第二章 小腦模型控制器

小腦模型控制器(CMAC)為類神經網路的一支系。CMAC 理論首先由 Albus 於 1971 年提出[3]，其主要以 Marr 於 1969 年所提出的人類小腦模型架構為基礎，其架構可視為聯想記憶體(Associative Memory Neural Network, AMN)與倒傳遞類神經網路(Back Propagation Neural Network, BP)的結合，它具有 AMN 的輸入映射特性，與 BP 監督式學習的輸出特性。Albus 利用此架構成功的控制機械手臂[1-2]，也因此引起人們的關注。在接下來的 5 小節中，將分別介紹 CMAC 的相關研究、基本架構、記憶體之分割方式、數學表示法及有關學習與回想之演算法。

2.1 有關小腦模型控制器的相關研究

長久以來，人們在面對控制上的問題，便習以對受控體建立數學模型，進而選取控制法則來處理問題，較為常見的有古典控制中的 PID 控制；近代控制中的狀態變數控制法、可變結構控制法、自適應控制法等。

但漸漸的，人們發現，當我們檢驗生物的活動方式時，例如：松鼠在樹梢間的跳躍、鳥兒的飛翔、或是人們打網球等，我們會直覺的想到，以數學所獲得的解，無法完全地展現出運動的複雜行為。也因此人們開始思考，為什麼要避免利用數學來建立模型，其基本理由可歸納以下四點：1.以數學所建立的模型，需要深厚的工程背景，要知道受控體的運作程序，與受控體如何運作等相關知識。2.有很多傳統程序，無法用數學來建立模型，因此需要人們來控制。3.對於要求快速的場合，若以數學的方式來建立，會變得很複雜，因此演算法會很難達成，例如：參數的估測，多項式的近似等。4.以數學所建立的模型，會隨使用的時間而喪失功用[4]。

早期雖然有人嘗試以數學式子去建構生物的運動行為，但均未成功；後來以生物本質為理論基礎的人工智慧群體，投入心力去研究，卻也都不如預期理想。基於此，我們換個角度來看待問題，嘗試不去建立這些複雜的模型，而只用函數模仿人類頭腦的控制方式來達成，CMAC 有關的發展，便是基於此精神而孕育而生。以下小節，我們將討論 CMAC 的優缺點與至今的相關研究成果。

2.1.1 CMAC 的優點

自從 1975 年 Albus 提出小腦模型控制器以來，至今已有許多學者陸續為 CMAC 的性能進行各項實驗，以下僅列舉數篇，說明有關 CMAC 的優點。

Kraft 等人於 1989 年，提出古典控制中的自動調節器與模型參考控制系統來和 CMAC 作比較。其中的自動調節器係使用最小平方建模技術為演算法，來估測未知的受控體參數，然後利用回饋控制來調整參數。模型參考控制系統其主要原理為，先設計好參考模式的系統，如此，受控體將依循參考模式的輸出作調整，一旦自適應調整完成，則受控體將吻合參考模式所設計的特性。而其中自適應的演算法則，是應用 Liapunov 函數的正定法則來達成。最終的結論為，在對受控體作誤差追蹤、學習時間和控制作用力等三項比較後，顯示 CMAC 對非線性與干擾之問題皆能克服且效果良好[5]。

Kraft 等人復於 1990 年，分別對 CMAC 測試兩種不同的控制架構：1. 採用前饋式架構，來學習倒置的控制系統。在此 CMAC 將學習倒置的受控體動態特性。用以預測下一次的控制信號。2. 將 CMAC 運用於模式參考架構。使用誤差追蹤來調整權重，CMAC 根據過去的學習經驗誤差值，送出信號來修正控制信號。其結果顯示兩者皆有良好的特性，但似乎第二種方式的效果較佳[6]。

Glanz 等人於 1991 年，為 CMAC 摘要出最為人們所熟知的特性有以下數點：1.輸入為整數向量，而輸出為實數向量，其輸出向量通常為 1 到 10。2.快速的學習收斂速度。3.具處理非線系統的能力。4.優良的類化能力。5.構造簡單，硬體易於實施。6.運算法則簡單，軟體編寫容易[7]。

2.1.2 CMAC 的缺點

相對於上述所提的優點，CMAC 也存在許多的缺點，其中有學者指出實際應用上的問題；亦有許多學者以理論的觀點為出發，為 CMAC 的限制作出建言。以下就列舉數篇說明有關 CMAC 的施用限制。

對於一高維度的函數近似問題，一直是控制動態系統的一大問題。例如：訓練資料的不足，或是取樣過多。因為資料的缺乏，會造成建模的不準；而取樣過多，則控制器就需大量的計算與大量的儲存空間。是以有一些研究者利用演算法避開維度高，而以對等的資訊原理在低維度擷取所需資料，例如：使用分量分析原理(principle component analysis)；使用輸入空間加入基礎函數的演算法；以漸增的方式建立聯想記憶體神經網路(AMN)等[8]。

Brown 等人於 1994 以最小平方誤差的理論提出證明，指出 CMAC 在高維度的輸入映射時，其所採用的方法為多對少的映射方式，因此會有數學建模上的缺陷。Brown 等人藉由飛機的控制系統，引用實例來加以說明。此外，作者亦舉出正交函數的例子來加以說明，最後為此學習的控制系統提出了兩點建議：1.改變學習率。2.選取不動作區(dead-zones)兩種方式，用來改善 CMAC 本質上缺失[8-9]。

Wang 等人於 1996 的研究指出，在使用雜湊編碼後，CMAC 無法增進函數近似的能力，而且會因所選的雜湊編碼方式之不同，造成 CMAC 學習精度很大的影響。研究並指出對於多數的應用，使用費不納西(Fibonacci)的雜湊編碼方式，可獲得更好的結果[10]。

Zhong 等人於 1997 年的研究指出，雜湊編碼會造成碰撞問題，且在收斂時會有三點缺點：1.會干擾收斂，甚至造成發散。2.造成收斂速度變慢。3.產生收斂的行為變差。例如：造成突然的昇降。這是因為雜湊編碼，會改變 CMAC 係數矩陣之特徵根所引起的。其改善之方法為，對 CMAC 的輸出關聯相量作補償，Zhong 等人提出使用 Window Sequence 方式，來克服雜湊編碼所帶來的問題[11]。

Hu 等人於 1999 年的研究指出，Albus 所提出之 CMAC 的主要缺點有：1.需要很大的記憶體。雖然最後由他本人提出雜湊編碼方式來改善，但仍存在碰撞問題。2.對於函數近似需要嚴密的理論[12]。

2.1.3 CMAC 對記憶體需求之改善

由上節的探討可知，CMAC 所面臨的最大問題，就是有關記憶體管理之問題，無論是使用雜湊編碼或是其他的演算法則，其目的多在降低記憶體之使用量，以下就列舉數篇相關文獻來加以說明。

Albus 指出，若函數的變化太大(在相鄰的區域內)，則 CMAC 將可能無法達到期望的精度。建議的改善方法為—增加解析度[2]。而其他的神經網路，如 MNN & RBFN 為靜態映射所常用的兩種神經網路。對於多層的神經網路(MNN)常會遭遇到的問題是學習的困難，例如：MNN 無法預期多長的時間將會收斂；也無法知道其所收斂的結果是否合乎要求。RBFN 常使用高斯函數為其基礎函數，這是因為使用高斯函數在區域擬合(fitting)時，收斂會非常容易[13]。但對於多變數的問題，卻會造成基礎函數快速的增加。因此很多的研究者建議，在輸入空間將每一維度的高斯函數予以比例的改變，然後再旋轉。其結果是損失了學習的效能。

針對 CMAC 的記憶體會隨輸入變數(維度)的增加，呈現指數的增加，因此限制了 CMAC 只能使用於低維度的應用。Menozzi 等人於 1997 年提出了，用多解析度的 CMAC 量化演算法。由於此演算法具備自我判斷輸入量

化的需求，因此改善了 Kim 等所提以自適應感應場在高維度時，會有的記憶體浪費的情形[14]。

Lin 與 Li 於 1996 年提出階層式自組織演算法，利用分類的方式，將一大問題分成若干的小模組，例如：Albus 之 CMAC 在 Ne=8，Nb=8，Nv=6 時需記憶體為 2097152；其可產生的模組相當於有 1365 個；但在階層化之後，由於具備自組織的能力，其所需之模組只要很少便可。並且經實驗顯示，其精度亦比 Albus 之 CMAC 為好，改善了擬合誤差的缺點[15]。

Hu 等人於 1999 年提出資料叢聚演算法，用以改善記憶體的需求(將稀疏矩陣變小，進而找出最大的資料聚集點)。施用此方法，可大量的減少記憶體的需求量，使得設計二足機器人變為可行[12]。

2.1.4 CMAC 學習速度的改善

Huang 等人發現傳統的 CMAC 在輸入空間量化時，採用均等量化的原則，這不僅造成學習干擾的問題，也造成了精度無法提昇。於是 Huang 等人於 1997 年以 Moody 的階層式架構，與多重解析度的輸入空間為基礎，配合動態解析度調整，採用自我成長的串級式階層架構，用於改善掃瞄器與列印輸出的誤差，獲得了精度與學習速度的雙重改善[16]。

2.1.5 CMAC 學習精度的改善

Pallotta 等人觀察 CMAC 的學習結果，發現記憶體內存值具有很多鋸齒狀的情形。這不僅輸出粗糙，也使得微分特性不佳。Pallotta 等人於 1999 年提出使用最佳化的觀點，建立一套新的權重平滑化演算法，除了以電腦模擬圖形來加以證明，改善了 CMAC 學習精度外，亦以振動控制來測試此演算法(傳統 CMAC 用來作振動控制是一項困難的問題)，發現在施用後，其學習速度、函數近似品質穩定能力和殘數(residual)所造成的高頻雜訊等問題均獲得改善，並且也改善了 CMAC 的微分特性[17-18]。

2.1.6 CMAC 的應用例子

Shiraishi 等人於 1995 年提出，利用 CMAC 的即時控制與快速的學習收斂特性，對高轉速的引擎作供油控制。此外，作者亦利用 CMAC 不需事先建模的優點，用於即時學習與產生自適應的效果[19]。

Hu 等人於 1999 年的研究中，闡述難以設計二足機器人的原因為：1.分析與控制困難。2.高維度。3.高耦合。4.複雜的非線性動態運動。這使得近代的離散控制理論很難對此複雜的行為，給出明確的控制量。最後 Hu 等人針對以上的數個缺點，提出了全面的改善措施，並成功的完成二足機械人研究[12]。

2.1.7 具微分特性的 CMAC

Lane 等人早期研究指出，傳統 CMAC 不具微分之特性。但 1992 年渠等提出植入基礎樣條函數(B-spline)，發現在植入基礎樣條函數後，對系統除了可施用於函數近似外，還具有微分特性，而使用基礎樣條函數有三個優點：1.確實(Positivity)：可消除很小的正數與負數，在運算時所產生的運算不穩定問題。2.由最基本的圖形所組成(Compact Support)：因為是由最基本的圖形所組成，讓樣條函數獲得高效率的運算。即使 x 是在[a,b]閉區間內，頂多也只有 n 個樣條函數的值不為零。3.正規化(Normalization)：此特性讓常數函數近似於樣條函數的值，可等同於單一常數係數[20]。

Chiang 與 Lin 研究 RBF 和 CMAC 此二類神經網路時發現，皆具有區域類化的能力，而彼此間最大的不同為 CMAC 在輸入空間植入高原函數(Plateau)，而 RBF 在輸入空間則是植入高斯函數(Gaussian)。1995 年 Chiang 與 Lin 的研究指出，在植入高斯函數後，不但可改善傳統 CMAC 的微分能力，且依然保有快速的學習效率(較 Albus 為慢)[21]。

2.2 小腦模型控制器的基本架構

傳統 CMAC 的架構圖如圖 2-1 所示，其架構可視為聯想記憶體與倒傳遞類神經網路的結合，它具有 AMN 的輸入映射特性，與 BP 監督式學習的輸出特性。

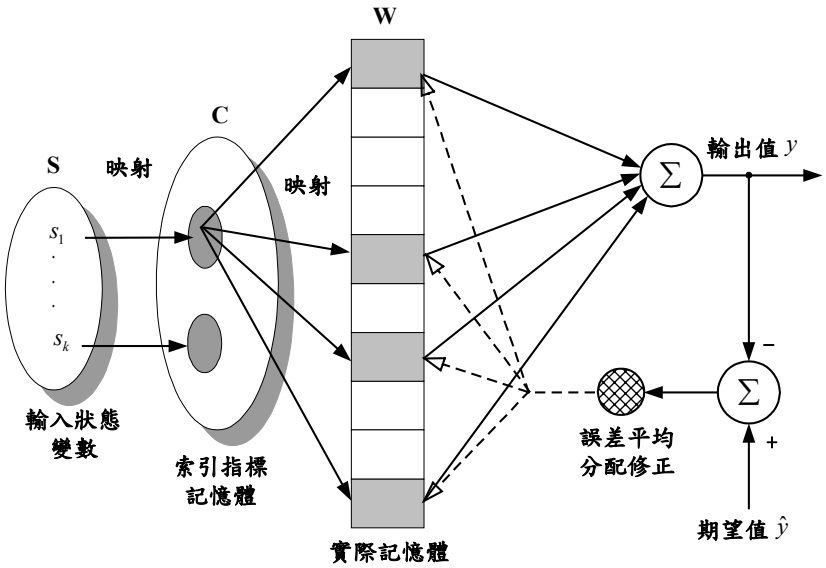

圖 2-1 小腦模型的基本架構

由圖 2-1 所示可知，CMAC 的架構是由四個部份所組成，其數學表示，可由 3 個映射來表示：

$$f:\quad \mathbf{S}\rightarrow\mathbf{C}$$

$$g:\ \mathbf{C}\rightarrow\mathbf{W}$$

$$h:\ \mathbf{W}\rightarrow\mathbf{Y}$$

$\mathbf{S}$＝{感知輸入向量群}

$\mathbf{C}$＝{索引指標向量群}

$\mathbf{W}$＝{記憶體向量群}

$\mathbf{Y}$＝{回應輸出向量群}

函數f、g通常是固定的，而函數h則會隨儲存(訓練)過程之係數值改變。當任何一個輸入向量$\mathbf{S}=(s_1,s_2,\mathrm{L}\ s_N)$出現在數個感知細胞時，此時$s_i$將會被映射到相對應的索引指標向量$\mathbf{C}$。將其中的係數(權重)加總起來，產生了輸出向量$\mathbf{Y}$。在此，只有非零的單元$C^*$會影響這個總和。所以輸入向量$\mathbf{S}$可視為一個位址，而回應輸出向量$\mathbf{Y}$，則可視為位址的內涵值。故對任何$s_i$，若我們想改變$\mathbf{Y}$的內容，那我們只要改變$C^*$所對應到的權重即可，在此$C^*$表示索引指標向量$\mathbf{C}$所致能或是不為零的單元。

因為 CMAC 的輸入樣式(Pattern)，會同時被許不同的輸入樣式所致能，因此對於某些輸入樣式而言，會有重疊、互相干擾的情形發生。這種情形，在訓練期間有時是有益的，但有時卻是致命的問題。例如：當有兩個輸入樣式s_1與s_2，他們分別致能兩組索引指標記憶體$C_1^* \cap C_2^*$，如果s_1與s_2的期望輸出相同，則當s_1的輸出已被適當的儲存時，那麼即使s_2完全沒有經過任何的修改，s_2亦可獲得十分接近的輸出。此特性類似生物將所學到的經驗類推至其他尚未學的事物上，所以這個特性便稱為類化能力。反之，若輸入樣式s_2與相鄰的輸入樣式s_1需產生一個差異很大的輸出結果，那麼對重疊的$C_1^* \cap C_2^*$而言，則將會有困難產生。因為我們期望獲得s_2所對應的輸出，但修改C_2^*所致能之記憶體的同時，我們亦修改了s_1所對應的大部份記憶體。這種情形與生物在學習時所遭遇的情況十分相似，所以我們稱之為學習干擾[1-3]。

關於 CMAC 學習干擾的問題，可藉由對s_1與s_2儲存的資料，進行迭代演算法來加以克服，藉由往復的迭代，最終將可在一小部份的記憶體中，獲得足夠大的係數，也就是說，只存在C_1^*或C_2^*中，但卻不存在$C_1^* \cap C_2^*$。如此相較於s_1，大差異的輸出s_2便可獲得。

CMAC 的類化能力來自於重疊或者可說是交集 $C_1^* \cap C_2^*$ 的特性。如果交集 $C_1^* \cap C_2^*$ 為零。那麼 CMAC 所相對應的輸出，將是各自獨立的，而且其類化的能力也將不發生。反之，如果 $C_1^* \cap C_2^*$ 不為零，則對 s_1 與 s_2 而言，就交集 $C_1^* \cap C_2^*$ 的部份其對應的輸出，將會相互影響。因此，對於輸入模式 s_1 與 s_2 將會產生相似的輸出。是故，CMAC 類化的程度亦將取決於 $C_1^* \cap C_2^*$ 的程度。CMAC 對 s_1 與 s_2 會產生不相似的輸出，乃取決於 $C_1^* \cap C_2^*$ 不相同所造成。所以，一般而言，$C_1^* \cap C_2^*$ 交集愈少，則愈容易產生不相似的輸出。

在本節中，我們可清楚的知道，CMAC 是由一連串的映射來運作，藉由重疊的記憶體來獲得類化能力，而類化能力與學習干擾是一體之兩面，最終還是可以藉由迭代方式獲得改善。在下節中我們將繼續討論有關 CMAC 的記憶體分割方式。

2.3 小腦模型控制器的記憶體分割方式

Albus 雖建議使用雜湊編碼方式，來改善記憶體不足之問題，雜湊編碼是計算機科學中常用的技巧，用於儲存稀疏矩陣的內容和一些散佈在記憶體中，卻相對量少的資料，用以減少記憶體的需求量。雜湊編碼的運作方式為藉由存取記憶體位址方式，將這些片斷的資料從記憶體中抽取。其原理為在一較小的記憶體中，算出資料的位置，再將此運算的結果做為參數，然後取得資料。例如在一個記憶體的任何一個位址，可應用一個隨機的虛擬亂數產生器，利用小的記憶體之位址，來代表函數產生器輸出的數個整體範圍。其結果為將一個大的記憶體，化成一個小的記憶體區塊，是多對少的映射。在此我們假定記憶體足夠大，而可免除雜湊編碼問題。在下節中，我們將分別探討一維及二維的例子，至於更高維的 CMAC 記憶體分割方式，可由二維的例子來加以推衍。

2.3.1 一維小腦模型控制器記憶體的分割方式

一維小腦模型控制器記憶體的配置方式，可由圖 2-2 來解釋，CMAC 首先將設定輸入變數(輸入空間)，設定成符合要求的空間，$\mathbf{S}=(s_1,s_2,\mathrm{L}\ s_N)$。然後在此空間中，將輸入變數**S**做均等分割，也就是設定解析度(resolution)大小，對於所有的s_i係定義在此空間中，$\forall_{s_i}\in\mathbf{S}$，解析度愈高，表示在區間$S$取樣愈多，如在圖 2-2 所示，輸入變數$S$被分割為 8 個不連續的狀態，將每一不連續的狀態映射給 2 個記憶體，此種映射為一線性之映射，可想像為，當狀態s_i致能時，則 CMAC 將分別對所欲學習的函數取樣，然後再將取樣的結果，分散儲存於不同的記憶體，如圖 2-2 所示垂直軸y對應到 CMAC 所欲學習之樣式。

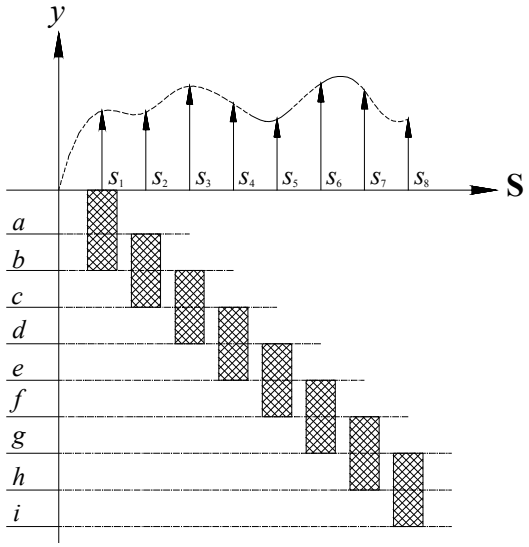

圖 2-2 一維小腦模型控制器記憶體分割圖

將圖 2-2 的每一狀態加以對應，整理可得表 2-1 的索引矩陣配置表。在本例中，參考狀態s_1其對應之記憶體為a與b兩個記憶體，所以a與b兩個記憶體，可對應在索引矩陣中。由表中清楚可見，狀態s_i與記憶體相互的關係，在表中“1”表示為致能(Enable)，而其餘未對映到的，則以“0”表示，

狀態 s_2 其對應之記憶體為 b 與 c 兩個記憶體，以此類推建立表 2-1。值得特別注意的是，在其中的記憶體 b，它分別儲存著參考狀態 s_1 與 s_2 的資訊，稱之為重疊(Overlap)。依據 Albus 的說法[1-3]，重疊的記憶體應愈多愈好，如此可將樣式分散儲存於不同的記憶體。例如，若重疊數為 100，則理想上，每一記憶體將各自分得 1%的值，如此即可擁有良好的類化能力。

表 2-1 記憶體與樣式對照表

記憶體 / 樣式	a	b	c	d	e	f	g	h	i
s_1	**1**	**1**	0	0	0	0	0	0	0
s_2	0	**1**	**1**	0	0	0	0	0	0
s_3	0	0	**1**	**1**	0	0	0	0	0
s_4	0	0	0	**1**	**1**	0	0	0	0
s_5	0	0	0	0	**1**	**1**	0	0	0
s_6	0	0	0	0	0	**1**	**1**	0	0
s_7	0	0	0	0	0	0	**1**	**1**	0
s_8	0	0	0	0	0	0	0	**1**	**1**

一維 CMAC 的輸出，可視為一線性的加總，h $\mathbf{W} \rightarrow \mathbf{Y}$，$h$ 的映射就是將對應到的記憶體致能，以供 CMAC 加總做輸出。透過監督式的學習，CMAC 將取得的誤差，平均分配誤差，送回給索引矩陣對應到的記憶體，週而復始的學習，誤差一直修正到預先所設定之閥值時，學習階段才算完成。

一維 CMAC 對於不同參考狀態樣式的切割非常簡單，所對應配置的記憶體大小，可用下列表示式來加以說明：

$$\text{Memory Size} = s_N + \rho - 1 \qquad (2\text{-}1)$$

式(2-1)中 s_N 表示樣式的個數，如圖 2-2 與表 2-1 所示，在本例中 $s_N = 8$，ρ 表示每個樣式所對應之記憶體個數，也可稱為類化指標，在本例中 $\rho = 2$，所以可知，對一維 CMAC 所需之記憶體數為 $8 + 2 - 1 = 9$。

2.3.2 二維小腦模型控制器記憶體的分割方式

函式 $f: \mathbf{S} \rightarrow \mathbf{C}$ 的映射，可被施用於任何維度 $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, \mathrm{L}\ \mathbf{S}_N)$ 的輸入向量。當經過函數映射後，不管輸入的維數多寡，$C_1^* \cap C_2^*$ 的數目，大約等比例於輸入空間向量 $\mathbf{S}_1$ 與 $\mathbf{S}_2$ 的接近程度。

本節以圖 2-3 所示之圖形，來說明二維 CMAC 之記憶體定址方式。例如考慮二維的輸入樣式 $s_i = (x_1, x_2)$。假設 $5 < x_1 < 6$ 與 $1 < x_2 < 6$，接下來我們將選擇 $\rho = 4$，將每個輸入變數軸均等量化(Quantizing)成 13 段不連續的小單元(Resolution)，所以可知我們將可得 $13 \times 13 = 169$ 個方格，此方格稱為不連續輸入狀態。

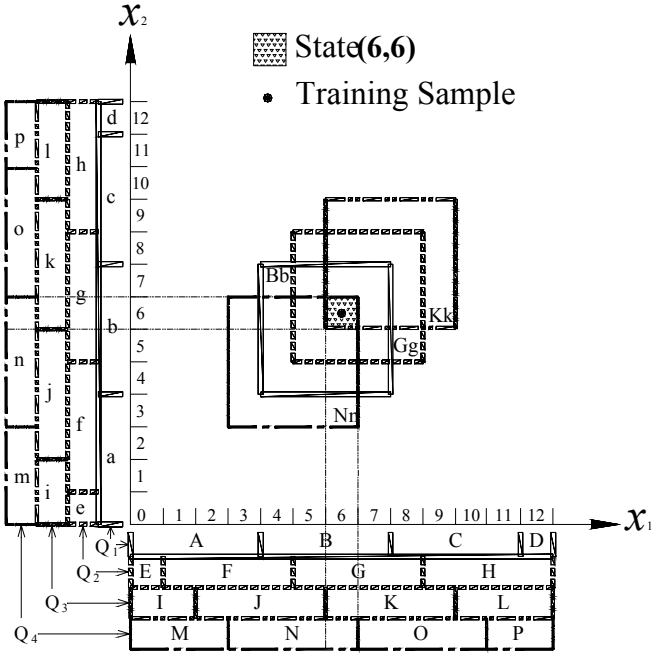

圖 2-3 二維 CMAC 學習空間量化示意圖

在本例中，參考圖 2-3 可知，在此選取的每一完整的區段，是由 4 個元素形所組成，在每次位移(Shift)一個單元後，經 4 次的位移，可得 Q1、Q2、Q3 和 Q4 等 4 層的量化層(Quantizing Layer)。相同的，對 x_2 軸的量化層產生方法亦和 x_1 軸的一樣。將這些交錯的方格依次排列，其將可完全的包圍此一空間。由圖 2-3 可得，狀態 s_i =(6 , 6)可映射到{ Bb, Gg, Kk , Nn }等四個平面，而這個平面，便稱為超立方體(Hypercube)。因此就本例而言，4 層× 16　(hypercube/層)= 64 個超立方塊，如表 2-2 所示。

表 2-2 關於圖 2-3 各量化層所形成之超立方體名稱

層名	超 立 方 體 名 稱															
Q_1	Aa	Ab	Ac	Ad	Ba	Bb	Bc	Bd	Ca	Cb	Cc	Cd	Da	Db	Dc	Dd
Q_2	Ee	Ef	Eg	Eh	Fe	Ff	Fg	Fh	Ge	Gf	Gg	Gh	He	Hf	Hg	Hh
Q_3	Ii	Ij	Ik	Il	Ji	Jj	Jk	Jl	Ki	Kj	Kk	Kl	Li	Lj	Lk	Ll
Q_4	Mm	Mn	Mo	Mp	Nm	Nn	No	Np	Om	On	Oo	Op	Pm	Pn	Po	Pp

對多維度的輸入空間來說，任何的輸入變數 $\mathbf{S}_i$ ，它的鄰域可能會依據 $\mathbf{S}_i \rightarrow \mathbf{C}$ 的映射，使其在座標軸上，會有伸長或縮短的情形產生。例如，若 $\mathbf{S}_{x_1} \rightarrow \mathbf{C}$ 為一高解析度的映射，而 $\mathbf{S}_{x_2} \rightarrow \mathbf{C}$ 為一低解析度的映射，那麼對於輸入空間 $\mathbf{S}$ 的鄰域而言，將會在 x_2 軸的方向延長，而在 x_1 軸的方向縮短。一個高解析度的映射 $\mathbf{S}_{x_1} \rightarrow \mathbf{C}$ ，使得 C^* 的組合，會強烈的依賴輸入變數 $\mathbf{S}_{x_1}$ 。(例如，在 $\mathbf{S}_{x_1}$ 只要輸入很少，就可以影響到 C^* 的一個或數個單元)。一個低解析度的映射 $\mathbf{S}_{x_2} \rightarrow \mathbf{C}$ ，使得 C^* 的組合不強烈的依賴 $\mathbf{S}_{x_2}$ ，(例如，大的改變 $\mathbf{S}_{x_2}$ ，才會影響到 C^* 的單元)。若將 $\mathbf{S}_{x_2} \rightarrow \mathbf{C}$ 的解析度調的很低，那麼 C^* 的組合，將會與 $\mathbf{S}_{x_2}$ 是各自獨立的關係，(不管 $\mathbf{S}_{x_2}$ 如何改變 C^* 都不會受影響)。所以當 C^* 與 $\mathbf{S}_{x_2}$ 是相互獨立時，我們可以說，延伸在 x_2 軸上的輸入空間，其鄰域為無限大。

將$\mathbf{S}_i \rightarrow \mathbf{C}$的映射，設定成不均勻的區間是可行的，(例如在第 i 軸上，將某區設定成高解析度，而其他則設定成低解析度)，且藉由此種方法，我們可將輸入空間區域，設定成不同的大小與形狀，這個特色在實際的應用上非常有用。

2.4 小腦模型控制器之數學表示法

由表 2-2 可知，總共有 4 層，而每層有 16 個超立方體，在此若對每一超立方體予以順序編號，則我們將可得 64 個獨立位置，例如圖 2-3 所示，狀態$s_i = (6,6)$在x_1軸，所對應到的區段為{B, G, K, N}；而x_2軸，其所相對應的區段為{c, g, k, n}，依據 Albus 的原則[1-3]，CMAC 只在有不同軸的同層區段才相結合。是故覆蓋狀態$s_i = (6,6)$之超立方體為{Bb , Gg , Kk , Nn}等 4 個，經由表 2-2 可得{ Bb, Gg , Kk , Nn }此 4 個超立方體，其分別位於第 1 層(Q1)至第 4 層(Q4)的第 6、27、43 及 54 位置，因此 CMAC 對狀態$s_i = (6,6)$之輸出，即可由這 4 個超立方體之內容值加總而得。最後，將此以數學的方式呈現如下：

$$\begin{aligned} y_{(6,6)} &= \begin{bmatrix} a_1 & a_2 & \Lambda & a_{64} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \mathrm{M} \\ w_{64} \end{bmatrix} \\ &= \sum_{j=1}^{Nh} a_j \cdot w_j \end{aligned} \tag{2-2}$$

在圖 2-3 中，狀態 $s_i=(6,6)$ 被 a_6、a_{27}、a_{43} 及 a_{54} 索引到，所以值為 1，其餘皆為 0，所以其至能的 4 個平面為{ Bb, Gg, Kk , Nn }。在此 a_j 表示超立方體選擇向量中的元素；w_j 代表真實記憶體陣列內的元素；N_h 則為超立方體的總數，亦即真實記憶體個數，故由表 2-2 可知，N_h=64。我們再將式(2-2)化為通式，可得：

$$y_s=\begin{bmatrix} a_{s,1} & a_{s,2} & \text{L} & a_{s,Nh} \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \\ \text{M} \\ w_{Nh} \end{bmatrix}=\sum_{j=1}^{Nh} a_{s,j}\cdot w_j=\mathbf{A}_s^T\mathbf{w} \qquad (2\text{-}3)$$

其中 y_s 表示，CMAC 對輸入樣本 s 所產生的輸出；$\mathbf{A}_s^T$ 表示真實記憶選擇向量，又稱聯想向量(association vector)；**w** 表示實體記憶體列向量。

2.5 小腦模型控制器之學習與回想演算法

我們已經知道，CMAC 的組成架構，以及 3 個 CMAC 的基本映射。觀察這些映射可發現，CMAC 在處理這些運算時，函數 f 、g 與 h 只是利用簡單的加法運算，函數 f 是將輸入樣本予以線性的分配給索引記憶體，函數 g 是將索引記憶體予以非線性的分配給實體記憶體，而函數 h 則是線性的將記憶體的內容相加輸出而已，這是 CMAC 的最大特點之一，其不需複雜的數學運算，由式(2-3)可求得 CMAC 的輸出值，在經由監督式學習所給予的期望值輸出值相互比較，可得學習之誤差值，經 CMAC 將誤差平均分配給其相對應之記憶體，如此便完成一個樣本的學習。當有 N 個樣本學習完成後，稱為一個週期(Epoch)，由於 CMAC 為前饋式類神經網路，其記憶體內所儲存之權值(Weights)，乃是透過迭代逐次學習來獲得，所以真實記憶體之內容，係數向量群 **w** ，會隨著迭代次數的增加而改變，最後將以最小平方差(LMS)的方式收斂[24]。

在此我們採用 delta 誤差修正法，來對記憶體內含值加以修正，其表示式為：

$$w_s = w_{s-1} + \Delta w_{s-1} \tag{2-4}$$

$$\Delta w_{s-1} = \frac{\alpha}{\rho} C_{s-1}\left(\hat{y}_{s-1} - y_{s-1}\right) \tag{2-5}$$

其中 w_s 表示在進行迭代演算時的現在狀態。式(2-4)其意義為—現在記憶體的向量值，為上一次迭代演算的向量值 w_{s-1}，再加上修正向量值與學習率(α)的乘積。學習率α，有時也稱為學習步距，主掌訓練程序腳步的大小。式(2-5)為誤差修改向量值，在傳統 CMAC 學習運算中，定義為在參考狀態 s 時，我們將期望值 $\hat{y}_s$ 與 CMAC 的輸出值 y_s 兩者之差，再乘以相對應的類化向量。

在真實生活的世界中，大多數的物理量是處於連續的狀態，例如：速度的變化、溫度的變化，以及 Albus 所觀測到的機械手控制函數等，為典型的物理量，其變化是相當平滑且連續的。這意謂著，對所有輸出而言，其相對應的輸入，也應該十分的接近。換個角度說，既然信號都十分相近，那我們也就不需要完全擷取所有的狀態，我們可只取適當、足夠的信號就好。CMAC 之學習與回想演算法，可視為一連串的映射而來，而映射的演算法為 CMAC 具有的特性，函數 f 將輸入向量彼此間的距離，轉換成重疊的程度，然後分散儲存，將函數值存於記憶體中。所以，CMAC 可說是一種記憶體管理技術，使得相似的輸入產生類化，因而產生相似的輸出；如果為不相似的輸入，則可產生成獨立的輸出。