神經網路與損失函數

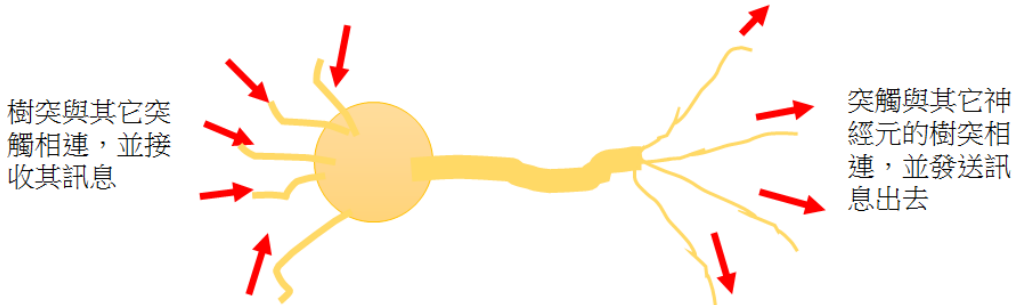
黃志勝（Tommy Huang）
義隆電子 人工智慧研發部
國立陽明交通大學 AI學院 合聘助理教授
國立台北科技大學 電資學院合聘助理教授

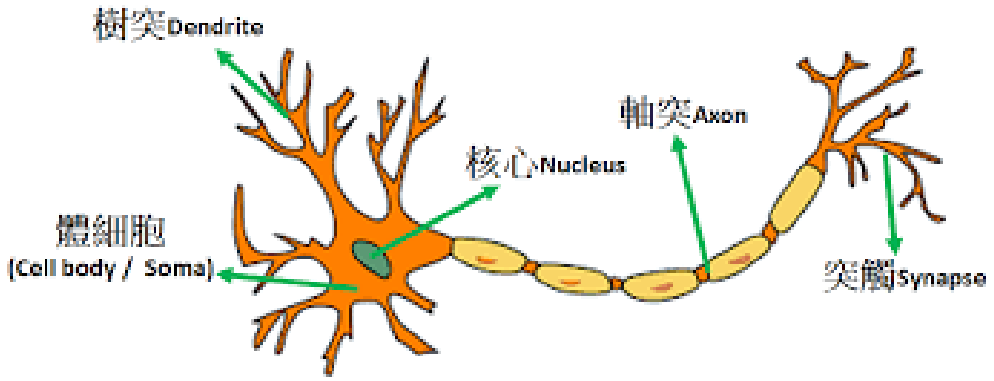
Outline

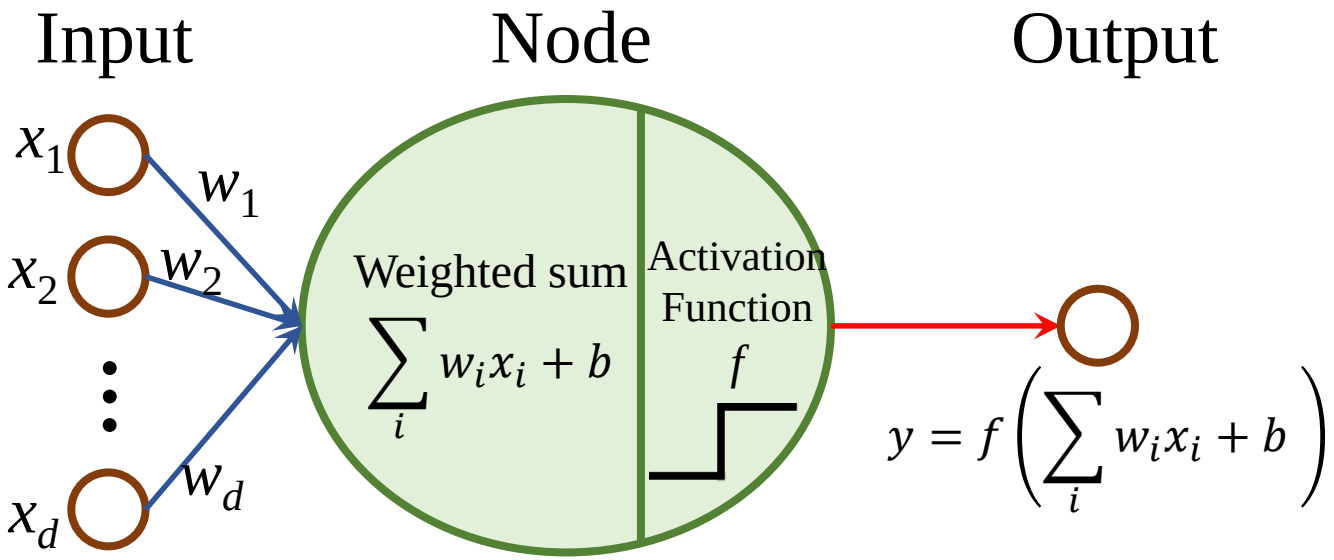
- 1.類神經網路(Neural Network, NN)
- 2.感知機(Perception)
- 3. Multi-layer perception (MLP)
- 4. How NN work?
- 5. Loss function

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。







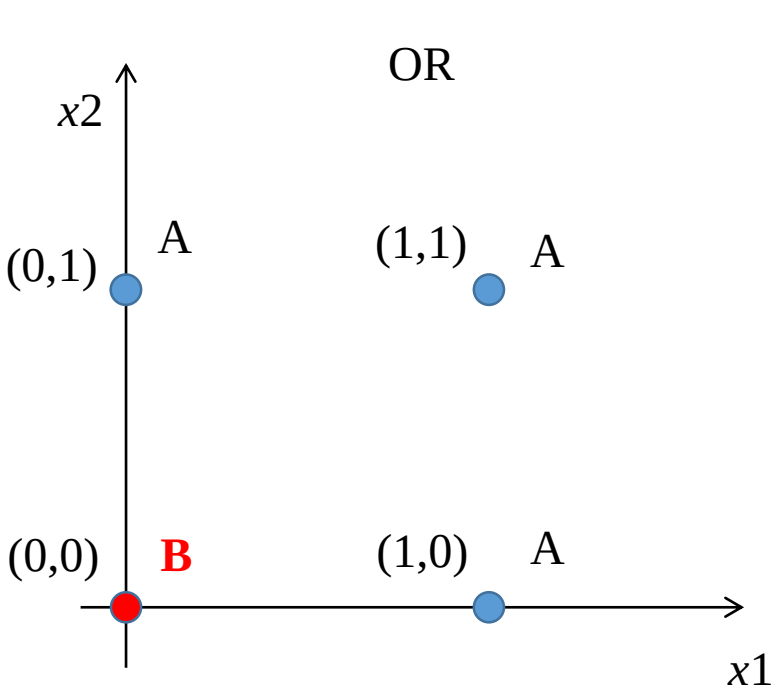
權重(w_i): Dendrite

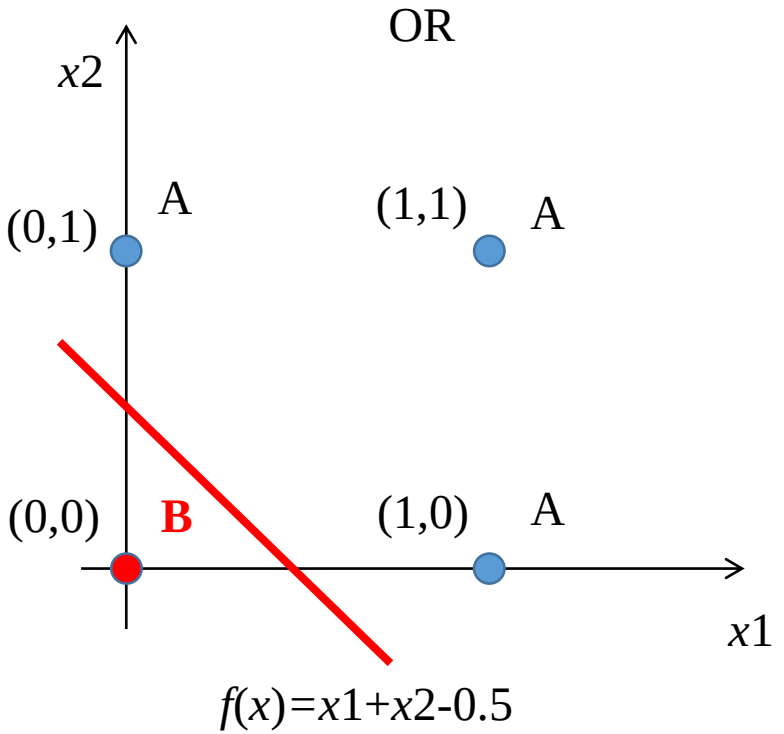
Input(x_i) and output (y) node: Synapse

Node: Cell body

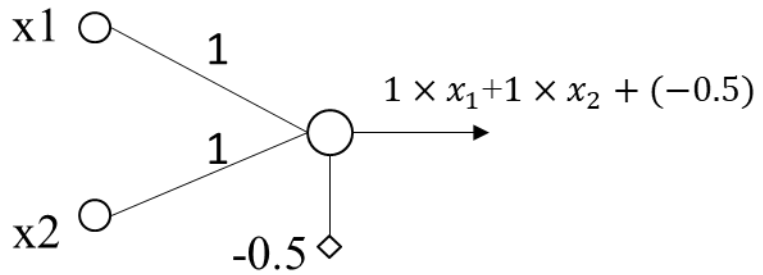
Output: Axon

NN for classification problem



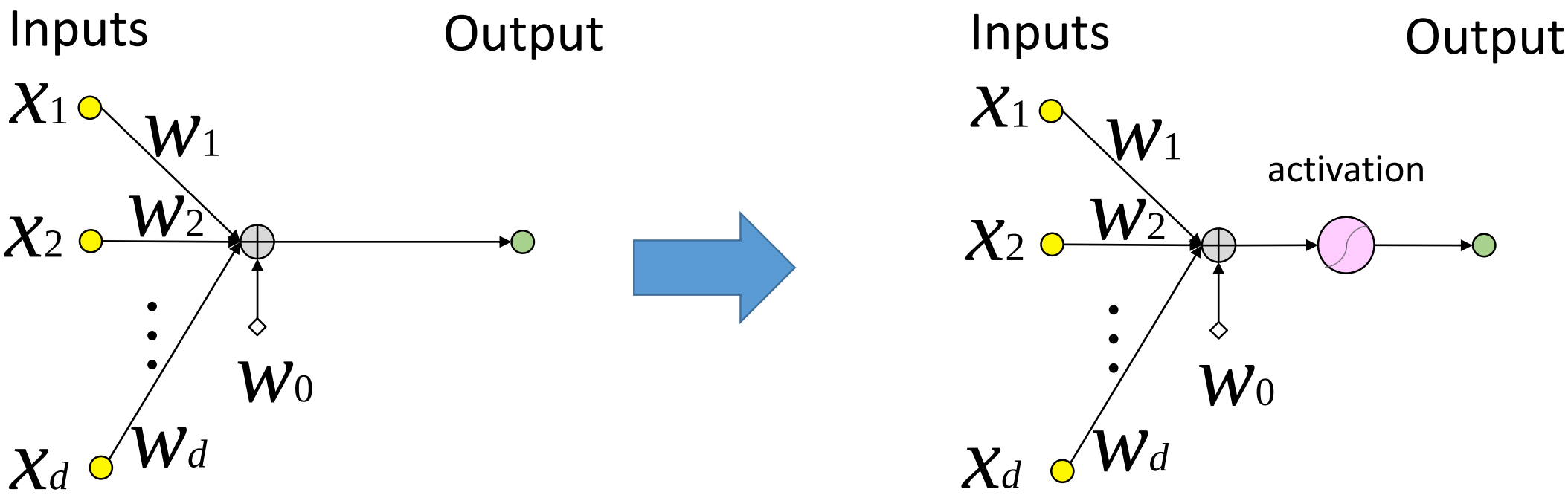


$f(x)=x1+x2-0.5$

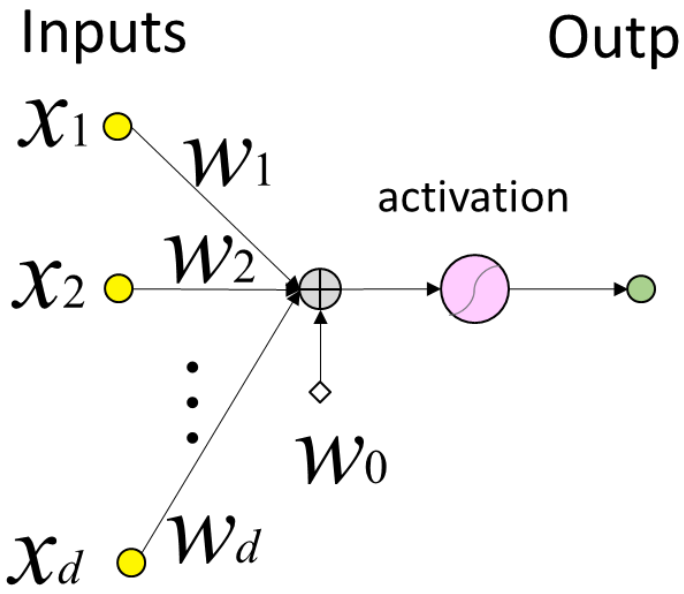


Perception

Perception can learn the nonlinear representation by activation function.



Perception



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

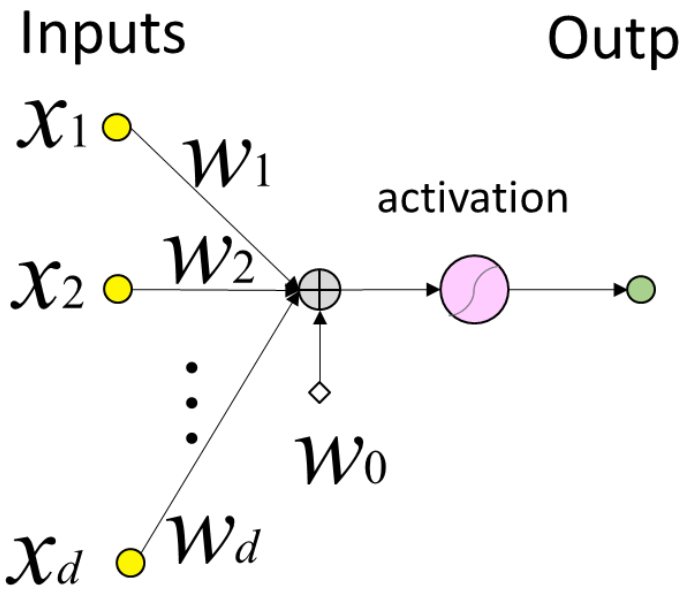
Classification: $f = \begin{cases} 1 & \boldsymbol{W}^T\boldsymbol{x} \geq 0 \\ 0 & O.W. \end{cases}$

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.

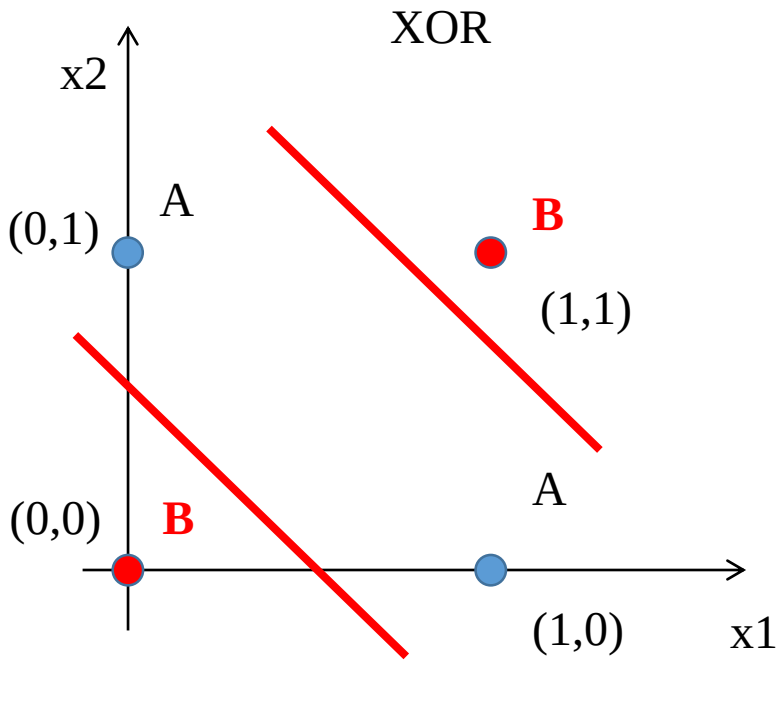
NN: backpropagation

Regression: OLSE (ordinary least squares estimator).

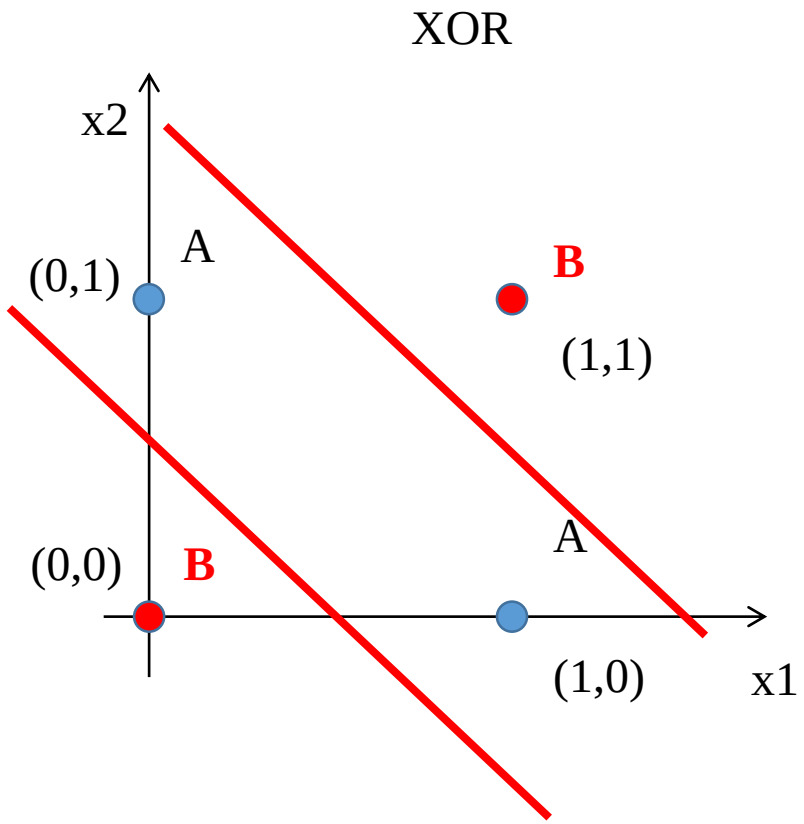
NN for XOR problem

- 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	XOR	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	0	B



NN for XOR problem



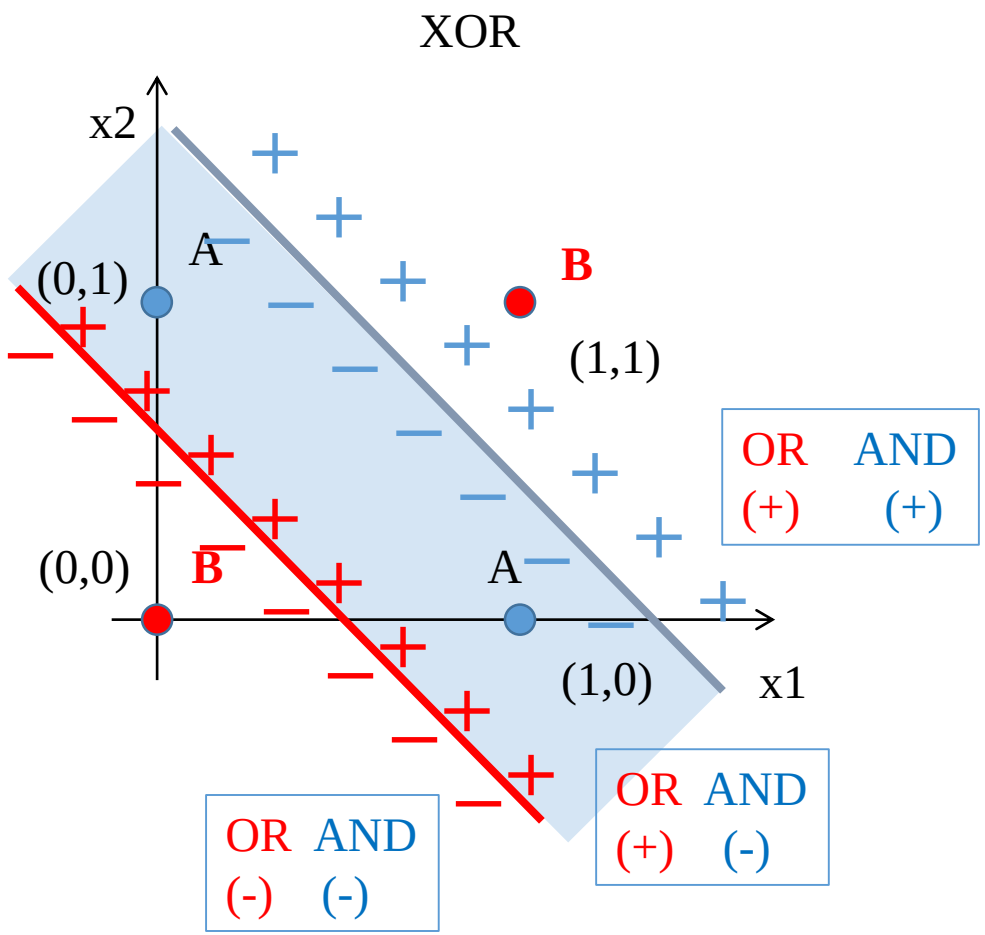
OR

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

AND

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

NN for XOR problem



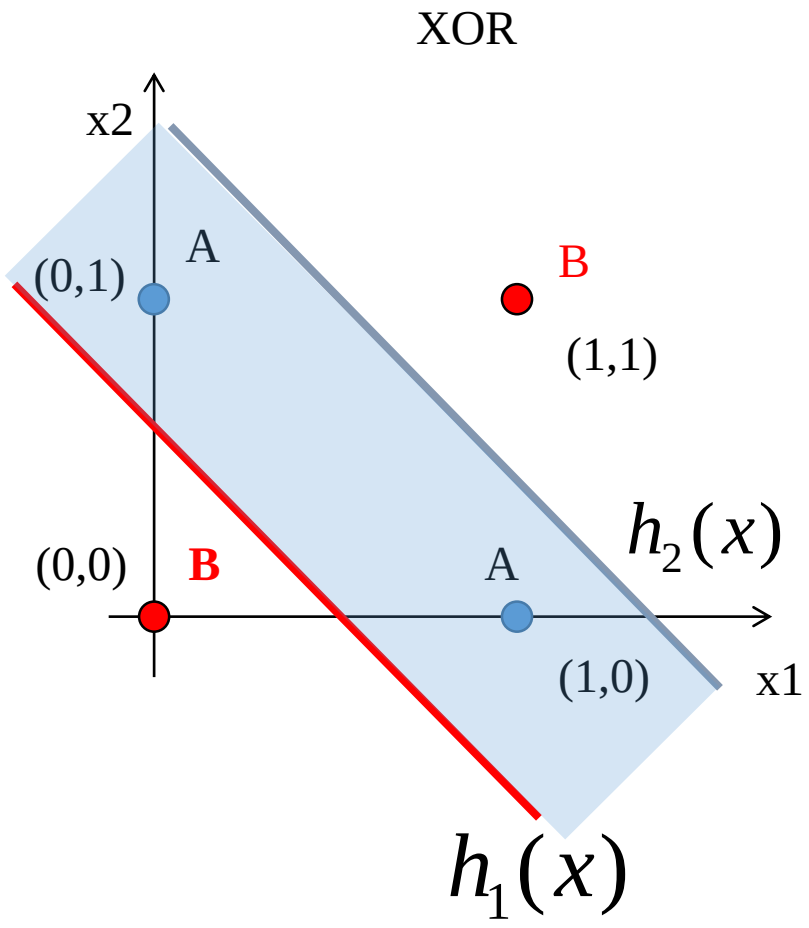
OR

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

AND

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

NN for XOR problem



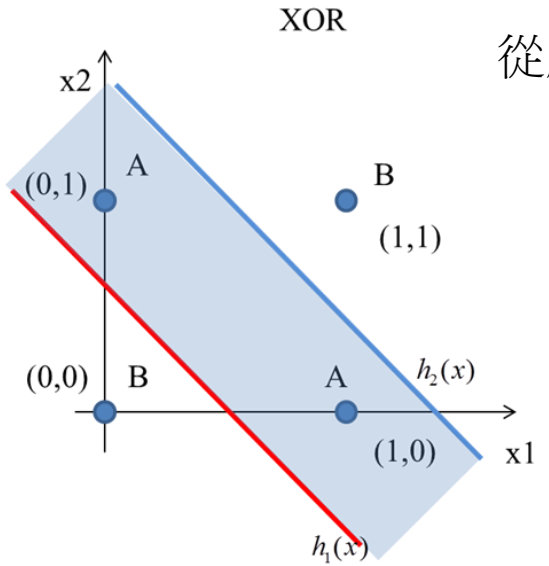
OR

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

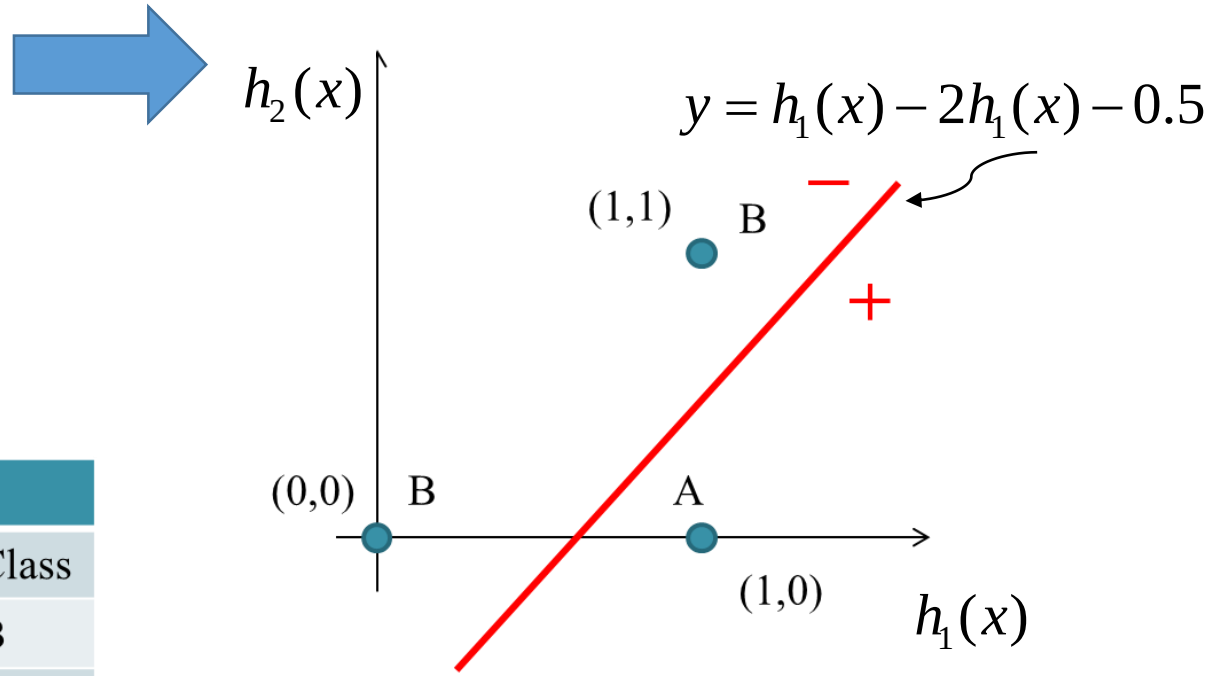
AND

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

NN for XOR problem



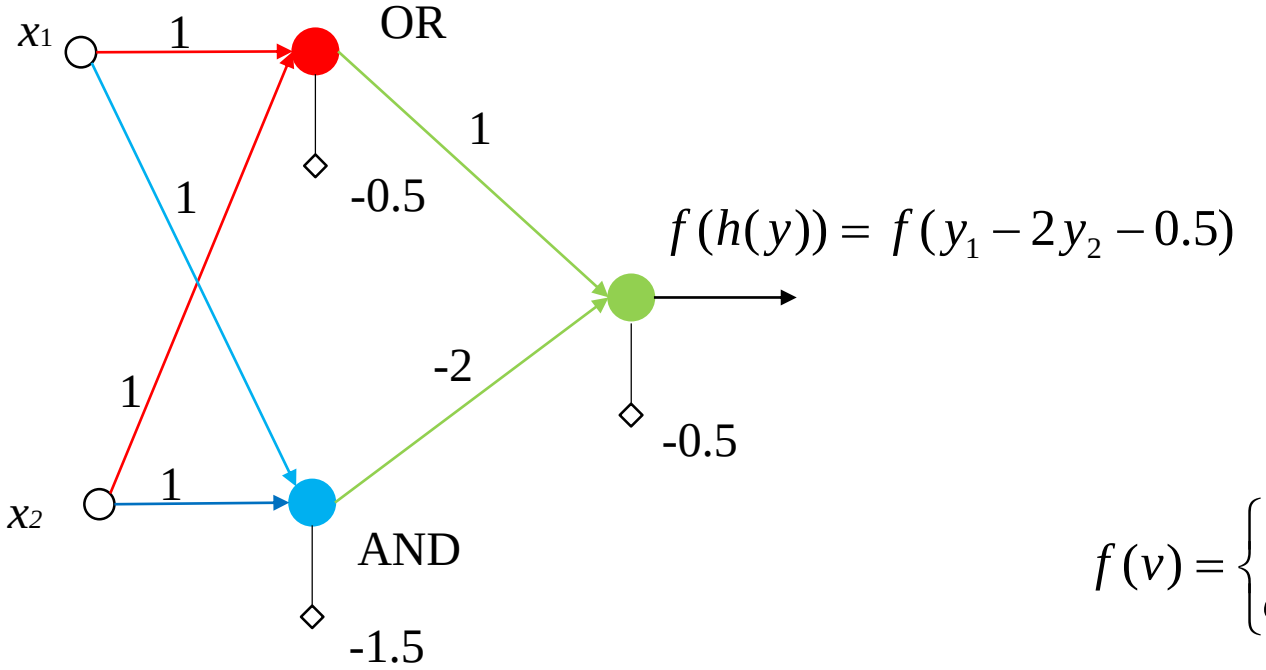
從原始特徵空間做特徵轉換(特徵萃取)



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

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



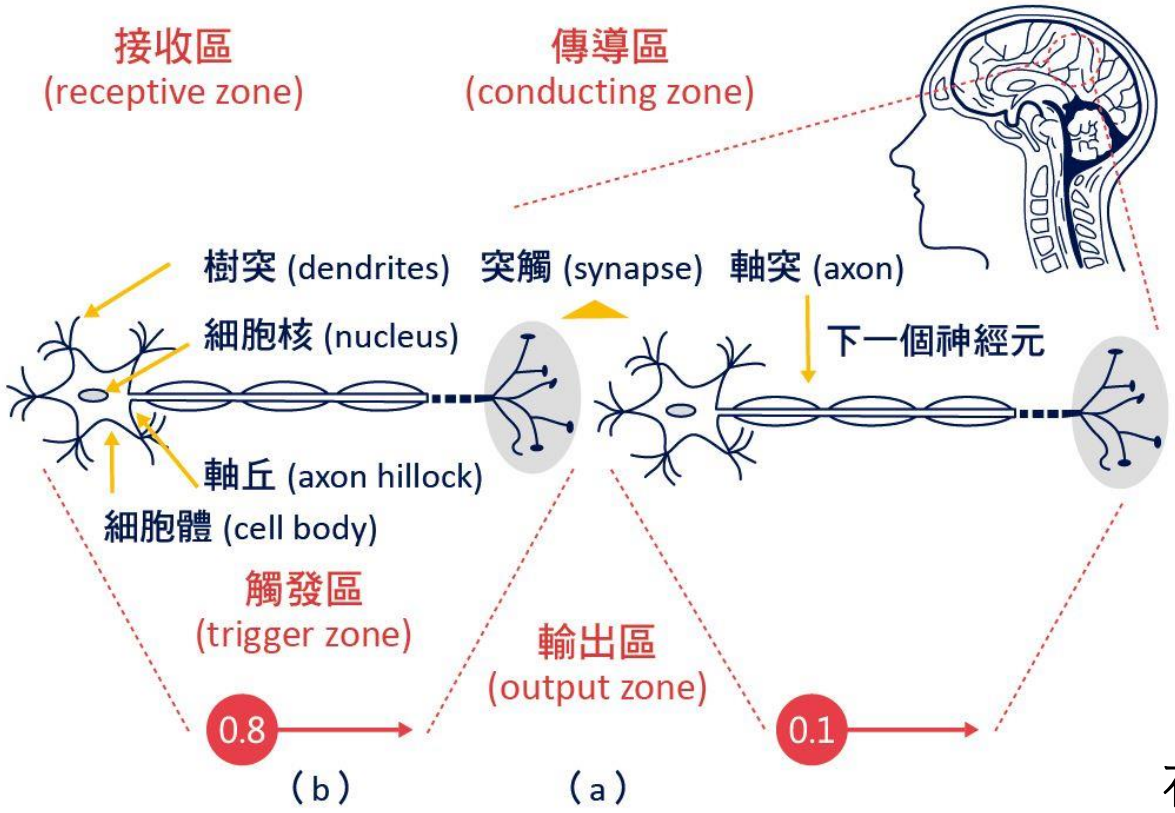
$$y_2 = f(h_2(x)) = f(x_1 + x_2 - 1.5)$$

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

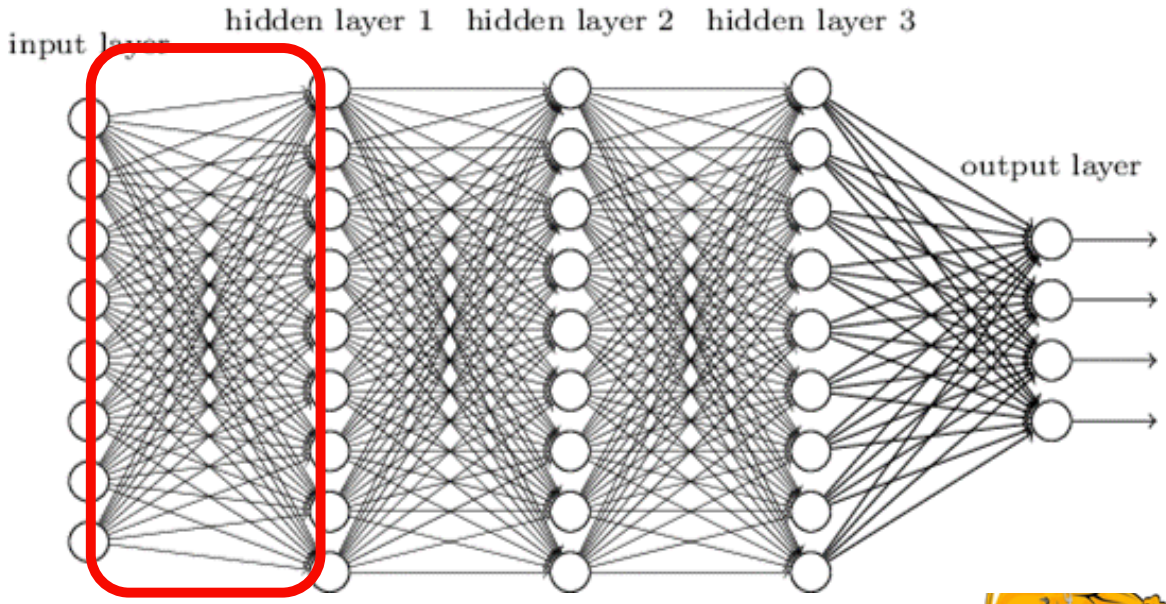
Step activation function

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。



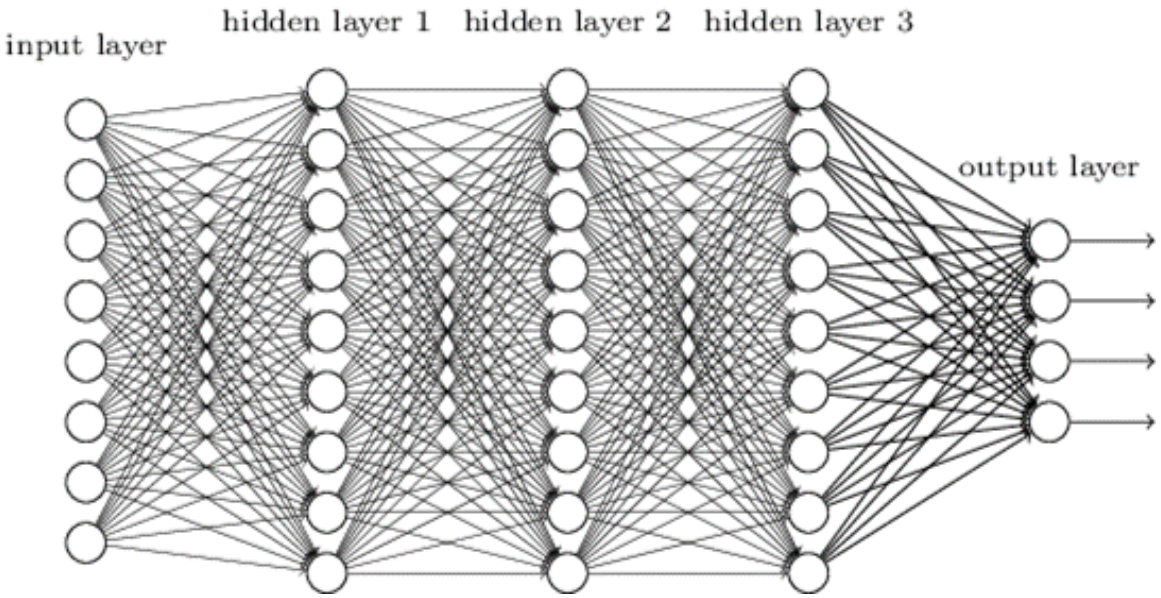
MLP



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



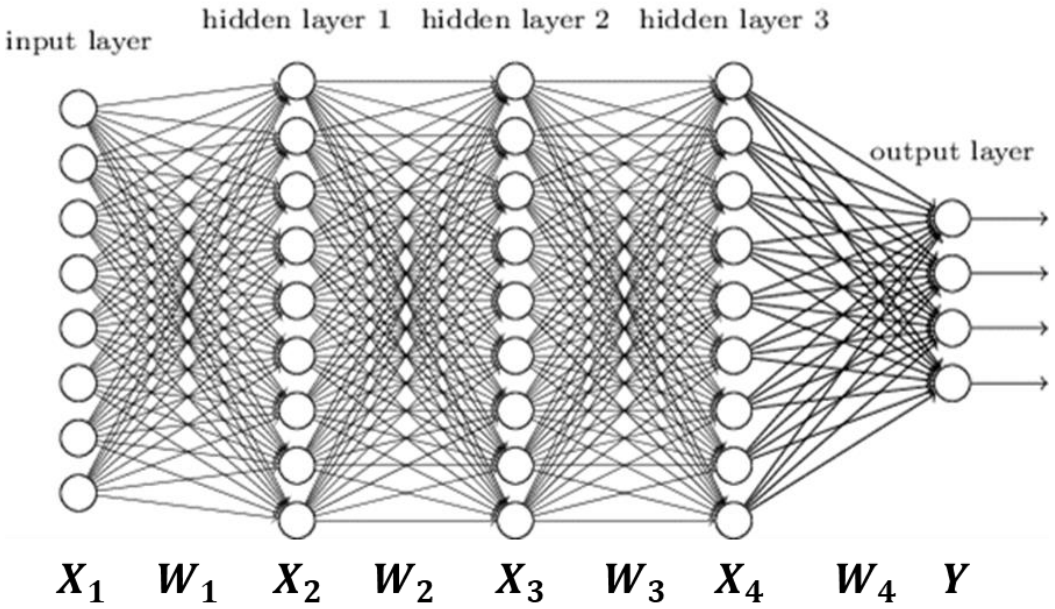
$$X_2 = W_1 X_1$$
$$X_3 = W_2 X_2$$
$$X_4 = W_3 X_3$$
$$Y = W_4 X_4$$

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

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



$$X_2 = W_1X_1$$
$$X_3 = W_2X_2$$
$$X_4 = W_3X_3$$
$$Y = W_4X_4$$

$$X_2 = f(W_1X_1)$$
$$X_3 = f(W_2X_2)$$
$$X_4 = f(W_3X_3)$$
$$Y = f(W_4X_4)$$

$$Y = W_4W_3W_2W_1X_1$$

$$Y = f\left(W_4f\left(W_3f(W_2f(W_1X_1))\right)\right)$$

$$\underline{Y = WX_1}$$

$$W = W_4W_3W_2W_1X_1$$

Activation Function

- Sigmoid、Tanh。

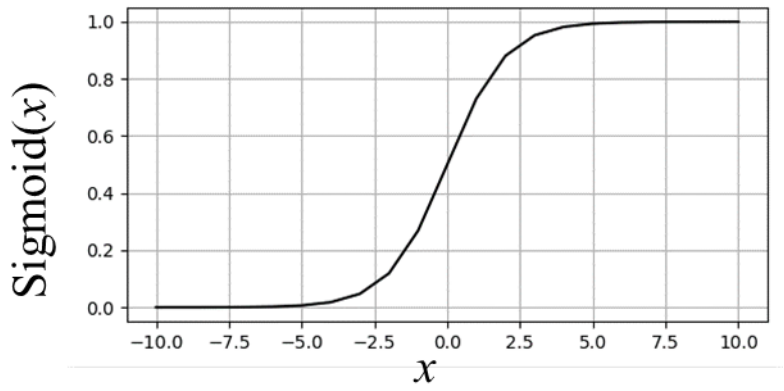
ReLU(Rectified Linear Unit)系列：

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

Activation function

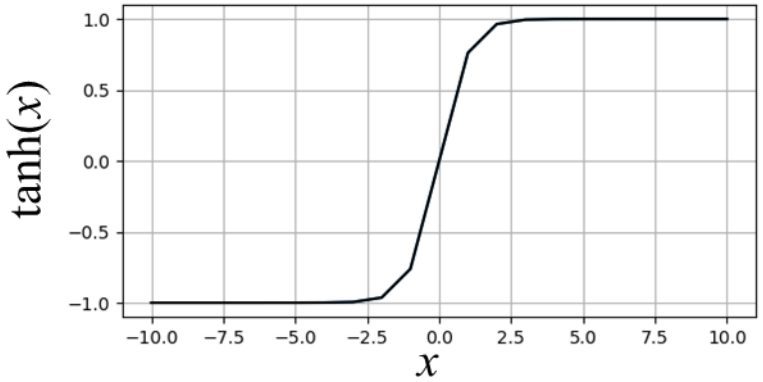
- $\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$

將值壓到0~1之間



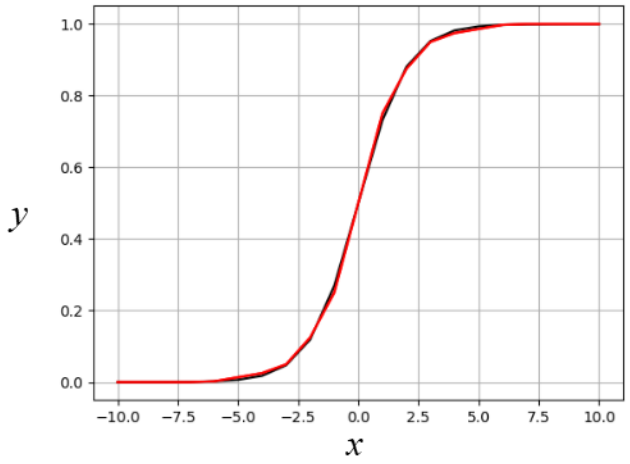
- $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

將值壓到-1~1之間



Sigmoid近似法

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



Black: sigmoid
Red: approximate sigmoid

```python
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.)
```

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)
- $\mathrm{ELU}(x) = \max(0, x) + \min(0, \alpha * (e^x - 1)) = \begin{cases} x & if\ x > 0 \\ \alpha * (e^x - 1) & O.W. \end{cases}$
- $\mathrm{SELU}(x) = \mathrm{scale}\big(\max(0, x) + \min\big(0, \alpha * (e^x - 1)\big)\big) = \mathrm{scale} \begin{cases} x & if\ x > 0 \\ \alpha * (e^x - 1) & O.W. \end{cases}$

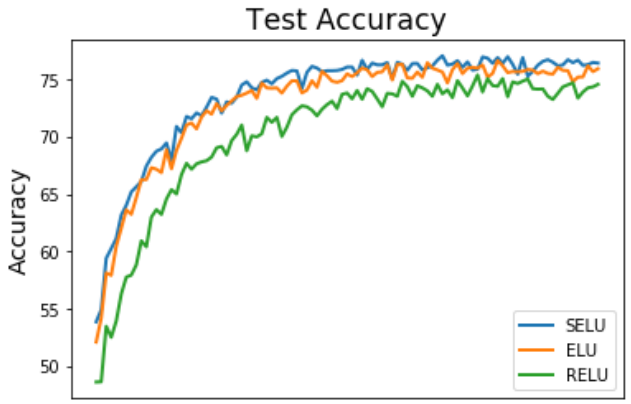
  α=1.67326324235437728481704299167 17
  scale=1.05070098735548049341933498529 46

  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

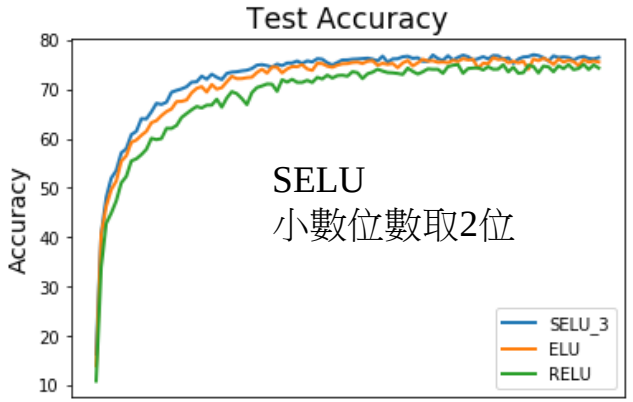


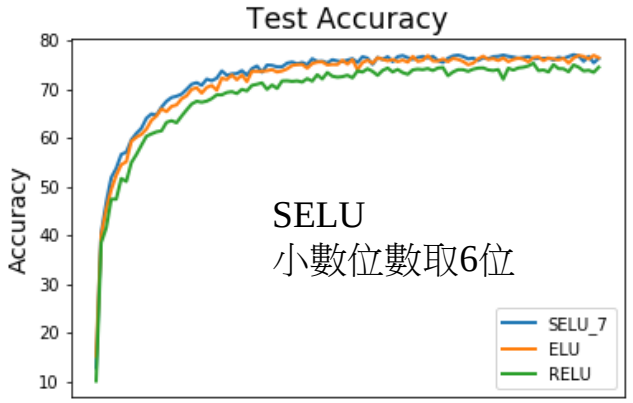
$$\mathrm{SELU}(x) = \mathrm{scale}\big(\max(0,x) + \min\big(0, \alpha * (e^x - 1)\big)\big)$$

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

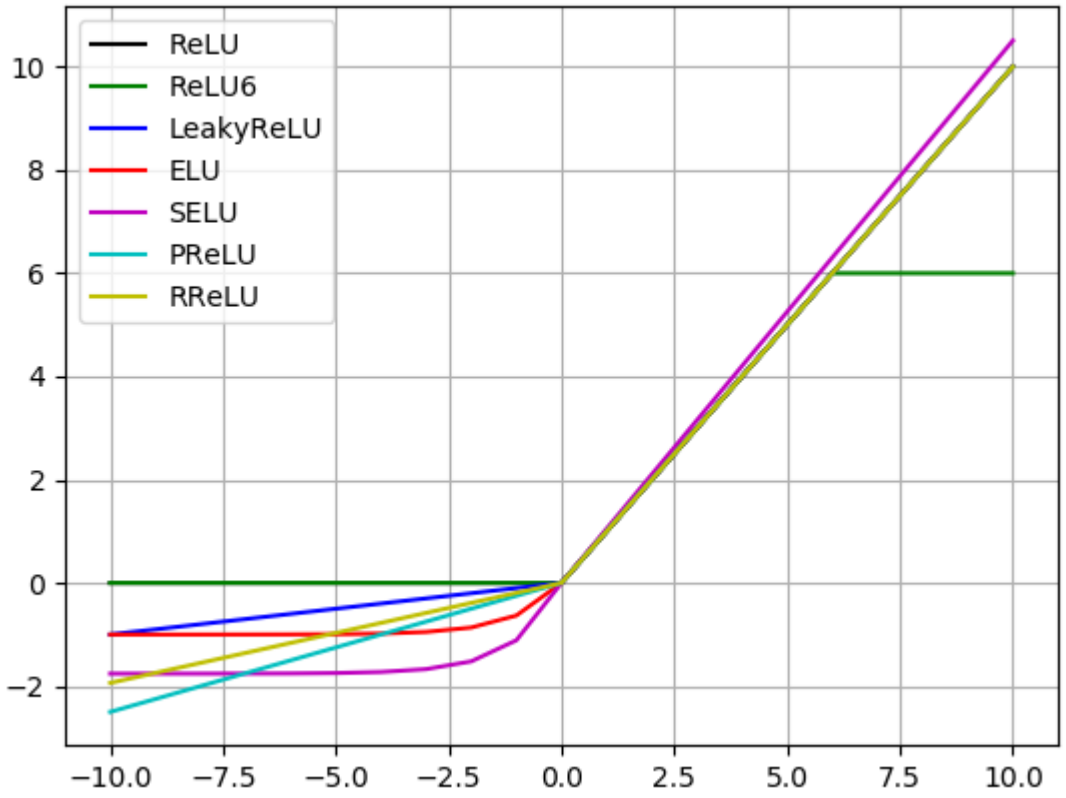
α=1.67326324235437728481704299167 17

scale=1.05070098735548049341933498529 46

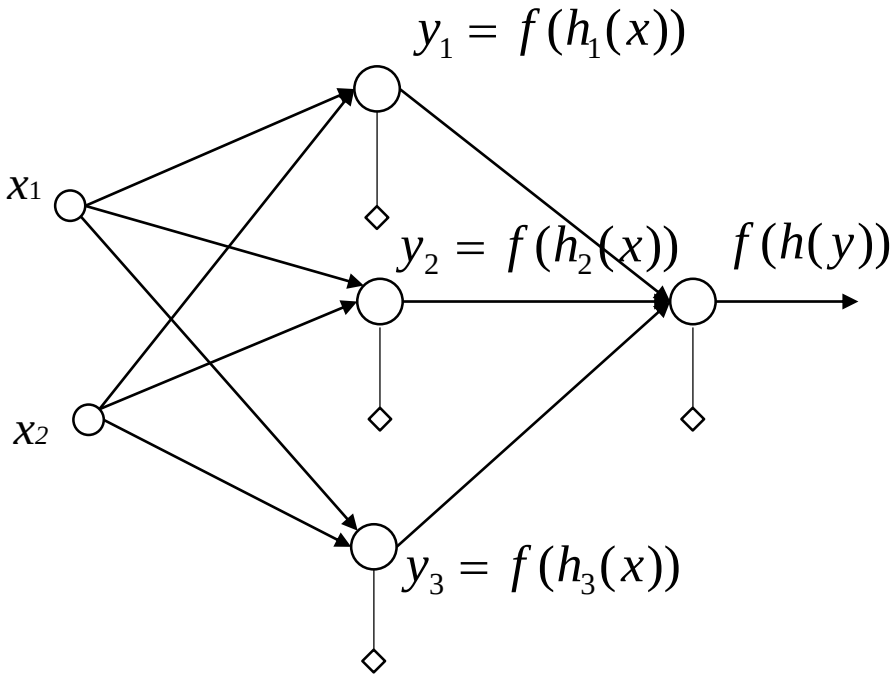




Activation function: ReLU系列

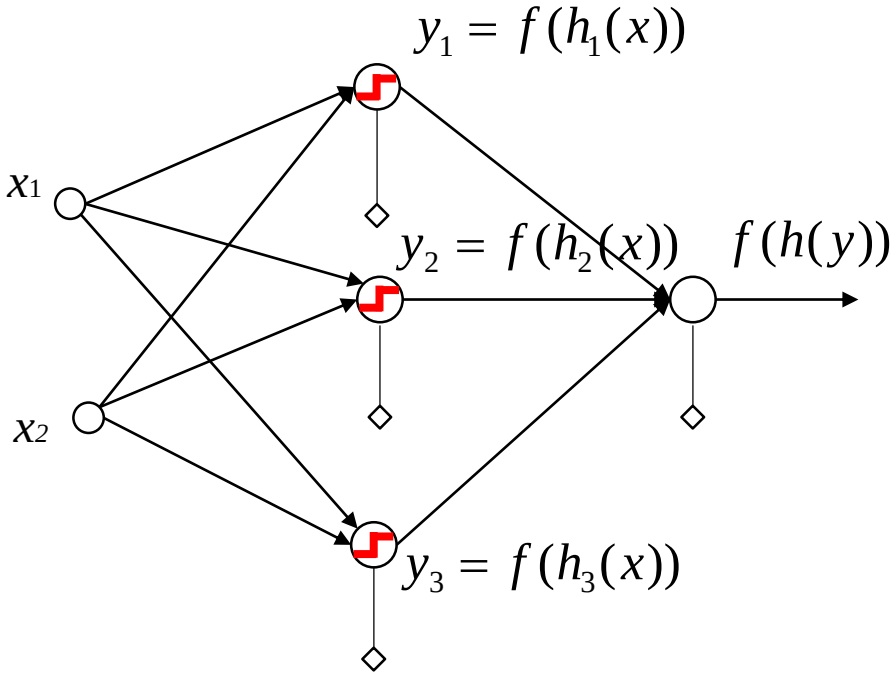


Polyhedral Regions

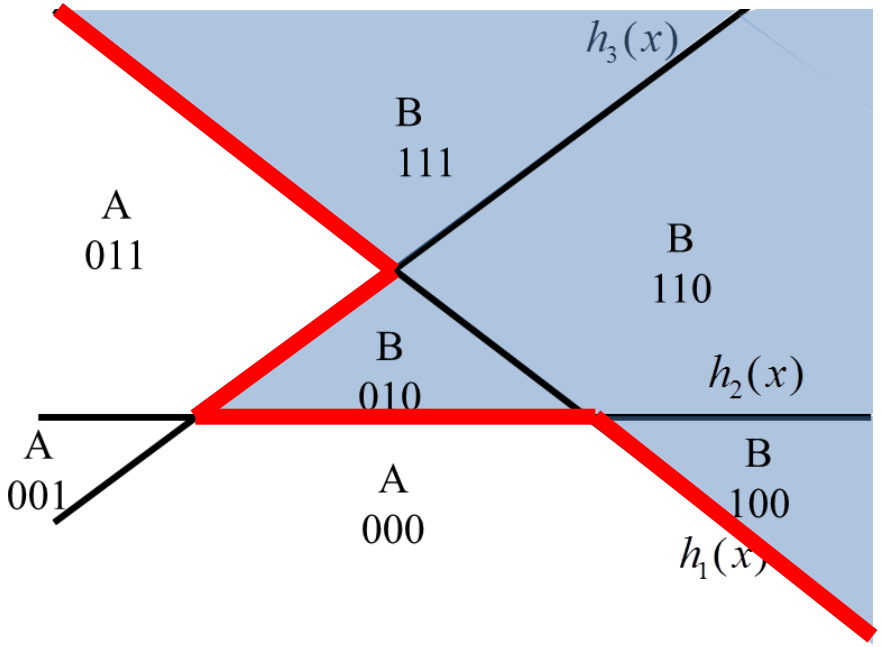


Polyhedral Regions

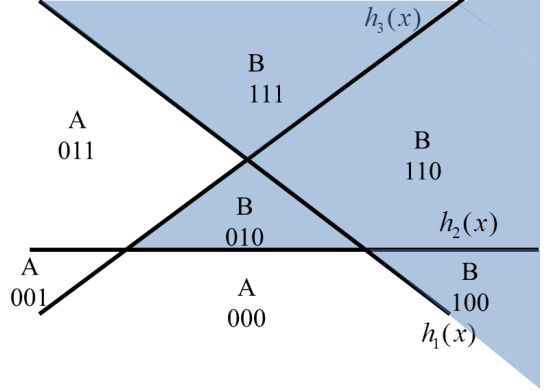
activation function = step function

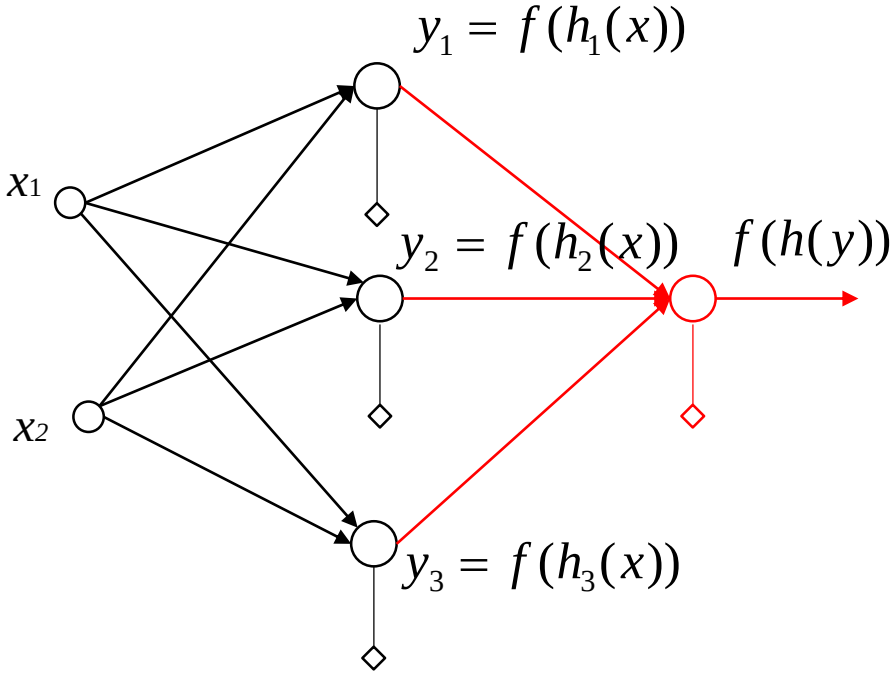


The first layer of neurons divides the input d-dimensional space into polyhedral, which are formed by hyperplane intersections.

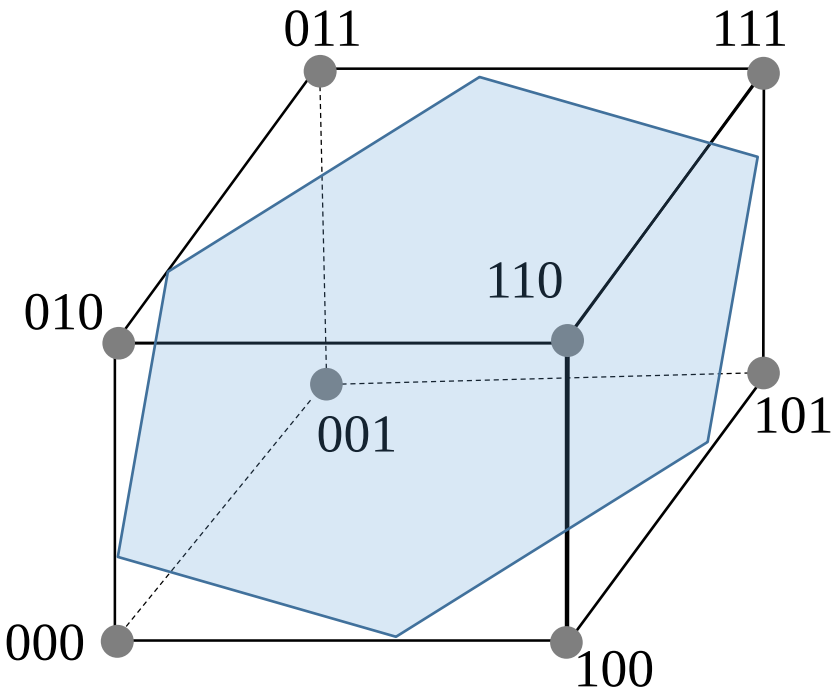


Polyhedral Regions

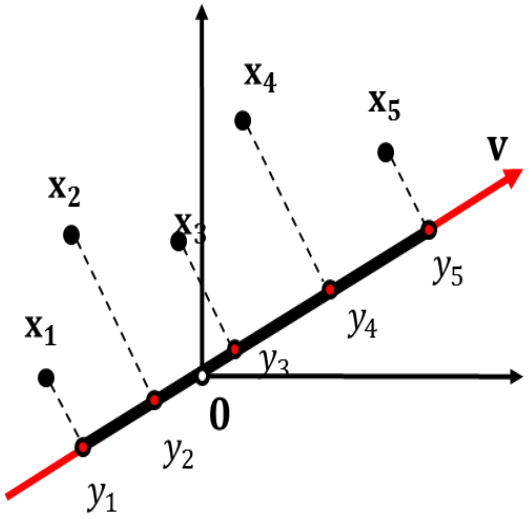


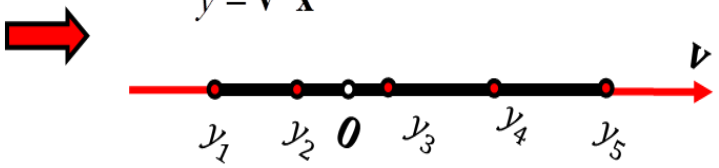


All vectors located within one of these polyhedral regions are mapped onto a specific vertex of the unit hypercube.



Dimension Reduction(Feature Projection)



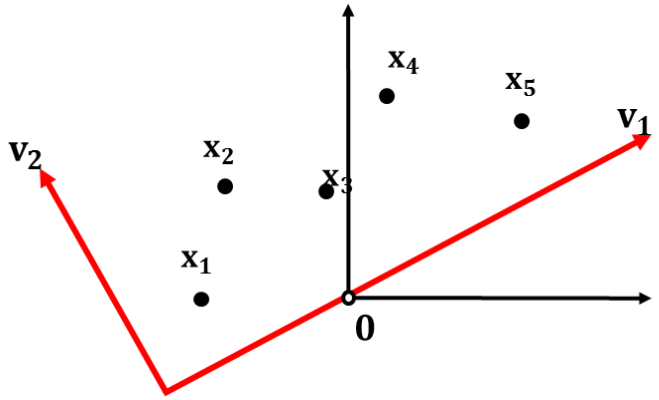


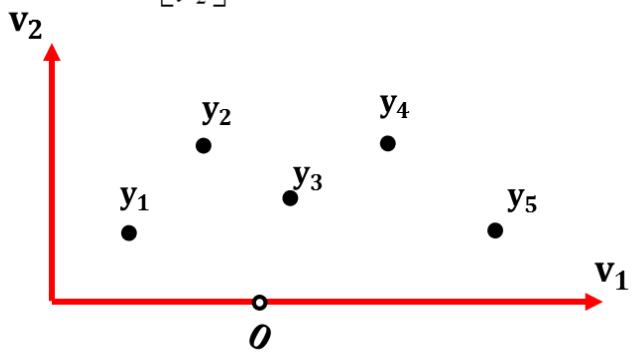
X空間轉換到特徵空間

$$y = \mathbf{v}^T \mathbf{x}$$

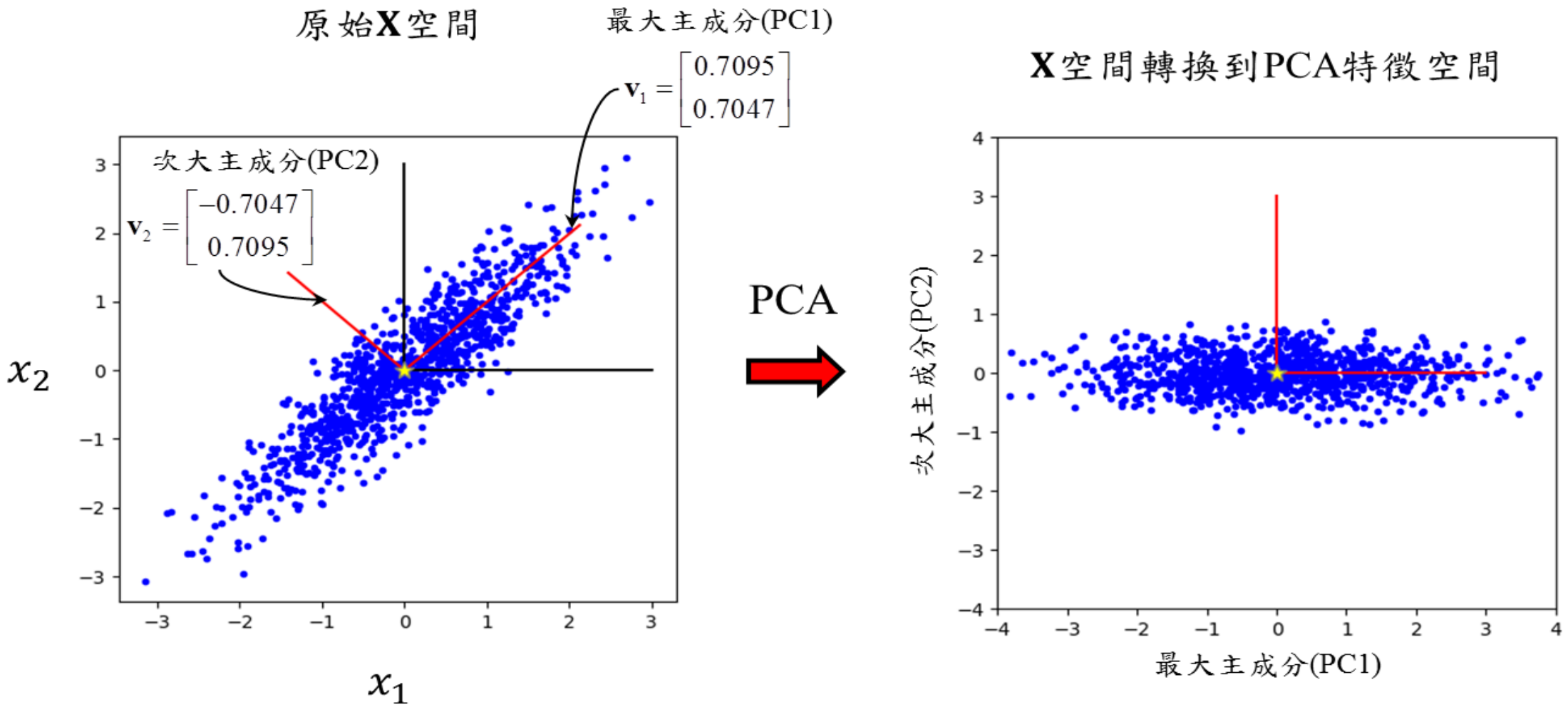
X空間轉換到Y空間

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathbf{V}^T \mathbf{x} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} \mathbf{x}$$

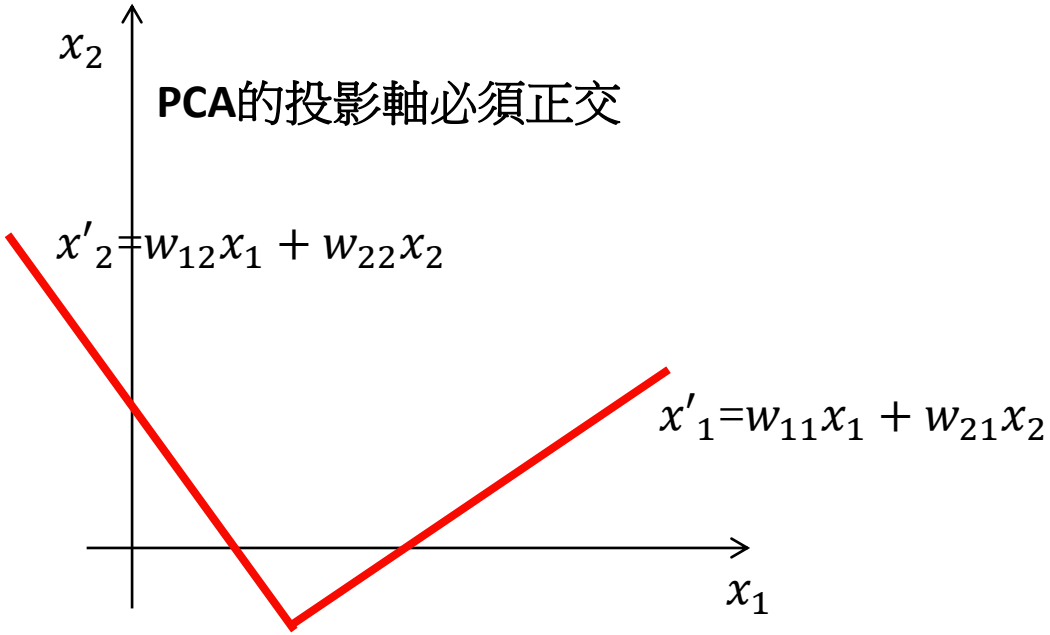




Principle Component Analysis



NN and Dimension Reduction



PCA的投影軸必須正交

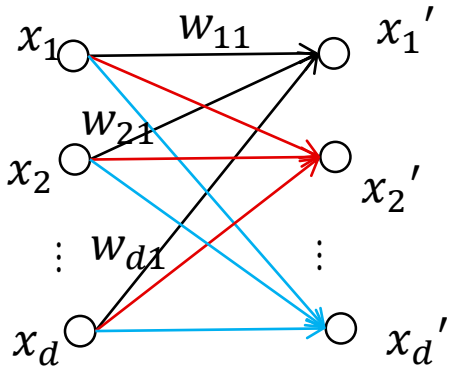
$x'_2 = w_{12}x_1 + w_{22}x_2$

$x'_1 = w_{11}x_1 + w_{21}x_2$

$$\boldsymbol{x'} = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}^{\boldsymbol{T}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boldsymbol{W^T x}$$

拓展為d維

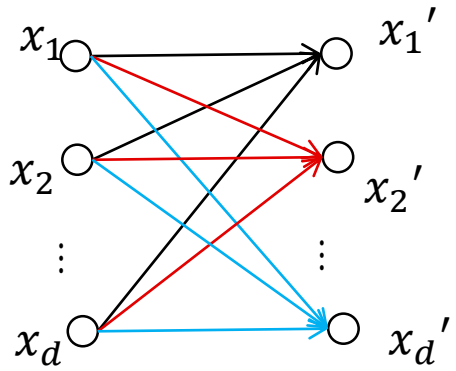
$$\boldsymbol{x'} = \begin{bmatrix} x_1' \\ \vdots \\ x_d' \end{bmatrix} = W^T \boldsymbol{x} = \begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{d1} & \cdots & w_{dd} \end{bmatrix}^{\boldsymbol{T}} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

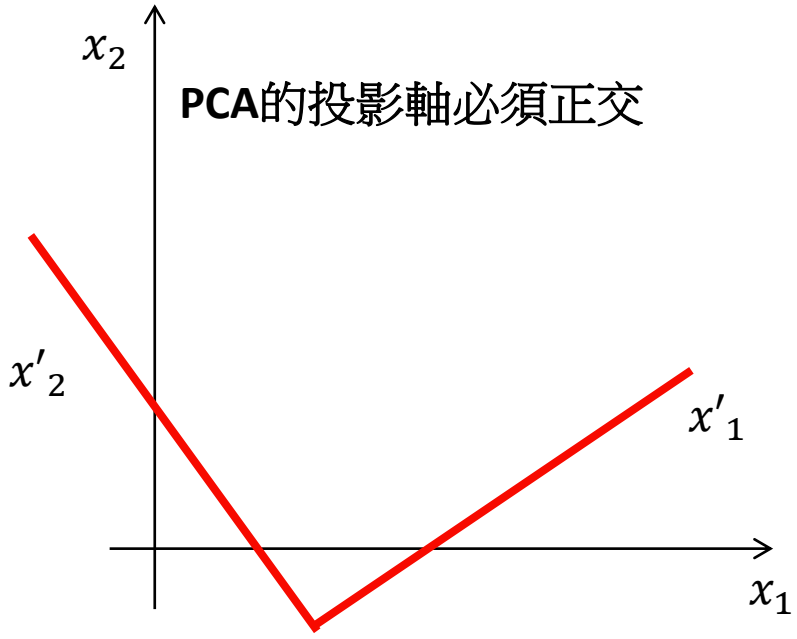


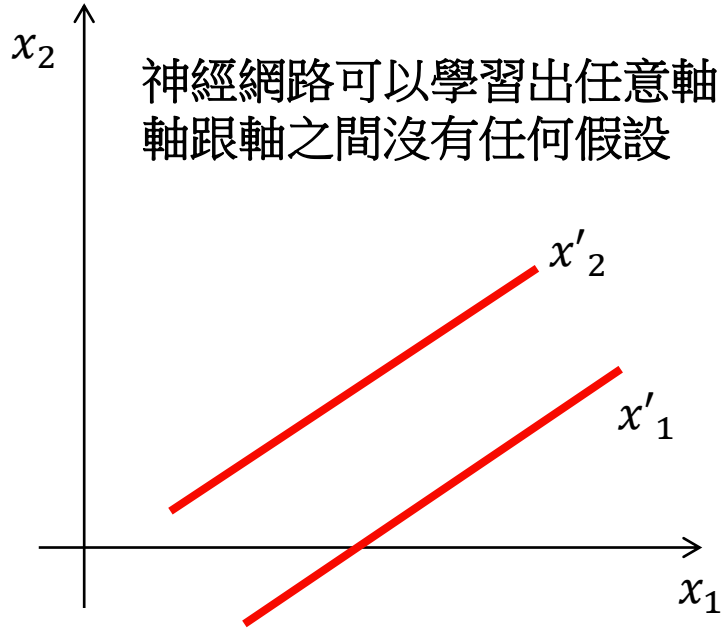
NN and Dimension Reduction

- **Principle Component Analysis**

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



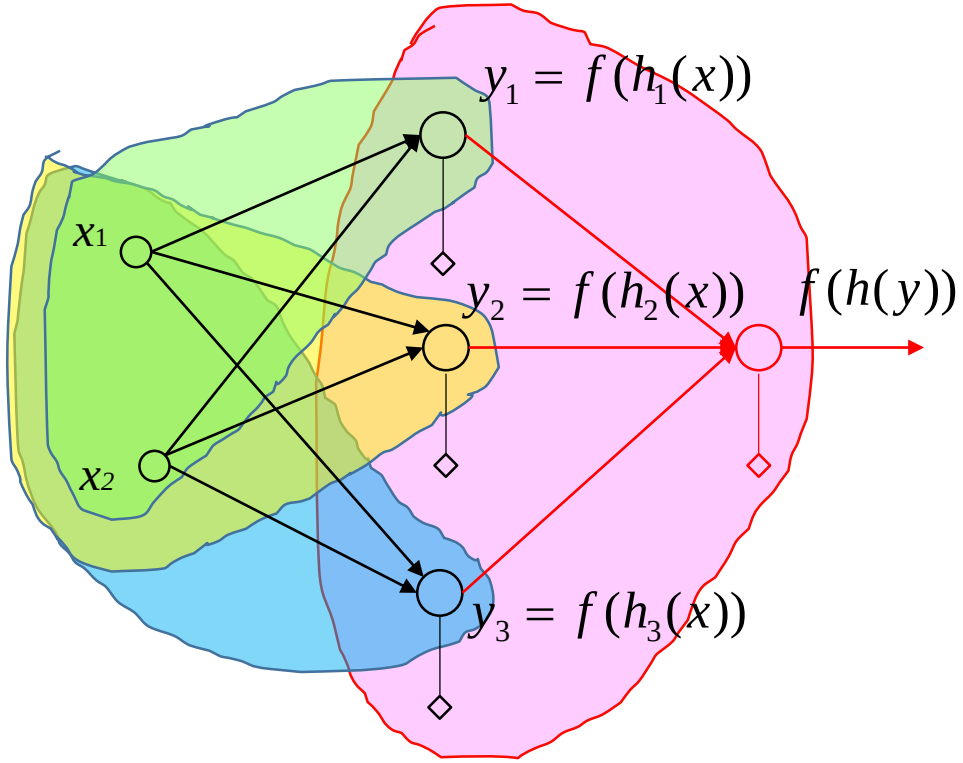




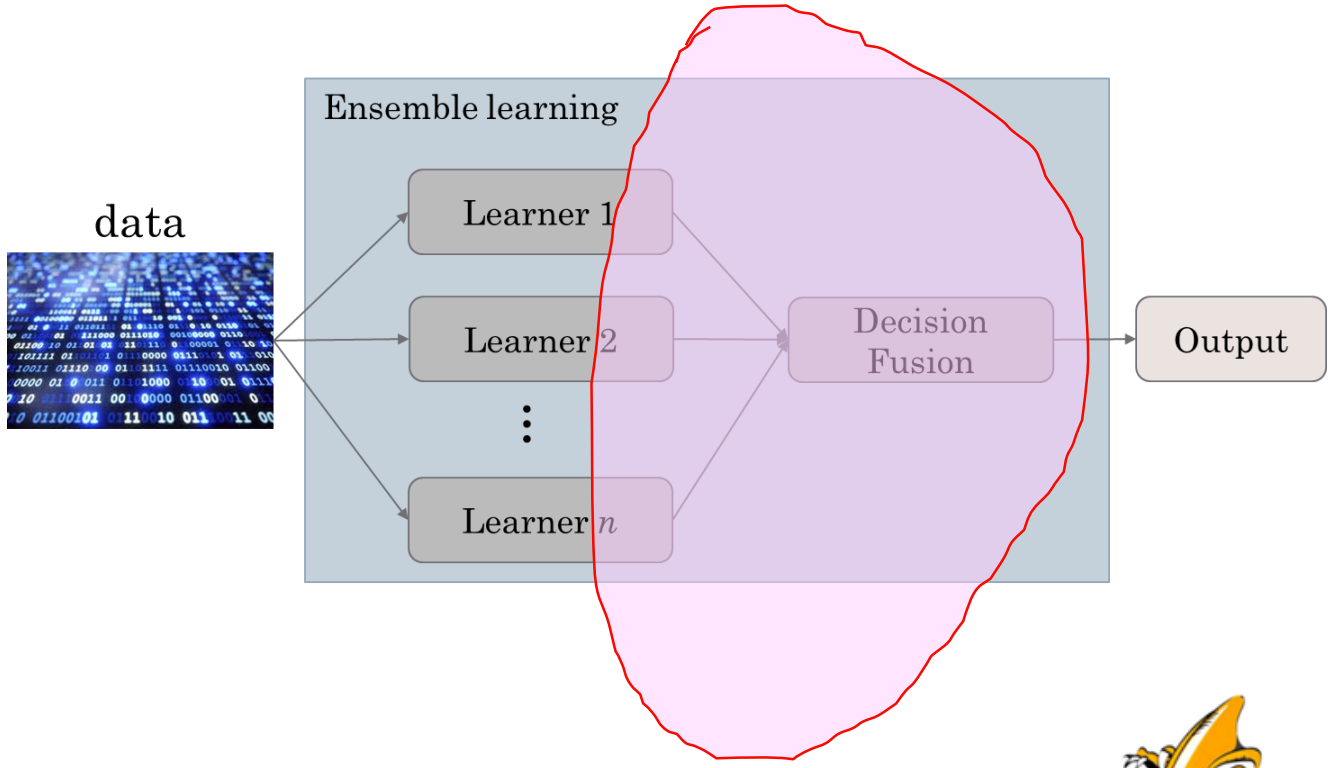
PCA是NN的一個special case

NN and Ensemble Learning

NN



Ensemble Learning



Introduction (loss function)

In learning algorithm, there is an assumption, which may accompany with an object function.

Regression: minimizing the mean square errors (MSE)

K-mean: minimizing the mean of square error between data and centers.

PCA: maximizing the variance of project data.

SVM: maximizing the margin.

Sometimes the same model but different object function can lead different results. (Linear regression and ridge regression)

Regression

$$\min_{\beta}\{\mathrm{MSE}(\hat{y}, y)$$

Ridge regression

$$\min_{\beta}\{\mathrm{MSE}(\hat{y}, y) + \lambda L_2\mathrm{norm}(\beta)\}$$

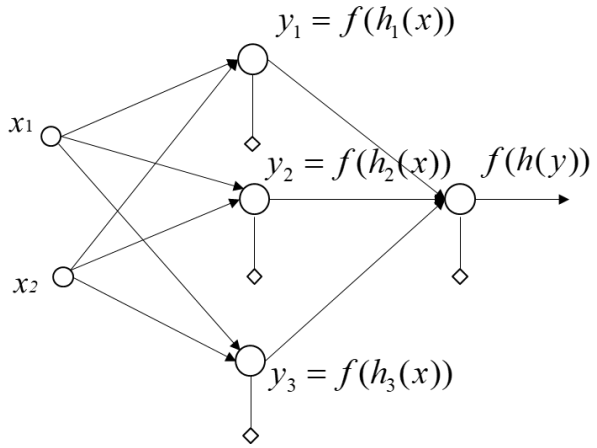
$$L_2\mathrm{norm}(\beta) = \sum_i \beta_i^2$$

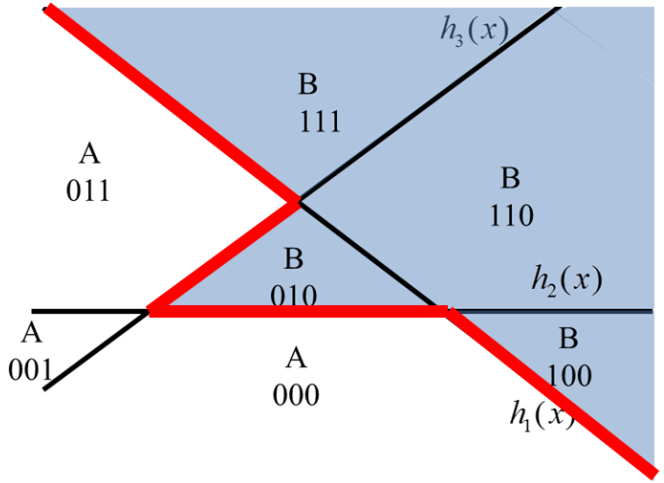
Neural Network (Number of output node)

Regression case: Number of output node of NN architecture is 1.

Classification case: Number of output node of NN architecture is *L* (Number of class in define problem). ->**memberships (call logits in DL) for each class (such as posterior probability in ML).**

Q: Why number of output node in this toy case is 1?

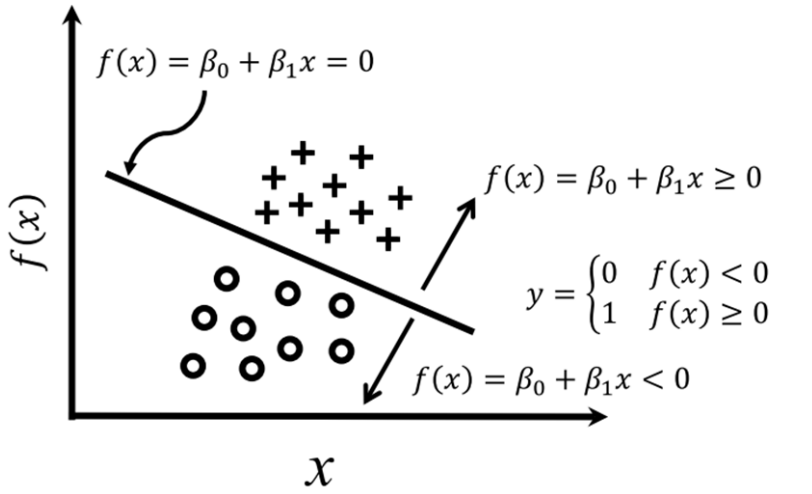




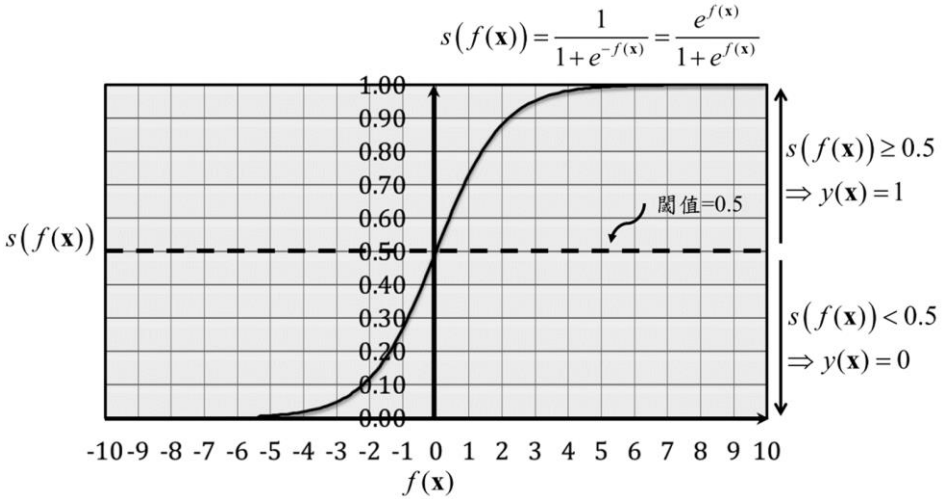
ANS: It's a binary classification problem.
A simple threshold can be defined as decision boundary, see the Logistic regression.

Neural Network
(Number of output node)

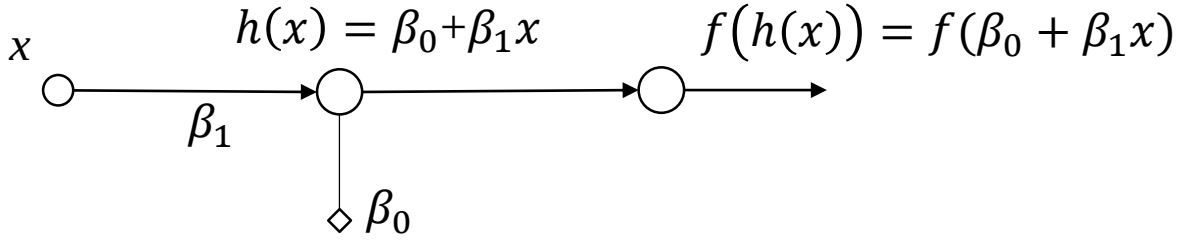
羅吉斯迴歸



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



Neural Network



Logistic Regression	NN	Activation
$f(x)$	$h(x)$	None
$\sigma(f(x))$	$f(h(x))$	Sigmoid

Introduction

1. Regression

   MSE, MAE, Huber Loss

2. Classification

   Cross entropy, Focal loss

3. Triple loss

Residual

- Residual: predicted value v.s. target value.

Regression:

$$y - \hat{y}$$

Classification (error):

$$sign(\hat{y}, y) = \begin{cases} 1 & \hat{y} = \hat{y} \\ 0 & \hat{y} \neq \hat{y} \end{cases}$$

$$error\ rate = \frac{1}{n}\sum_{i=1}^{n} sign(\hat{y}_i, y_i)$$

MSE & MAE

Mean Square Error (MSE)

Mean Absolute Error (MAE)

Why square or absolute?

Target value: $y_1 = 0, y_2 = 1$

Predicted value: $\hat{y}_1 = 100$, $\hat{y}_2 = 99$

$$\text{Residual 1}=y_1 - \hat{y}_1 = 0 - 100 = -100$$

$$\text{Residual 2}=y_2 - \hat{y}_2 = 1 - (-99) = 100$$

$$\text{Residual 1+ Residual 2} = -100 + 100 = 0$$

MSE & MAE

Mean Square Error (MSE)

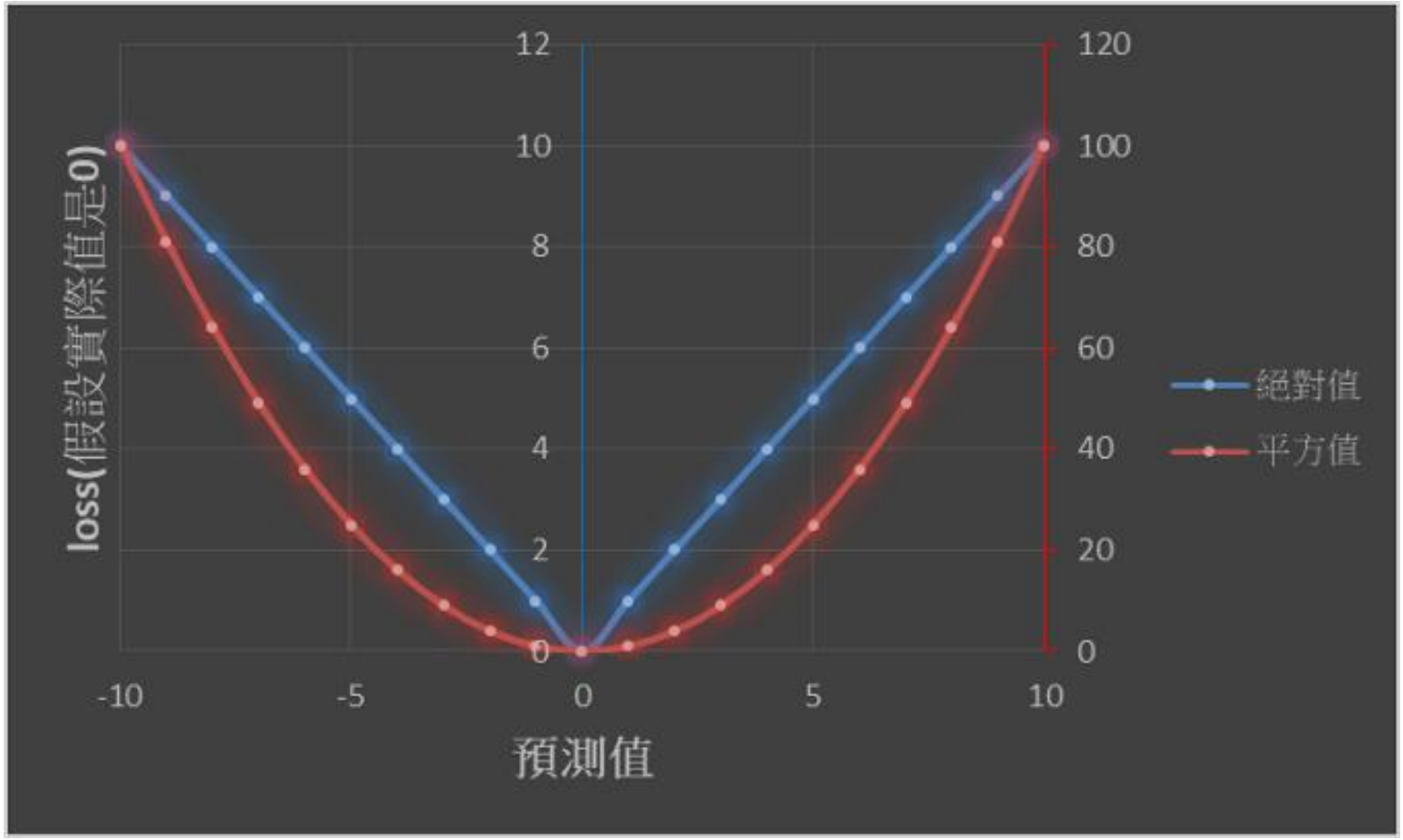
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MSE & MAE

Trend of residual (target value =0)



Comparison (MSE&MAE)

With a fair baseline, RMSE (root MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Comparison (MSE&MAE)

ID	residual	| residual |	residual^2
1	-10	10	100
2	-5	5	25
3	0	0	0
4	5	5	25
5	10	10	100
MAE=6, RMSE=7.07			

Change ID5 with a outliner

ID	residual	| residual |	residual^2
1	-10	10	100
2	-5	5	25
3	0	0	0
4	5	5	25
5	100	100	10000
MAE=24, RMSE=45.06			

Problem of MSE: more outlier sensitivity.

Comparison (MSE&MAE)

- Problem of MAE: same gradient value.
- When loss is small, it's difficult to reach the optimal target.

$$f_1(x) = x^2, f_1'(x) = 2x$$

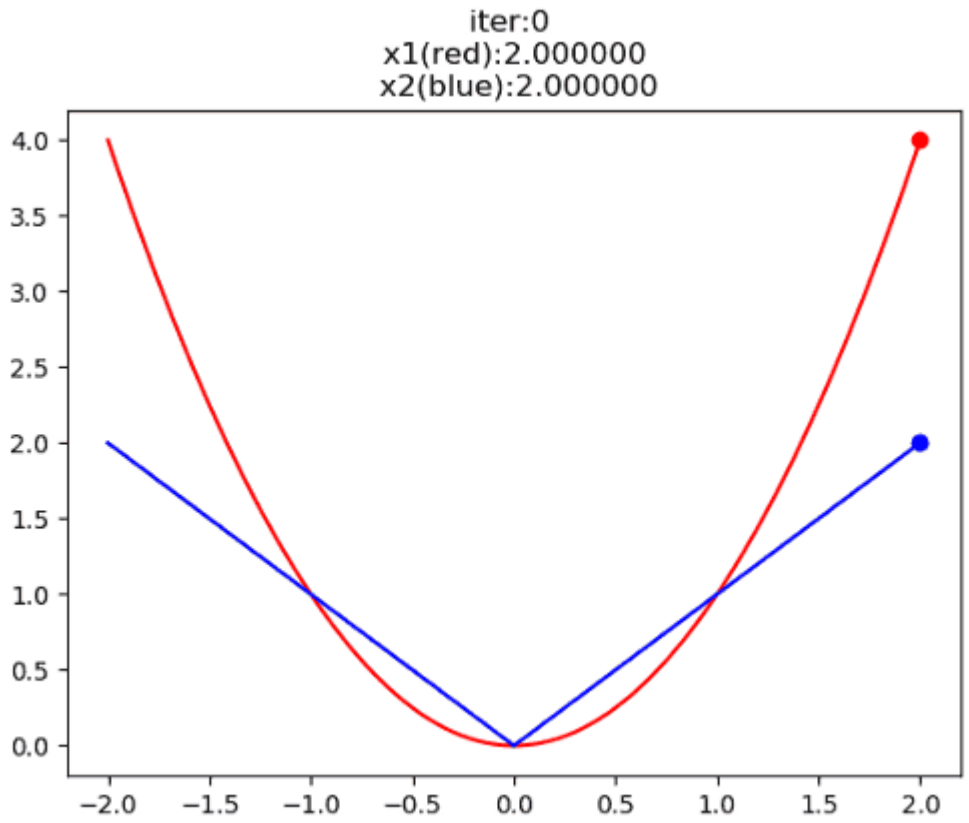
$$f_2(x) = |x|, f_2'(x) = \frac{x}{|x|}$$

Gradient update:

$$x^{t+1} \to x^t - rf'(x)$$

Suppose: $x^0 = 2, \quad r = 0.3$

Comparison (MSE&MAE)



$f(x)=x^2$				$f(x)=\|x\|$			
t	x^t	$f'(x)$	x^{t+1}	t	x^t	$f'(x)$	x^{t+1}
1	2	4	0.8	1	2	1	1.7
2	0.8	1.6	0.32	2	1.7	1	1.4
3	0.32	0.64	0.128	3	1.4	1	1.1
4	0.128	0.256	0.0512	4	1.1	1	0.8
5	0.0512	0.1024	0.02048	5	0.8	1	0.5
6	0.02048	0.04096	0.008192	6	0.5	1	0.2
7	0.008192	0.016384	0.003277	7	0.2	1	-0.1
8	0.003277	0.006554	0.001311	8	-0.1	-1	0.2
9	0.001311	0.002621	0.000524	9	0.2	1	-0.1
10	0.000524	0.001049	0.00021	10	-0.1	-1	0.20
11	0.000210	0.000419	0.000084	11	0.20	1	-0.1
12	0.000084	0.000168	0.000034	12	-0.10	-1	0.20
13	0.000034	0.000067	0.000013	13	0.20	1	-0.10
14	0.000013	0.000027	0.000005	14	-0.10	-1	0.20
15	0.000005	0.000011	0.000002	15	0.20	1	-0.10
16	0.000002	0.000004	0.000001	16	-0.10	-1	0.20
17	0.000001	0.000002	0.000000	17	0.20	1	-0.10
18	0.000000	0.000001	0.000000	18	-0.10	-1	0.20
19	0.000000	0.000000	0.000000	19	0.20	1	-0.10
20	0.000000	0.000000	0.000000	20	-0.10	-1	0.20

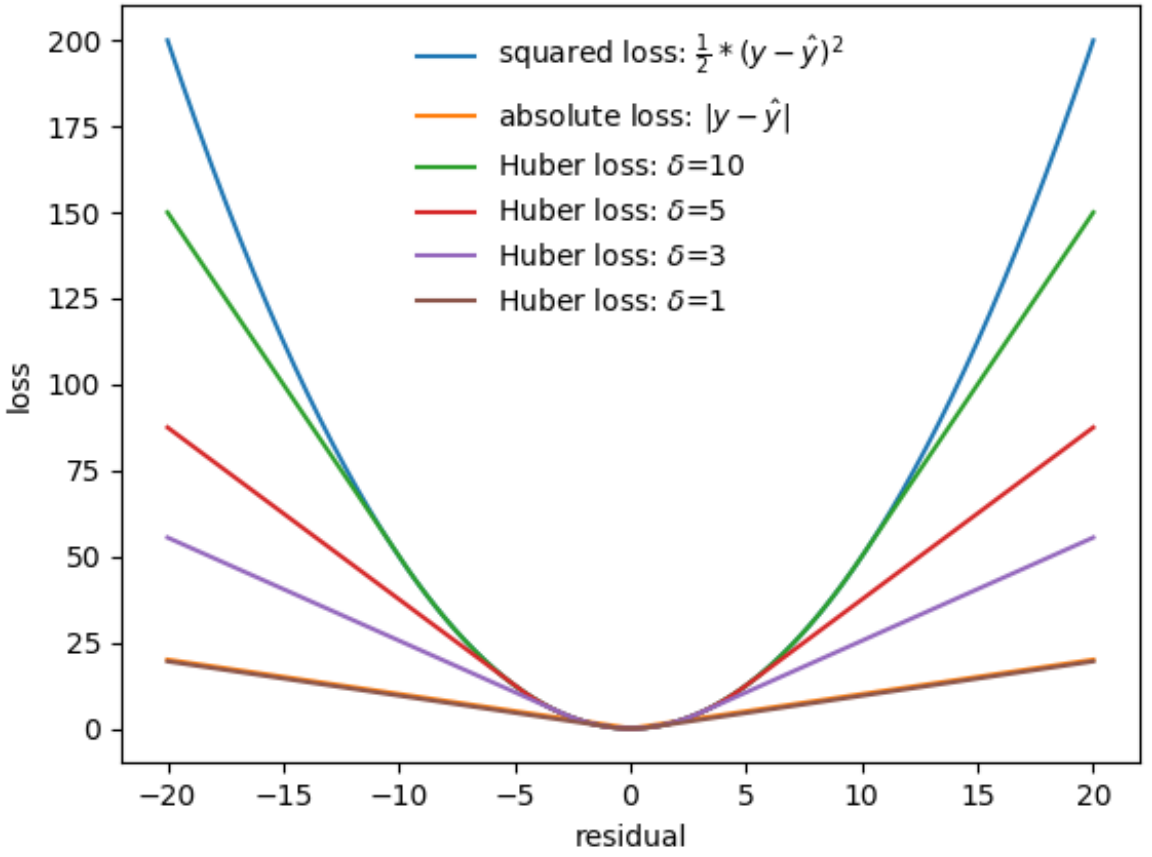
Huber Loss

Huber loss:

$$Loss(y, \hat{y})$$

$$= \begin{cases} \frac{1}{2}(y - \hat{y})^2, & |y - \hat{y}| \leq \delta \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta), & O.W. \end{cases}$$

δ: parameter of Huber loss.



MAE, MSE & Huber Loss

ID	residual	| residual |	residual^2	Huber (δ=1)	Huber (δ=10)
1	-10	10	100	9.5	50
2	-5	5	25	4.5	12.5
3	0	0	0	0	0
4	5	5	25	4.5	12.5
5	10	10	100	9.5	50
MAE=6, RMSE=7.07 MeanHuber(δ=1)=5.6, MeanHuber(δ=10)=25					

ID	residual	| residual |	residual^2	Huber (δ=1)	Huber (δ=10)
1	-10	10	100	9.5	50
2	-5	5	25	4.5	12.5
3	0	0	0	0	0
4	5	5	25	4.5	12.5
5	100	100	10000	99.5	950
MAE=24, RMSE=45.06 MeanHuber(δ=1)=23.6, MeanHuber(δ=10)=205					

Classification

Classification:

$$sign(\hat{y}, y) = \begin{cases} 1 & y = \hat{y} \\ 0 & y \neq \hat{y} \end{cases}$$

$$error\ rate = 1 - \frac{1}{n}\sum_{i=1}^{n} sign(\hat{y}_i, y_i)$$

We hope less error rate more better in classification.

Can we use the classification error rate/accuracy as loss function?

Classification

		Model 1 (輸出)				Model 2 (輸出)			
		機率輸出			判斷	機率輸出			判斷
	Target (Label)	男生	女生	其他		男生	女生	其他	
data 1	男生	0.4	0.3	0.3	男生 (正確)	0.7	0.1	0.2	男生 (正確)
data 2	女生	0.3	0.4	0.3	女生 (正確)	0.1	0.8	0.1	女生 (正確)
data 3	男生	0.5	0.2	0.3	男生 (正確)	0.9	0.1	0	男生 (正確)
data 4	其他	0.8	0.1	0.1	男生 (錯誤)	0.4	0.3	0.3	男生 (錯誤)
		模型1錯誤率: 1/4=0.25				模型2錯誤率: 1/4=0.25			

Can we observe any difference between model 1 & 2 from **error rate**?

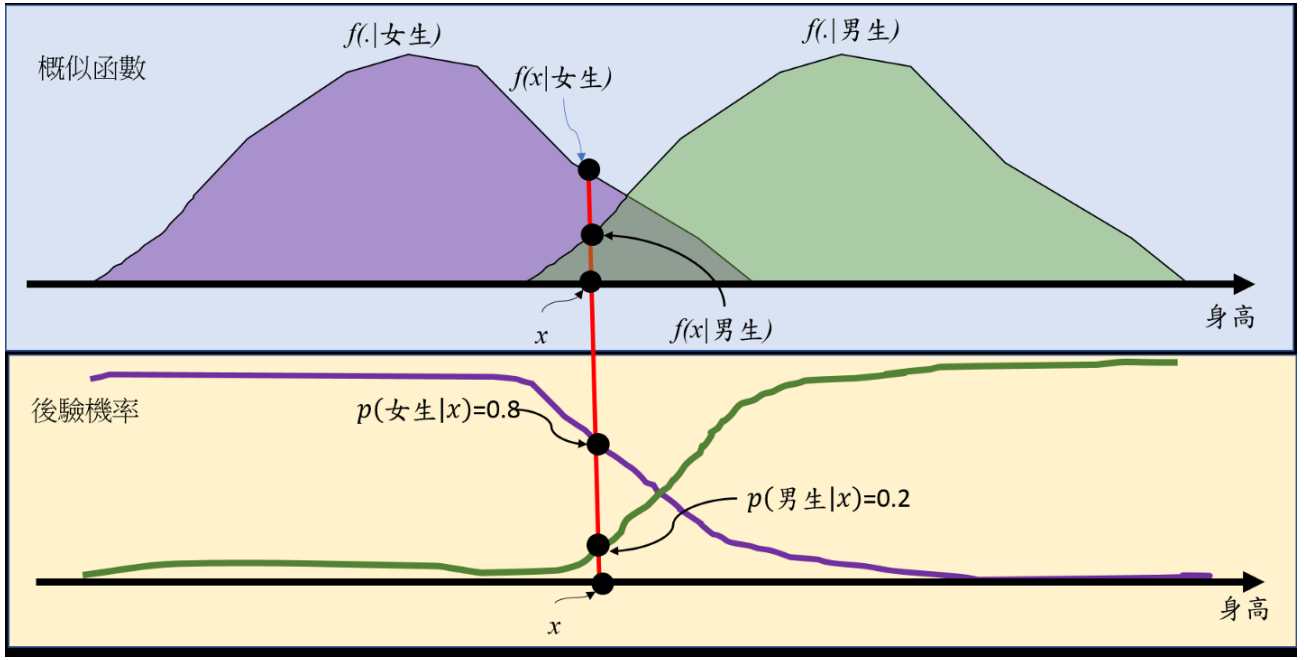
NO…

BUT we can observe that model 2 has better probability outputs than model 1.

Error rate cannot as learning object for learning updating, it's just a metric for evaluating model performance.

Classification

- How do we make decision for a new sample in classification model?
- **ANS: posterior probability.**



Cross-entropy

- Cross-entropy is usually used in classification loss.
- **Entropy**: the average of information which is produced by a stochastic source of data.
- **Information gain**: (suppose X is a random variable)

$$I(x) = -log_2(p(x))$$

Information gain

A is stupid, and his grades usually are around 50 marks.

B is smart, and his grades usually are almost 100 marks.

Probability to pass the exam for A: $p(x_A) = 0.4$

$$I(x_A) = -log_2\big(p(x_A)\big) = 1.322$$

Probability to pass the exam for B: $p(x_B) = 0.99$

$$I(x_B) = -log_2\big(p(x_B)\big) = 0.014$$

Entropy

Entropy: the average of information which is produced by a stochastic source of data.

In information theory,

Entropy = Shannon entropy

$$H(X) = \sum_i -p_i log_2(p_i)$$

Generally, entropy refers to uncertainty for the random variable *X*.

Entropy

$$p(x_A = pass) = 0.4, \qquad p(x_A = fail) = 0.6$$

$$p(x_B = pass) = 0.99, \qquad p(x_B = fail) = 0.01$$

$$H(X) = \sum_i -p_i log_2(p_i)$$

$$H(X_A) = -0.4 \log(0.4) - 0.6 \log(0.6) = 0.971$$

$$H(X_B) = -0.99 \log(0.99) - 0.01 \log(0.01) = 0.081$$

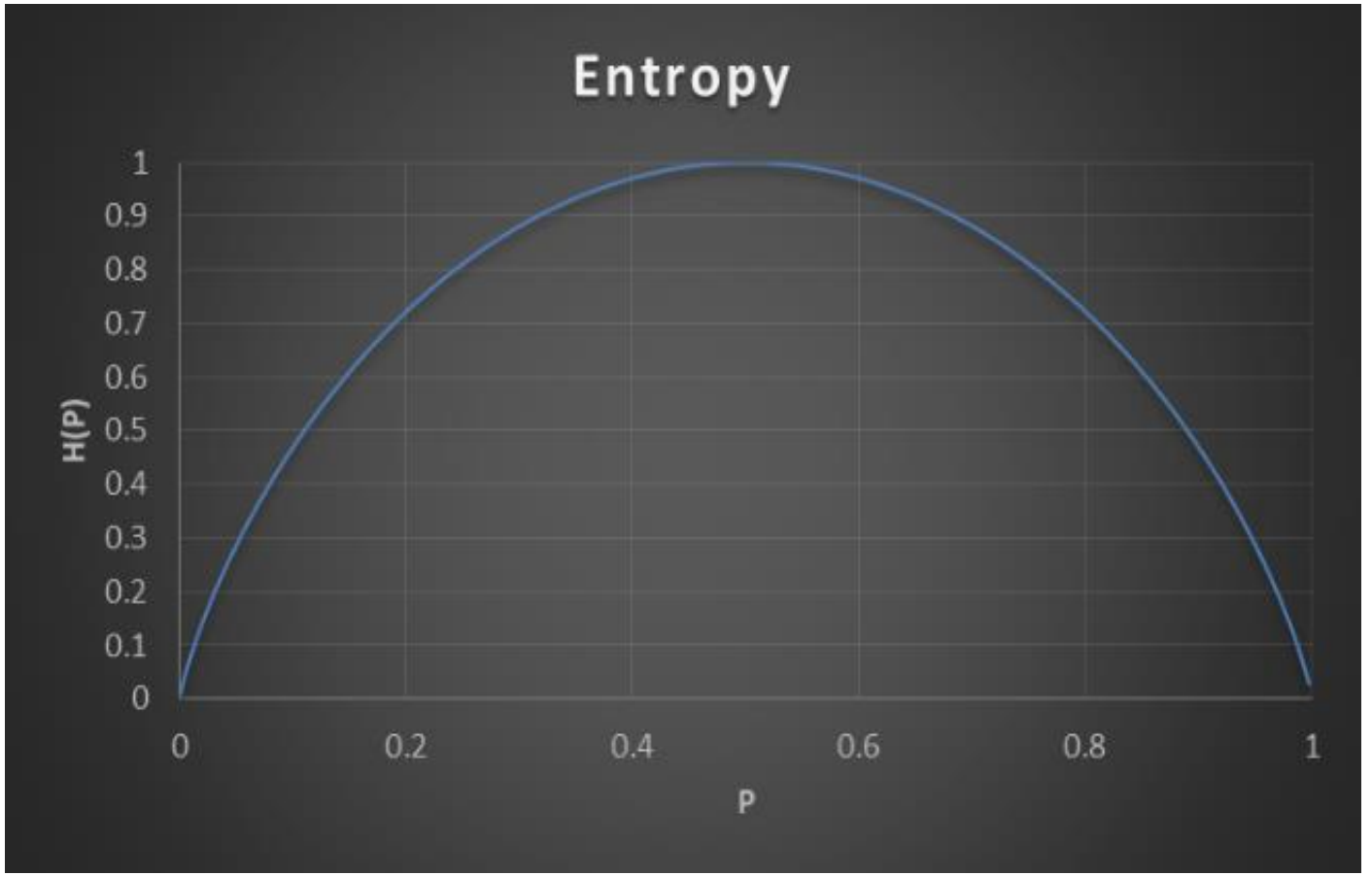
Same conclusion for information gain.

$$I(x_A) = -log_2\big(p(x_A)\big) = 1.322$$

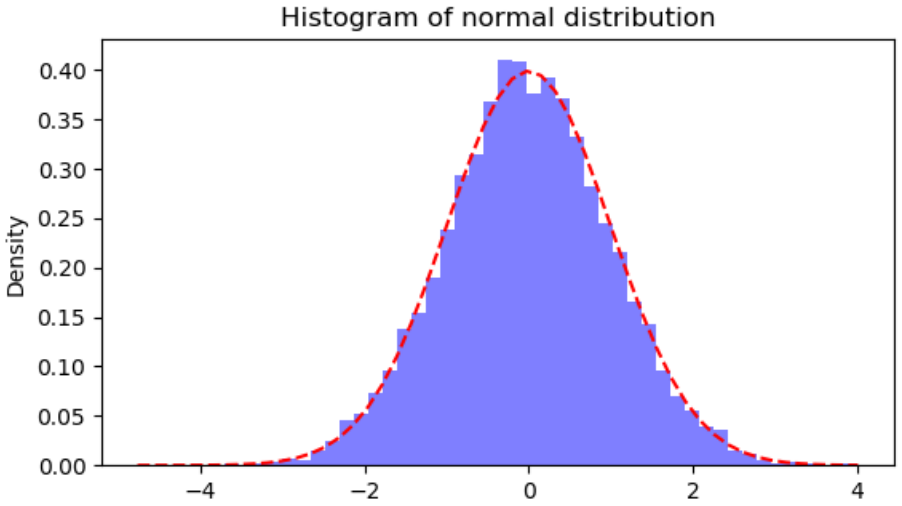
$$I(x_B) = -log_2\big(p(x_B)\big) = 0.014$$

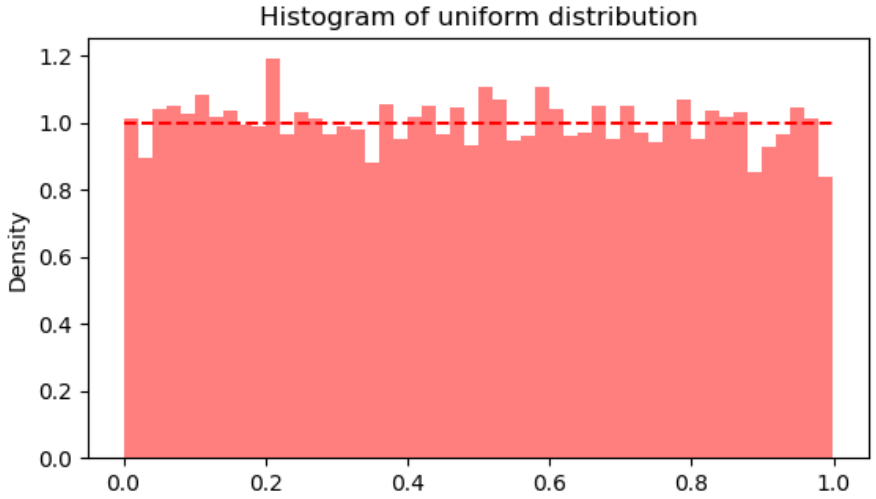
Entropy

When p=0.5 has the largest entropy.



Entropy





$p(X_A) = \begin{cases} 0.1 & x = 1 \\ 0.15 & x = 2 \\ 0.5 & x = 3 \\ 0.15 & x = 4 \\ 0.1 & x = 5 \end{cases}$	$p(X_A) = \begin{cases} 0.01 & x = 1 \\ 0.09 & x = 2 \\ 0.8 & x = 3 \\ 0.09 & x = 4 \\ 0.01 & x = 5 \end{cases}$	$p(X_B) = \begin{cases} 0.2 & x = 1 \\ 0.2 & x = 2 \\ 0.2 & x = 3 \\ 0.2 & x = 4 \\ 0.2 & x = 5 \end{cases}$
$H(X_A) = 1.985$	$H(X_A) = 1.016$	$H(X_B) = 2.322$

Cross-entropy

Formula of cross- entropy:

$$CE = \sum_{i=1}^{n} \sum_{c=1}^{C} -y_{c,i} log_2(p_{c,i})$$

C: number of class (male, female, other)

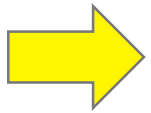
n: number of data

$y_{c,i}$: binary indicator (0 or 1) from one hot encode (*i*-th data assigns to *c*-class)

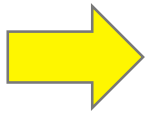
$p_{c,i}$: probability of *i*-th data assigns to *c*-class

One hot encode (Dummy variable)

Data 1	Male
Data 2	Female
Data 3	Male
Data 4	Other



Male
Female
Other



	Male	Female	Other
Data 1	1	0	0
Data 2	0	1	0
Data 3	1	0	0
Data 4	0	0	1

Cross-entropy

$$CE = \sum_{i=1}^{n} \sum_{c=1}^{C} -y_{c,i} log_2(p_{c,i})$$

		Model 1 (輸出)					
		機率輸出			實際One-hot encode		
	Target (Label)	男生	女生	其他	男生	女生	其他
data 1	男生	0.4	0.3	0.3	1	0	0
data 2	女生	0.3	0.4	0.3	0	1	0
data 3	男生	0.5	0.2	0.3	1	0	0
data 4	其他	0.8	0.1	0.1	0	0	1
		模型1錯誤率: 1/4=0.25 Cross-entropy=6.966					

Data 1:

$$\sum_{c=1}^{C} -y_{c,1} log_2(p_{c,1})$$

$$= -1 * \log(0.4) - 0 * \log(0.3) - 0 * \log(0.3) = 1.322$$

Data 4:

$$\sum_{c=1}^{C} -y_{c,4} log_2(p_{c,4})$$

$$= -0 * \log(0.8) - 0 * \log(0.1) - 1 * \log(0.1) = 3.3219$$

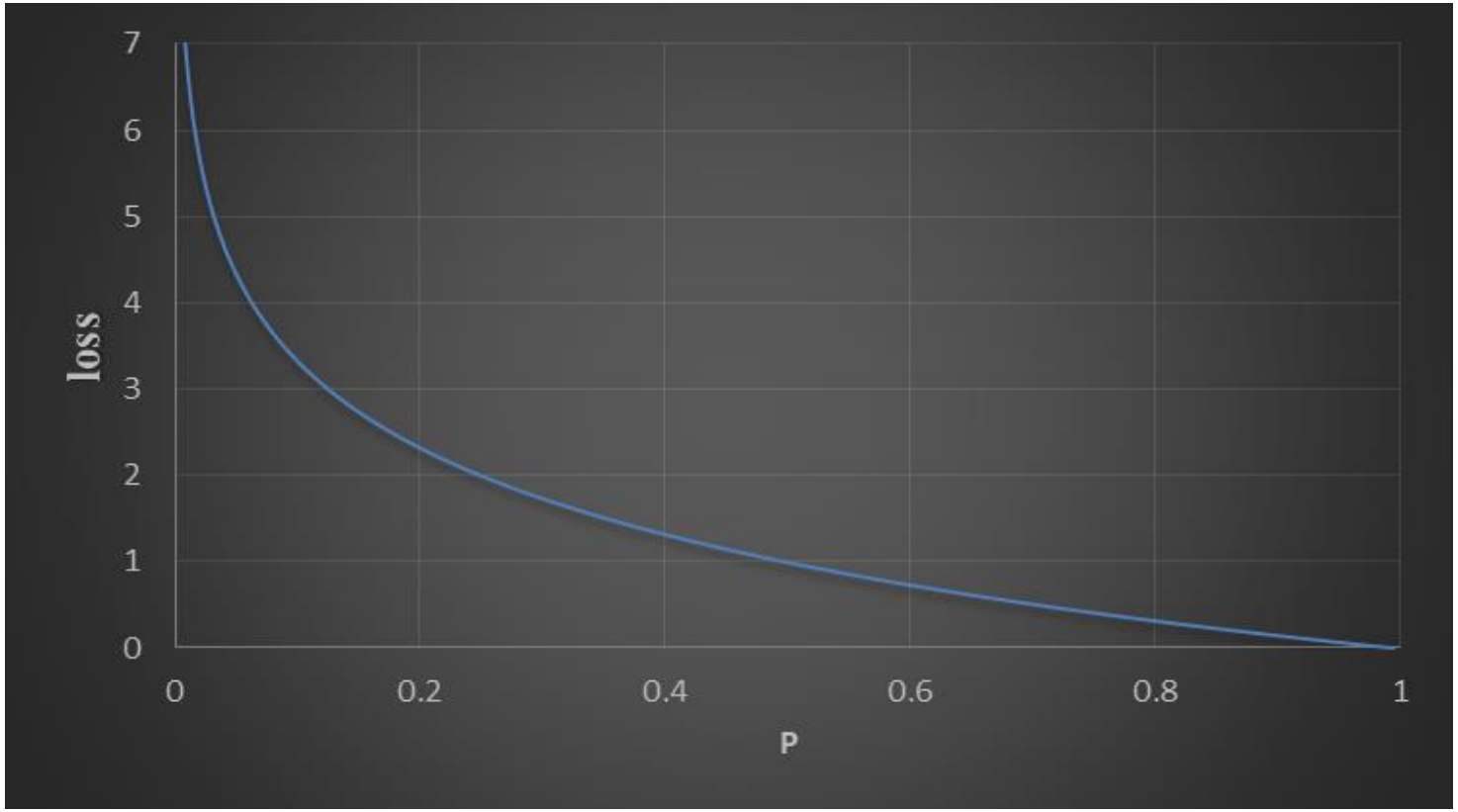
SO less probability data has larger loss function (entropy value)→learning target.

Cross-entropy
for evaluating the model performance

		Model 1 (輸出)					
		機率輸出			實際One-hot encode		
	Target (Label)	男生	女生	其他	男生	女生	其他
data 1	男生	0.4	0.3	0.3	1	0	0
data 2	女生	0.3	0.4	0.3	0	1	0
data 3	男生	0.5	0.2	0.3	1	0	0
data 4	其他	0.8	0.1	0.1	0	0	1
		模型1錯誤率: 1/4=0.25 **Cross-entropy=6.966**					

		Model 2 (輸出)					
		機率輸出			實際One-hot encode		
	Target (Label)	男生	女生	其他	男生	女生	其他
data 1	男生	0.7	0.1	0.2	1	0	0
data 2	女生	0.1	0.8	0.1	0	1	0
data 3	男生	0.9	0.1	0	1	0	0
data 4	其他	0.4	0.3	0.3	0	0	1
		模型1錯誤率: 1/4=0.25 **Cross-entropy= 2.310**					

Cross-entropy for loss function



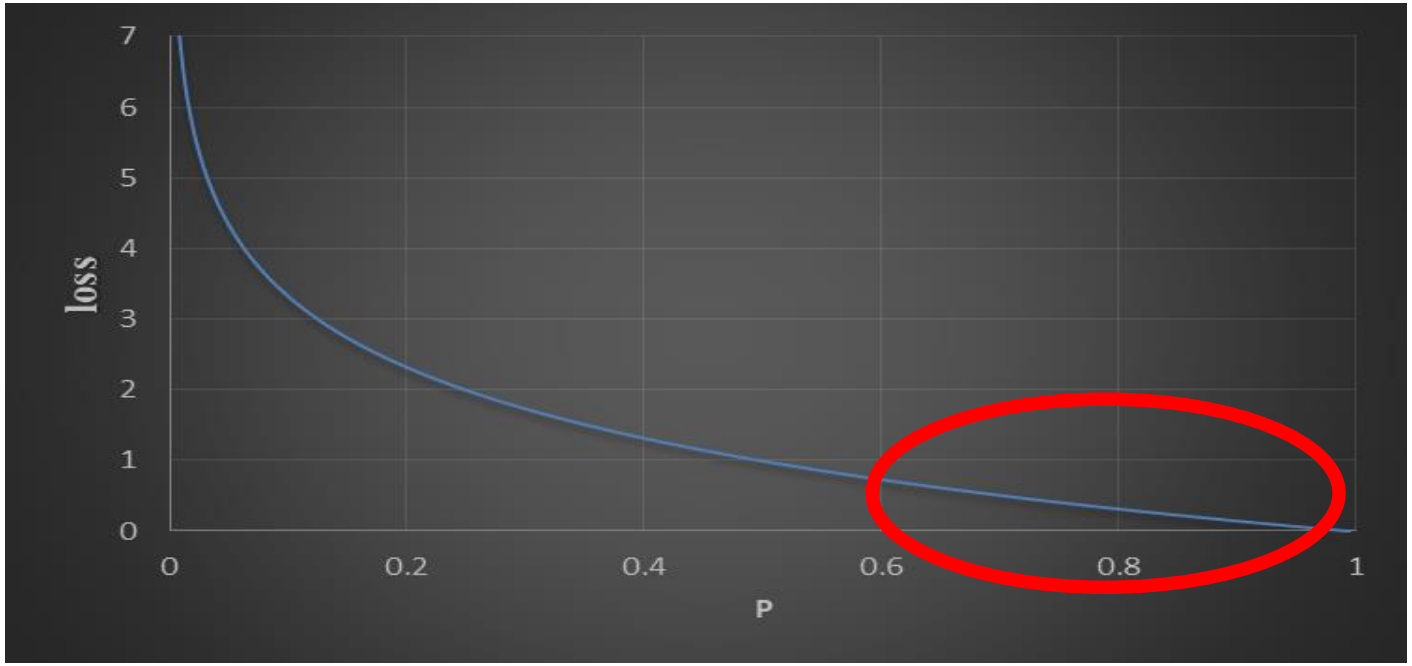
$$CE\ loss = \sum_{i=1}^{n} \sum_{c=1}^{C} -y_{c,i} log_2(p_{c,i})$$

Focal loss (1/3)

Cross-entropy (CE) for $y \in \{\pm 1\}$

$$CE(p,y) = \begin{cases} -\log(p), & if\ y = 1 \\ -\log(1-p), & if\ y = -1 \end{cases}$$

$$CE(p,y) = CE(p_t) = -\log(p_t), \qquad p_t = \begin{cases} p, & if\ y = 1 \\ 1-p, & if\ y = -1 \end{cases}$$



Focal loss (2/3)

α-balanced cross-entropy:

$$CE(p_t) = -\alpha \log(p_t)$$

BUT it's not effect for larger class unbalance problem.

Modulating factor:

$$(1 - p_t)^r$$

r: focusing parameter, $r \geqq 0$.

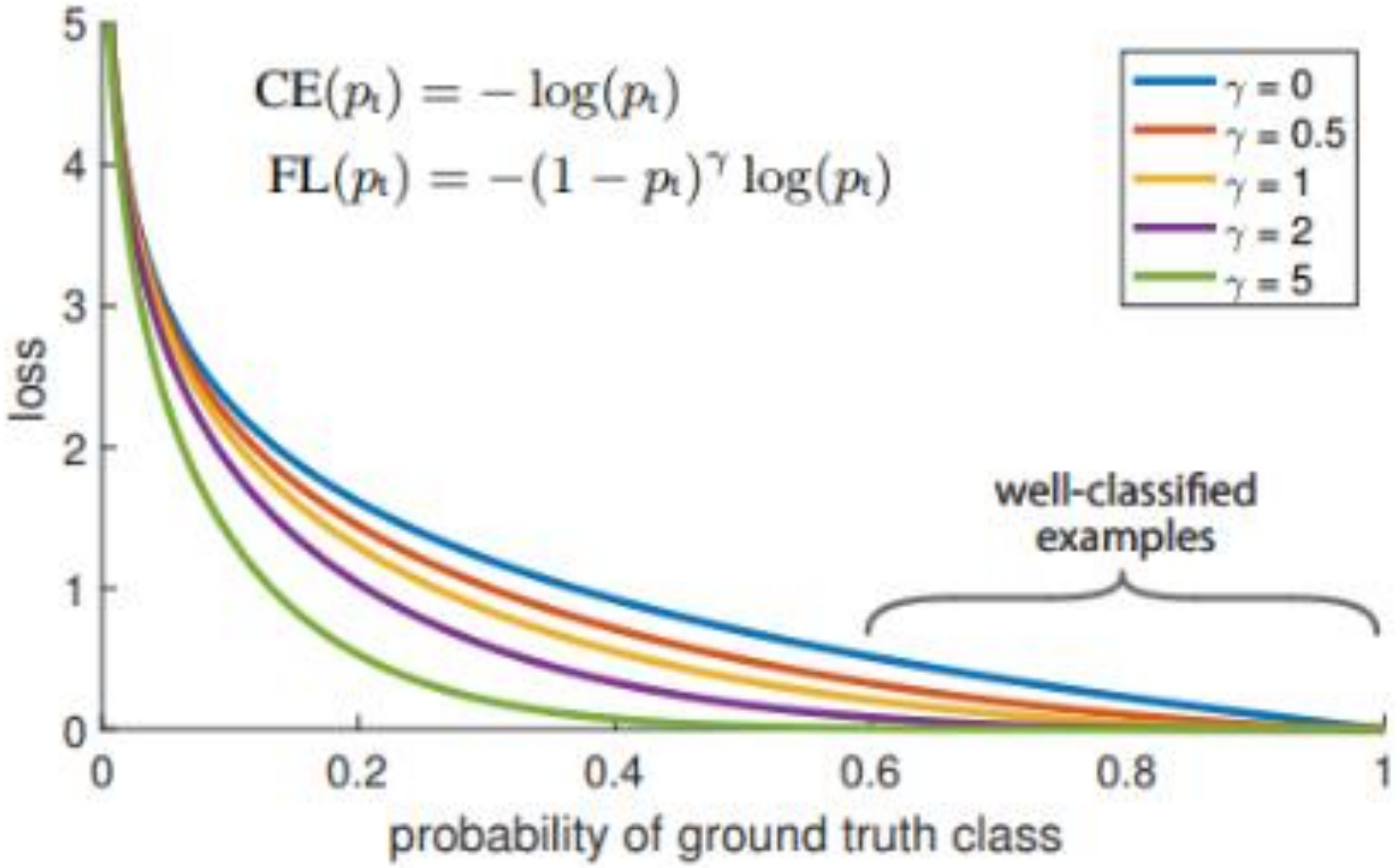
Focal loss:

$$FL(p_t) = -(1 - p_t)^r \log(p_t)$$

α-balanced focal loss:

$$FL(p_t) = -\alpha(1 - p_t)^r \log(p_t)$$

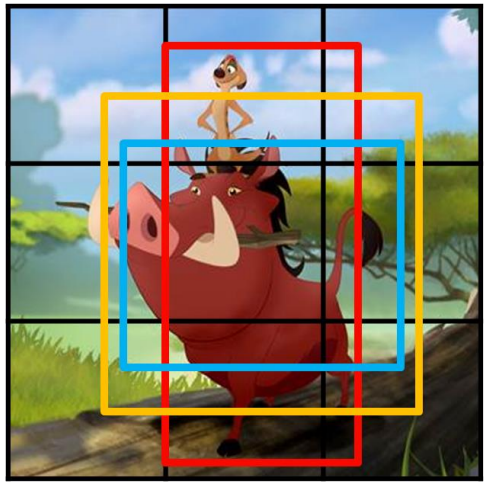
Focal loss (3/3)



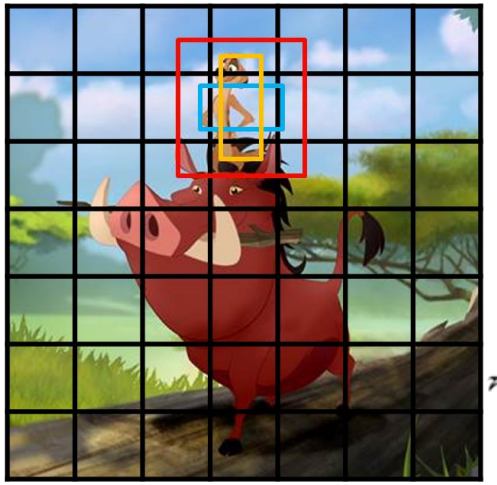
Why Focal loss? (Proposed in RetainNet)

在訓練階段的特徵圖對應的Anchor
有90%都是沒用的背景anchor ，所以要怎麼抑制背景的anchor。
怎麼強化判錯的Anchor的loss?
Focal loss。

3*3的特徵圖



7*7的特徵圖



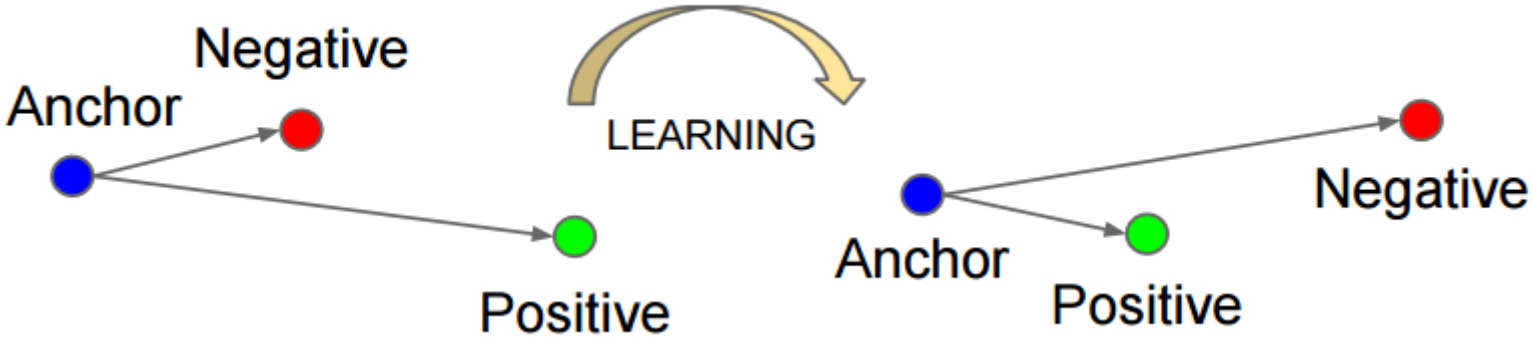
Triple Loss

Triple

Anchor: a randomly training data with label c

Positive: training data in label c

Negative: training data in other labels



Triple Loss

$Anchor : x_i^a$

$Positive : x_i^p$

$Negative : x_i^n$

Encoder network : $f(x)$

$Anchor : f(x_i^a)$

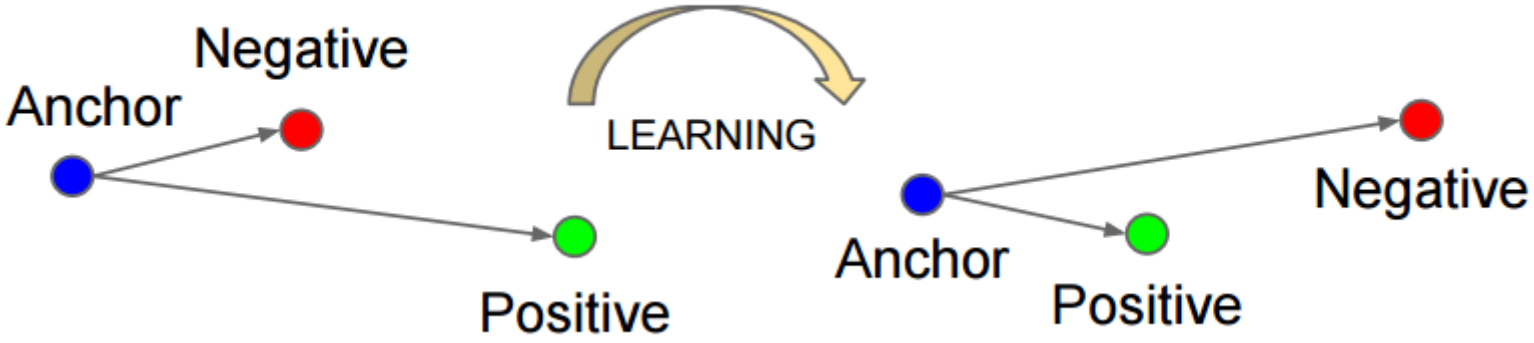
$Positive : f(x_i^p)$

$Negative : f(x_i^n)$

triple loss aims to

$dist(f(x_i^a), (x_i^p)) \downarrow$

$dist(f(x_i^a), (x_i^n)) \uparrow$



Triple Loss

$$dist(f(x_i^a), f(x_i^p)) + \alpha < dist(f(x_i^a), f(x_i^n))$$

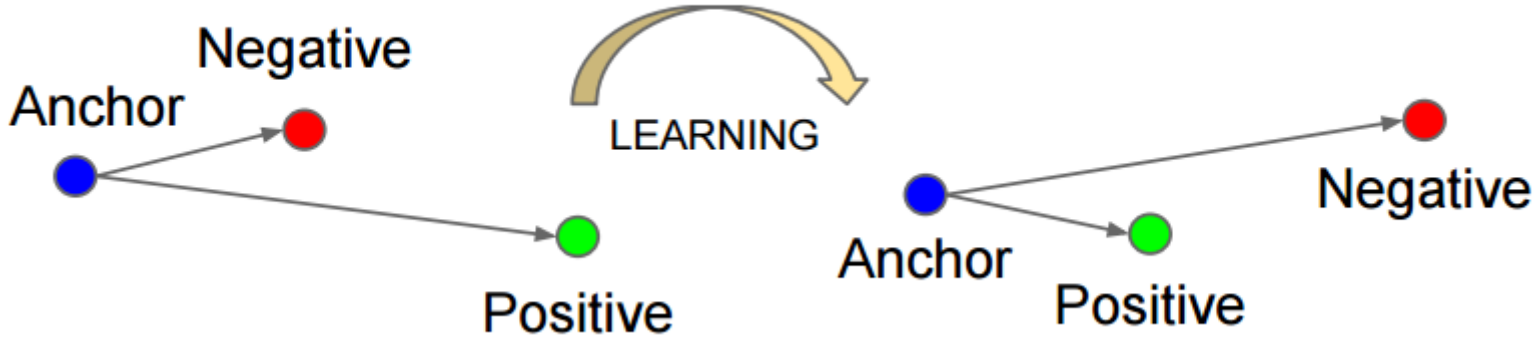
$$\Rightarrow \left\| f(x_i^a) - f(x_i^p) \right\|_2^2 + \alpha < \left\| f(x_i^a) - f(x_i^n) \right\|_2^2$$

$$\arg\min\{\sum_i \left(\left\| f(x_i^a) - f(x_i^p) \right\|_2^2 + \alpha - \left\| f(x_i^a) - f(x_i^n) \right\|_2^2 \right)\}$$

$$\arg\min \sum_i \left[\left\| f(x_i^a) - f(x_i^p) \right\|_2^2 + \alpha - \left\| f(x_i^a) - f(x_i^n) \right\|_2^2 \right]_+$$

Triple Loss

$$\left[d_p - d_n + \alpha\right]_+ = \begin{cases} d_p - d_n + \alpha & d_p - d_n + \alpha > 0 \\ 0 & d_p - d_n + \alpha < 0 \end{cases}$$

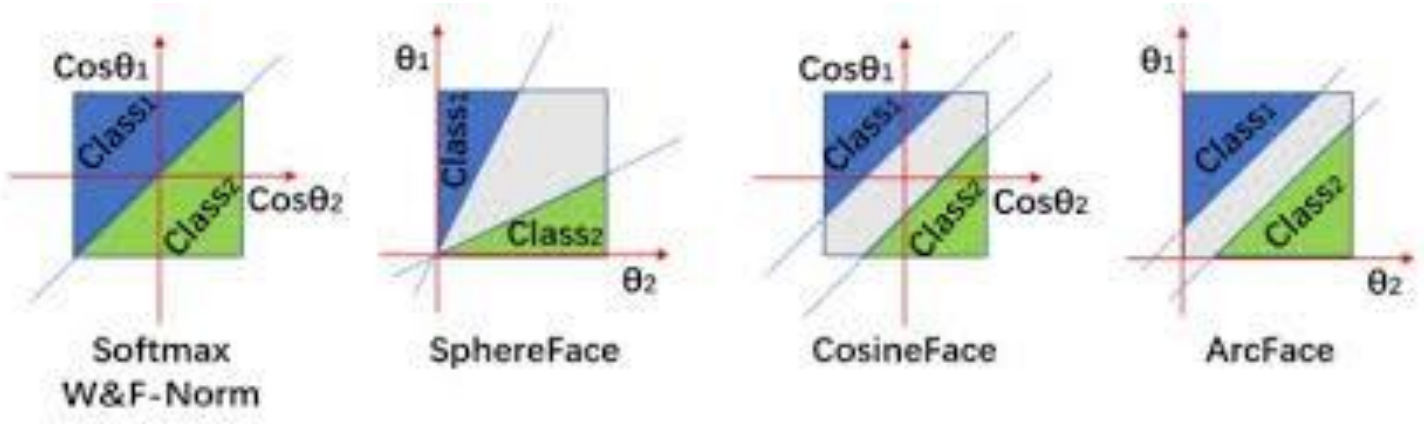


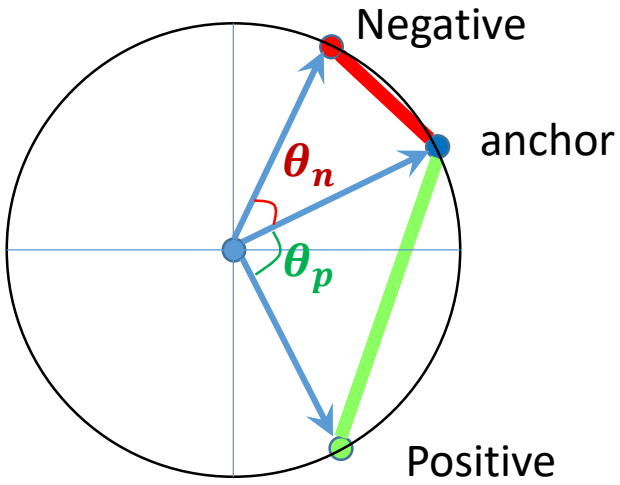
Conclusion

MSE, MAE, Huber loss, triple loss do the same thing.

Similarity measurement.

Cosine loss





Can MSE be a loss for classification?