = \frac{1}{1+e^{-\boldsymbol{\beta}^T \boldsymbol{x}}}$$

Perception with linear output is the linear regression.

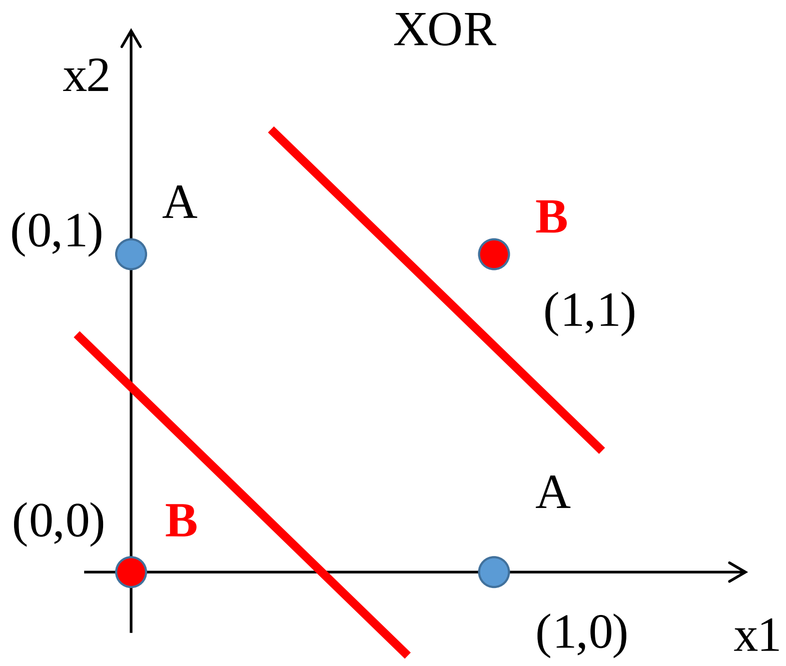
NN: backpropagation

Regression: OLSE (ordinary least squares estimator).

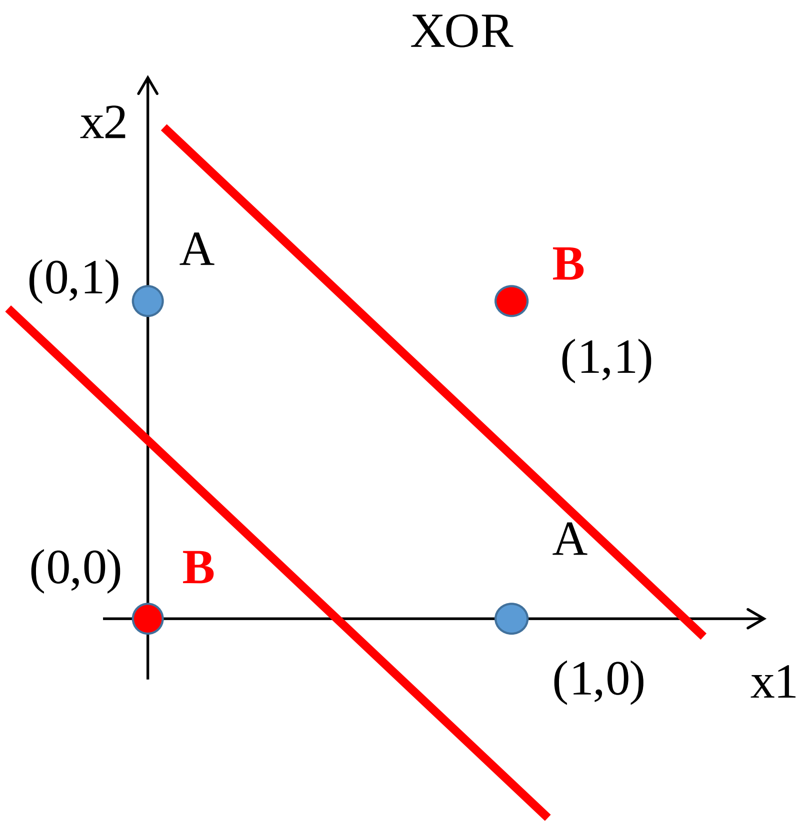
NN for XOR problem

- Exclusive OR (XOR Boolean function)
- It's impossible to find a single straight line to separate two classes.

Truth Table for the XOR problem			
x1	x2	AND	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	0	B



NN for XOR problem



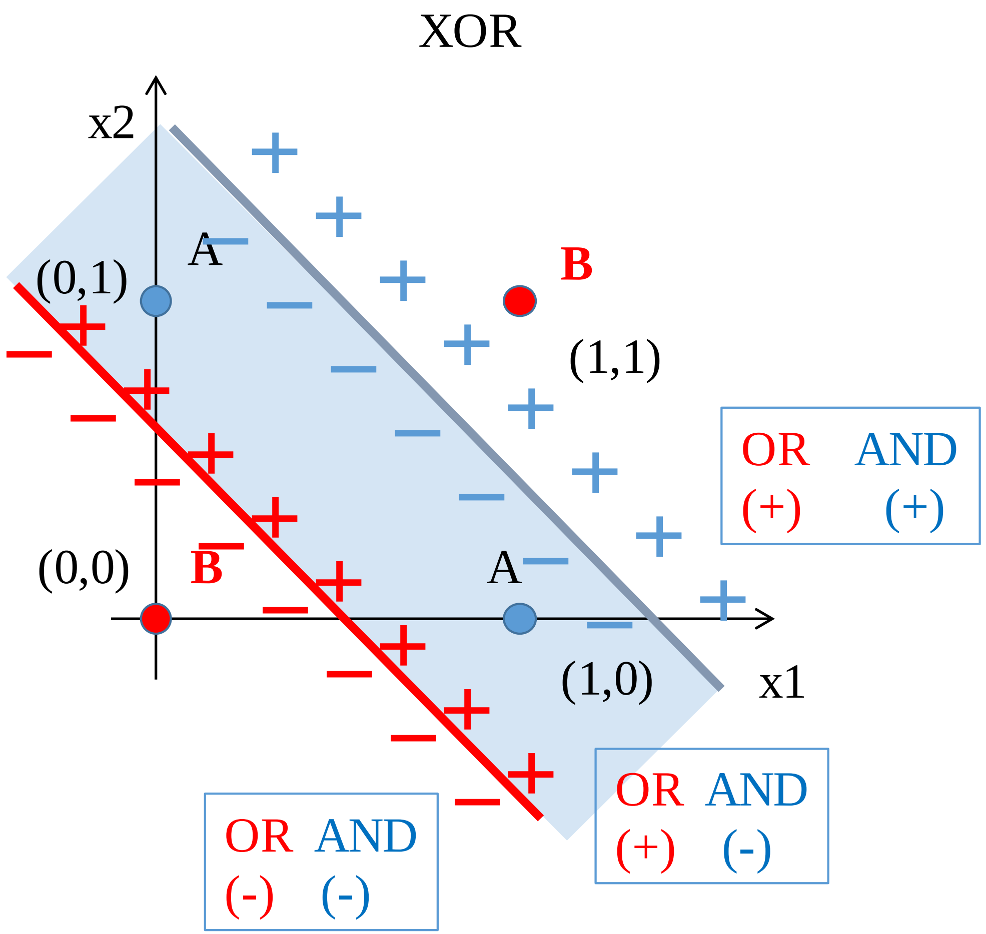
OR

$$h_1(x) = x_1 + x_2 - 0.5 = 0$$

AND

$$h_2(x) = x_1 + x_2 - 1.5 = 0$$

NN for XOR problem



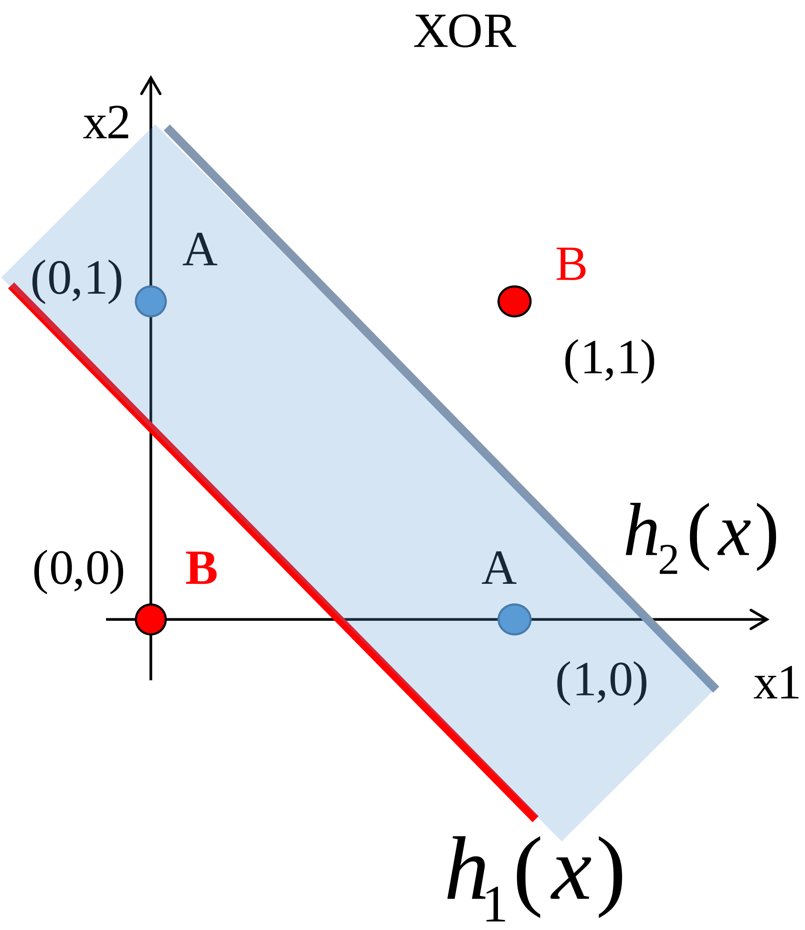
OR

$$h_1(x) = x_1 + x_2 - 0.5 = 0$$

AND

$$h_2(x) = x_1 + x_2 - 1.5 = 0$$

NN for XOR problem



OR

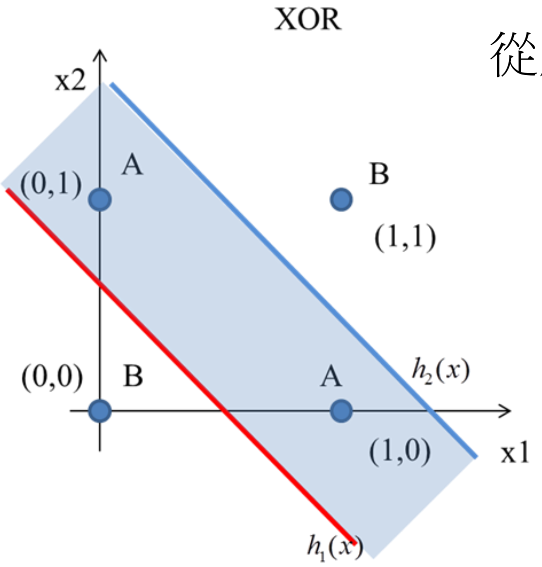
$$h_1(x) = x_1 + x_2 - 0.5 = 0$$

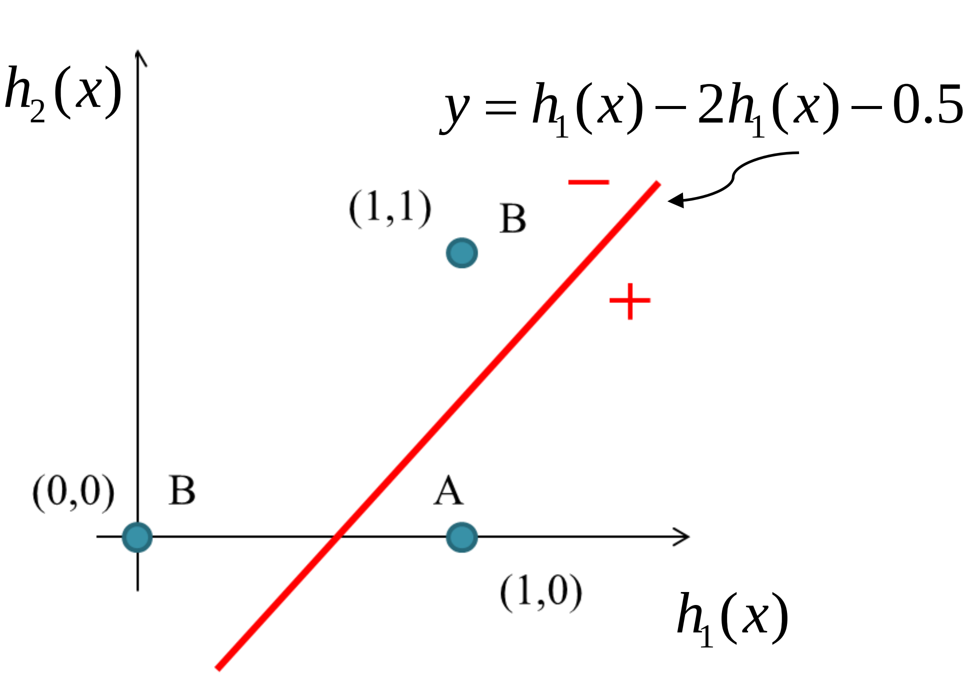
AND

$$h_2(x) = x_1 + x_2 - 1.5 = 0$$

NN for XOR problem

從原始特徵空間做特徵轉換(特徵萃取)

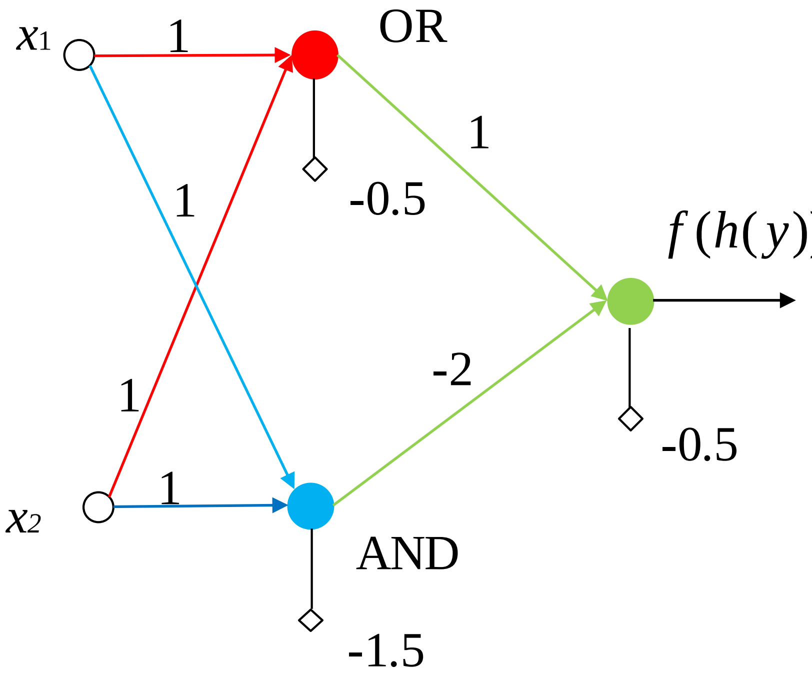




Truth Table for XOR problem				
x1	x2	$h_1(x)$	$h_2(x)$	Class
0	0	0(-)	0(-)	B
0	1	1(+)	0(-)	A
1	0	1(+)	0(-)	A
1	1	1(+)	1(+)	B

Two Layer Perception

$$y_1 = f(h_1(x)) = f(x_1 + x_2 - 0.5)$$



$$f(h(y)) = f(y_1 - 2y_2 - 0.5)$$

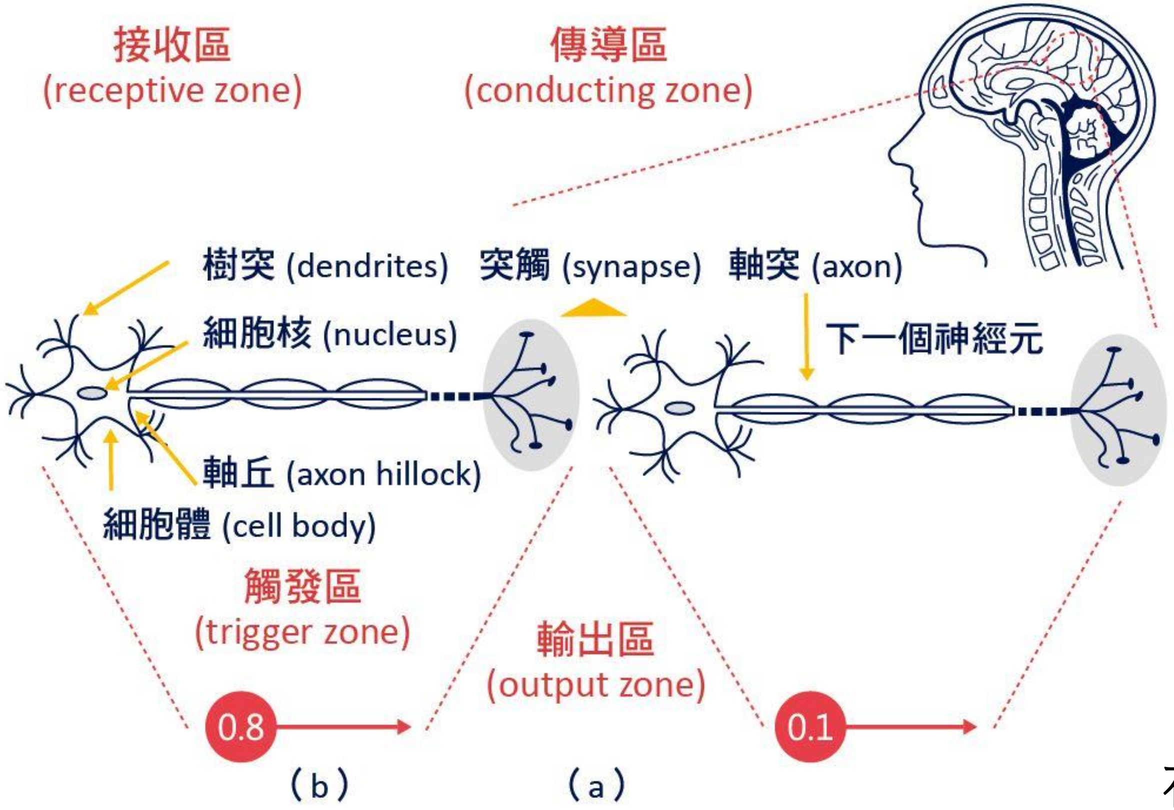
$$y_2 = f(h_2(x)) = f(x_1 + x_2 - 1.5)$$

$$f(v) = \begin{cases} 1 & if \ \ v \geq 0 \\ o & if \ \ v < 0 \end{cases}$$

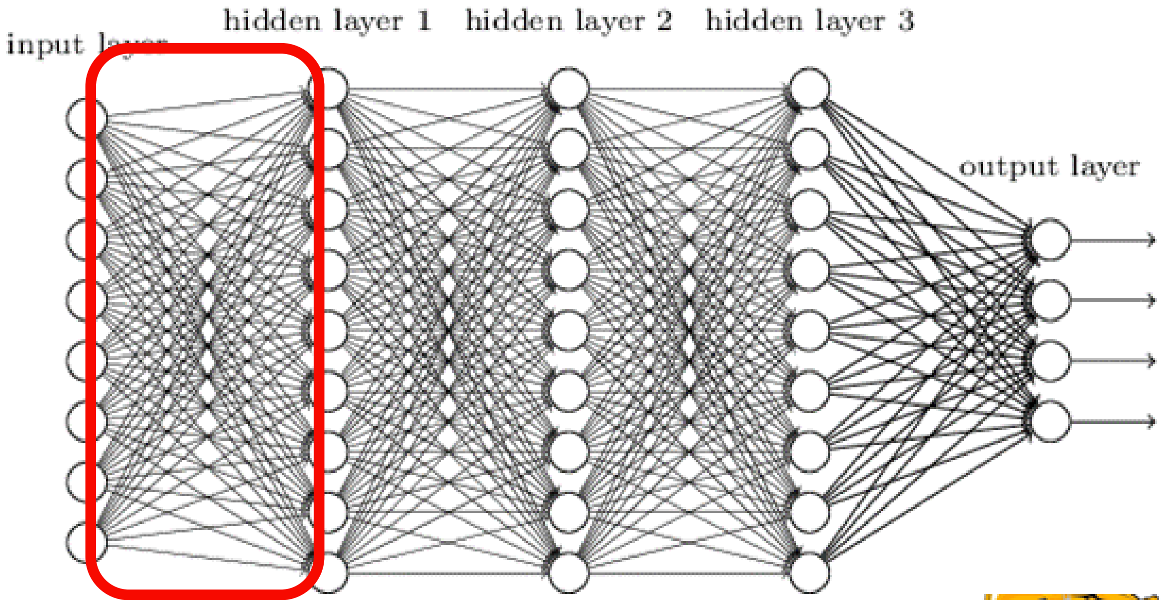
Step activation function

類神經網路

但神經訊息傳遞不會像只有一層Single layer perceptron運作，神經網路應該是多個細胞部段將訊息傳遞下去運作的模式，這就是Multilayer perception (MLP)，也就是一般認知的類神經網路。



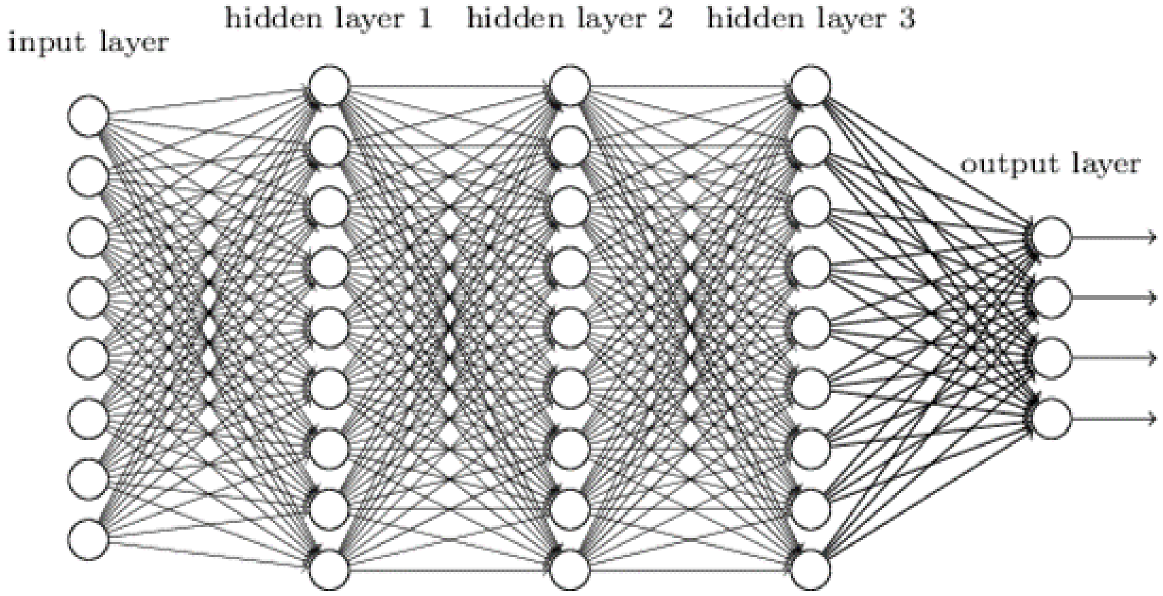
MLP



在深度學習這樣的層跟層叫做**fully connection**

Activation

- 如果沒有activation function會有什麼影響



$X_1 \quad W_1 \quad X_2 \quad W_2 \quad X_3 \quad W_3 \quad X_4 \quad W_4 \quad Y$

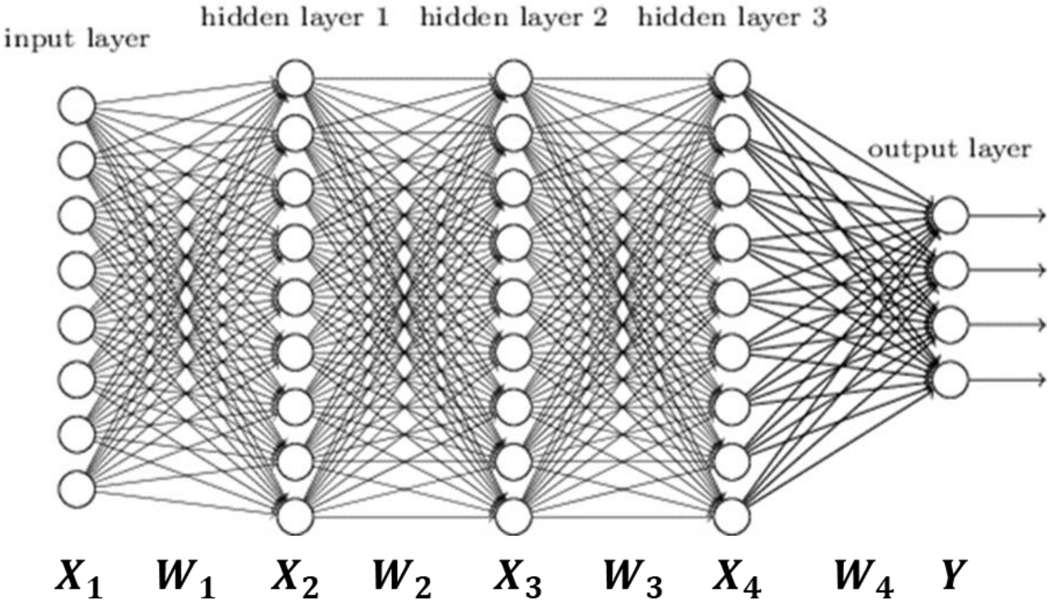
$$X_2 = W_1 X_1$$
$$X_3 = W_2 X_2$$
$$X_4 = W_3 X_3$$
$$Y = W_4 X_4$$

$$Y = W_4 W_3 W_2 W_1 X_1$$
$$\underline{Y = W X_1}$$
$$W = W_4 W_3 W_2 W_1 X_1$$

結果就只是一層的神經網路而已。

Activation

- 如果沒有activation function會有什麼影響



$$X_2 = W_1 X_1$$
$$X_3 = W_2 X_2$$
$$X_4 = W_3 X_3$$
$$Y = W_4 X_4$$

$$Y = W_4 W_3 W_2 W_1 X_1$$
$$\underline{Y = W X_1}$$
$$W = W_4 W_3 W_2 W_1 X_1$$

$$X_2 = f(W_1 X_1)$$
$$X_3 = f(W_2 X_2)$$
$$X_4 = f(W_3 X_3)$$
$$Y = f(W_4 X_4)$$

$$Y = f\left(W_4 f\left(W_3 f\left(W_2 f(W_1 X_1)\right)\right)\right)$$

Activation Function

- Sigmoid、Tanh。

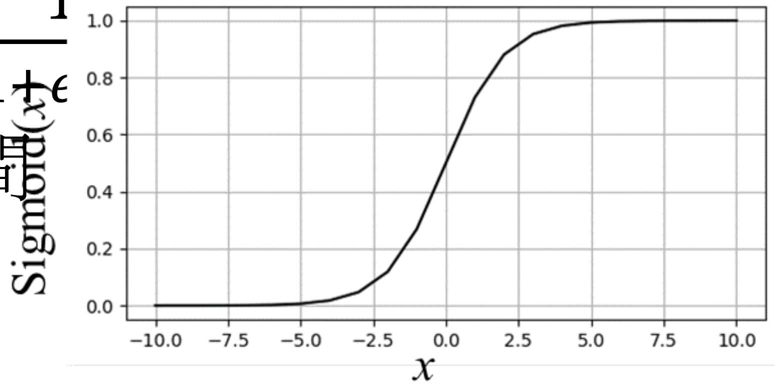
ReLU(Rectified Linear Unit)系列：

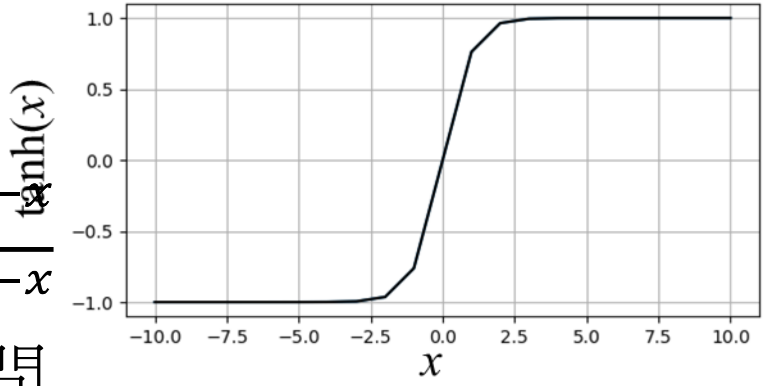
- ReLU
- Leaky ReLU
- ReLU6
- PReLU
- RReLU
- ELU (Exponential Linear Unit)
- SELU (Scaled Exponential Linear Units)

Activation function

- $\mathrm{Sigmoid}(x) = \frac{1}{1+e^{-x}}$

將值壓到0~1之間



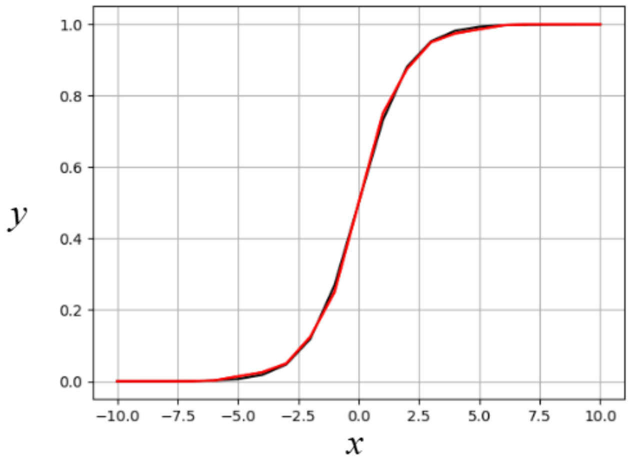


- $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

將值壓到-1~1之間

Sigmoid近似法

$$y = approxSigmoid(x) = \begin{cases} 0.5 * \left(1.5 * \left(\dfrac{\frac{x}{2}}{1+\frac{x}{2}}\right) + 1\right) & if\ \ 0 \le x < 3.4 \\ 0.5 * (0.9444 + 0.0459 * (\frac{x}{2} - 1.7) + 1) & if\ \ 3.4 \le x \le 5.8 \\ 0.9997 & x \ge 5.8 \end{cases}$$



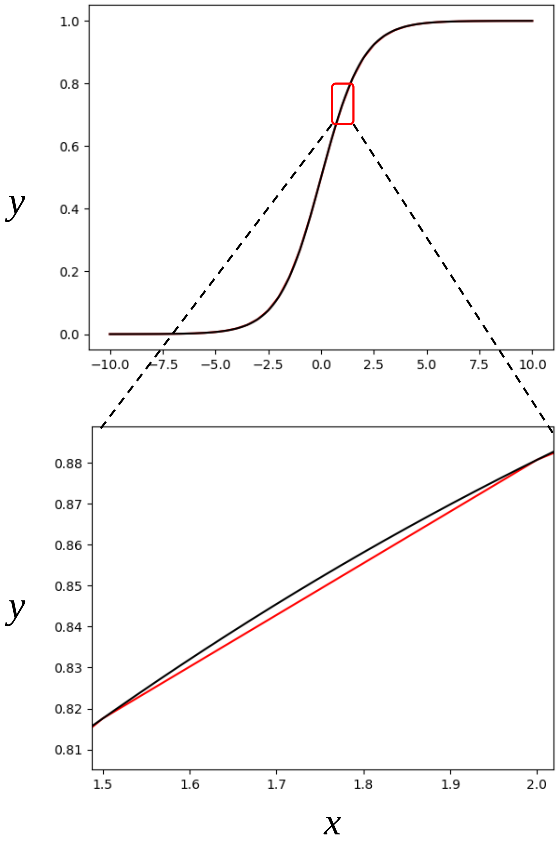
Black: sigmoid
Red: approximate sigmoid

```
def fast_sigmoid2(x):
    x = 0.5 * x
    flag_neg=0
    if x<0:
        x=-x
        flag_neg=1

    if x < 1.7:
        z = (1.5 * x / (1 + x))
    elif x < 2.9:
        z = (0.9444 + 0.0459 * (x - 1.7))
    else:
        z = 0.9997
    if flag_neg==1:
        z=-z
    return 0.5 * (z + 1.)
```

Sigmoid (look up table)

x	sigmoid(x)
0	0.5000
0.5	0.6225
1	0.7311
1.5	0.8176
2	0.8808
2.5	0.9241
3	0.9526
3.5	0.9707
4	0.9820
4.5	0.9890
5	0.9933
5.5	0.9959
6	0.9975
6.5	0.9985
7	0.9991
7.5	0.9994
8	0.9997
8.5	0.9998
9	0.9999
9.5	0.9999
10	1.0000



Black: sigmoid
Red: partial sigmoid

Activation function: ReLU系列

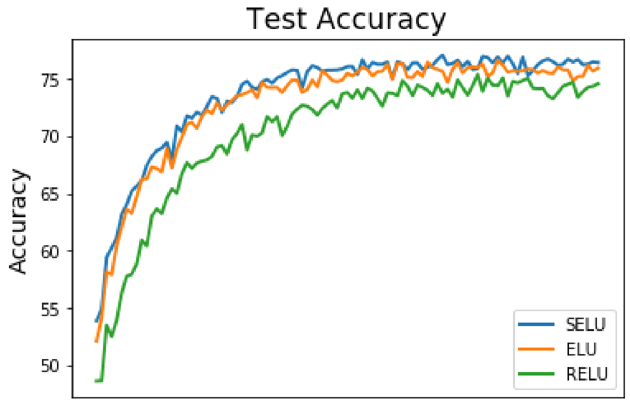
- $\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} x & if\ x > 0 \\ 0 & O.W. \end{cases}$
- $\mathrm{ReLU6}(x) = \min(\max(0, x), 6) = \begin{cases} 6 & if\ x \geq 6 \\ x & if\ 0 < x < 6 \\ 0 & O.W. \end{cases}$
- $\mathrm{LeakyReLU}(x) = \max(0, x) + a * \min(0, x) = \begin{cases} x & if\ x > 0 \\ ax & O.W. \end{cases}\ (default\ a = 0.1)$
- $\mathrm{PReLU}(x) = \max(0, x) + a * \min(0, x) = \begin{cases} x & if\ x \geq 0 \\ ax & O.W. \end{cases}$ (a是訓練得到，$init$設定:0.25)
- $\mathrm{RReLU}(x) = \max(0, x) + a * \min(0, x) = \begin{cases} x & if\ x > 0 \\ ax & O.W. \end{cases}$ (a是隨機$U(lower = \frac{1}{8}, upper = \frac{1}{3})$選取)
- $\mathrm{ELU}(x) = \max(0, x) + \min(0, \alpha * (e^x - 1)) = \begin{cases} x & if\ x > 0 \\ \alpha * (e^x - 1) & O.W. \end{cases}$
- $\mathrm{SELU}(x) = \mathrm{scale}\big(\max(0, x) + \min\big(0, \alpha * (e^x - 1)\big)\big) = \mathrm{scale} \begin{cases} x & if\ x > 0 \\ \alpha * (e^x - 1) & O.W. \end{cases}$

  α=1.6732632423543772848170429916717
  scale=1.0507009873554804934193349852946
  實驗驗證，小數取兩位數即可，α=1.67, scale=1.05

SELU係數精度實驗

- Model: LeNet 加強版
- Database: Cifar-10
- 看分類正確率隨著learning epoch變化的影響

SELU係數小數位數全取

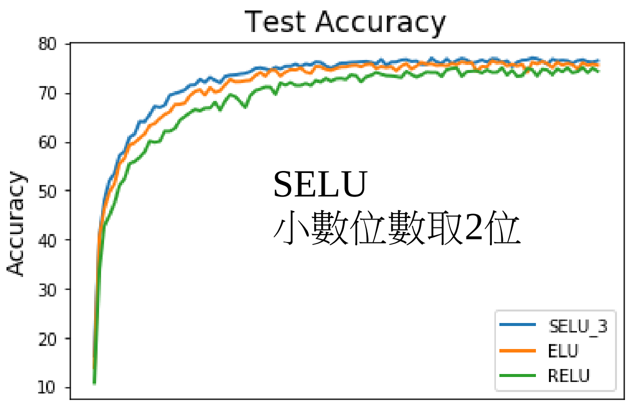


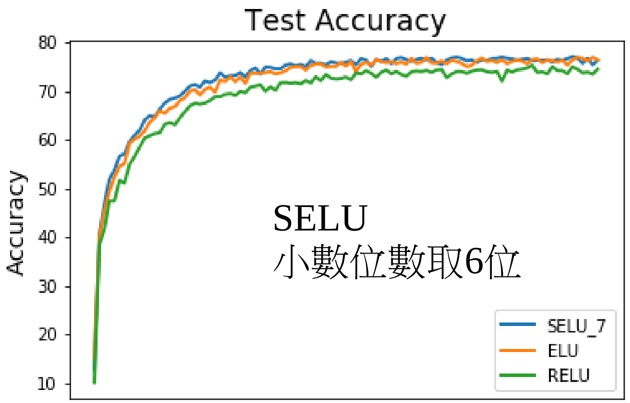
$$\mathrm{SELU}(x) = \mathrm{scale}\big(\max(0, x) + \min(0, \alpha * (e^x - 1))\big)$$

$$= \mathrm{scale}\begin{cases} x & if\ x > 0 \\ \alpha * (e^x - 1) & O.W. \end{cases}$$

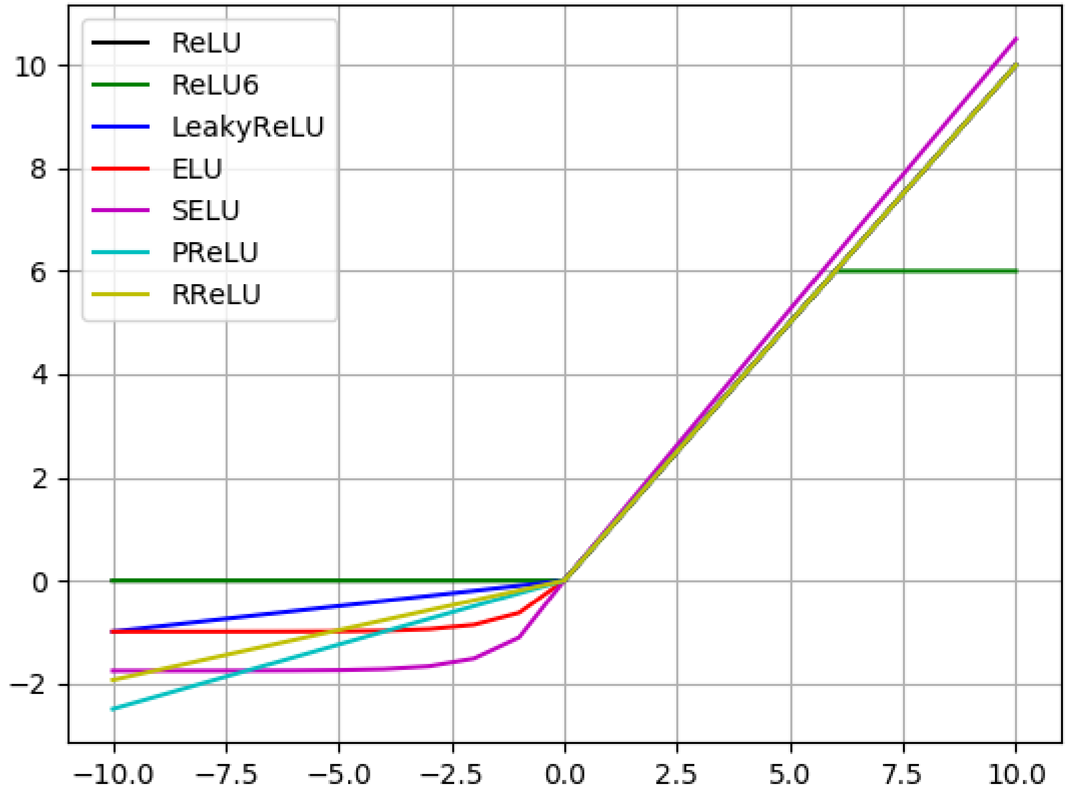
α=1.6732632423543772848170429916717

scale=1.0507009873554804934193349852946





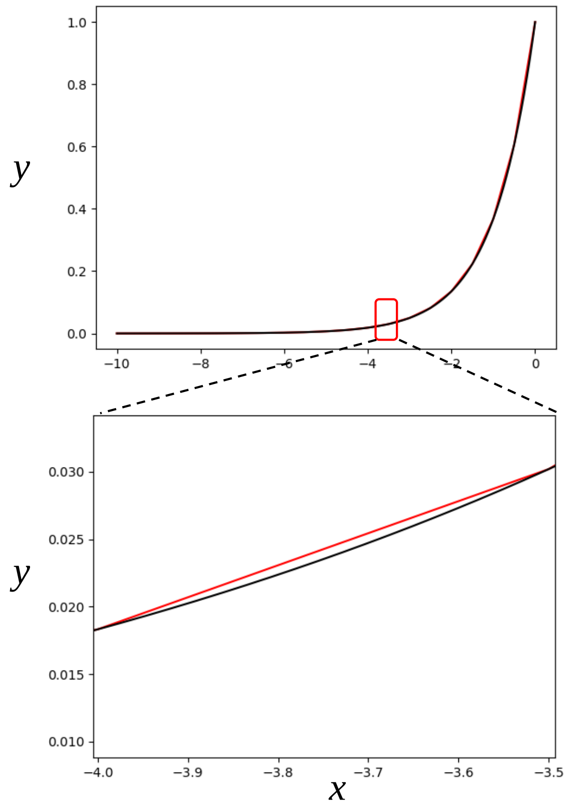
Activation function: ReLU系列



Exp (look up table)

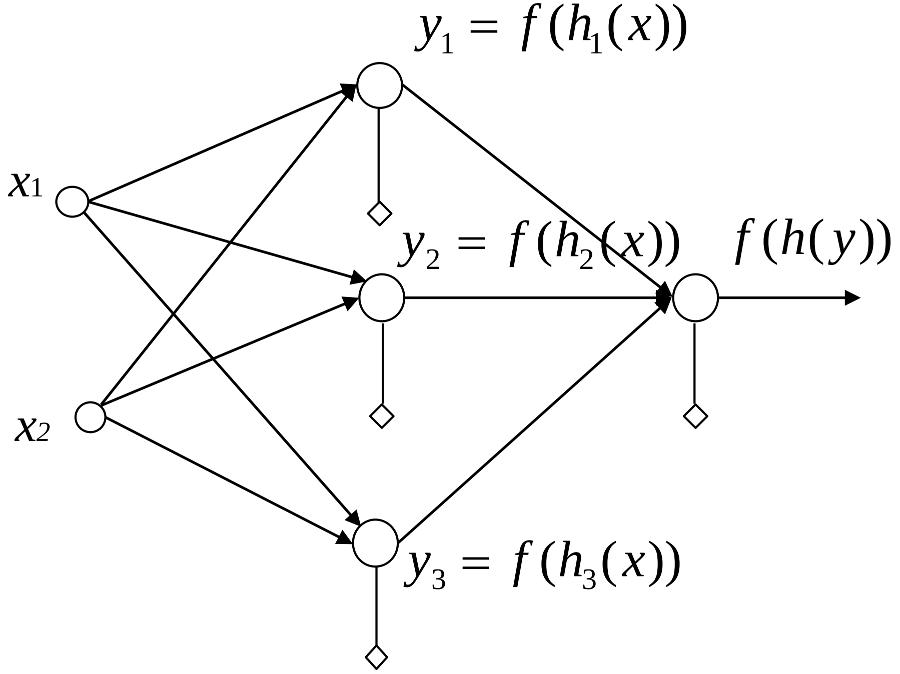
因為在activation (ELU)系列只有輸入是負號才會算exp，所以列出

x	exp(x)
-20	0
-10	0.00005
-9.5	0.00007
-9	0.00012
-8.5	0.00020
-8	0.00034
-7.5	0.00055
-7	0.00091
-6.5	0.00150
-6	0.00248
-5.5	0.00409
-5	0.00674
-4.5	0.01111
-4	0.01832
-3.5	0.03020
-3	0.04979
-2.5	0.08209
-2	0.13534
-1.5	0.22313
-1	0.36788
-0.5	0.60653
0	1.00000



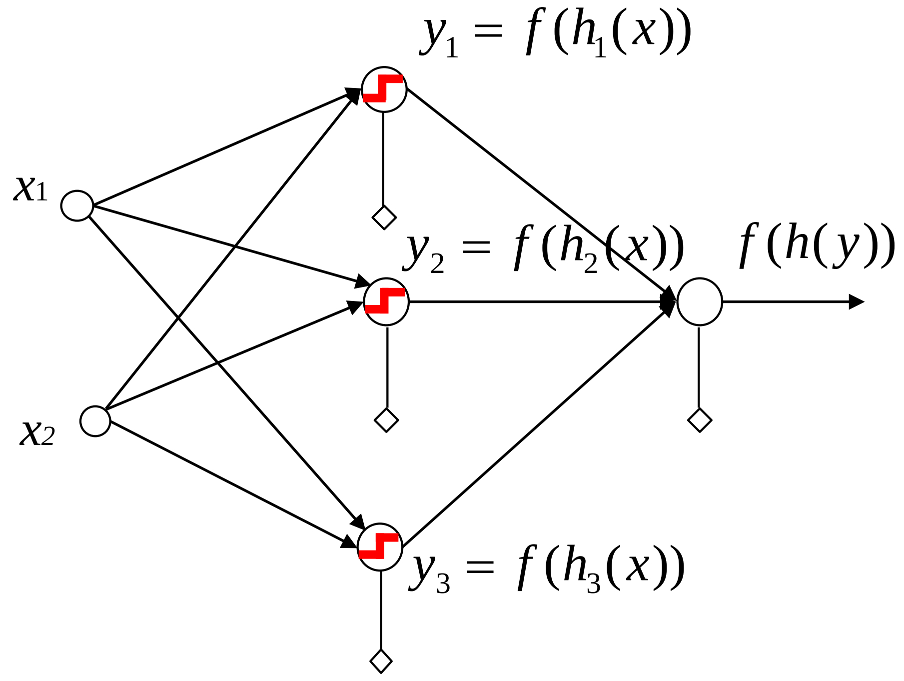
Black: exp
Red: partial exp

Polyhedral Regions

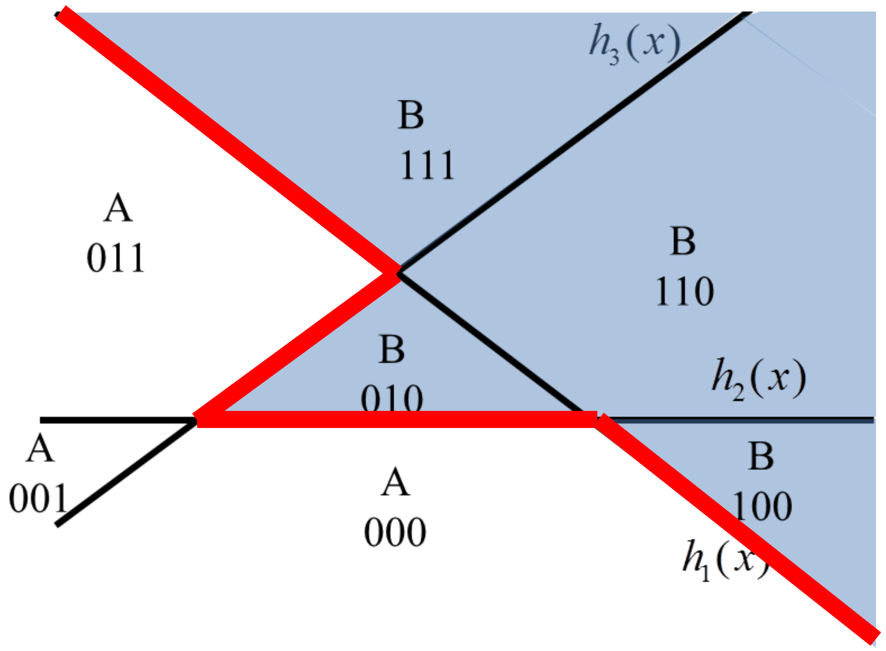


Polyhedral Regions

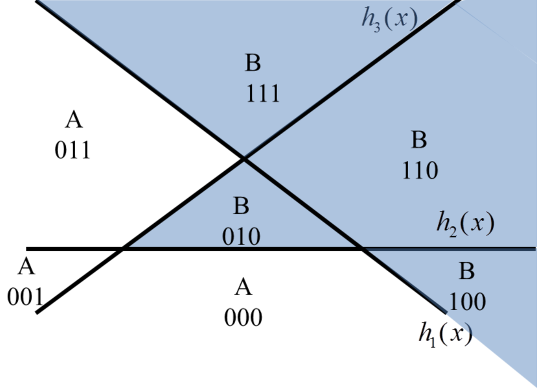
activation function = step function



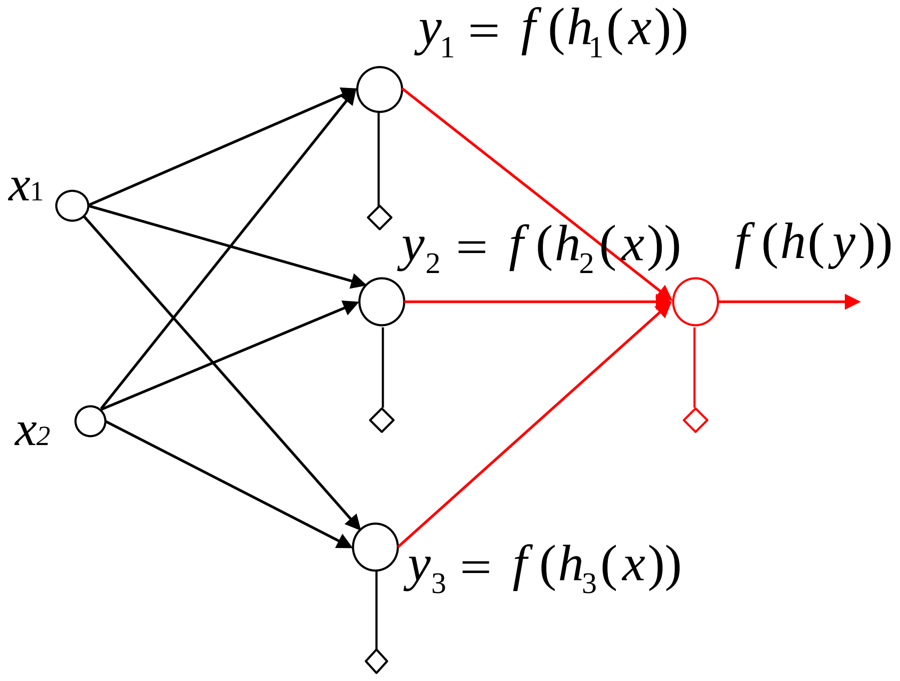
The first layer of neurons divides the input d-dimensional space into polyhedral, which are formed by hyperplane intersections.

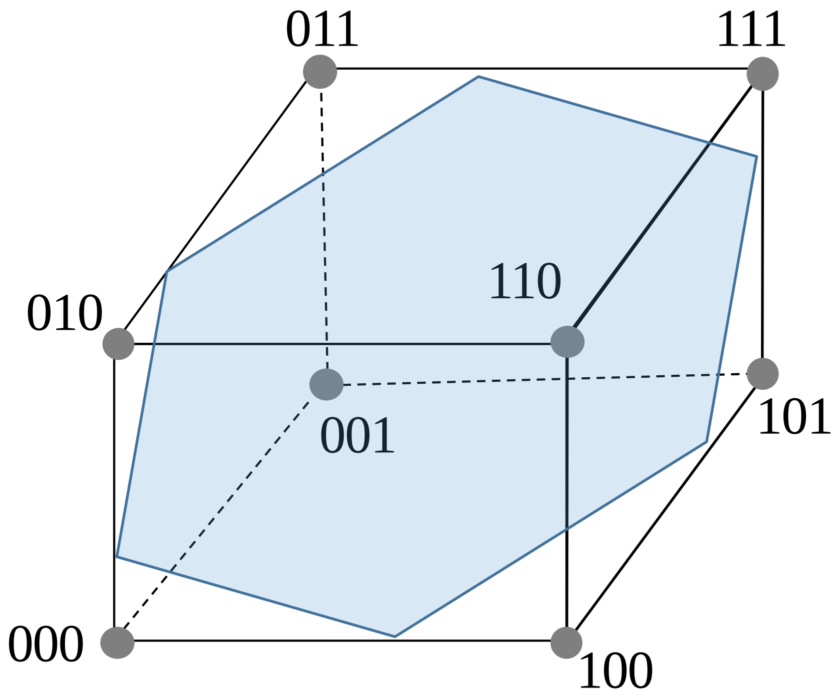


Polyhedral Regions



All vectors located within one of these polyhedral regions are mapped onto a specific vertex of the unit hypercube.

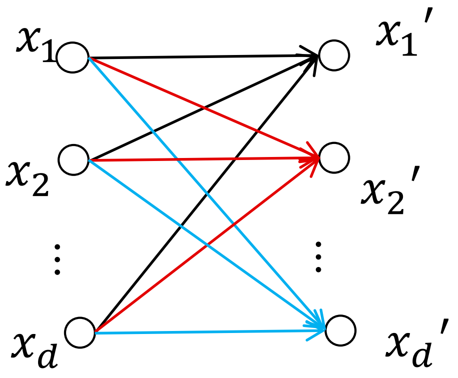


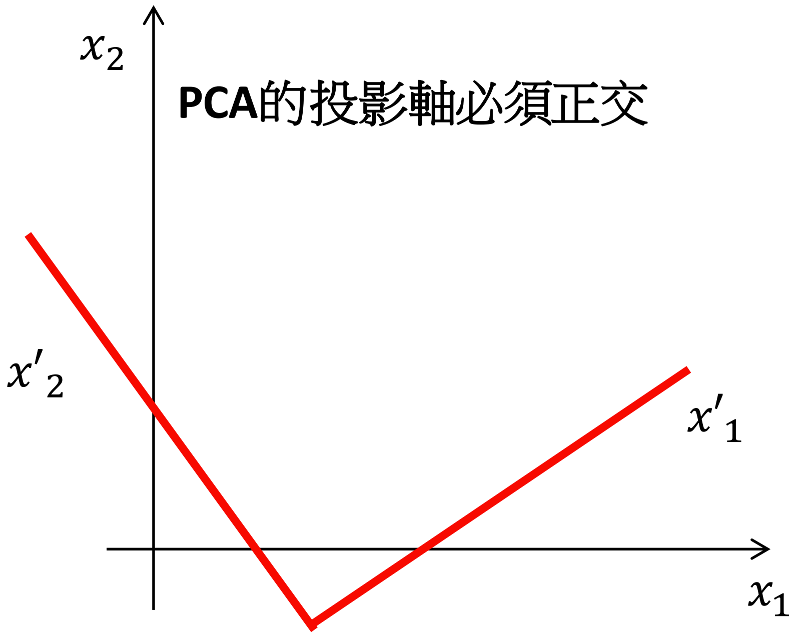


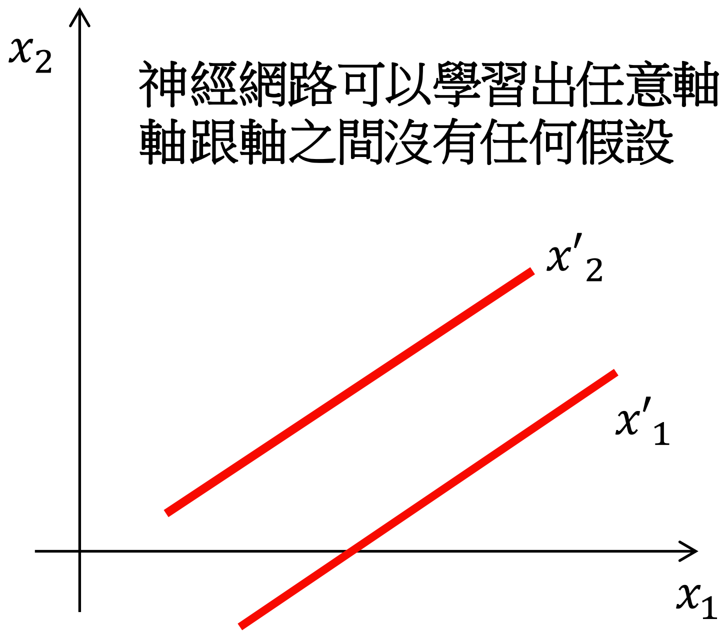
NN and Dimension Reduction

- **Principle Component Analysis**

$$\boldsymbol{x}' = \begin{bmatrix} x_1{}' \\ \vdots \\ x_d{}' \end{bmatrix} = W^T\boldsymbol{x} = \begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{d1} & \cdots & w_{dd} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$



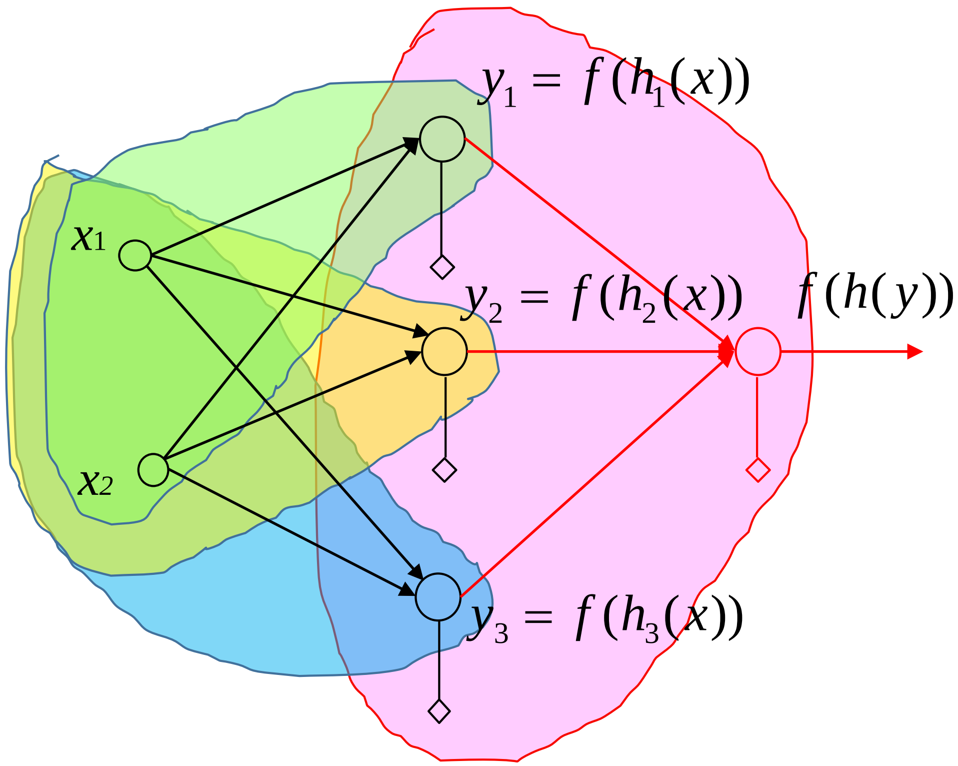




PCA是NN的一個special case

NN and Ensemble Learning

NN



Ensemble Learning