第三章　小腦模型控制器理論

CMAC最早於1975年由美國學者J.S. Albus根據Marr所發表小腦架構[16][17]，並提出數學函數模型，發展至今已有30年的歷史，CMAC最早被Albus用於機械手臂的控制上，其主要特點是不需經過繁瑣的數學運算處理與分析來達成控制的目標，而是事先將量化的數學函數資料，建立記憶體樣本資料，透過檢索方式取出數個對應於某些特定輸入的記憶體內容值，經過簡單的算術累加後，達到對複雜系統的控制[16][17]。CMAC是模仿人類小腦皮質分層儲存訊息的架構得而來，在過去的研究中，成功的運用於控制[18][19][20][21]、信號處理[22]、圖像辨識[23]與錯誤診斷[24]等工程領域，相較於類神經網路而言，CMAC沒有牽涉到艱深的數學運算，構造簡單易於處理，且在學習的過程中，改善了類神經網路學習收斂速度慢的缺點，而產生一種有別於類神經網路的學習架構，它具有結構簡單、學習行為可靠、良好的區域量化能力、學習速度快速收斂及可學習非線性系統等特性，近年來已普遍獲得各類研究領域人員的重視。

3.1 傳統小腦模型控制器

傳統CMAC所強調的特性，顧名思義是模仿人類小腦皮質分層儲存訊息的資料儲存架構，人類的小腦的記憶容量大，對於各種訊號的學習與記憶各自使用一部份區域的記憶體，而對相似信號的輸入，則產生反射性的輸出動作。例如：當在學習騎腳踏車的過程中，聽取別人的學習經驗與騎乘方法，就好比是CMAC的學習樣本，在經過多次的失敗與跌倒後，會修正騎乘時的平衡感、重心位置與輸出力量，最後修正到適當的輸出力量，學會了如何騎乘腳踏車，就算換了一台不同特性的腳踏車，人們也會依循過去所累積的經驗與方法，在剛開始騎乘時或許需要適應一下，經過一段學習時間後，便可以很穩定的騎乘。而CMAC在控制受控體的方法，就是模仿人類的小腦學習方法，首先將學習樣本量化後輸入記憶體中，對CMAC作訓練，當實際輸出信號(實際值)與期望值有所出入時，便將此誤

差平均值回存至剛才資訊檢出的記憶體位置，作為修正誤差，在經過多次的迭代過程後，即可到正確的輸出。

3.2 傳統小腦模型控制器演算法

傳統 CMAC 之學習及回想（recall）演算法基本上是經由一連串的映射方式與迭代來達成。在 CMAC 的技術中，可分為 4 個主要步驟。

1. 規劃出CMAC訓練樣本的學習空間：學習空間的多寡關係著學習結果的精密度，此樣本空間切割的越細密，使得在同樣的學習範圍內，則學習空間則越多，控制的精密度越高；當輸入的控制變數為1個時，學習空間則為一維，若輸入控制變數為多個時，學習空間則為多維。一般而言，CMAC學習能力並非全域性的，僅能針對所規劃的學習空間進行學習，CMAC對於學習空間以外的樣本是無法進行正確學習的。
2. 量化輸入信號：當學習空間規劃完成後，接著便將此空間量化（quantize）成一塊塊不連續的狀態，且每個狀態均對應到數個超立方塊（hypercube）覆蓋。一般而言，CMAC輸入的信號為類比信號，須先將此信號量化之後，方能進行編碼分群。全部超立方塊所在的空間稱為聯想空間，且每個不連續狀態均有自己所屬的一組超立方體所構成的聯想向量，這些超立方塊會各自對應到一個專屬的真實記憶體位置。當CMAC由學習空間中獲得一組輸入信號後，系統會先找出這組輸入在學習空間中的所在位置，並判斷它位於哪個不連續狀態中。而每一個學習空間會對應到聯想空間中的一組索引指標記憶體。
3. 學習演算法：當CMAC訓練的過程中，對應於某一訓練樣本的CMAC輸出值可能會和此樣本被期望的輸出值有所出入，將此樣本的期望輸出與CMAC實際輸出的誤差平均分配回剛才資料檢出的記憶體位置，經過多次迭代後，CMAC對這個相同的訓練樣本輸出反應會加以修正，直到輸出值收斂到容許的誤差範圍以內，才會停止學習，此即為CMAC的學習

過程。如3-1、3-2式，表示目前輸入CMAC的樣本所對應之記憶體內容，為前一個樣本所對應之記憶體內容加上其修正量。

w：權重

α：學習率

s：目前學習的樣本編號

$\bar{y}_s$：第s樣本的實際值輸出

i：目前學習的週期數

N_e：每一個樣本所對應的真實記憶體數量(層數)

a：真實記憶體選擇向量(聯想向量 association vector)

$$w_s^{(i)} = w_{s-1}^{(i)} + \Delta w_{s-1}^{(i)} \tag{3-1}$$

$$\Delta\, w_{s\text{-}1}^{(i)} = \frac{\alpha}{N_e} a_{s\text{-}1} (\bar{y}_{s-1} - a_{s\text{-}1}^{T}\, w_{s\text{-}1}^{(i)}) \tag{3-2}$$

4. 回想演算法：當期望值或訓練樣本輸入CMAC後，經過一連串的量化、映射後會產生一組索引指標，每一組索引指標記憶體指向真實記憶體，故若所指向的真實記憶體的內容加總起來，則即為CMAC對此不連續狀態的反應輸出。如3-3式。

j: 真實記憶體編號。

Nh：真實記憶體數量(超立方塊的數量)

$$\bar{y}_s = [a_{s,1}, a_{s,2}, \ldots, a_{s,Nh}] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{Nh} \end{bmatrix} = \sum_{j=1}^{Nh} a_{S,j}\, w_j = a_S^T w \tag{3-3}$$

3.3 一維小腦模型之基本架構

如圖3-1 所示，當輸入的參考狀態只有一組時，CMAC只需規劃出一維的學習空間，每一個參考狀態會對應到一個索引指標，每一個索引指標會對應到一組真實記憶體，每一組真實記憶權重的總和，即是此參考狀態的輸出。

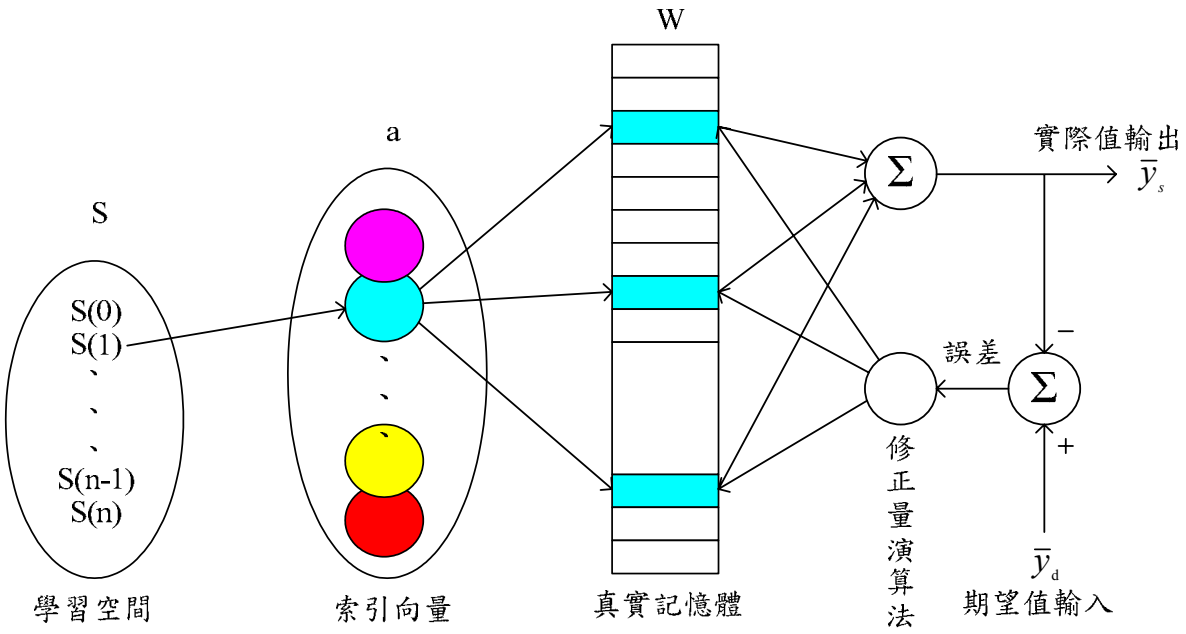

圖3-1 一維小腦模型控制之基本架構

3.3.1 一維小腦模型記憶體單元的分割方式

每一個輸入狀態最後會對應到一組實體記憶體，此記憶體的數量所代表的是量化層的數量，且此量化層會有重疊的現象，如圖3-2所示，具有4層量化層之CMAC，當量化層的數量越多時，則CMAC學習的效果越好，但所需的記憶體空間與運算時間也會隨之增加。以圖3-2為例，規劃4層量化層，所以每一組索引指標會對應到4個實體記憶體，所需使用到的記憶體總數如3-4 式。

k：參考狀態的數量(學習空間)。

m：量化層的數量、索引指標所對應實體記憶體的數量。

Memory size= k+m-1 (3-4)

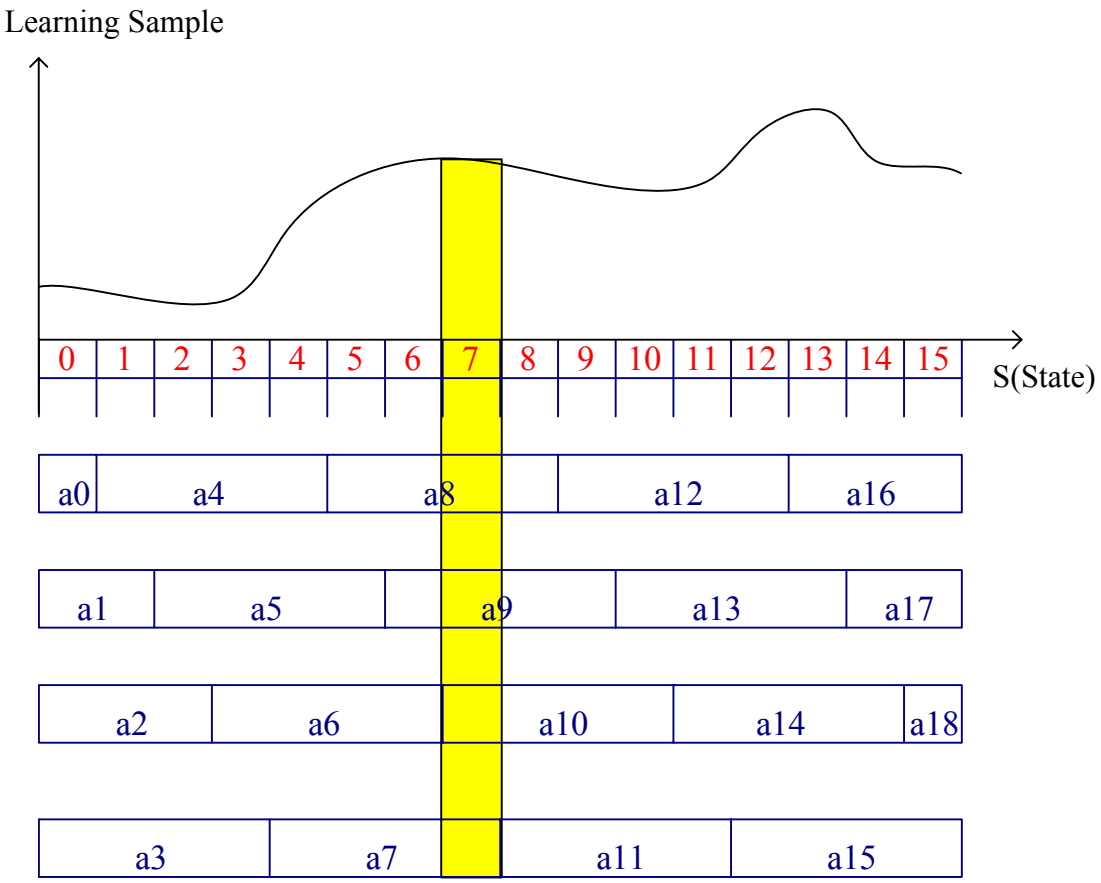

圖3-2 一維小腦模型記憶體單元的分割方式

3.3.2 一維小腦模型記憶體單元索引指標建立表

每一個索引指標會對應到一組實體記憶體，若CMAC設計4層量化層，則每一個索引指標會對應4個實體記憶體，被對應到的實體記憶體，索引指標會設為1，其餘皆為0，如表3-1 所示，將輸入狀態分割成16等分，每一個狀態對應到4個實體記憶體，共使用19個實體記憶體。

如3-3式，當輸入狀態維S(7)時，所對應到的索引指標a7, a8, a9, a10會被設為1，其餘皆被設為0，S(7)的輸出就是將a7, a8, a9, a10所對應到的實體記憶體內容

相加。

表3-1 記憶體單元索引指標建立表

	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	a17	a18
S(0)	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S(1)	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S(2)	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
S(3)	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
S(4)	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
S(5)	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
S(6)	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
S(7)	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
S(8)	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
S(9)	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
S(10)	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0
S(11)	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
S(12)	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
S(13)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
S(14)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
S(15)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

3.4 二維小腦模型控制之基本架構

當輸入參考狀態有2組時，輸入的參考狀態可以規劃成二維學習空間，如圖3-3 所示，除了學習空間不同外，基本架構接與一維CMAC相同，本節以說明CMAC如何進行二維學習空間之量化以及輸入狀態之記憶體定址方式。如圖3-4所示之二維學習空間可看出，每個輸入變數軸（x_1及x_2）均被量化成16段不連續的小單元，在此稱為一個元素，且每個元素的寬度稱為解析度（resolution）。每一個軸切割的元素越多時，解析度越高，學習的效果越好，兩輸入變數軸的不連續小單元所圍成的一塊塊小方塊稱為參考狀態，由圖可知此學習空間總共被量化成16×16=256個不連續參考狀態。

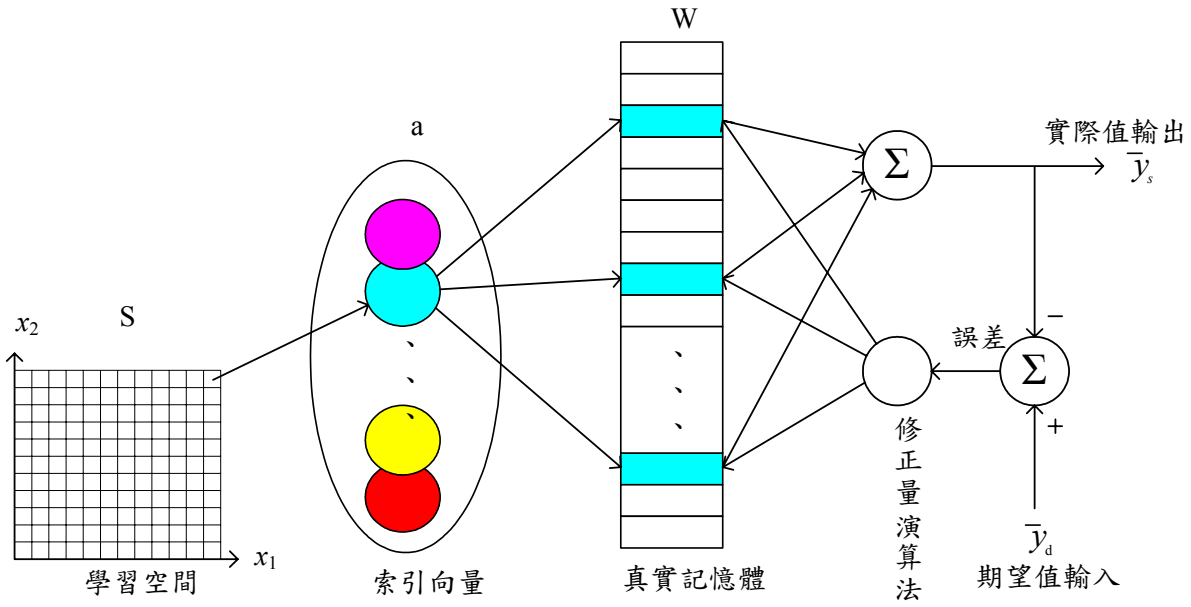

圖3-3　二維 CMAC 學習空間基本架構

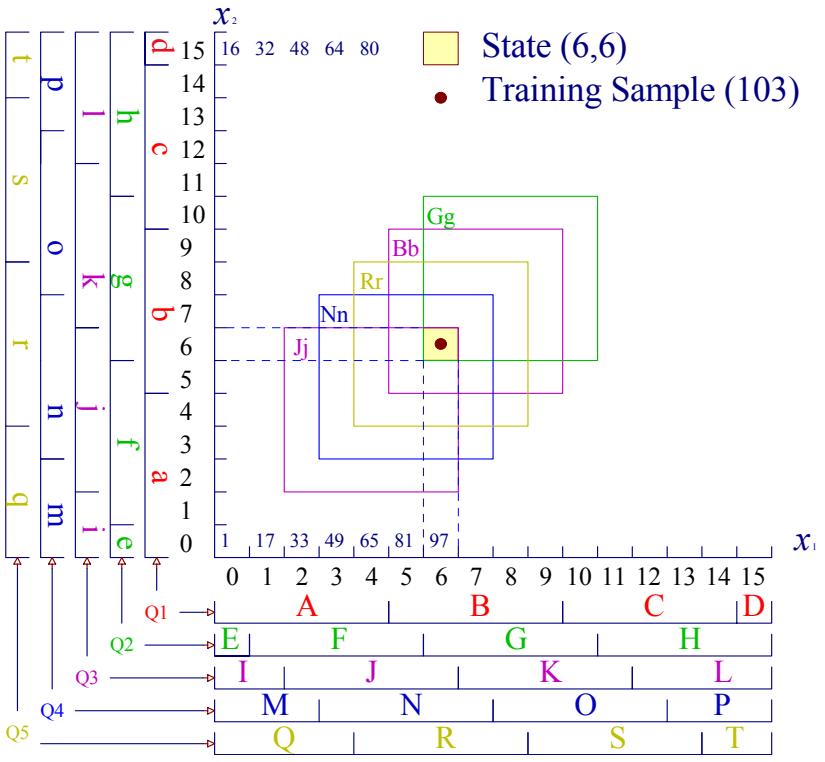

圖3-4　二維 CMAC 學習空間記憶體單元分割方式

若將此空間規劃5層量化層，每一個區段（block）由5個參考狀態所組成，由圖3-4 可知共可形成ceil(16/5)=4個區段，其中ceil() 為Matlab中無條件進位的指令，第1層量化層（quantization layer, Q_1）總共由4個區段所形成，其中包含三個完整區段（A、B及C）及一個不完整區段（D）；將第1層量化層中的區段向右平移一個元素以形成第2層量化層Q_2及另外4個區段（E、F、G及H），以此類推，繼續平移兩次後可分別得到第3、4、5層量化層Q_3、Q_4及Q_5，因為繼續平移的話會產生與Q_1相同大小及位置分佈的區段，為避免重複之區段配置造成記憶體浪費之情形，故本例中產生第5層量化層後即可停止（量化層數與完整區段所含的不連續單元個數相等）。x_2軸的量化層產生方法亦和x_1軸一樣，因圖3-4 設定x_2軸之完整區段元素亦為5個，故x_2的量化層及區段個數與排列均和x_1軸相同。當兩軸的各量化層之區段分佈均確定後，吾人限制只有"不同軸的同層區段"才能兩兩形成超立方塊（hypercube），由此可知針對圖3-4 的例子而言，共可產生5層，每層4個區段，共5×4 =20個區段，最後超立方塊總數共有4×20=80個，而每一個超立方塊代表一個記憶體空間，所以必需使用到80個記憶體空間。此例中所有超立方塊名稱如表3-2 所示。

表3-2 量化層所形成之超立方塊名稱

層名	超立方塊名稱															
Q_1層	Aa 1	Ab 2	Ac 3	Ad 4	Ae 5	Ba 6	Bb 7	Bc 8	Bd 9	Be 10	Ca 11	Cb 12	Cc 13	Cd 14	Ce 15	Da 16
Q_2層	Ee 17	Ef 18	Eg 19	Eh 20	Fe 21	Ff 22	Fg 23	Fh 24	Ge 25	Gf 26	Gg 27	Gh 28	He 29	Hf 30	Hg 31	Hh 32
Q_3層	Ii 33	Ij 34	Ik 35	Il 36	Ji 37	Jj 38	Jk 39	Jl 40	Ki 41	Kj 42	Kk 43	Kl 44	Li 45	Lj 46	Lk 47	Ll 48
Q_4層	Mm 49	Mn 50	Mo 51	Mp 52	Nm 53	Nn 54	No 55	Np 56	Om 57	On 58	Oo 59	Op 60	Pm 61	Pn 62	Po 63	Pp 64
Q_5層	Qq 65	Qr 66	Qs 67	Qt 68	Rq 69	Rr 70	Rs 71	Rt 72	Sq 73	Sr 74	Ss 75	St 76	Tq 77	Tr 78	Ts 79	Tt 80

且因為有 5 層量化層，故每個不連續狀態皆有 5 個超立方塊覆蓋，以狀態(6,6)為例，因 x_1 軸對應到它的區段為{B, G, J, N, R}且 x_2 軸對應到它的區段為{b, g, j, n, r}，根據只有同層區段才能結合成超立方塊的原則，可清楚地看出覆蓋到它的超立方塊為{Bb, Gg, Jj, Nn, Rr}5 個，且由表 1 可知，{Bb, Gg, Jj, Nn, Rr}5 個超立方塊分別位於超立方塊集合的第 7、27、38、54 及 70 個記憶體上，故 CMAC 對狀態(6,6)之輸入產生實際的輸出就是將這 5 個記憶體的內容相加。

如 3-3 式，其中 a_j 表示超立方體選擇向量中的元素，其中只有 a_7、a_{27}、a_{38}、a_{54} 及 a_{70} 的值為 1（被選中），其餘皆為 0（不被選），這代表了對於狀態(6,6)而言，只有第 7、27、38、54 及 70 個超立方塊覆蓋到它，即這個狀態只用到了第 7、27、38、54 及第 70 個真實記憶體位置；w_j 代表真實記憶體陣列內的元素；N_h 則為全部的超立方塊數（真實記憶體數），此例中 N_h=80。