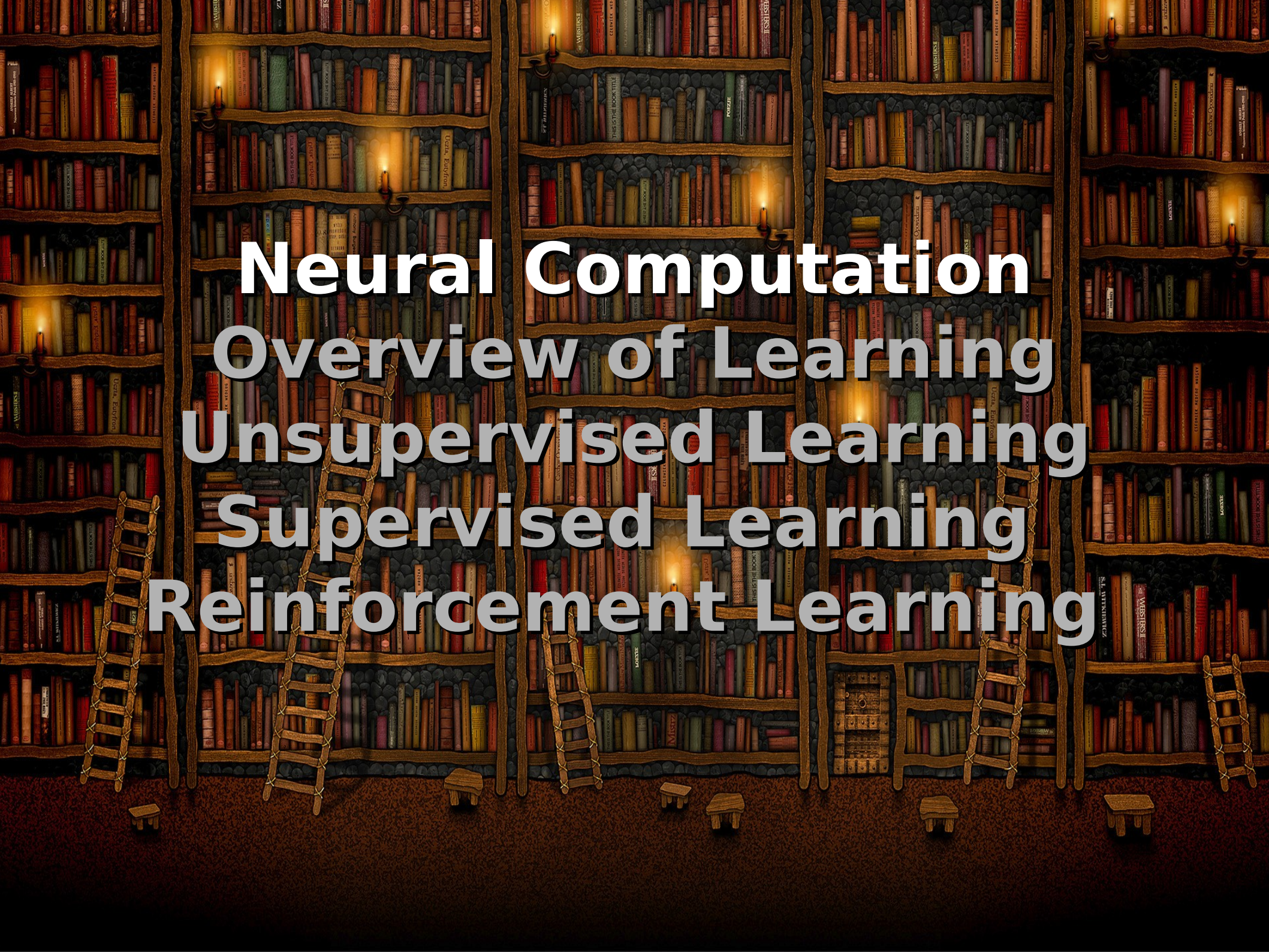
神經與行為模型建構
(Neural & Behavioral Modeling)

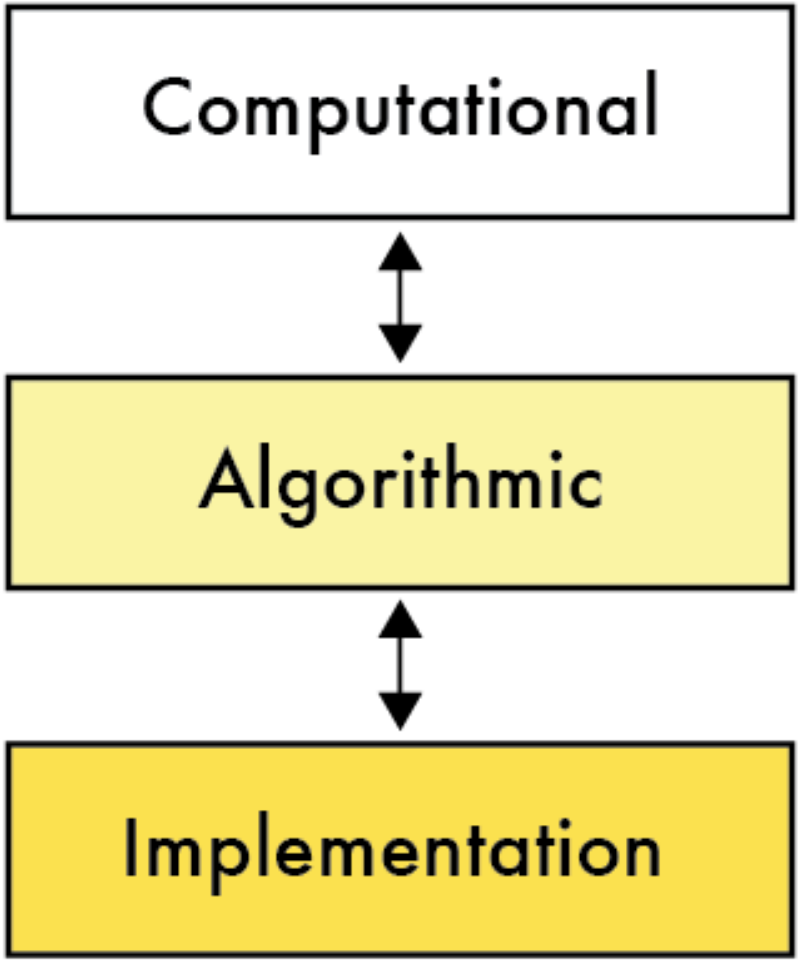
課號：Psy7277　　教室： 北 206
識別碼：227M9280　　時間： 五 234


學習的實驗 / 模型較複雜
!


Neural Computation
Overview of Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning

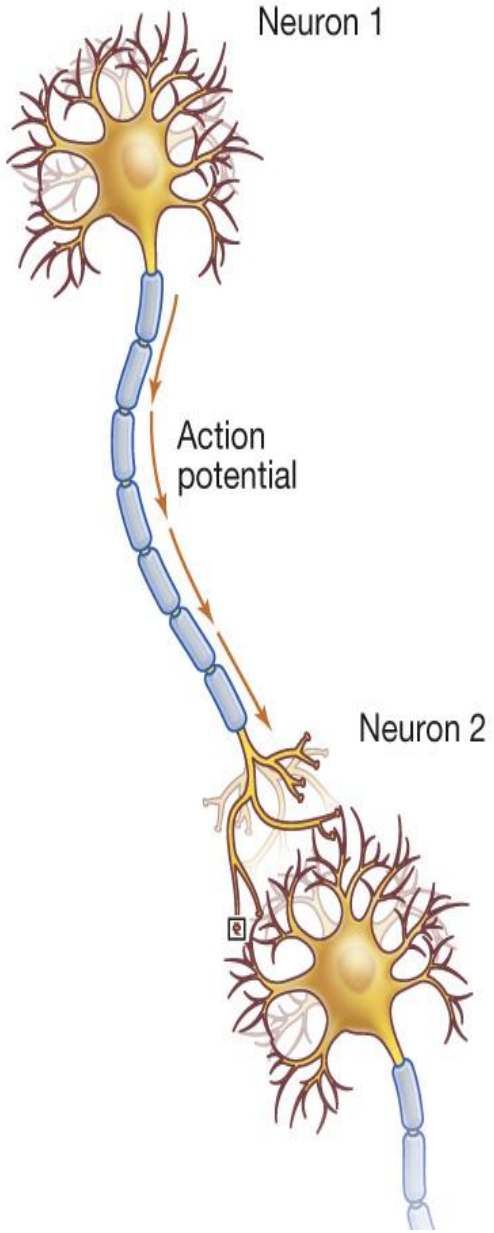
了解一個認知系統的特性與極限

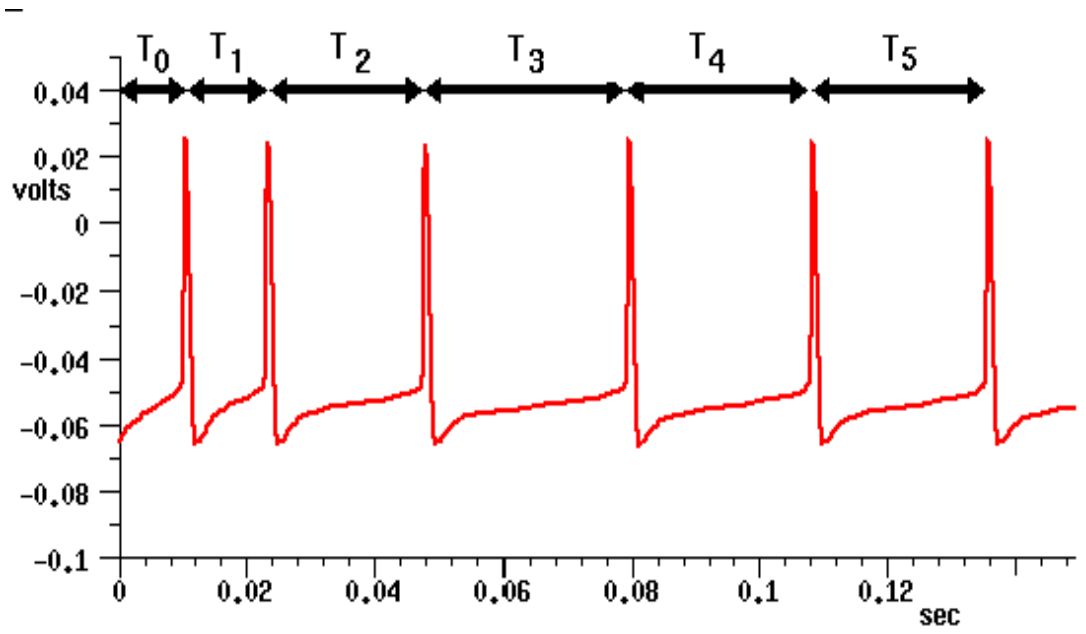
David Marr's 3 levels of analysis

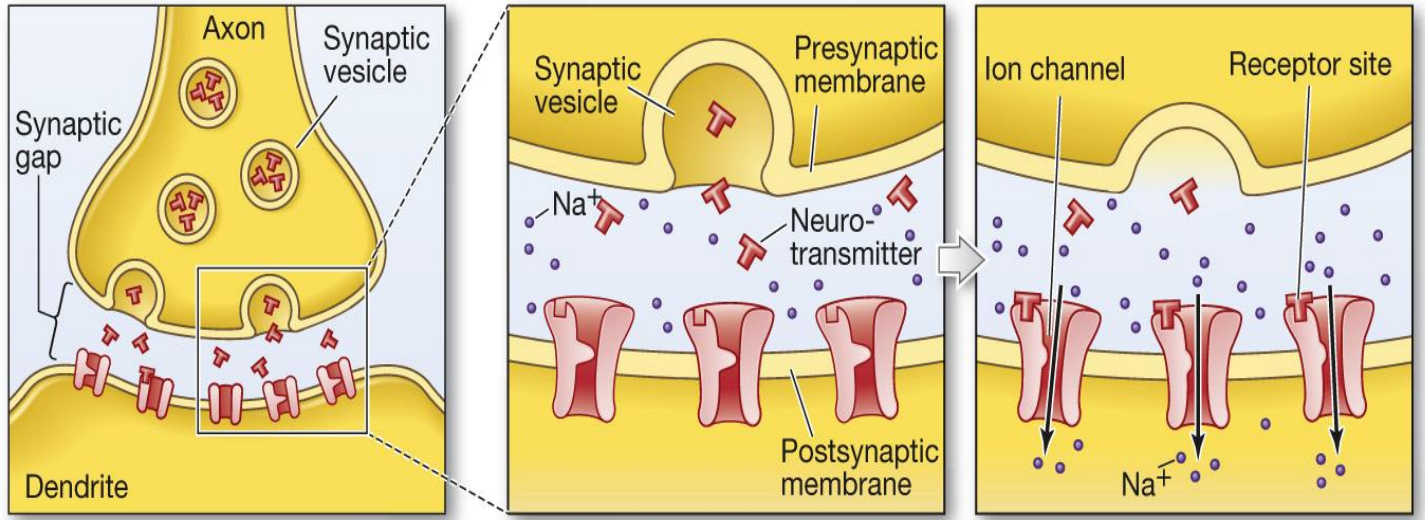


- 計算上的問題是什麼？
- 能夠用什麼演算法來解決？
- 要透過什麼硬體來實現算法？

Implementation: 有 0 與 1 的神經元

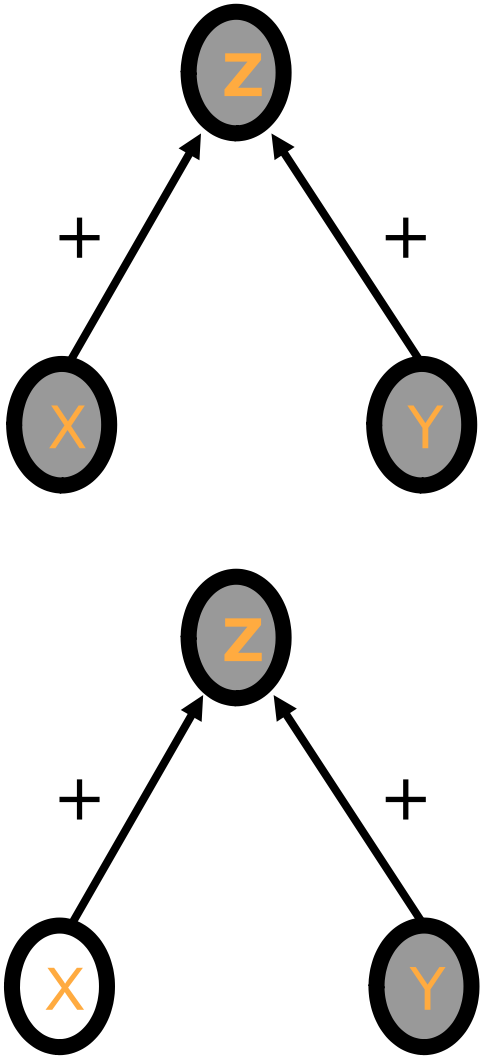






Implementation: 神經元做加法

如果接受刺激的神經元有個低閾值

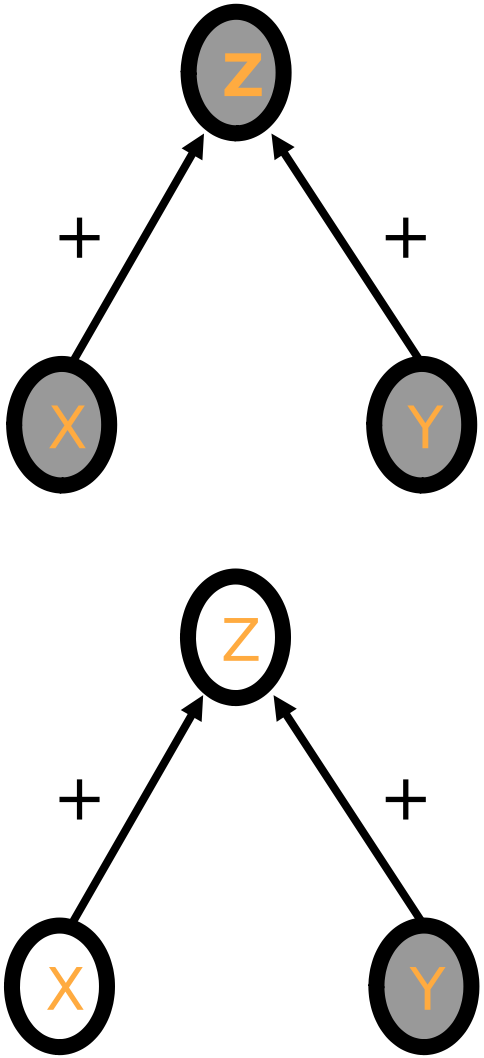


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Z=X+Y (OR)

Implementation: 神經元做乘法

如果接受刺激的神經元有個高閾值

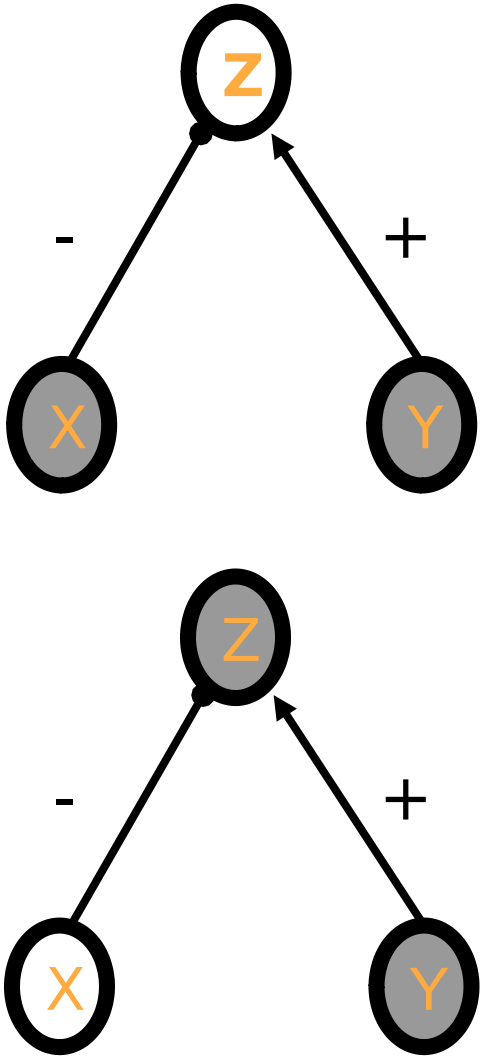


X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Z=X*Y (AND)

Implementation: 神經元做減法

如果開始考慮神經元彼此的抑制關係

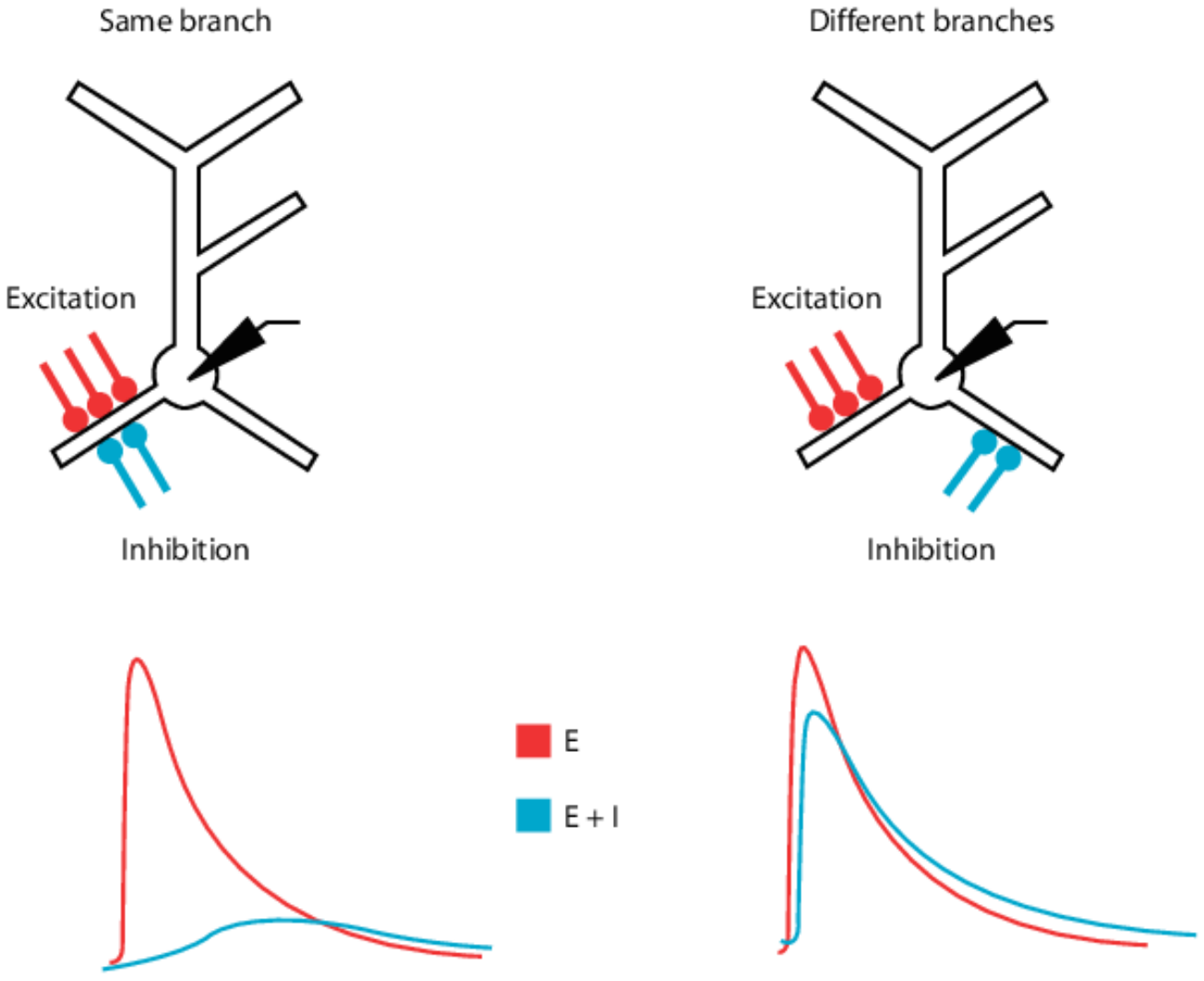


X	Z
0	1
1	0

Z=1-X (NOT)

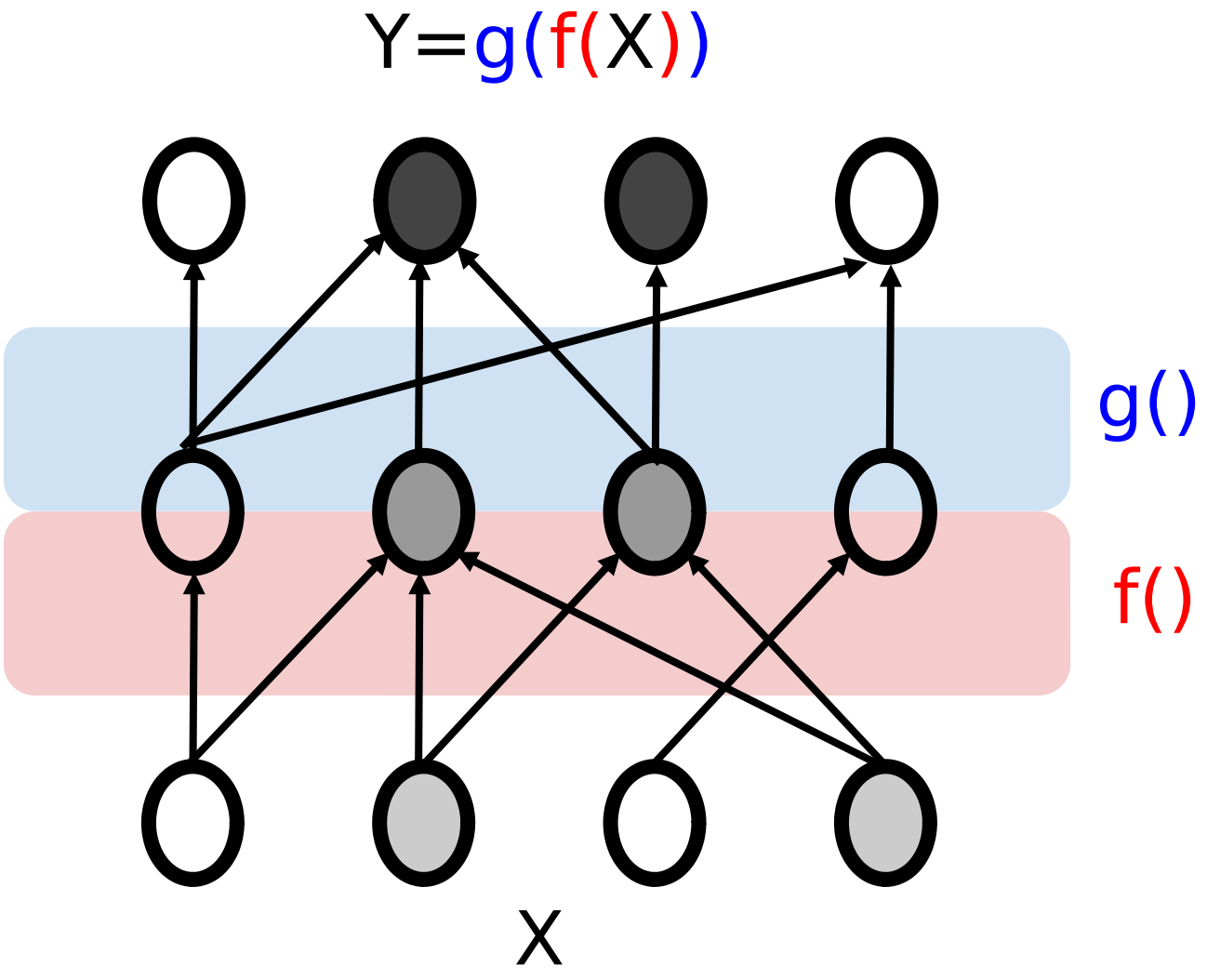
Implementation: 神經元做除法

神經元間的抑制可以導致減法或除法



Algorithm: 基本計算的組合

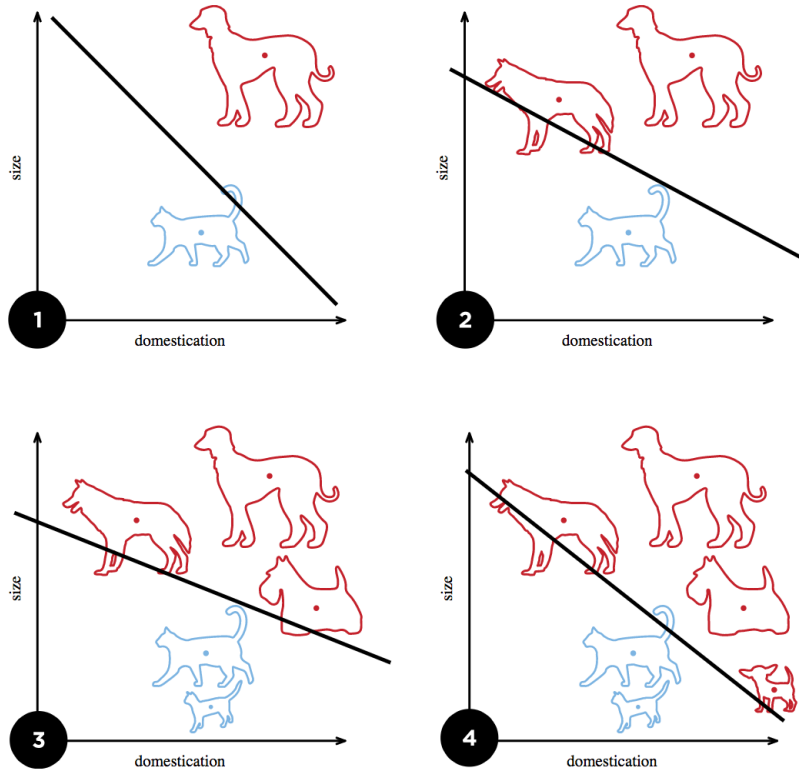
一個神經網路透過組合基本的計算來實現演算法

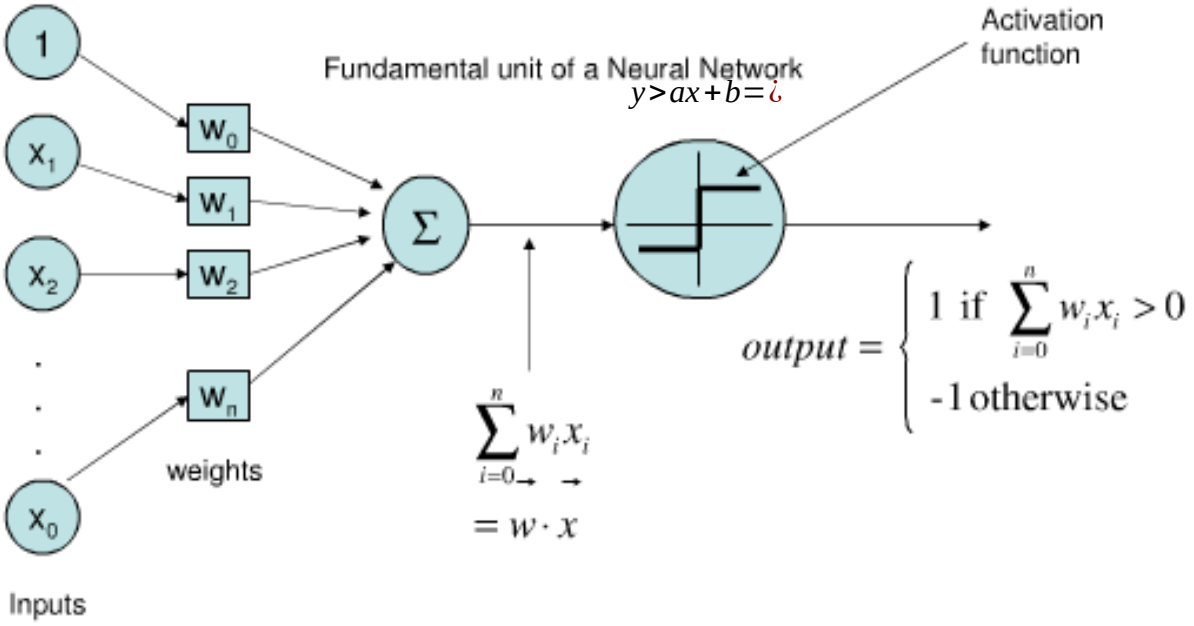


Computational Problem: 區分貓狗

Perceptron (Rosenblatt, 1958)
Cognitron (Fukushima, 1975)

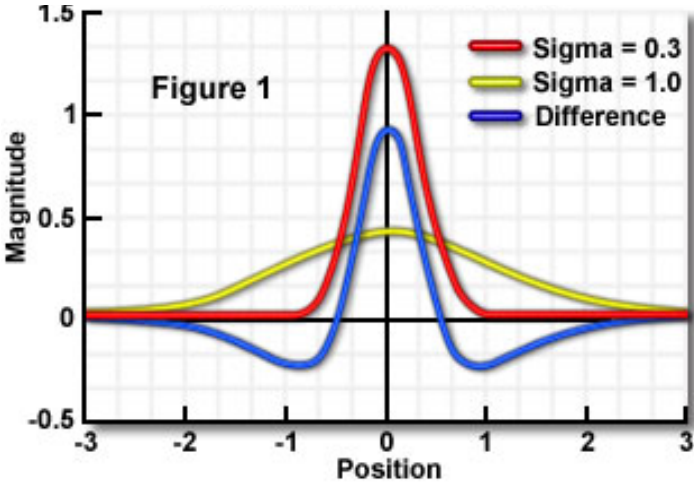
$$y > ax + b \Rightarrow y - ax - b > 0 \Rightarrow w_2 x_2 + w_1 x_1 + w_0 1 > 0$$

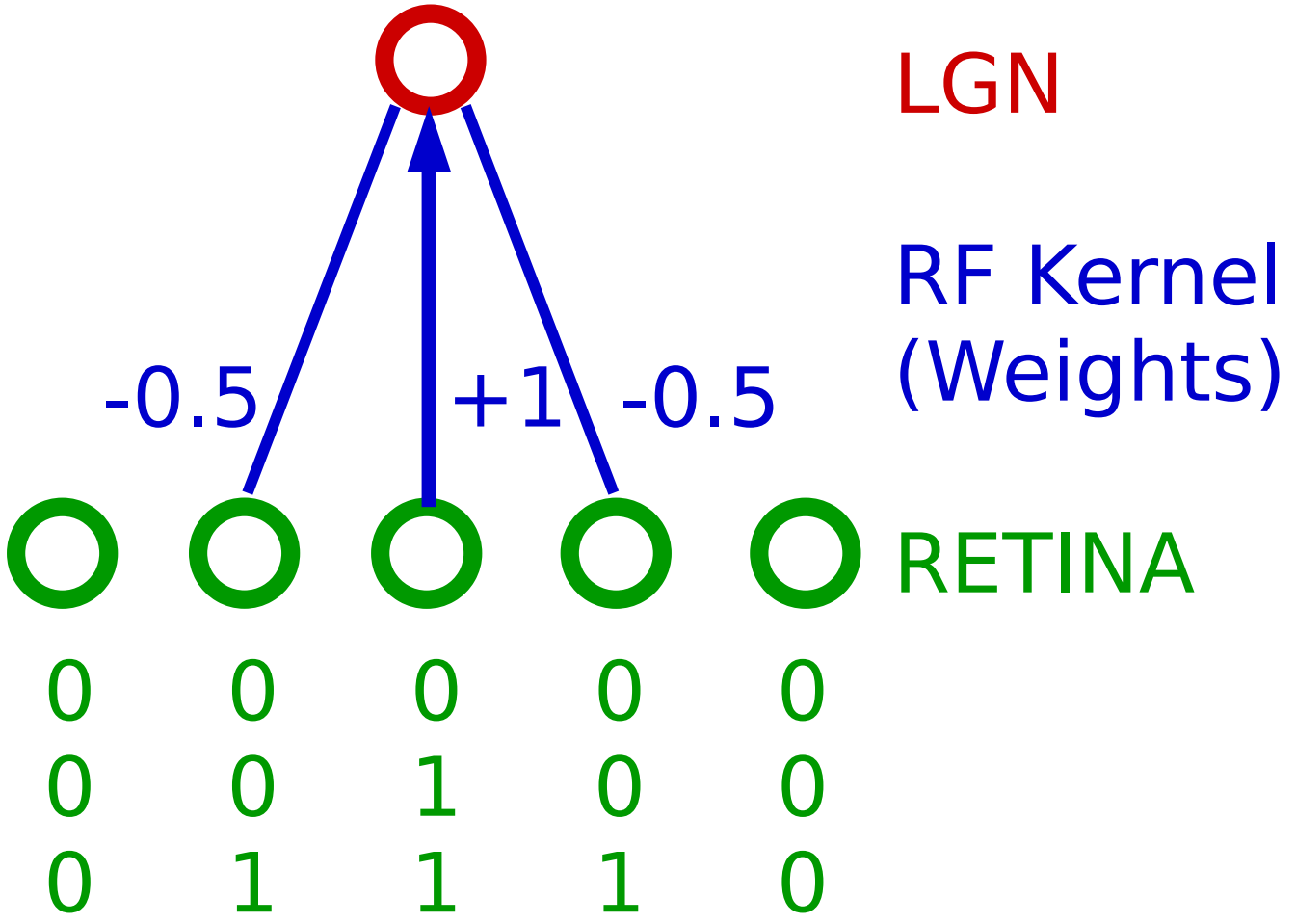




$$y < ax + b \Rightarrow y - a - b < 0 \Rightarrow w_2 x_2 + w_1 x_1 + w_0 1 < 0$$

Neuron as a Pattern Recognizer



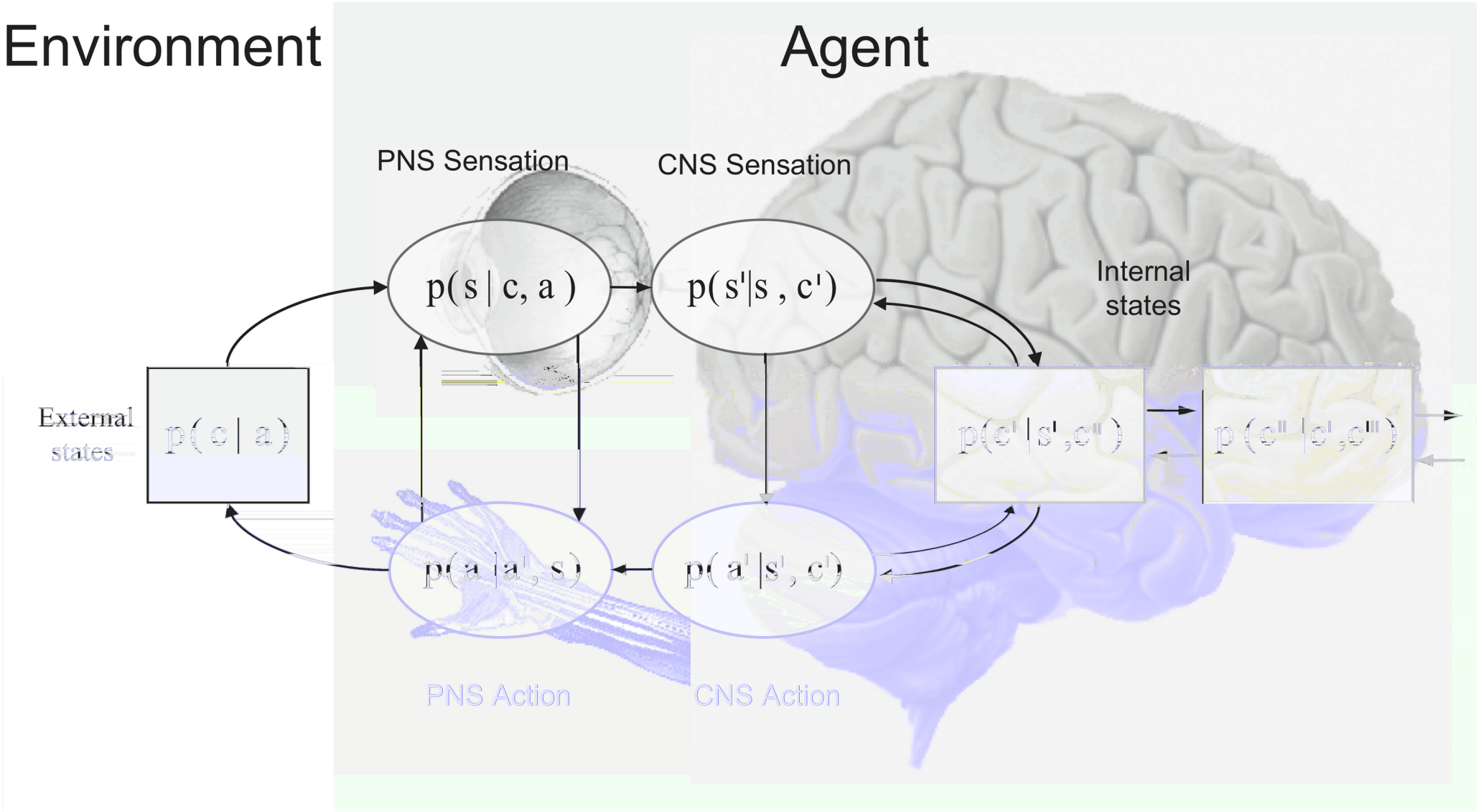



```
RETINA=matrix('0 0 0 0 0;0 0 1 0 0;0 1 1 1 0')
LGN_RF=matrix('0;-0.5;1;-0.5;0')
LGN_RESPONSE=RETINA*LGN_RF
```


Neural Computation
Overview of Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning

學習 / 記憶的目的：預測與控制

學習和演化的目的都是更適應環境

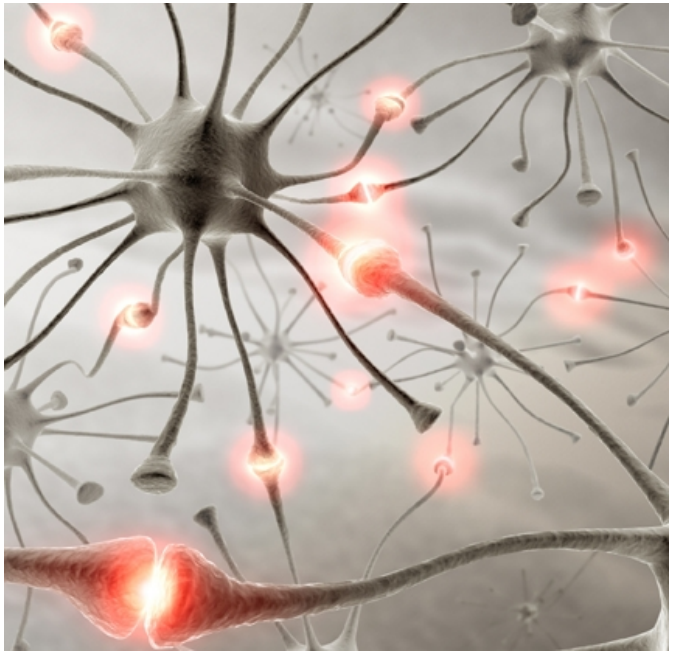


學習中的兩難 (Dilemma)

Learning vs. Performance

行為上，正在學習一件事時表現通常不好，表現好時表示其實已相當熟練沒學到什麼。

大腦學習神經路徑是平行的
有些迴路很 plasticity；有些則是保持 stability

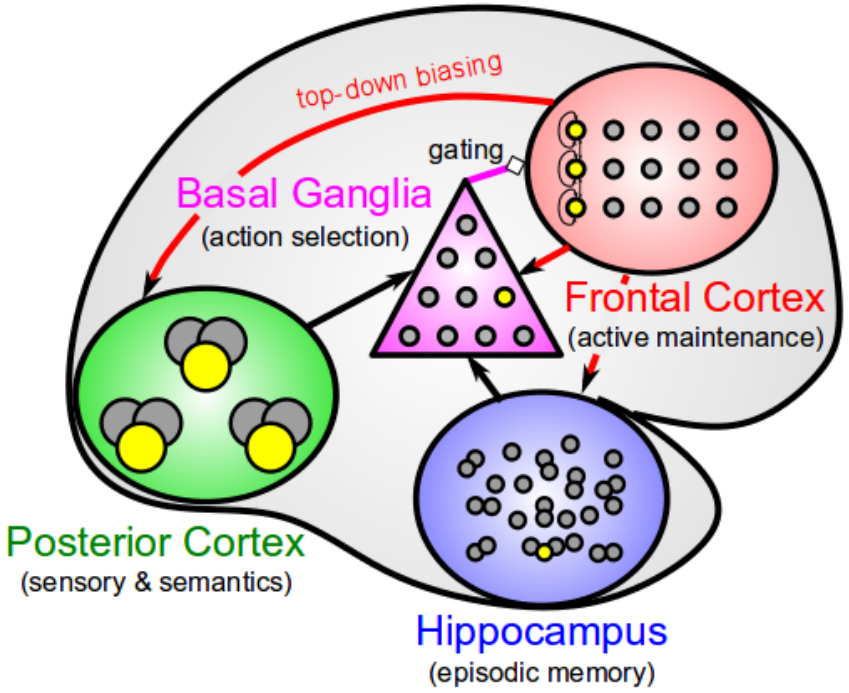


Stability vs. Plasticity

大腦中，神經結構的變化雖可幫助學習新事物，但這些結構上的改變也意味著對舊事物的遺忘。

stability 不一定不好。例子：紅燈行綠燈停遊戲的例子

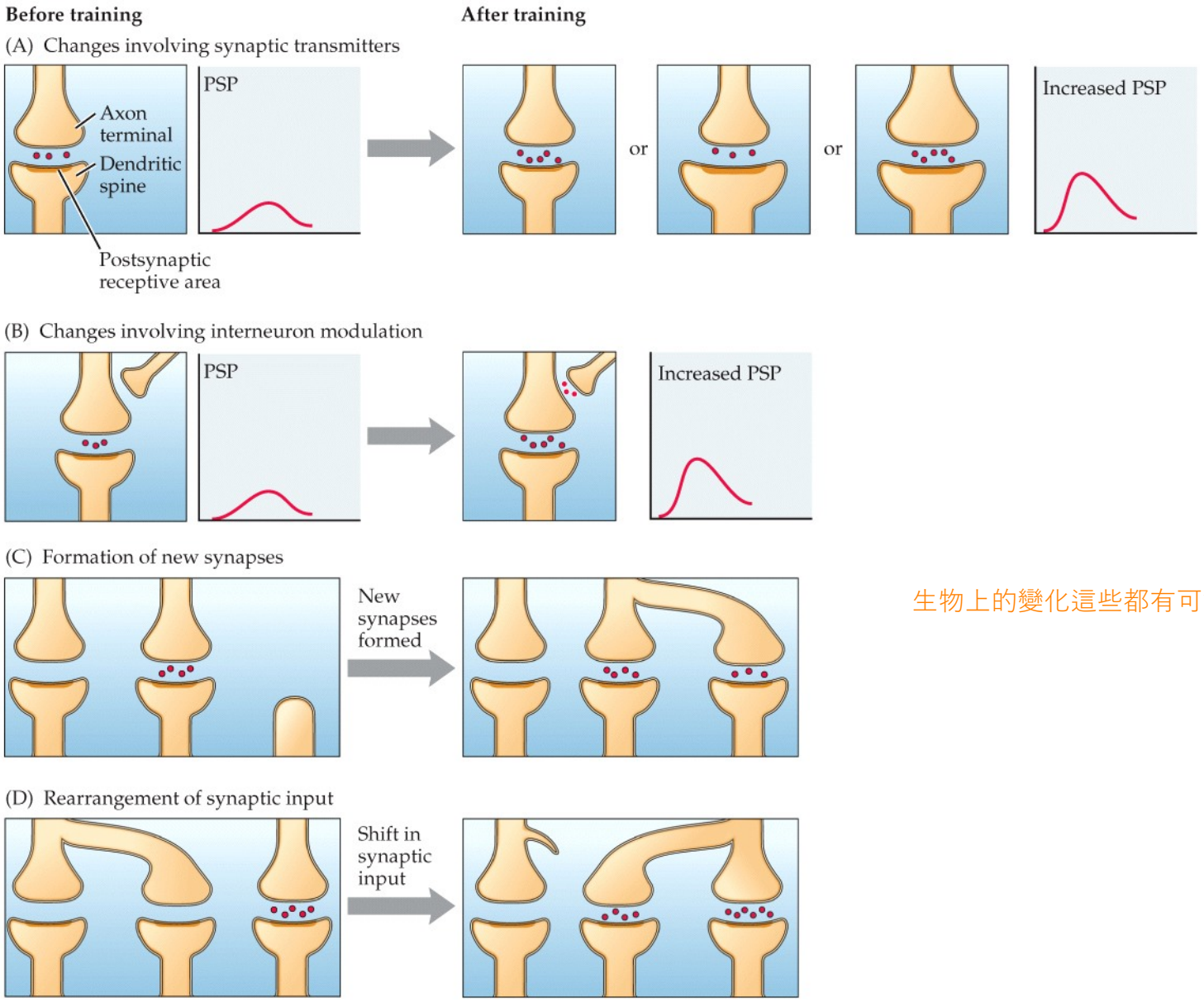
不同腦區的計算 / 學習性質



	Learning Signal			*Dynamics*		
Area	**Reward**	**Error**	**Self Org**	**Separator**	**Integrator** integrate 新舊資訊	**Attractor** 異中求同
Basal Ganglia	+++	---	---	++	-	---
Cerebellum	---	+++	---	+++	---	---
Hippocampus	+	+	+++	+++	---	+++
Neocortex	++	+++	++	---	+++	+++

突觸連結變強的生物機制

雖有不同機制但數學上都是 ΔW_{ij} 變大



生物上的變化這些都有可能

人與機器的歸納學習

induction learning

皆有以下三種情境

非監督式學習：事物沒有標籤（對錯或名字）

小孩發現有些生物會動，有些不會動

監督式學習：事物有明確標籤或該怎麼做

爸媽教小孩會動的叫動物，不會動的叫植物

強化式學習：只知道特定事件後會有賞罰

小孩發現打動物會被咬，打植物卻沒事

有事後的回饋(獎賞或逞罰)(知對錯但不確定哪邊要改),
但不向監督式的回饋(除了知道對錯,
妳還知道是那些x變項的weight要怎麼改進)那麼強


Neural Computation
Overview of Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning

Hebbian vs. Anti-Hebbian

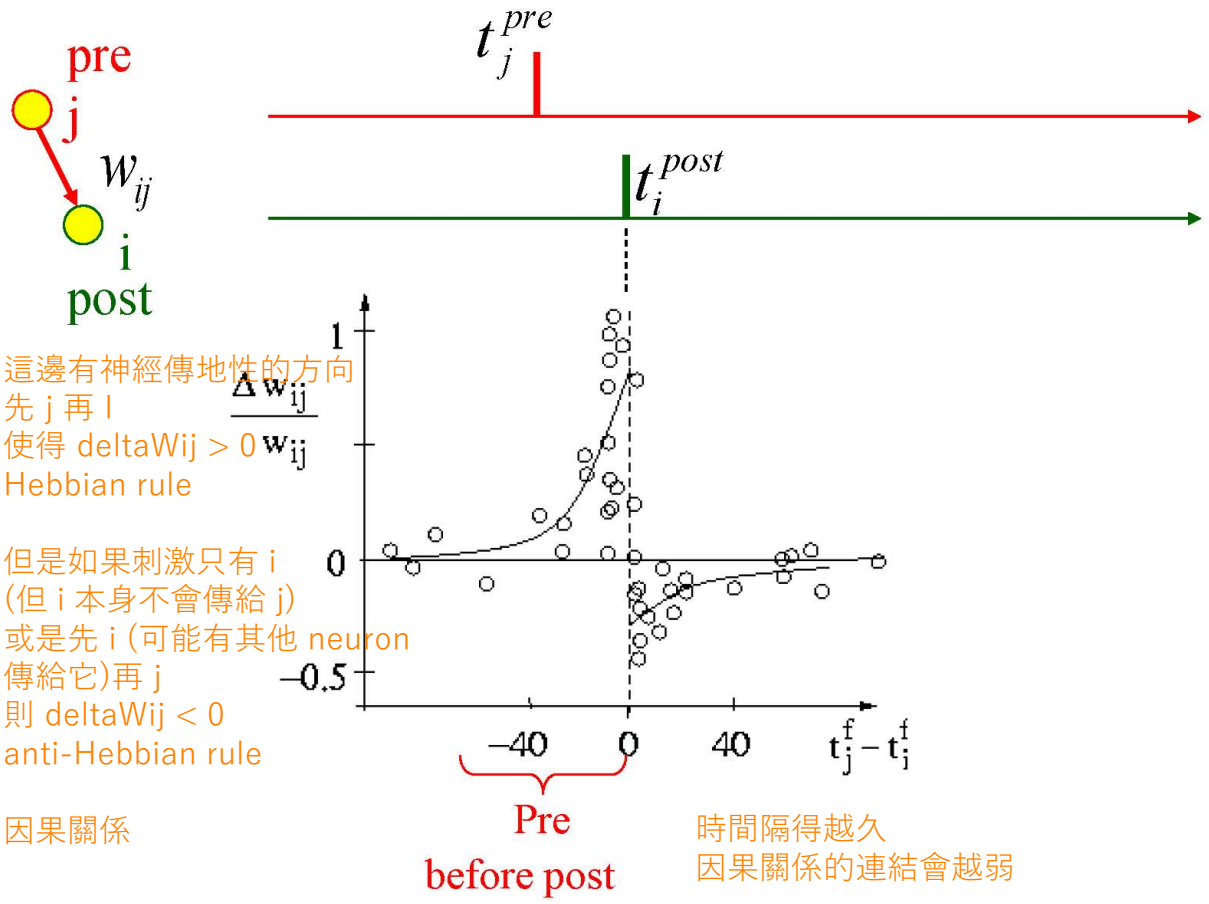
Fire together, weird together

e.g. x = (0, 1, 0)'
y = (1, 0, 0)'

$$\Delta w_{ij} = x_i y_j \quad \text{vs.} \quad \Delta w_{ij} = -x_i y_j$$

生物上發現兩者並存：STDP（左）& BCM rule（右）

spike timing depedece



這邊有神經傳地性的方向
先 j 再 I
使得 deltaWij > 0
Hebbian rule

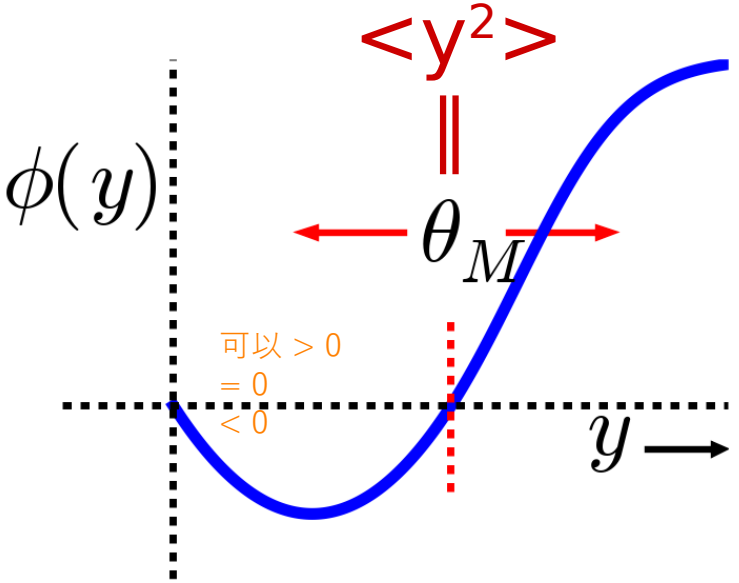
但是如果刺激只有 i
(但 i 本身不會傳給 j)
或是先 i (可能有其他 neuron
傳給它)再 j
則 deltaWij < 0
anti-Hebbian rule

因果關係

時間隔得越久
因果關係的連結會越弱

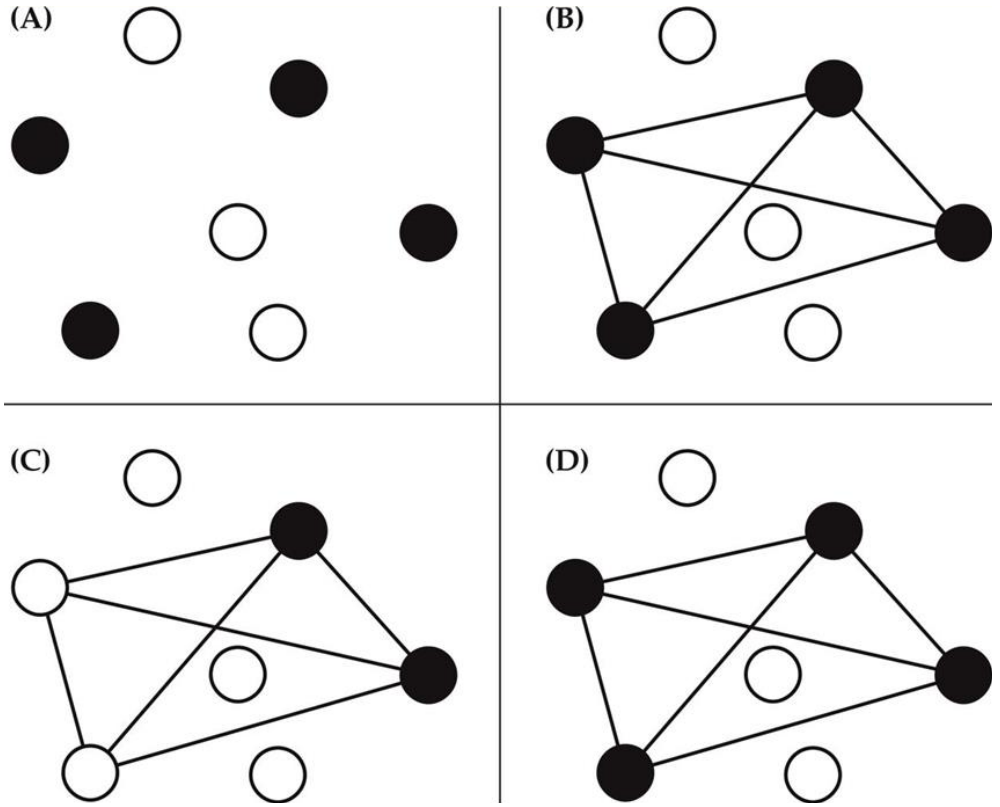
這邊是看 firing rate

$$\Delta w_{ij} = x_i \Phi(y_j)$$
$$= x_i [y_j (y_j - \theta)]$$



Hopfield Network

類 hippocampus 的 auto-associative memory



Memorize:

$$w_{ij} = \frac{1}{n}\sum_{p=1}^{n} \Delta w_{ij} = \frac{1}{n}\sum_{p=1}^{n} x_i^p x_j^p$$

Cued Recall:

$$x_i = \begin{cases} +1, \sum_j w_{ij} x_j \geq \theta_i \\ -1, otherwise \end{cases}$$

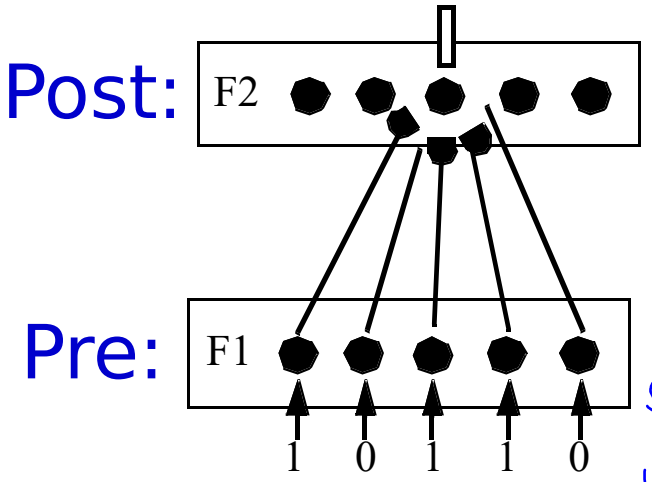
本質上是用大家來投票來實行

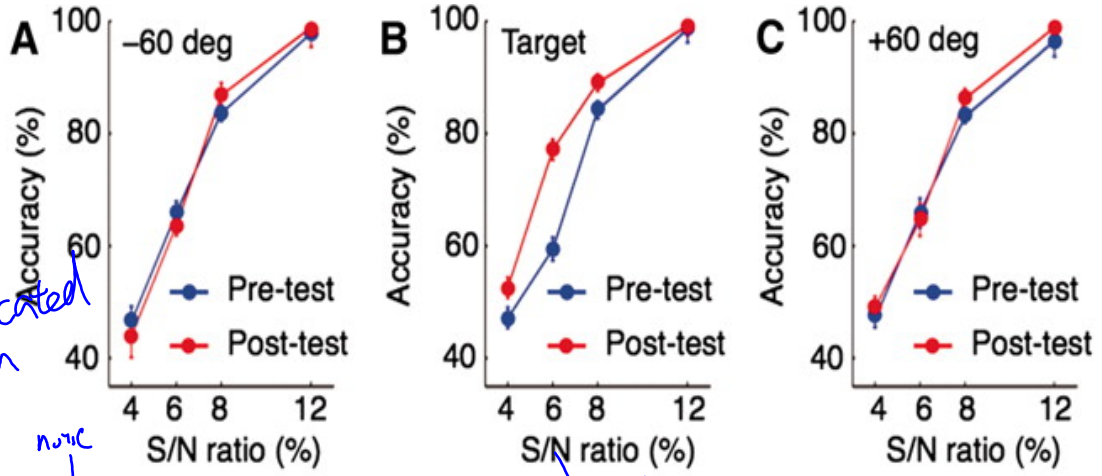
constraint satisfaction / convergent thinking

Boltzmann Machine是 Hopfield Net 的隨機版

InStar Learning

InStar: 突觸後神經元學習辨認突觸前的樣式





$$\Delta \vec{W}_i^{1\to 2} = y_j\left(\vec{x} - \vec{w}_i^{1\to 2}\right)$$

self-decayed term

Hebbian

$$\sum_{x=1}^{n} \left(\quad + \quad + \quad \right) / n = \quad$$

noise

```
x=array([0,1,0]) # repeated stimulus
W=random.rand(3) # random initial weights
for i in range(10): # 10 trials
    y=dot(W,x) # the only postsynaptic neuron
    W=W+y*(x-W) # postsynaptically gated InStar
    print(W,y)
```

Self-Organizing Map

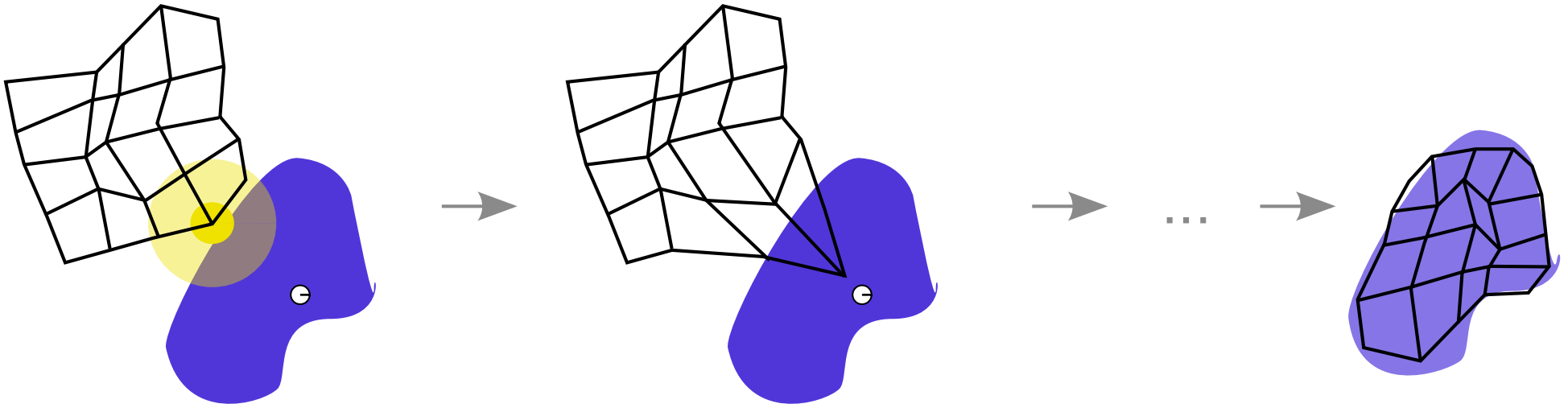
只是 InStar 的變形：每次學習的是一個贏者 y_j 的近鄰

贏者全拿，weight = 1

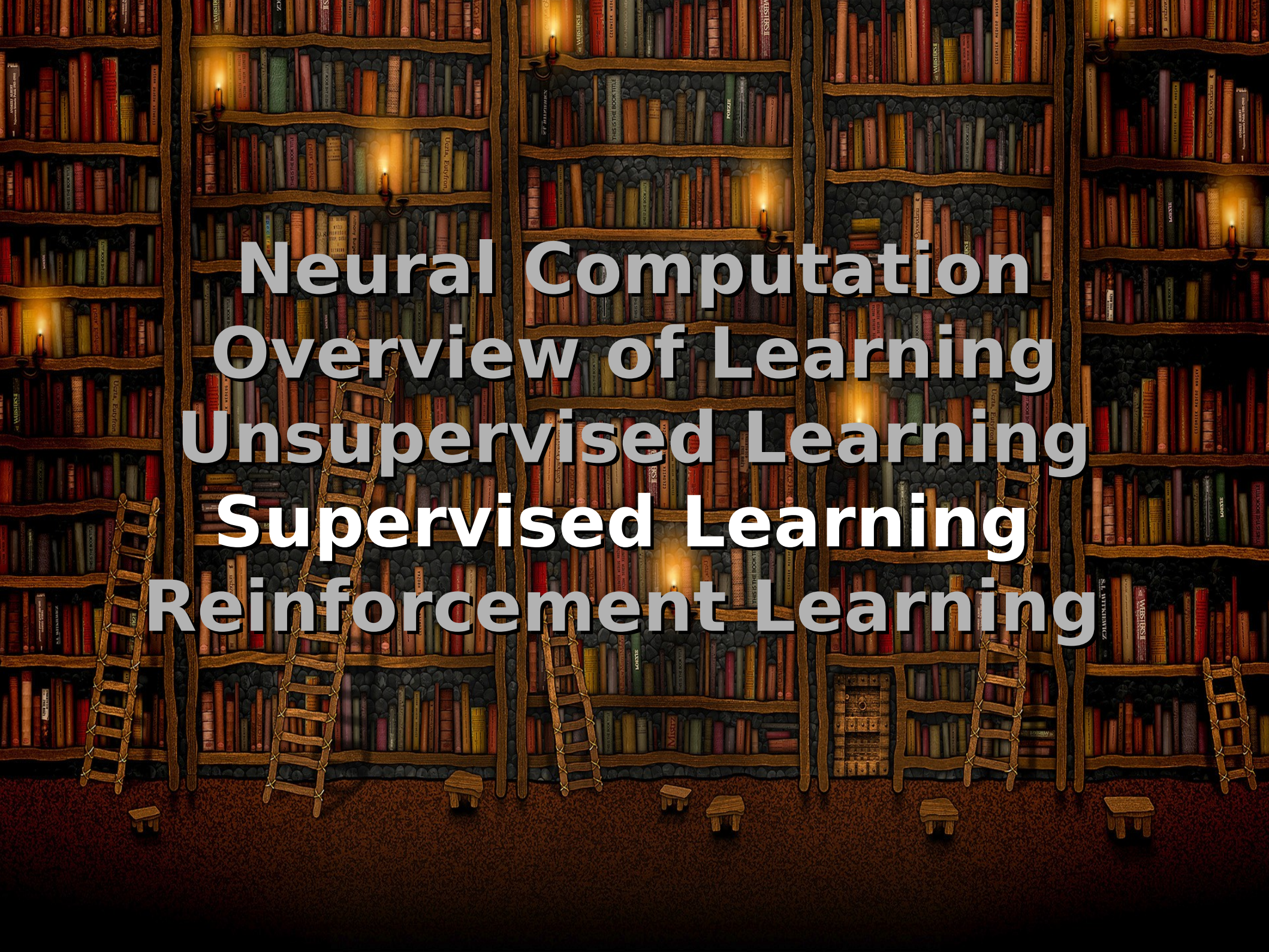
但這邊是贏者附近的也有 learning rate，只是隨贏者距離越遠會越少

$$\Delta \vec{w}_i = N(y_j, y_i)(\vec{x} - \vec{w}_i)$$

N 是某種 neighborhood function（如常態分佈）



學辨認刺激但保留神經元間的拓撲關係 (V1/A1/SMC)


Neural Computation
Overview of Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning

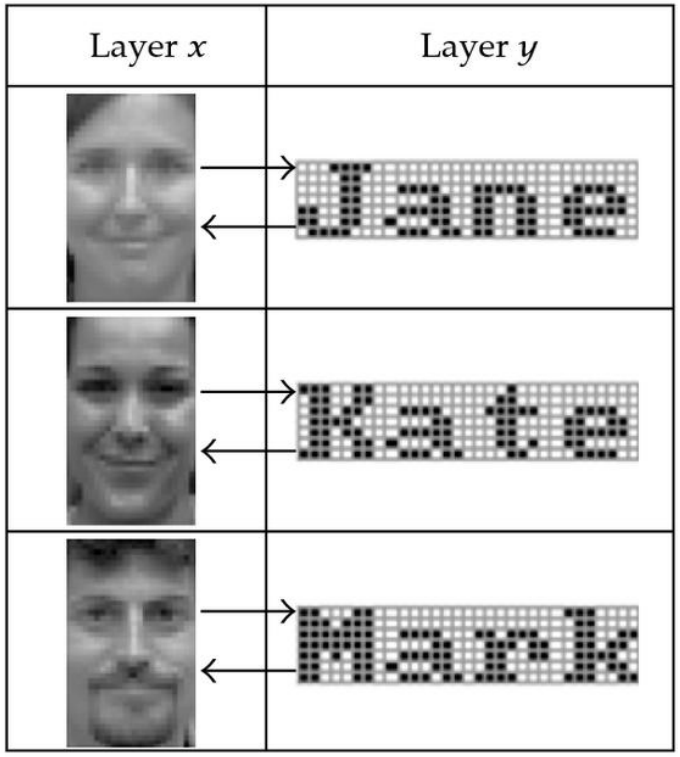
Supervised Hebbian Learning

輸入層的 x_i 和輸出層的 y_j 都 clamp 住來改變 w_{ij}

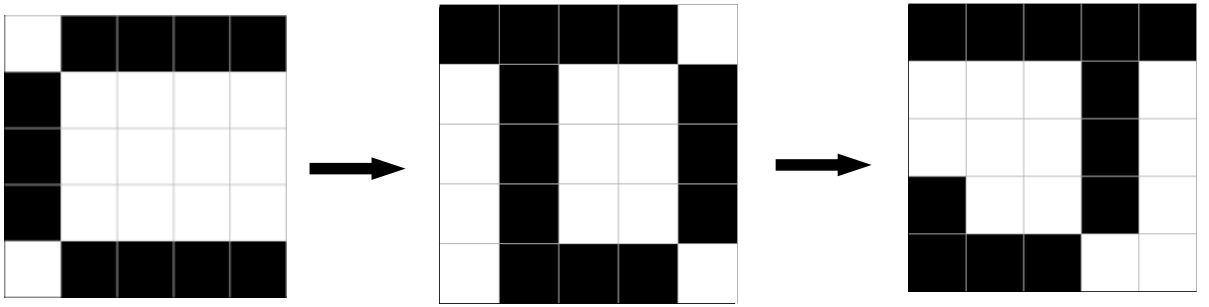
$$\Delta w_{ij} = x_i y_j \qquad w_{ij} = \frac{1}{n}\sum_{p=1}^{n} \Delta w_{ij} = \sum_{p=1}^{n} x_i^p y_j^p$$

x 是臉
y 是名字

Hopfield 網路可改為 Hetero-associative Memory
來連結不同本質 / 維度的 x 與 y 或產生 x 的序列

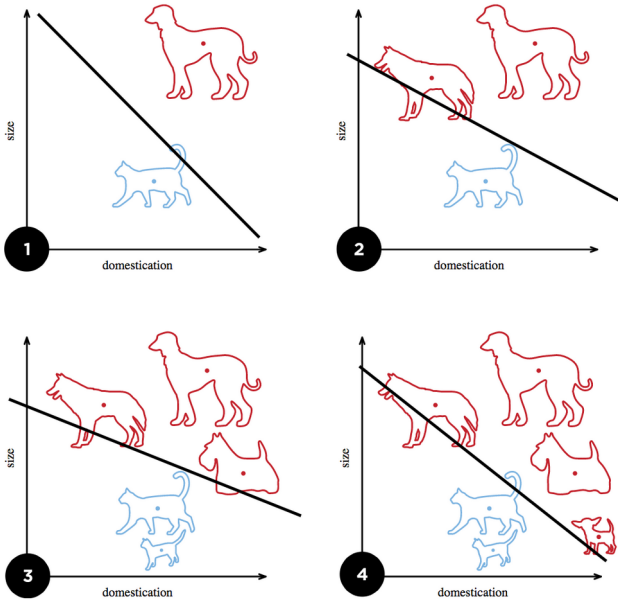


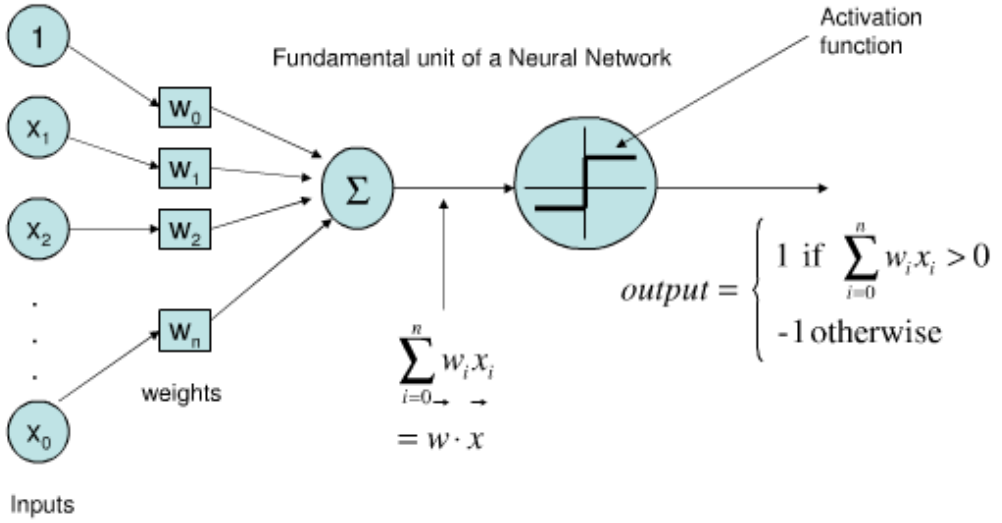
也可以去學這種序列式的資料



Perceptron: 分類

學習規則通常由 gradient descent 推導而來



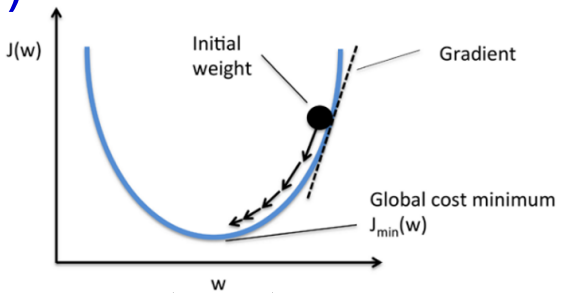


而不是這個 ??

$$E = -\left(WX^p\right)Y^p \neq \left(Y^p - sgn\left(WX^p\right)\right)^2$$

為何 error 是這個？
(cf. cross-entropy)

一些感想
統計模型其實有點是直接找出最佳解
但機器學習種點在於學習
我們是如何 "update (過程)" 到最佳解



這不就是 Supervised Hebbian 嗎 !?

$$W^{new} = W^{old} + \Delta W = W^{old} - \partial E\left(W\right)/\partial W = W^{old} + X^p Y^p$$

Delta Rule：迴歸

x_i驅使 y_j 產生的錯誤 $\delta_j=(t_j-y_j)$ 要透過修改 w_{ij} 來減小

$$y_j=\sum_i w_{ij}x_i$$

$$\Delta w_{ij}=x_i\delta_j=x_i(t_j-y_j)$$

何時deltaW = 0， 要馬x = 0 , 要馬 t = y
1. 沒有 error (t-y = 0)， 就不會學習
2. 上游神經沒反應,
即便有誤差我還是不會學習,
應未來源不甘它

上式只適用兩層且線性的網路
對任意的激發函數 f 則可由最小平方誤差法推得：

$$y_j=f\left(\sum_i w_{ij}x_i\right)$$

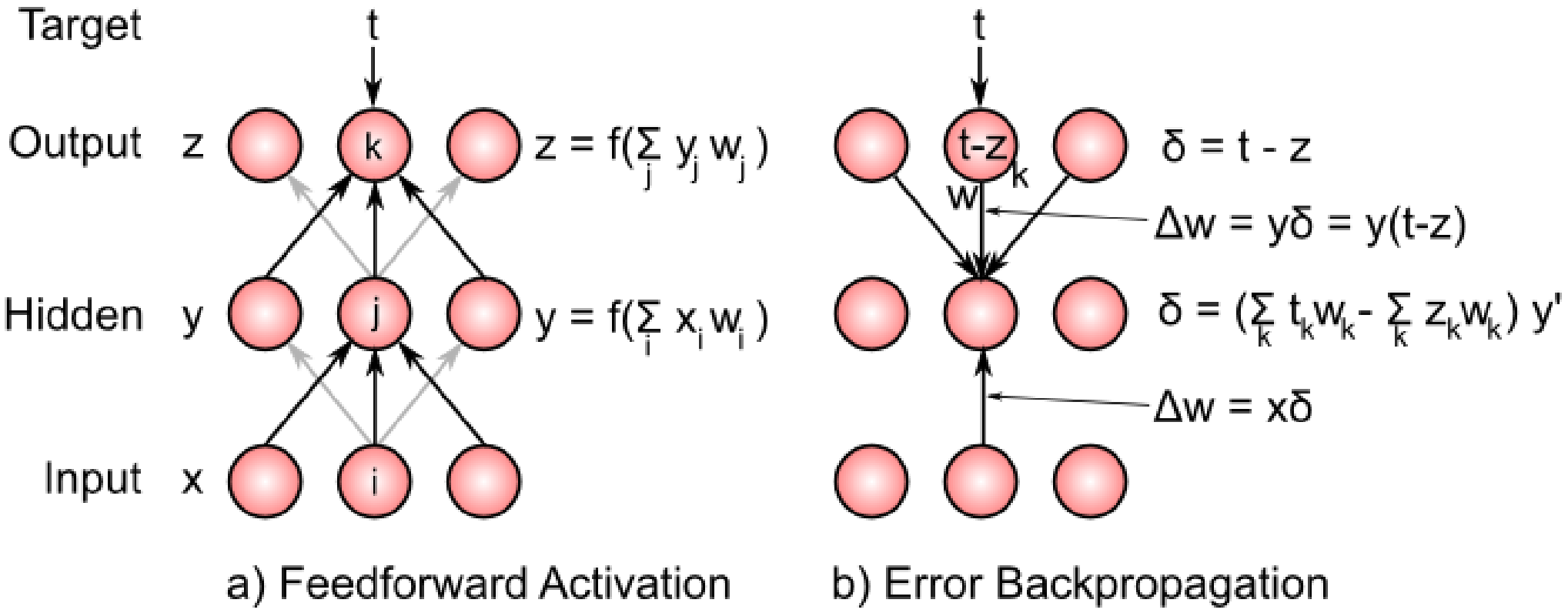
可視為非線性網路的 δ_j

$$\Delta w_{ij}=x_i\delta_j f'(h_j)=x_i(t_j-y_j)f'(h_j)$$

注意：若 $x_i=0$(不在場)，則 $\Delta W_{ij}=0$(不是它的錯)

Backpropagation

多層 NNs 常用的非生物性演算法（因違反 locality）



當激發函數是 sigmoid 時 BP 的推討請參考這裡

Revisiting Biological Plausibility

大腦內可能有接近計算 Back Prop 的方法

Review

Theories of Error Back-Propagation in the Brain

James C.R. Whittington[1,2] and Rafal Bogacz[1,*]

This review article summarises recently proposed theories on how neural circuits in the brain could approximate the error back-propagation algorithm used by artificial neural networks. Computational models implementing these theories achieve learning as efficient as artificial neural networks, but they use simple synaptic plasticity rules based on activity of presynaptic and postsynaptic neurons. The models have similarities, such as including both feedforward and feedback connections, allowing information about error to propagate throughout the network. Furthermore, they incorporate experimental evidence on neural connectivity, responses, and plasticity. These models provide insights on how brain networks might be organised such that modification of synaptic weights on multiple levels of cortical hierarchy leads to improved performance on tasks.

Highlights

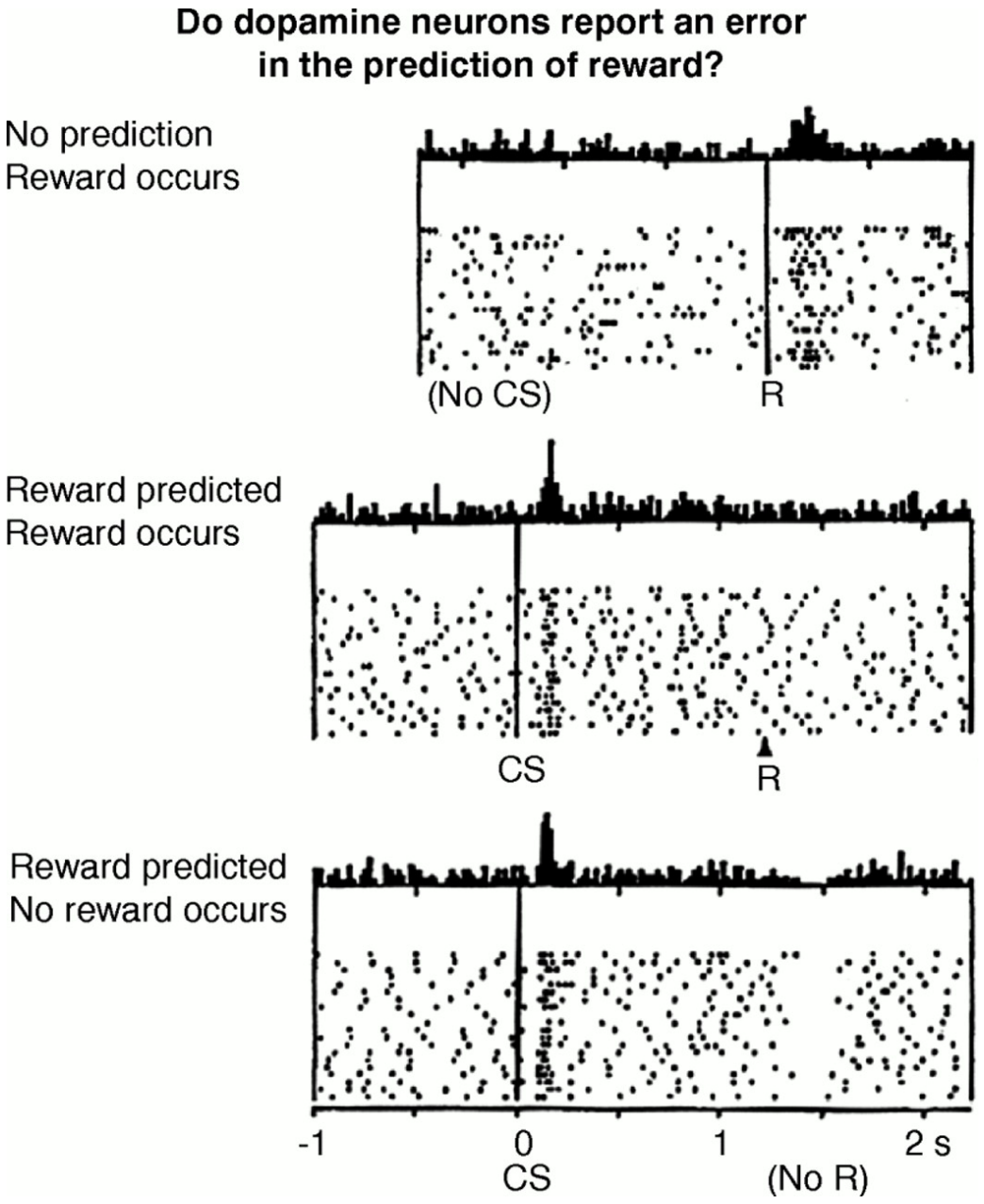
The error back-propagation algorithm can be approximated in networks of neurons, in which plasticity only depends on the activity of presynaptic and postsynaptic neurons.

These biologically plausible deep learning models include both feedforward and feedback connections, allowing the errors made by the network to propagate through the layers.

The learning rules in different biologically plausible models can be implemented with different types of spike-time-dependent plasticity.


Neural Computation
Overview of Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning

獎賞與多巴胺 (Dopamine)



古典制約學習
可看成是學 S-S 連結
或是刺激替代

注意左圖神經元反應的
是獎賞的預測錯誤
而非獎賞本身

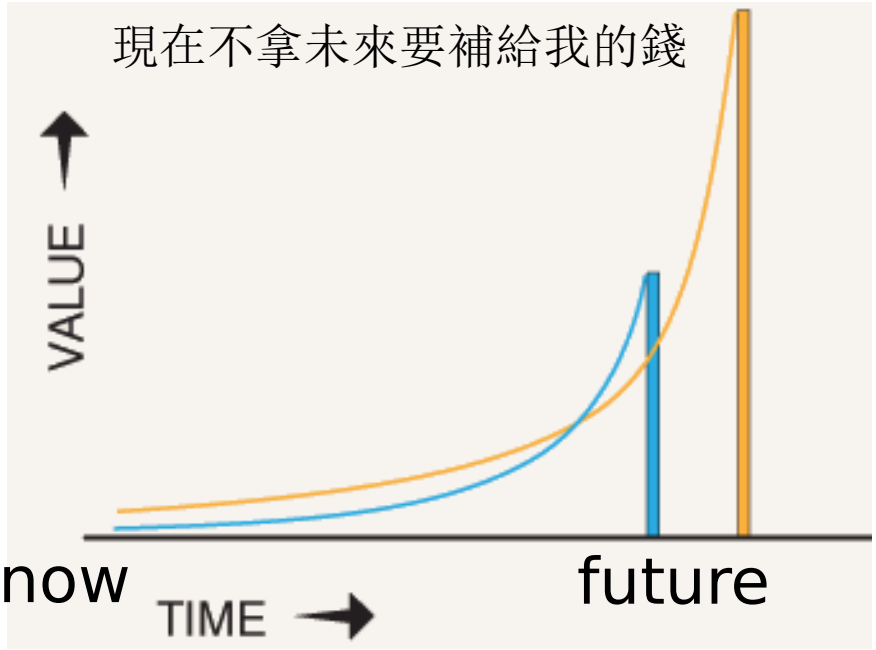
可用 temporal δ rule
可解釋 DA dip=r-r̂

$$\hat{r}^t = \sum_i w_i^{t-1} x_i^{t-1}$$

$$\Delta w_i = x_i^{t-1} \delta^t = x_i^{t-1} \left(r^t - \hat{r}^t \right)$$

更精確的獎賞估計應考慮長遠的未來

沒帶來立即獎賞的 CS 其實預測之後的各種獎賞



獎賞的時間離當下
愈遠價值愈低
(temporal discounting)

$$\hat{V}^t = \sum_i w_i^{t-1} x_i^{t-1}$$

$$\Delta w_i = x_i^{t-1}\left(V^t - \hat{V}^t\right) = x_i^t \delta V^t$$

因不知 V 無法計算 δV

$$V^t = r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} \ldots$$

若 γ=0 回到上頁公式
上式可改寫成遞迴形式：

$$V^t(t-1) = r^t(t) + \gamma V^{t+1}(t)$$

學習目標： 上面等式成立

學習過程：讓 $\delta V \equiv r(t) + \gamma \hat{V}(t) - \hat{V}(t-1) \to 0$

古典制約的 TD Learning

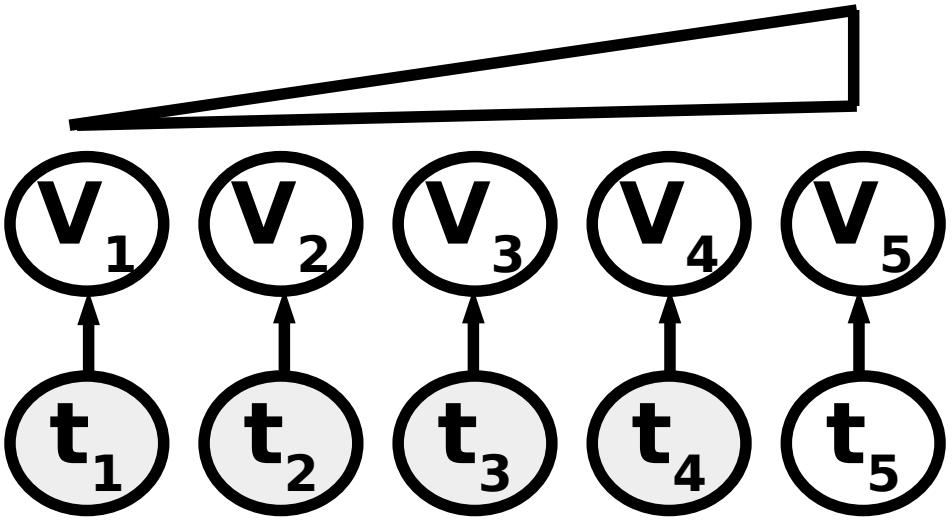
TD Learning $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$

Previous estimate

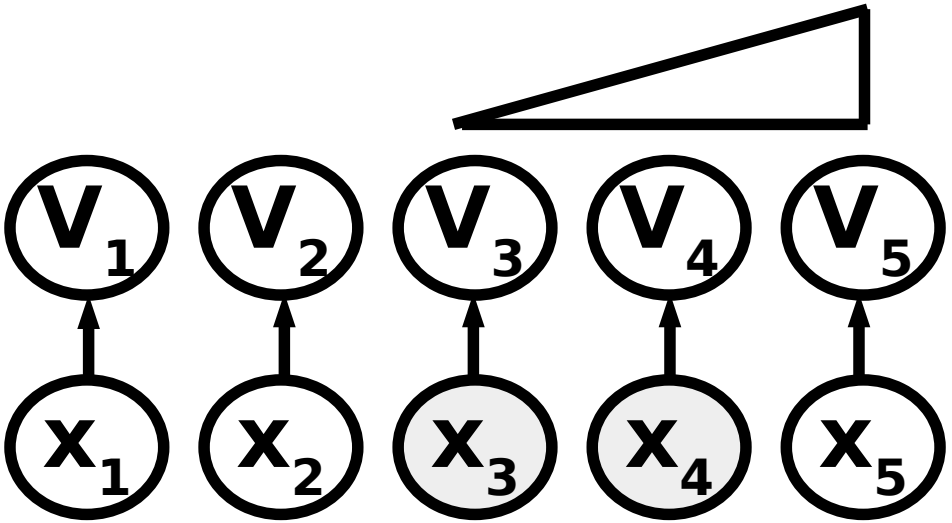
Reward t+1

Discounted value on the next step

TD Target



若 s=t 則 V(t) 變為
無邊界的時間折價函數
（如無刺激出現但
固定 5 秒後出現酬賞）



若 s=x(t) 則 V(x) 變為
有邊界的時間折價函數
（如有刺激出現，且之後
固定 3 秒後出現酬賞）

操作制約的 Q-Learning(1/2)

Q(s,a) 是狀態 s 與行動 a 的函數

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}} \right)$$

在 t 時要選什麼動作變成下一個狀態要根據既定 policy

計算上常離散化 s 與 a 來做成一個 Q(s,a) 的查找表

Game Board:

Current state (*s*): 000 010

Q Table: γ = 0.95

	000 100	000 010	000 001	100 000	010 000	001 000
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	0.4	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

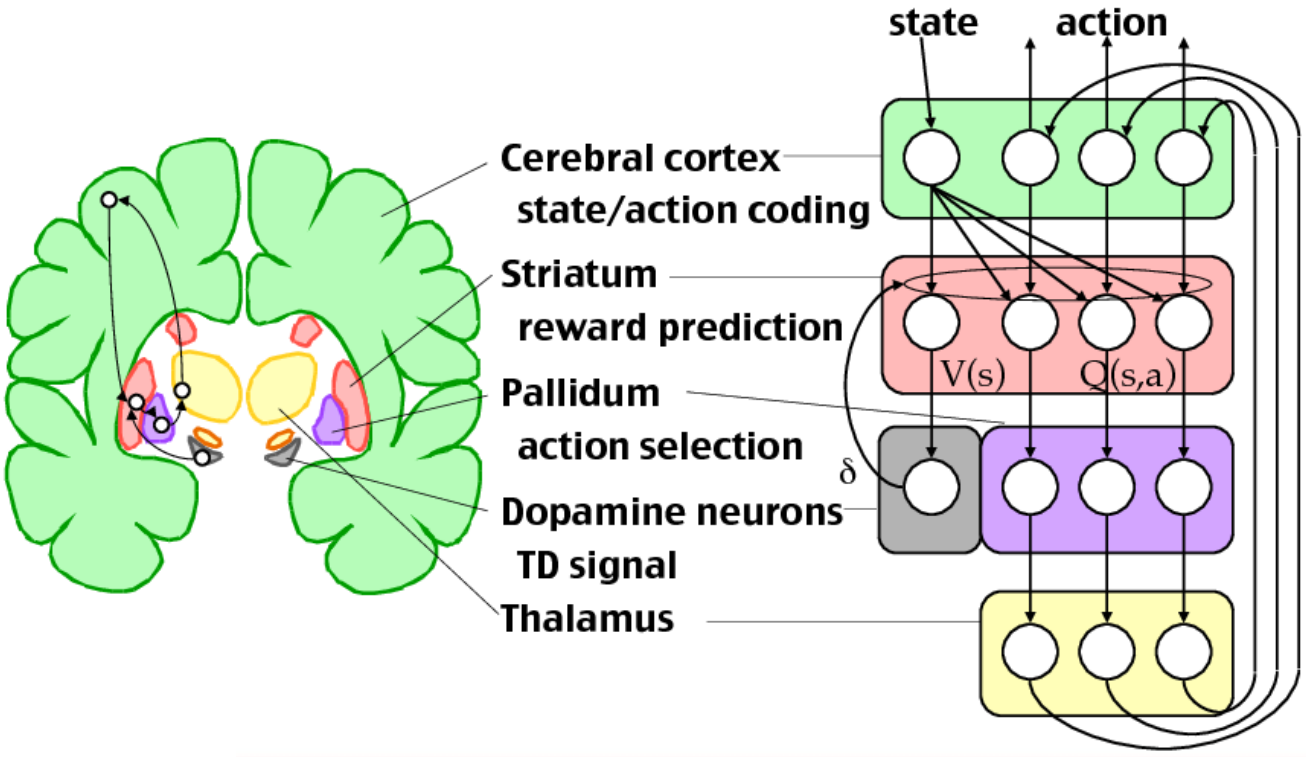
操作制約的 Q-Learning(2/2)

Q(s,a) 是狀態 s 與行動 a 的函數

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}} \right)$$

在 t 時要選什麼動作變成下一個狀態要根據既定 policy

計算上常離散化 s 與 a 來做成一個 Q(s,a) 的查找表



Game Over