基於個人化距離之音樂索引系統設計

葉弘淇
國立屏東科技大學資訊管理系
研究生
e-mail：
M10056019@mail.npust.edu.tw

高誌隆
國立屏東科技大學資訊管理系
研究生
e-mail：
M10156005@mail.npust.edu.tw

劉寧漢*
國立屏東科技大學資訊管理系
副教授
e-mail：
gregliu@mail.npust.edu.tw

摘要

在網際網路蓬勃發展下，取得音樂資料的管道越來越多元，因此使用者將擁有龐大的音樂資料，而為了讓使用者在查詢相似歌曲時，能快速找尋到自己喜歡的音樂，我們提出一個基於個人化的音樂索引系統，其主要讓使用者在聆聽歌曲後，給予歌曲之間的相似度評分，再依據這個評分計算出使用者的個人化距離公式，導入到我們的階層式自組織特徵映射演算法 (Hierarchical Self-Organizing Maps, HSOM)，產生具個人化的音樂索引，讓使用者再查詢相似歌曲時，各自擁有符合自己個人化索引，且經實驗證實，個人化索引能夠快速且準確來提供使用者做相似歌曲查詢。

關鍵詞：音樂索引；個人化距離公式；Hierarchical Self-Organizing Maps

Abstract

Due to the rapid development of the Internet, there are becoming increasingly diversified music data pipelines. So the users will have a vast music data.

It is an issue worth exploring about how to come up with some rules from the music data which allowing the users can quickly find their favorite music. However, recently, there are more diversified kinds of music and more similar styles of songs. Hence, the results may be less ideal if we use the traditional clustering algorithm for the grouping.

Therefore, we proposed a personalized based music indexing system. This system allows a user mainly after listening to songs and giving a similarity score between songs. In accordance with those scores to derive his/her own personal distance parameter. Follow this parameter can be inserted into our class Hierarchical Self-Organizing Maps clustering algorithm as a criterion for grouping. In the end, it is resulted in producing a personal music index. Consequently, when different users query similar songs again which can have his/her own well-matched characteristic personal index for each one.

Our experiments confirmed that the personal index generated by our system is able to quickly and accurately provide users do similar songs query.

Keywords: Music index; personal distance; Hierarchical Self-Organizing Maps

1. 前言

現今網際網路的普及化，造就了使用者在取得音樂資料的管道越來越多元，可透過許多的線上音樂平台來購買音樂，因此使用者將會擁有更多的音樂資料。

但是在龐大的音樂資料中，使用者若希望在短時間內查詢一些自己喜歡特定歌曲的相似歌曲，則需要將資料庫內的資料做分群處理，而分群主要是透過音樂資料的特徵值來做處理，傳統的特徵擷取方式多半採用人工方式來擷取，將歌曲中的特徵，如：發行年代；歌手姓名；歌手性別；曲風、類型…等，透過文字型態的方式擷取出來並儲存在資料庫中。但是採用人工的方式相對較耗時，而現今的音樂種類多元化，許多音樂類型曲風較相近，透過這類擷取出的特徵將資料分群，在結果上可能沒有辦法快速且有效適合使用者做相似歌曲查詢。

因大型資料探勘技術一直是近來熱門的研究主題，在過去一些相關音樂分群的研究中，許多的研究是基於音樂的類型、曲風等資訊來透過這些特徵將其資料做分群處理。但現今的音樂種類較多元化，透過這樣的方式可能會導致結果不盡理想。

為此，本論文將提出一種基於個人化距離

的音樂索引系統，首先，在音樂特徵值部份，我們選擇了語音辨識當中較常用的梅爾倒頻譜係數 (Mel-scale Frequency Cepstral Coefficients, MFCC)作為我們分群的音樂特徵值，而我們所採用的分群演算法為階層式自組織特徵映射圖 (Hierarchical Self-Organizing Maps , HSOM)，因 SOM 的特性是可將高維度的特徵向量映射在二維平面當中，且保持資料的拓樸特性，將相似的資料群聚在一起，而我們主要希望在查詢時能夠加快系統處理速度，所以我們將透過階層式的架構來加速且提升分群演算法整體的效能。所以，為了計算出個人化距離，我們透過使用者在聆聽歌曲後，給予歌曲之間的相似度評分，再依相似度的評分計算出使用者內心隱含的個人化距離公式，作為我們分群法的依據，經由系統處理後，產生符合使用者本身個人特質的索引，當使用者在作相似歌曲查詢時，可輸入一首範例歌曲做查詢，快速找到與自己偏好相似的歌曲索引。

2.相關研究

本研究主要透過不同使用者對於音樂的感官程度不同，透過個人化距離公式，做為分群演算法的依據，產生個人化的音樂索引，在此將針對相關研究進行探討，包含梅爾倒頻譜係數、相似歌曲之音樂推薦、音樂分群與 *k*-NN 分群演算法以及本研究核心方法 SOM 分群演算法及階層式自組織映射圖。

2.1 梅爾倒頻譜係數

用音樂資料在做資訊檢索時，會先將代表該首音樂的某些較具代表性的特徵值擷取出來，如:演唱者姓名；音樂年份、類型；歌曲曲風…等，將代表歌曲的資訊做特徵擷取處理，這些特徵值往往都透過人工的方式來擷取，在處理上非常耗時。

音樂中有許多具代表的特徵值可做為分群、分類的依據，而在語音或語者辨識中，最常用到的語音特徵就是「梅爾倒頻譜係數」以下稱為 MFCC ，此一參數主要考慮到人耳對不同頻率的感受程度，因此用不同領域的語音辨識都有良好的效果。MFCC 近來已經廣泛地應用在語音識別領域[1][2]，MFCC 在計算上先已快速傅立葉轉換 (Fast Fourier Transform,FFT)將時域訊號轉化為頻域，之後對頻譜能量用三角帶通濾波器進行旋積，最後對各個三角濾波器所輸出的向量進行離散餘弦轉換(Discrete Cosine Transform,DCT)。在[6]研究當中，特別將權重的概念加入了濾波器中(weighted filter bank analysis, WFBA)，按照 MFCC 的流程[3]，音訊在取音框時經由傅立葉轉換(FFT)後，將其頻譜用三角帶通濾波器進行旋積，接著把各個三角濾波器輸出的能量取對數並進行離散餘弦轉換；而在最後進行離散餘弦轉換(DCT)時，在取對數後乘上一個權重，權重的值隨著三角濾波器組中濾波器的編號而改變，因此，加入權重概念來的 MFCC 在結果上有著更佳的辨識效果。

2.2 相似歌曲之音樂推薦、音樂分群

現階段有許多的音樂推薦、分群系統，其主要都以歌曲的某些屬性特徵，作為推薦或分群的特徵值，再搭配演算法將其相似的歌曲提供給用者，而其中的判斷相似度接透過特徵向量之間的距離作為相似判定的依據，將距離較近的歌曲特徵判定為相似度極高，而較遠的則可視為極不相似。其中這些屬性往往都從音樂當中擷取出有規則性，可具體代表音樂的某一些特徵，如音高；音長；音差；或語音特徵的訊號等(如 MFCC)，從這些特徵當中，找出歌曲與歌曲之間特徵的距離，來計算距離遠近[9]。

而所謂的距離度量（Distance measure）是所有辨識系統上最基本的一塊，透過不同的度量方式，亦會影響到輸出的結果，而在空間中的點與線、面之間都存在著距離，在學術上較常用的就是傳統的歐幾里德距離公式，歐幾里德距離主要是計算空間中的兩個點，可表示為 A、B，其中 $A=(A_1,A_2,\cdots,A_n)$與 $B=(B_1,B_2,\cdots B_n)$，而再 A、B 中的 *n* 表示其特徵值的維度，所以為了計算 A、B 兩點之間的距離，透過公式(1-1)來計算：

$$distance\ (A,B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2} \qquad (2\text{-}1)$$

透過點與點之間距離的計算，可將其應用在許多資料分群、分類的計算上，在人工智慧的類神經網路的一些相關研究也常常使用到歐幾里德距離公式，如較常見的演算法 *k*-means，*k*-NN 等皆有歐幾里德距離公式的應用。

2.3 *k*-NN 分群演算法

k-NN 全名是 *k*-th nearest neighbor，意指「*k* 個最鄰近的鄰居」，而 *k* 所代表最接近群心周圍的鄰近節點數量。如:5-NN 意指找出離群心最接近的 5 個節點(特徵點)，如圖 2-1 所示，

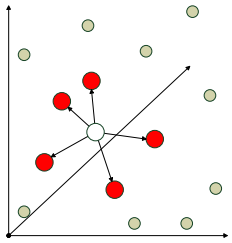

圖 2-1　*k*-NN 空間分布示意圖

在 *k*-NN 演算法中，初始會以某一個點作為種子，然後與空間中其他的點作距離計算，並出與種子距離最接近的 *k* 個點。因此 *k*-NN 具有容易推測結論，且能夠運用在各式各樣的資料型態上，甚至是沒有關聯性的資料，皆可以透過 *k*-NN 來做處理，但也因其空間複雜度與距離的計算量較大，特徵選擇上需要較嚴謹，若特徵不好，則分群後的結果也不盡理想，如分辨男女生的特徵下，若特徵只有頭髮長度與臉型輪廓，如遇到長頭髮的男生或輪廓較大的女生則可能會將其判斷錯誤，所以在這樣的特性下，*k*-NN 在特徵選取上勢必需要更嚴謹才可更準確的將資料分群。

2.4 SOM 分群演算法

在 1980 年由學者 Kohonen 所提出自組織特徵映射圖 (Self-Organizing Maps，SOM)[7]是一種非監督式演算法，也可稱其為競爭式學習法。當輸入一個數值時，在神經網路上的類神經元會彼此競爭，如圖 2-2 所示，以便獲得被活化的機會，而這個機會只賦予對輸入值最有反應的那個類神經元，即輸出值最大者，這個被稱作得優勝神經元，其與周圍的權值會被調整，以便更增加其與此時輸入之間的相似性。

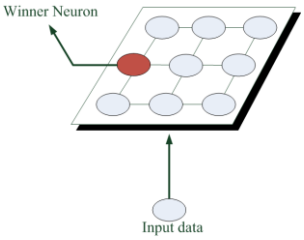

圖 2-2　SOM 示意圖

在 SOM 的概念中，輸入資料 X 中，都是以 n 個維度所構成的特徵向量即 $X = (x_1, x_2, \cdots x_n)$，而 n 代表著輸入特徵的各個屬性，輸出的結果皆以二維的方格呈現。

在方格當中的神經元也是由 n 個維度所組成的權重 W_i，即 $W_i = (W_{i1}, W_{i2}, \cdots, W_{in})$，而當中的 i 則代表每一個類神經元(neuron)，所以每一個資料都會與方格上的神經元作相似程度的運算，所以在維度數量上都必須式相同的。在 SOM 處理的過程上，主要是為了找出優勝的神經元，在調整優勝神經元與周圍鄰居的權值向量[10]。

首先，為了找到與輸入資料最近距離的優勝神經元，會先透過隨機的方式產生權重向量，並與輸入值計算其距離，而在 SOM 中所使用的距離公式為歐幾里德距離公式，將資料計算後，可以找到最接近輸入資料的神經元，稱其為優勝神經元(Winner neuron)。

在找出優勝神經元後，必須要調整他的權值，而且也會一併調整其周圍類神經元的權值，而調整權值時還必須要從學習率與訓練的次數來決定，這個學習率是為了將神經元可以隨著訓練次數與時間達到收斂的結果，在學習的過程中可以將神經元的權值趨於穩定狀態，之後若是有更多的輸入資料，在權值上也不會有過大的變動。所以在結束訓練的過程後，輸出的資料會以一個二維的方格圖呈現，與其他分群演算法的最大不同在於，SOM 存在著一個拓樸圖(Topological map)，此拓樸圖用來表達每個輸出值(output/cluster)的分布狀況。

所以，SOM 在輸出後可透過視覺化的方式呈現資料分布，用低維度空間來表達原本的高維度空間的資料，因存在著拓樸的特性，在視覺化後的結果亦能有效說明分群後各個資料間的相似性，若越接近的神經元或是群聚在一起的神經元，代表這些資料是極為相似的。

2.5 階層式自組織映射圖

階層式自組織映射圖 (Hierarchical Self-Organizing Maps , HSOM)，亦可稱為多層式(Multilayer)自組織映射圖，主要是基於 SOM 的演算法所延伸的一項分群技術，可以透過階層式的架構，讓每一層都各自擁有一個拓樸空間[4]，如圖 2-3 所示。階層式架構的概念，是藉由某一個 SOM 的輸出值，可以作為下一層的輸入值，而在多層的架構上可將其表示為一個樹狀的結構，藉由量化誤差(Quantization Error)來控制其訓練過程。

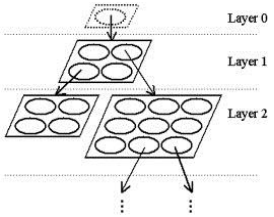

圖 2-3　階層式分群示意圖

在 SOM 中量化誤差就是輸入資料(input)與優勝神經元之間的差異程度，在計算上也是使用傳統的 SOM 中的歐幾里德距離公式來計算，在將量化誤差計算其平均值，表示其輸入資料與優勝神經元當中的變異程度，可將其表

示為：

$$mqe_i = \frac{1}{n_c} \times \sum_{x_i \in c_i} \|m_i - x_i\| \qquad (2\text{-}2)$$

n_c 代表輸入向量的樣本數量，而 C_i 代表第 i 個輸出模型的向量集合，而 m_i 則為模型的向量。在 SOM 訓練結束後，針對每個神經元去計算每一群的量化誤差，將映射在神經元 i 當中的輸入資料向量神經元 i 的權重向量作距離計算，在求其平均值，即可完成平均量化誤差的計算。

最上層的平均量化誤差可以將其表示為：

$$mqe_0 = \frac{1}{n_T} \times \sum_{x_i \in T} \|m_0 - x_i\| \qquad (2\text{-}3)$$

mqe_0 即為了計算所有輸入資料的變異程度，當中的 n_T 則是所有輸入資料的數量，因在階層的處理上，採用的平均量化誤差 mqe_0 作為往下一層分群的條件，所以為了控制其門檻，而採用了一個門檻的計算：

$$mqe_i < \tau * mqe_0 \qquad (2\text{-}4)$$

若平均量化誤差的 mqe_i 比 $\tau \times mqe_0$ 大，則可能是因為出現一些過於極端的值，或是資料過於集中在某一個神經元當中，所以必須要反覆的將資料重新做分群，映射在新的 SOM 上，直到滿足上列式子為止。

3. 系統設計

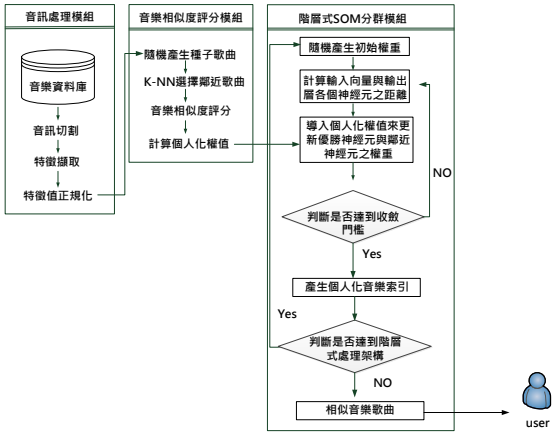

圖 3-1 系統架構圖

在系統設計部分，如圖 3-1，首先，我們採用音訊辨識中較常被使用的 MFCC 係數作為本研究的特徵擷取的依據，將音訊檔案切割後副歌的部分透過 MATLAB 工具將歌曲的特徵值擷取出來，並計算其平均值與標準差。接下來，引用過去所做的研究[I]，在系統初始時，將隨機從資料庫中選取數首歌曲作為種子歌曲，再透過 *k*-NN 找尋種子歌曲鄰近的數首歌曲，播放這些歌曲的副歌片段給使用者聆聽，並設計簡單的評分機制，讓使用者在評分過後，依據此評分計算出使用者的個人化權值，透過個人化權值做為分群的依據，在分群後產生出個人化之音樂索引，讓使用者在作查詢時，透過一首輸入歌曲，快速找到與其相似的歌曲。

3.1 音樂特徵擷取

在音樂分群前，必須先使用音樂中特定的特徵值來做為分群依據(如:歌手性別、歌曲的曲風、歌曲發行年代…等)，所以我們在系統前置處理時，必須先將每一首音樂先做特徵值擷取處理，而我們所選擇的音樂片段，主要擷取歌曲中人聲較明顯部分，因為多數歌曲當中副歌部分人聲較為明顯且重複多次，令人印象深刻所以採用副歌部分作為特徵擷取的依據。

在此我們所使用梅爾倒頻譜係數(MFCC)做為我們的音樂特徵值，我們先透過音樂切割工具，將音樂 Wave 檔案(.wav)中副歌的部分進行切割，再利用 MATLAB 工具將切割過的音樂進行特徵擷取處理，在每一首音樂 Wave 檔案中，採用的取樣率為 44.1kHz，取樣解析度為 16 bits，並以每 20 ms 進行音框的切割，而音框重複部分為音框長度的一半(1/2)。

我們透過 MATLAB 工具將每一首歌曲的音訊檔案擷取出其 MFCC 係數，將其作量化，而 MFCC 係數中主要包含 13 個係數(包含 12 組倒頻譜係數及 1 組對數係數)，因此我們再進一步處理歌曲中的 MFCC 特徵值，將每首歌曲的 MFCC 係數計算其平均數與標準差，並儲存於資料庫中，因此每一首歌曲共可得到一個由 26 維度構成的音樂特徵值。

而音樂特徵值擷取完畢後，為了讓特徵值不出現極端值，而影響到實驗結果，必須進行資料正規化處理的動作，如(3-1)所示：

$$N_{data} = \frac{data - D_{min}}{D_{max} - D_{min}} \qquad (3\text{-}1)$$

經由正規化後的 N_{data}，所有資料的特徵值皆將介於 0 到 1 之間。

3.2 音樂相似度評分

在音樂相似度評分模組中，我們透過不同使用者對於不同的音樂間觀感程度不同，將透過一個簡單的評分設計，讓使用者透過簡易的網頁問卷，在聆聽音樂後給予一組評分，我們在做特徵擷取時，每首歌曲皆透過人工方式剪輯其副歌片段，所以使用者必須聆聽這些歌曲之該片段後，給定一組用來判斷相似度距離分數，系統並將評分記錄，計算其個人化權值，

儲存於資料庫中。

在選擇聆聽的歌曲上，系統隨機選取 N 首歌曲做為種子音樂，並以 *k*-NN(*k* Nearest Neighbor Algorithm)演算法找出種子歌曲 *N* 鄰近的 *k* 首歌曲，所以我們進一步設計使用者評分時所需要聆聽的歌曲，我們將種子歌曲 N 表示為 $N_1,N_2,N_3,\cdots N_n$(共有 *n* 首)，且由 k-NN 所找出來的歌曲 *k* 可表示為 $k_1,k_2,k_3\cdots k_n$(共有 *n* 首)，所以使用者必須聆聽在 N_n 中的$(k_1,k_2,k_3\cdots k_n)$這些歌曲。

而考量計算上的合理性，我們將分數設定為 1 至 10 分。若使用者給予的分數越高分，則表示兩首音樂之越相似，反之則越不相似，而這個相似的程度我們可依據表 3-1，來將分數與距離作定義，若分數越高，則表示音樂在空間分布中的距離越相近，反之則越遙遠。

表 3-1　音樂相似度評分量表

分數	1	2	3	4	5	6	7	8	9	10
距離	10	9	8	7	6	5	4	3	2	1

因此，我們用 *k*-NN 演算法來找尋種子音樂 N 周圍最鄰近的 *k* 個點，來提供使用者作相似度評分，在空間分布當中，每一個點都是一首音樂資料，若是點與點之間越接近可視為這兩首音樂相似程度較高，所以我們透過這個簡易的評分，可以找出使用者內心隱含對於音樂距離的標準。

3.3 個人化權值計算

經上述的音樂評分完成後，我們依據使用者對於歌曲之間的相似度評分，來調整其空間中音樂之間的距離，使歌曲在分群時可獲得維度的加權。

我們在計算使用者權重的方法，採用最大概似估計法(Maximum Likelihood Estimation，MLE)。而傳統計算空間中的距離，使用的公式為歐幾里德距離公式，但傳統歐幾里德距離在計算距離時，將每個維度的重要性視為一致，但在音樂中每個維度對於音樂的相似度計算都有不同的影響，像是節奏在用來計算搖滾音樂相似歌曲時，就較抒情歌曲來得重要。因此我們將每個維度的權重納入計算，稱其為權重式歐幾里德距離 (Weighted Euclidean Distance)。為了降低計算複雜度，故使用權重式歐幾里德平方距離公式，如公式(3-2)所示：

$$Distance\ (X,Q)=\sum_{n=1}^{d} w_n\,(x_n-q_n)^2 \qquad (3\text{-}2)$$

首先，我們假設系統隨機挑選出的種子歌曲為 *Q*，且 *Q* 可視為 *n* 維度空間上的一點。接下來選擇周為鄰近的 *X* 首歌曲作為距離計算的對象，其中 x_n 及 q_n 分別為歌曲 *X* 與 *Q* 之第 *n* 維的特徵值，各維度權值的初始值皆為 1(因為每個維度都經由上述的正規化處理)，假定 *M* 為利用 *k*-NN 搜尋出與 *Q* 距離最接近的 *k* 首歌曲集合，$M=\{S_i \mid S_i \in k\text{-}NN\ of\ Q,\ i=1,2,\cdots,k\}$，使用者接下來針對由 *k*-NN 找出的歌曲集合 *Si* 與種子歌曲 *Q* 的相似程度進行評分，如果使用者認為 S_i 與 *Q* 較相似則給予較高分，反之則給予較低的分數。

假定 r_i 為其相似度評分，r_i 值越高表示 S_i 與 *Q* 之相關性越高。假定 M_p 為具分數之集合，$M_p=\{m_i \mid m_i \in M\ and\ r_i \neq \phi\}$，$m_i$ 之間假設為相互獨立的事件，則發生 M_p 事件集合的機率可表示為(3-3)，其中 $W=<W_1, W_2, \cdots, W_d>$，$r_i$ 為使用者針對歌曲 m_i 的評分，r_i 的值越大表示該分數對於機率之大小有較大程度之影響。

$$P(M_p \mid W,Q)=\prod_{m_i \in M_p} P(m_i \mid W,Q)^{r_i} \qquad (3\text{-}3)$$

接著調整 *W* 向量中各維度之權值，使得 $P(M_p|W,Q)$之機率值最大化：

$$W'=\arg\max_{W} P(M_p \mid W,Q) \qquad (3\text{-}4)$$

假設 $P(m_i|W,Q)$ 為高斯分佈 (Gaussian Distribution)可得：

$$P(m_i \mid W,Q)=\frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{\prod_{n=1}^{d}\frac{1}{w_n}}}\exp\left(-\frac{1}{2}D(m_i,Q)\right) \qquad (3\text{-}5)$$

求取對應於每個 w_n 的 $P(M_p|W,Q)$的偏微分 (Partial Derivatives)：

$$\nabla\{P(M_p \mid W,Q),W\}\equiv\left[\frac{\partial P(M_p \mid W,Q)}{\partial w_1},\frac{\partial P(M_p \mid W,Q)}{\partial w_2},\ldots,\frac{\partial P(M_p \mid W,Q)}{\partial w_d}\right] \qquad (3\text{-}6)$$

為使 *W* 能讓 $P(M_p|W,Q)$最大化，各維度偏微分後值應為零。另為求解，先針對 $P(M_p|W,Q)$取對數值(log)：

$$\ln(P(M_p \mid W,Q))=\sum_{m_i \in M} r_i\left[\frac{1}{2}\left(\sum_{n=1}^{d}\ln w_n - D(m_i,Q)-d\times\ln(2\pi)\right)\right] \qquad (3\text{-}7)$$

且

$$\frac{\partial \ln(P(M_p \mid W,Q))}{\partial w_n}=0 \qquad (3\text{-}8)$$

因此

$$\sum_{m_i \in M_p} r_i \frac{1-w_n(m_{i,n}-q_n)^2}{2w_n}=0 \qquad (3\text{-}9)$$

$m_{i,n}$ 及 q_n 分別為歌曲 m_i 及種子歌曲 *Q* 在維度 *n* 的特徵值。

最後，維度 *n* 的新權值 w'_n 為：

$$w_n^{'} = w_n = \frac{\sum_{m_i \in M_p} r_i}{\sum_{m_i \in M_p} r_i (m_{i,n} - q_n)^2} \quad (3\text{-}10)$$

由於一次的評分不一定有足夠資料能算出使用者對音樂距離之判斷，因此可重複上述評分步驟數次，藉以算出較具代表性的權值。假設共進行 T 次評分程序，且每次均為獨立事件，則維度 n 的最後權值為：

$$\overline{w}_n = \frac{\sum_{t=1}^{T} w_{n,t}^{'}}{T} \quad (3\text{-}11)$$

在完成上述計算之後，我們即可得到一個 26 維度的個人化權重特徵向量，可表示為 $(w_1, w_2, w_3 \cdots w_{26})$，此一向量即表示使用者對於音樂間距離的加權函數，我們將透過這個加權函數作為我們階層式 SOM 演算法的依據。

3.4 SOM 分群流程與階層式分群機制

在我們的階層式 SOM 演算法中，將其分為下面幾個步驟:

Step 1.初始化:

在初始化時，需要設定一些參數來讓 SOM 運作，首先，設定初始神經元的數量為 V (Set V=5x5)，學習率 α 為 0.9，α 的縮減率為 0.95，即經一次迭代後將縮減原本的 0.95，最小的學習率設為 0.1，而訓練次數 p 為 1000 次，以及設定調整半徑 R(R=0.95) 與半徑縮減率 r(r=0.9)，並隨機產生初始權重 w，初始權重透過亂數方式產生介於 0 到 1 之間的數值，設定好以上這些初始參數後即須進入下一步驟尋找優勝神經元。

Step 2.從訓練資料中尋出優勝神經元:

接下來，為了找出優勝神經元（Winner neuron），並且調整優勝神經元及其鄰居的權值，所以我們必須從所有的訓練資料當中計算其與各個神經元之間的距離，接著我們將找最小距離的優勝神經元，在此我們將透過權重式歐幾里德距離來用以計算，而我們將輸入資料表示為 S 計算距離即透過權重式歐幾里德距離公式，找出在訓練資料當中，滿足最小權重式歐幾里德距離優的勝神經元 j_x，如(3-12)所示：

$$j_x = argmin\|S - v\| \ , \quad v \epsilon V \quad (3\text{-}12)$$

j_x 表示訓練資料 S 對所有的訓練資料，透過權重式歐幾里德平方距離計算最短距離，而權重式歐幾里德平方距離中的 w_n 為上一節中透過相似度評分所計算出的個人化權值，藉此找出最短距離的神經元 j_x，我們稱其為優勝神經元。因為每筆資料所對應到的優勝神經元都不同，所以必須透過優勝神經元權值得加權調整使其達到與所有訓練資料處於穩定狀態，因此必須執行下一步驟。

Step 3.學習:

找出優勝神經元 j_x 後，需要調整優勝神經元周圍鄰居的鏈結權值，此過程稱為學習，而更新鄰居權重的公式如(3-13)所示：

$$w_{iu}(p+1) = w_{iu}(p) + \Delta w_{iu}(p) \quad (3\text{-}13)$$

u 表示在優勝神經元 jx 調整半徑 R 內的訓練資料，而 $\Delta w_{iu}(p)$的調整則依據下列公式：

$$\Delta w_{iu}(p) = \alpha \times [S_i - w_{iu}(p)] \times w_i \quad (3\text{-}14)$$

我們須經由訓練次數 p 來調整周圍鄰居的鏈結權值。

Step 4.迭代

最後透過迭代的方法，每次將更新學習率 α 及半徑 R 一次，如(3-15)所示：

$$\begin{cases} \alpha = \alpha * 0.95 \\ R = r \times R \ , r\text{為半徑縮減率 } R\text{為半徑} \end{cases} \quad (3\text{-}15)$$

迭代的終止條件為滿足訓練次數 p=1000，否則須重複執行步驟 2，直到滿足設定迭代次數 p，表示訓練資料以分群完畢產生特徵映射圖。在經由上述步驟過後，即產生一個由訓練資料所構成的二維拓樸圖，而分佈在圖上各個群內所包含的資料，即為訓練資料經 SOM 分群處理後，將特徵相似的歌曲群聚在一起的分佈情況，讓使用者在查詢歌曲時，選擇一首輸入歌曲資料，依據此拓樸圖來將輸入資料對應到特定的群當中，在將該群內的資料依據使用者查詢條件呈現給使用者。

3.5 階層式分群架構

首先，請使用者輸入一首歌曲來查詢與這首歌曲相似的歌曲，但輸入歌曲所對應到的群中，資料有可能未能夠滿足使用者查詢的條件，則我們將透過階層式分群的架構來處理歌曲查詢時數量過多的群，將這些群內的資料進行下一層的SOM分群處理。

但我們所提出的個人化索引系統中，主要讓使用者在作查詢時，系統需快速的找到符合使用者查詢的歌曲，所以為了避免多餘的處理運算，我們將階層的數量限制在三層，接下來我們將詳細敘述階層式的運作流程。

首先我們將使用者查詢歌曲數量設定為C，而輸入歌曲對應到所屬群內歌曲數量設定為 m_n，而n則代表透過階層式分群後第n層群內的歌曲總數，在下列的Case中將詳細敘述使用者查詢時所面臨的一些情況，以及階層式的處理模式：

Case 1 – 使用者所查詢到群內數量過多時：

第一種情況，若使用者在查詢相似歌曲時，需要先輸入一首歌曲，透過系統處理後來查詢與其相似的歌曲，若輸入歌曲所對應的群中的歌曲數量過多時表示：$C>m_1$(在Layer 0)，我們將透過階層式的架構，將該群內資料(m_1)在做一次SOM分群的處理，例如使用者查詢歌曲數量為5首歌曲時，在所對應的群當中共有10首歌曲，則我們必須再從這10首歌曲在做一次SOM分群，在第二層當中，找到符合使用者的查詢的歌曲數量，但在下一層當中還是歌曲數量大於查詢歌曲時表示：$C>m_2$(Layer 1)，則再將該群(m_2)進行一次SOM分群，因為我們所限制的層數為3，所以在最底層所屬的群內數量還是過多時表示：$C>m_3$(Layer 2)，則我們需要透過使用者所查詢的輸入歌曲C與該群內(m_3內所有的歌曲資料)的歌曲特徵進行k-NN的距離計算，找出與使用者查詢歌曲最鄰近的5首歌曲，呈現給使用者。

Case 2 – 使用者查詢到的數量過多，但在下一階層數量過少：

另一種情況是，在分群過後，若輸入歌曲對應到第一層所屬群內資料大於使用者所查詢的數量時表示：$C>m_1$(Layer 0)，則我們透過階層式架構，將該群內資料(m_1)再作一次SOM分群處理，但在下一層，若所對應到的群內資料過少時表示：$C<m_2$(Layer 1)，則需要計算所屬群與其他周圍鄰近群的距離，並找出最短距離的群，因為每個群內的數量不一定，因此不一定只有在隔壁的某一群，所以必須再計算與查詢數量所差距的歌曲數，所以我們先透過k-NN演算法找出最近距離的群，在將群內的歌曲資料與輸入歌曲計算空間中的距離，找出剩餘不足的歌曲數量，例如使用者輸入一首歌曲並查詢相似的15首歌曲，但分群之後所屬於的群當中只有10首歌曲，所以我們找到最鄰近的群，而最鄰近的群內共有6首歌曲，所以我們必須從這6首歌曲中，計算與輸入歌曲最接近的5首歌曲，並與所屬群內的10首歌曲一併呈現給使用者，若鄰近歌曲只有3首，則我們需再計算第二鄰近的群，在將其透過上述的方式計算剩餘兩首歌曲，以此類推，最後滿足使用者查詢條件即完成。

3.6 群組合併設計

每位使用者若都建置個人化索引，將使得系統存有過多的索引資料，為了避免系統儲存過多的使用者資料，我們透過使用者在作音樂相似評分後，若個人化權值距離較相近的使用者，可視為同一群，並共用一份索引。

在此我們將透過 k-means 分群演算法來將使用者的權重向量來分群，透過這個步驟來縮減系統的使用者索引量，將相似的使用者群聚在一起，共同使用一份索引，

k-means 的流程，我們將其分為下列步驟：

Step 1.隨機產生群心點：

我們隨機選取 k 個使用者(k 為縮減的量，若 k=10，即將索引量縮為共 10 組)，作為群組合併的初始群心。

Step 2.所有使用者計算與群心的最短距離：

我們隨機選擇 k 個使用者作為初始的群心中，計算所有使用者與各群心 k 之間的距離，計算距離的公式，我們透過歐幾里德距離來計算，將與 k 的距離較小使用者的歸屬在同一族群。

Step 3.調整群心點：

將所有使用者對群心點 k 的距離計算一次之後，將距離 k 較小的使用者分為同一群，在將計算後屬於同一群內的使用者重新計算新的群心點。

Step 4.群心點達收斂，完成 k-means 分群：

計算 k 群中點的距離，我們可找到一個錯誤函數 U，如(3-16)所示：

$$U = min \sum_{i=1}^{k} \sum_{x_j \in \mu_j} \|x_j - \mu_j\|^2 \quad (3\text{-}16)$$

當計算後距離較小的點群歸類為同一群，在分為 k 群的情況下，隨機找出某位使用者 μ_j 作為初始群心，與其他使用者 x_j 來計算距離，若當錯誤函數 U 不在改變時則完成 k-means 的分群流程，否則需要返回 Step 1 依序執行步驟達收斂為止。

透過 k-means 找出使用者權重與群心點的距離，判別使用者歸屬於哪一群，在判斷屬於該群後，我們將以最接近群心的權重值，作為該群的領袖權值(最接近群心的使用者權值)，將領袖權值所產生的個人化索引，讓屬於該群的使用者共用這份索引，縮減系統資料量。

4. 實驗分析

4.1 實驗結果與分析

實驗對象我們挑選日常有聆聽音樂習慣的使用者，共計 30 名，男女性比率為 3:1，接著讓使用者自己選擇一首歌曲，作為自己所要查詢的輸入歌曲，經系統處理後，產生出個人化的音樂索引，並從這各索引中依照使用者所查詢歌曲數量來提供給使用者聆聽，使用者必須

透過簡易的網頁問卷來作答。

使用者只須在網頁問卷上選擇「是」或「否」，選擇「是」則代表與輸入歌曲相似，則判定為正確，若選擇「否」則代表該歌曲不相似，則判定為錯誤，因此使用者必須依據自己查詢的歌曲數量來聆聽這些音樂歌曲之片段，來判斷這些歌曲中共有 X 首與查詢歌曲相符，故可將正確率表示如下：

近似歌曲搜尋正確率(%) = (X /查詢歌曲數量) × 100% ，X：使用者認為「是」的曲目數量

而索引查詢回饋時間，可以將其表示如下：

索引查詢回饋時間 = 使用者查詢相似歌曲所花費的時間，單位為秒。

4.2 評分相關參數分析

首先，使用者進行音樂相似評分階段時，系統將隨機從音樂資料庫中選取數首歌曲作為種子歌曲，我們將種子歌曲表示為 N，再透過 *k*-NN 演算法找出種子歌曲 N 周圍最近距離的 *k* 首歌曲，讓使用者聆聽這些歌曲的片段之後，給予相似度評分，再透過最大概似估計法計算個人化權值。

表 4-1 評分次數對於正確率分析表

	N=1	N=2	N=3	N=4	N=5
***k*=1**	53%	56%	60%	65%	66%
***k*=2**	56%	59%	63%	67%	69%
***k*=3**	58%	65%	66%	69%	69%
***k*=4**	63%	68%	69%	69%	71%
***k*=5**	66%	70%	69%	70%	73%
***k*=6**	68%	72%	70%	71%	73%

※N 表示為種子歌曲，*k* 則表示鄰近歌曲數量，單位為%。

我們透過上表分析後，先從音樂資料庫中，隨機挑選 4 首歌曲作為種子歌曲，並以 *k*-NN 演算法找出最接近種子歌曲周圍的 6 首歌曲，分別讓使用者聆聽這些歌曲的片段過後進行相似度評分作為我們相似評分的基準。

4.3 階層式架構於正確率與查詢時間分析

為確認我們的系統在導入階層式的架構下，更能有效的加速查詢時間且提升其索引的正確率，我們將比較使用者在查詢相似音樂時，分別以我們所設計的階層式 SOM 與傳統的 SOM 作比較，將查詢出來的結果分析其差異性，在此我們將比較近似歌曲搜尋正確率與索引查詢的回饋時間。

表 4-2 階層式個人化 SOM 分群分析

	1 首	2 首	3 首	4 首	5 首	6 首	7 首	8 首	9 首	10 首
正確率 (%)	70%	68%	68%	70%	69%	69%	68%	70%	69%	70%
時間 (秒)	1.73	1.72	1.71	1.73	1.71	1.73	1.73	1.7	1.74	1.74

表 4-3 傳統 SOM 分群分析

	1 首	2 首	3 首	4 首	5 首	6 首	7 首	8 首	9 首	10 首
正確率 (%)	70%	68%	67%	69%	69%	69%	68%	70%	68%	70%
時間 (秒)	1.85	1.8	1.77	1.77	1.77	1.78	1.78	1.7	1.86	1.92

透過上列分析圖表可發現，本系統的個人化階層式架構在正確率表現上較傳統的 SOM 分群好上許多，因加入個人化距離公式能夠使得分群的結果滿足使用者做相似歌曲查詢，並能有效率的找出符合的歌曲，在索引時間的表現也較傳統的 SOM 來的好，所以我們所設計的階層式機制在處理更龐大的音樂數據資料上，必能較傳統的來的更好。

4.4 個人化距離於正確率與索引時間分析

為了確定我們的系統在查詢時正確率是否能滿足使用者查詢，我們將與 *k*-NN 演算法做正確率的比較，基於公平比較條件下，在此所比較的對象 *k*-NN，一樣也透過個人化距離公式，在此我們將 *k*-NN 視為我們最佳解，透過與 *k*-NN 相差的值來作正確率統計分析，

表 4-4 個人化 SOM 與 *k*-NN 正確差異分析

	1 首	2 首	3 首	4 首	5 首	6 首	7 首	8 首	9 首	10 首
相差比	15%	17%	16%	15%	16%	16%	17%	15%	16%	16%

透過上表的分析，我們的個人化 SOM 與 *k*-NN 比較的正確率表現上，在查詢歌曲數量越多時，正確率差異性較小，但差異約 15% 左右，所以我們在正確率表現上與 *k*-NN 不相上下，因此我們將進一步與 *k*-means 分群法來分析其正確率與查詢歌曲回饋的時間。

在這個階段我們也是將個人化權重導入 *k*-means 來與我們的系統作歌曲索引比較。因我們所用的階層式 SOM 可能會讓使用者在作查詢時，所查詢到的群因為階層架構關係，導致分群數量不固定，為了公平比較，我們系統初始化時，將分群群數設定為 25 群，在此我們也將權重式 *k*-means 資料分為 25 群來讓使用者查詢。

表 4-5 個人化 SOM 與 *k*-means 索引正確率分析

查詢歌曲數 / 分群演算法	1 首	2 首	3 首	4 首	5 首	6 首	7 首	8 首	9 首	10 首
個人化 SOM	70%	68%	68%	70%	69%	69%	68%	70%	69%	70%
權重式 *k*-means	66%	67%	65%	66%	66%	65%	66%	64%	64%	65%

透過上表得知，個人化 SOM 在索引正確率的表現上，較 *k*-means 的正確率來的優秀，而我們在探討查詢時所花費的時間，

表 4-6 個人化 SOM 與 *k*-means 索引查詢回饋時間分析

查詢歌曲數 / 分群演算法	1 首	2 首	3 首	4 首	5 首	6 首	7 首	8 首	9 首	10 首
個人化 SOM	1.73	1.72	1.71	1.73	1.71	1.73	1.73	1.7	1.74	1.74
權重式 *k*-means	1.98	1.96	1.96	1.94	1.9	1.94	1.96	1.95	1.94	1.94

由表 4-6 可得知，個人化 SOM 在歌曲查詢數量越多時，本系統在索引正確率的表現上，相較 *k*-means 的正確率普遍有著較佳的結果，接著我們在探討索引時所花費的時間。由表 10 可得知，本系統在歌曲查詢時所花費的時間，較權重式 *k*-means 要快。

4.5 合併索引差異性分析

經由本系統的分群處理過後，系統將存有使用者的索引資料，若在未來有更大量的資料儲存在資料庫中，可能會造成儲存空間上過多的負擔，因此我們將分析若縮減系統的索引量，是否會影響使用者在索引查詢時的正確率，首先，我們先將系統內所有使用者所建置的索引量設為 C，將索引量做縮減處理，即群組合併機制。

因此我們將索引空間依序縮減為原來的 1/6 來分析其正確率，而在此計算正確率是讓使用者輸入一首歌曲，查詢其相似的歌曲，而我們請使用者在合併後查詢歌曲數量分別設定為 1~10 首，探討縮減後之正確率，如表 4-7：

表 4-7 縮減索引量之正確率分析表

索引縮減量 / 查詢歌曲數	*C*	*C*/2	*C*/3	*C*/4	*C*/5
1 首	70%	66%	54%	46%	30%
2 首	68%	64%	52%	41%	27%
3 首	68%	64%	53%	43%	28%
4 首	70%	65%	54%	45%	33%
5 首	69%	66%	52%	45%	35%
6 首	69%	65%	52%	41%	33%
7 首	68%	63%	52%	40%	33%
8 首	70%	65%	54%	47%	37%
9 首	69%	64%	51%	42%	33%
10 首	70%	66%	53%	45%	30%

我們也將索引查詢回饋的時間作分析，也是依據上述條件依序分析縮減各個索引量對於查詢時間的影響，如表 4-8：

表 4-8 縮減索引量之索引查詢時間分析表

索引縮減量 / 查詢歌曲數	*C*	*C*/2	*C*/3	*C*/4	*C*/5
1 首	1.73	1.73	1.72	1.72	1.72
2 首	1.72	1.72	1.72	1.72	1.72
3 首	1.71	1.71	1.71	1.71	1.71
4 首	1.73	1.73	1.73	1.73	1.73
5 首	1.71	1.71	1.71	1.71	1.71
6 首	1.73	1.72	1.72	1.72	1.72
7 首	1.73	1.73	1.73	1.73	1.73
8 首	1.7	1.7	1.7	1.7	1.7
9 首	1.74	1.74	1.74	1.74	1.73
10 首	1.74	1.74	1.73	1.74	1.74

我們可發現縮減索引量對於正確率的影響，在 C/2 時，與原索引量 C 相比，正確率落差約為 3%，差異性相較於 C/3 時較平緩，且沒有較大幅度的落差，所以在基於縮減少空間的考量上，將索引量縮減為原來的 1/2，讓使用者來做個人化索引資料的建置。

5.結論與未來展望

我們透過實驗分析可確定，加入個人化距離公式導入分群法後，使用者作相似歌曲查詢時，索引的正確率較傳統來的高，可得出加入個人化的因素時，確實可有效找到使用者所期望呈現的歌曲，雖正確率不及於 *k*-NN 分群法來的優秀，但在歌曲查詢時系統所回饋的時間快上許多。

未來在處理更龐大的資料時，必能更快速找到較適合的結果，而也可透過更多的參數調整，查詢是否有更適合的索引方法，而未來在索引空間的縮減條件，也可以在資料量更多時，找到更適合的相關空間縮減條件，更有效的將節省系統儲存空間。

參考文獻

I. 吳育澤,(2012),"以個人化距離公式為基礎之音樂推薦系統─使用基因規劃法"。

[1] B. Logan, (2000), "Mel Frequency Cepstral Coefficients for Music Modeling", In International Symposium on Music Information Retrieval.

[2] B. Logan, A. Salomon, (2006), "A music similarity function based on signal analysis", In International Conference on Multimedia and Expo (ICME).

[3] B.Yegnanarayana, K.S.R. Murty, (2006), "Combining evidence from residual phase and MFCC features for speaker recognition", IEEE Transactions on Signal Processing Letters, 13(1), (pp. 52-55).

[4] D. Neto, J.A.F. Costa, M.L.A.Netto, (2001) "Hierarchical SOM Applied to Image Compression", In International Joint Conference on Neural Networks (IJCNN),

(pp. 442-447).

[5] F. Zhouyu, L. Guojun, M.T.Kai, Z. Dengsheng, (2011), "A Survey of Audio-Based Music Classification and Annotation", IEEE Transactions on Multimedia, 13(2), (pp. 303-319).

[6] J. Sungyun, S. Jongmok, B. Keunsung, “Telephone speech recognition with Data-Driven selective temporal filtering based on principal component analysis”, In International Conference on Electronics, Informations and Communications (ICEIC) , (pp. 764-767).

[7] J. Vesanto, E. Alhoniemi, (2000), ”Clustering of the Self-Organizing Map”, IEEE Transactions on Neural Networks, 11(3), (pp. 586-600).

[8] J. Wenxin, W.R.Zbigniew, (2013), “Multi-label automatic indexing of music by cascade classifiers”, Journal of Web Intelligence and Agent Systems, 11(2), (pp. 149-170).

[9] J.Schluter, T. U. Munchen, Munich , Germany, C.Osendorfer, (2011), “Music similarity estimation with the mean-covariance restricted boltzmann machine” , In International Conference on Machine Learning and Applications and Workshops (ICMLA), (pp. 118-123).

[10] K. B. Dickerson ; D. Ventura , (2009) , “Music recommendation and query-by-content using self-organizing maps”, In International Joint Conference on Neural Networks (IJCNN), (pp. 705-710).

[11] N.T.Thanh, R. Wehrens, M.C.L Buydens, (2006), ”KNN-kernel density-based clustering for high-dimensional multivariate data”, Journal of Computational Statistics and Data Analysis, 51(2), (pp. 513-525).