Music Information Retrieval

HW2 Report

102062209 邱政凱

這次的作業要我們實作的是 Key-Finding 的演算法。給定一段旋律或曲子，猜測他的 Key。而這次作業的 Data set 是使用 GTZAN DataSet，Data set 的 labels 則是使用 Alexander Lerch 的 label。

Q1

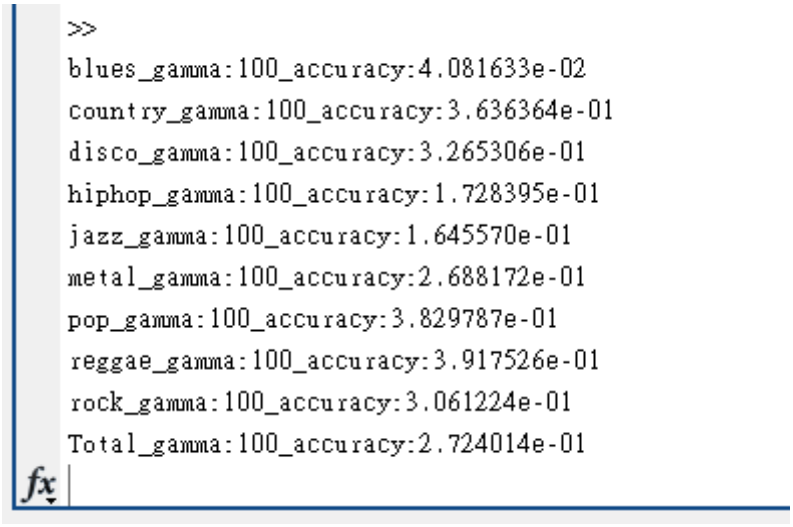
Q1~Q3 的部分是要我們實作第一種最基礎的演算法，也就是 Binary Template matching。首先先使用 Chroma toolbox 依照不同的曲風抽出各自的 Pitch，存起來之後再抽出每首曲子的 chroma gram，以供後續演算法使用。第一種 Binary template matching 的核心在於，把每首歌的 Chroma feature 做 sum pooling，先算出該首曲子哪個音出現最多次，便直接先猜該首歌的 key 就是該音(可能為小調或大調)。

接著，要決定究竟是小調還是大調，便是讓該首歌的 Chroma feature 去跟預先定義好的 binary template 去做 matching。這個 template 其實就是該音的 Major scale 跟 Minor Scale，是一個包含十二個非零即一元素的 vector。而對應不同 key 的 template 也就是以該音為起點(tonic)將對應位置的元素設成 1，在 diatonic scale 之中每隔全音就隔一個零的元素再放入下一個一，隔一個半音就直接在下個元素放入一，而 diatonic scale，大調為”全全半全全全半”，小調為”全半全全半全全”。而 Matching 的方法就是把 chroma feature 的 vector 跟對應的 template 一起算出 correlation coefficient。

實作基本上演算法老師都已經給得很清楚了，大方向上沒什麼問題，不過實務上有一些值得注意的小細節，像是 Alexander Lerch 的 label 是以 A 為 0 遞增上去的，但是 Chroma toolbox 算出來的 chroma feature 的第一個元素是 C。所以為了可以直接對應到 Lerch 的 tag，我把 chroma feature 全部都 circular shift 了三個元素來方便運算(另外，因為這邊 shift 了三個元素，所以 template 的元素也要跟著 shift)。

另外就是有些歌曲 Lerch 並沒有標出答案，這種時候在跑迴圈的時候就必須記得要把它跳掉。(另外，因為 classical 裡面的歌曲幾乎全部都是未標示的狀態-1，所以我在跑 Data set 的時候沒有用 classical 的歌曲)。

而第一題跑出來的結果如下：



Binary template matching for key finding

可以看到，單純使用最簡單的 binary template matching，而且相似的 key 都不計分的狀況下在 GTZAN Dataset 的 key finding 結果綜合準確率約是 27%，其實真的稱不上是個好看的準確率。(不過相較於隨機猜測的期望準確率 4%(1/24)已經可以看到這個演算法還是有發揮到功用的)。

如果依照音樂的分類來看的話，流行歌(Pop)和雷鬼音樂(Reggae)和鄉村音樂(Country)…等等的準確率都偏高，這也許可以歸功於這些音樂的曲調都比較單純明確，主旋律和和弦通常也會一直重複，在編曲上也經常依循著(西方本位的)樂理來編曲，因此使用這種(依循西方本位樂理開發出來的)演算法容易得到較高的準確率。

而偏低的藍調(Blues)、嘻哈(Hip Hop)、爵士(Jazz)等等曲風則通常會呈現比較隨興、即興等自由發揮的路線(另外值得一提的是這三種準確率最低的曲風都是源自於黑人音樂)，這些曲風的發源本來就不是依著縝密的樂理發展出來的，因此用這種基於西方本位的樂理發展出來的演算法去猜它的 Key 自然會偏低。(另外還有有趣的一點像是搖滾樂是結合鄉村音樂和爵士音樂融合演變而成的，Accuracy 剛好也是基於這兩者之間 XD)。

Q2

第二題則是要我們在跑一次第一題，但是要試驗在算 Chroma Feature 時就把頻譜上的能量用不同的參數去做 Log-Compression 的話效果會有何不同。

```
blues_gamma:1_accuracy:1.020408e-01
country_gamma:1_accuracy:3.434343e-01
disco_gamma:1_accuracy:2.551020e-01
hiphop_gamma:1_accuracy:1.234568e-01
jazz_gamma:1_accuracy:1.265823e-01
metal_gamma:1_accuracy:3.548387e-01
pop_gamma:1_accuracy:3.723404e-01
reggae_gamma:1_accuracy:3.608247e-01
rock_gamma:1_accuracy:3.265306e-01
Total_gamma:1_accuracy:2.676225e-01
```

Factor of compression = 1

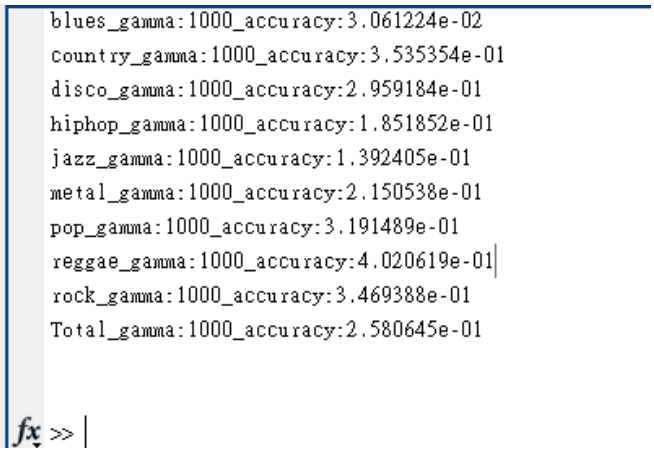
```
blues_gamma:10_accuracy:6.122449e-02
country_gamma:10_accuracy:3.636364e-01
disco_gamma:10_accuracy:3.265306e-01
hiphop_gamma:10_accuracy:1.481481e-01
jazz_gamma:10_accuracy:1.645570e-01
metal_gamma:10_accuracy:3.225806e-01
pop_gamma:10_accuracy:3.510638e-01
reggae_gamma:10_accuracy:4.020619e-01
rock_gamma:10_accuracy:3.163265e-01
Total_gamma:10_accuracy:2.771804e-01

>>
```

Factor of compression = 10

```
>>
blues_gamma:100_accuracy:4.081633e-02
country_gamma:100_accuracy:3.636364e-01
disco_gamma:100_accuracy:3.265306e-01
hiphop_gamma:100_accuracy:1.728395e-01
jazz_gamma:100_accuracy:1.645570e-01
metal_gamma:100_accuracy:2.688172e-01
pop_gamma:100_accuracy:3.829787e-01
reggae_gamma:100_accuracy:3.917526e-01
rock_gamma:100_accuracy:3.061224e-01
Total_gamma:100_accuracy:2.724014e-01
```

Factor of compression = 100



Factor of compression = 1000

Logarithmic Compression 是為了讓頻譜的能量分布呈現比較接近人類聽覺系統的感覺和能量之間的關係(log-scale)而誕生的。但是直接取 Log 在許多狀況下是不太實際的作法，因此使用 log(1+ γ |x|)的方式取代之。

從上面幾個比較可以看出，對總體準確率來說，1、10、100、1000 的參數其實都沒有太大的差別，不過還是可以看出會有比較好的選擇(10)，而且也不是說越大或是越小準確率就越高，因此要選取適當的 Log-compression 的 factor 也許是需要花時間去 tune 的。

不過，從上面還可以看出，Gamma = 1 的時候 blues 的準確率是四種最高的，儘管他的總體準確率沒有其他高。而在總體準確率最高的 Gamma = 10 中，Pop 類的準確度卻沒有 Gamma = 1、100 時還要高，因此我們可以合理認為改變 Factor of compression 對不同曲風的音樂來說都會有不一樣的效果。

Q3

這題讓我們依照 MIREX 競賽對於 Key Finding 的準確率的標準來做為計算 Accuracy 的標準。對於一首歌的 Key 來說，這個 Key 會容易被認為是該 Key 的 Perfect fifth、該 Key 的 Relative Major/Minor 或者是該 Key 的 Parallel Major/Minor。因此，在這三種情況下我們也依照不同權重給予不同的分數。 (Perfect fifth 0.5 分、Relative Major/Minor 0.3 分、Parallel Major/Minor 0.2 分)。

我的實作細節大致如下：

```
250 -       if key <= 11 && key>=0,
251 -           if key==correctAns(n+1)+12,
252 -              genreMatched = genreMatched + 0.2;
253 -              totalMatched = totalMatched + 0.2;
254 -           elseif key==mod(correctAns(n+1)+7,12),
255 -               genreMatched = genreMatched + 0.5;
256 -               totalMatched = totalMatched + 0.5;
257 -           elseif key==(mod(correctAns(n+1)+9,12)+12),
258 -              genreMatched = genreMatched + 0.3;
259 -              totalMatched = totalMatched + 0.3;
260 -           end
261 -        elseif key<=23 && key>=12,
262 -           if key == correctAns(n+1)-12,
263 -              genreMatched = genreMatched + 0.2;
264 -              totalMatched = totalMatched + 0.2;
265 -           elseif key == mod(correctAns(n+1)-9,12),
266 -              genreMatched = genreMatched + 0.3;
267 -              totalMatched = totalMatched + 0.3;
268 -           elseif key== mod(correctAns(n+1)-5,12) + 12,
269 -               genreMatched = genreMatched + 0.5;
270 -              totalMatched = totalMatched + 0.5;
```

依照我找出來的關係，對某一個 Key 來說，他跟其他三種容易被混淆的 Key 在 Lerch 的標示法上會有如上的關係。

```
blues_gamma:100_accuracy:1.683673e-01
country_gamma:100_accuracy:5.505051e-01
disco_gamma:100_accuracy:4.132653e-01
hiphop_gamma:100_accuracy:2.222222e-01
jazz_gamma:100_accuracy:2.721519e-01
metal_gamma:100_accuracy:3.279570e-01
pop_gamma:100_accuracy:5.159574e-01
reggae_gamma:100_accuracy:5.051546e-01
rock_gamma:100_accuracy:4.387755e-01
Total_gamma:100_accuracy:3.847073e-01

fx >>
```

以上則是我用這種新的 Accuracy 計算方式所重新算出來的第一題的結果。可以看到，總體準確率從 27%上升到 38%(這其實是理所當然的，畢竟這種算法只有額外的分數而沒有漸少的分數)。

各個音樂 genre 的準確度也都有了提升，原本就已經很高的幾種 genres 從 3x%的準確率一口氣上升到 5x%，原本最低的 blues 也一口氣暴增了四倍的準確率來到了 16%，可見其實在原本的預測中有很多 Key 都被找成這三種容易混淆的 Key 了。

我們可以合理推論經過這種新的 Accuracy 計算方式，獲得越大幅度的準確

率提升的曲風，在做 key finding 時越是容易被搞混成其他相似的 key。

總結來說，Binary Template matching 雖然還算是結果可看的演算法，但是在細節上還是有很多瑕疵，尤其是當我們就算用 circular shift 改變整個 template，準確率會在某些 shift 的值的狀況下有些有趣的變化。

```
blues_gamma:100_accuracy:2.500000e-01
country_gamma:100_accuracy:5.404040e-01
disco_gamma:100_accuracy:4.438776e-01
hiphop_gamma:100_accuracy:2.222222e-01
jazz_gamma:100_accuracy:3.101266e-01
metal_gamma:100_accuracy:3.602151e-01
pop_gamma:100_accuracy:5.159574e-01
reggae_gamma:100_accuracy:4.742268e-01
rock_gamma:100_accuracy:4.183673e-01
Total_gamma:100_accuracy:3.978495e-01
```

上面這個是我把整個 template shift 7 個元素之後的結果(例如：[1 0 1 0 1 1 0 1 0 1 0 1]變成[1 0 1 0 1 0 1 1 0 1 0 1])，也就是說，我們用該 Key 的 Perfect fifth 的 template 去做 Matching 的結果幾乎跟我們直接用該 Kay 的 template 去做 Matching 的結果差不多(甚至還更勝一點….)。

```
blues_gamma:100_accuracy:1.479592e-01
country_gamma:100_accuracy:5.505051e-01
disco_gamma:100_accuracy:3.724490e-01
hiphop_gamma:100_accuracy:2.222222e-01
jazz_gamma:100_accuracy:2.594937e-01
metal_gamma:100_accuracy:2.956989e-01
pop_gamma:100_accuracy:4.840426e-01
reggae_gamma:100_accuracy:4.639175e-01
rock_gamma:100_accuracy:4.183673e-01
Total_gamma:100_accuracy:3.620072e-01
```

上面則是把 template Shift 5 個元素的結果。也就是說該 Key 的 Perfect fourth 當作 template 去算出來的準確率也可以逼近原 Key。

從這些 shift template 比較我們可以發現，Binary template matching 其實是個有點模稜兩可的演算法，會把相似的東西混在一起，彼此之間的區別幾乎是非常的不明確。也因此，下一題我們將試驗另外一種有較高區別度的演算法 – Krumhansl Schmukler 演算法。

Q4

這題要我們把 Q1~Q3 都再做過一次，不過是要使用 Krumhansl Schmukler 演算法。Krumhansl Schmukler 演算法不同於 Binary template matching 不同的是，他並不需要先從 Chroma feature vector 中找出 tonic，而是直接把該曲子的 Chroma feature vector 跟所有可能的 key 的 template 去做 Matching(也就是取 Correlation Coefficient)。

然後直接把預測的 Key 設為相關係數最大的 key。使用的 template 也不再是 Binary，而是依照各種容易搞混的 key 給遇不同的權重。

以下稍微附上我的實作細節。

```
453 -    for i=1:24,
454 -         up = 0;down = 0;downX = 0;downY = 0;
455 -             if i<=12&&i>=0,
456 -                 t = tMajorAvg;
457 -             else
458 -                 t = tMinorAvg;
459 -             end
460 -        for j=1:12,
461 -             up = up + (sumVec(j) - xAvg) * (KSTemp(i,j) - (t));
462 -             downX = downX +  (sumVec(j) - xAvg)^2;
463 -             downY = downY +  (KSTemp(i,j) - (t))^2;
464 -        end
465 -         down = (downX * downY)^(0.5);
466 -         R = up/down;
467 -         if R>RMax,
468 -            RMax = R;
469 -            key = i-1;
470 -         end
471 -    end
```

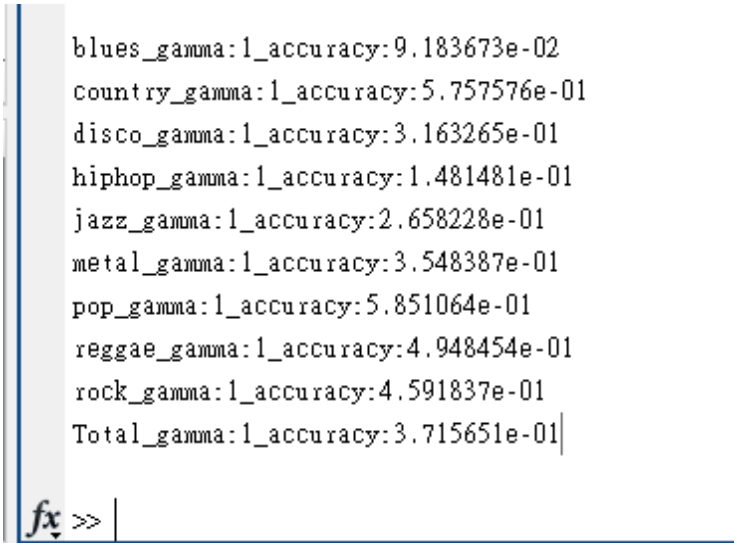
```
blues_gamma:100_accuracy:5.102041e-02
country_gamma:100_accuracy:5.353535e-01
disco_gamma:100_accuracy:3.367347e-01
hiphop_gamma:100_accuracy:1.111111e-01
jazz_gamma:100_accuracy:2.025316e-01
metal_gamma:100_accuracy:2.795699e-01
pop_gamma:100_accuracy:5.744681e-01
reggae_gamma:100_accuracy:4.845361e-01
rock_gamma:100_accuracy:4.795918e-01
Total_gamma:100_accuracy:3.464755e-01

fx >>
```

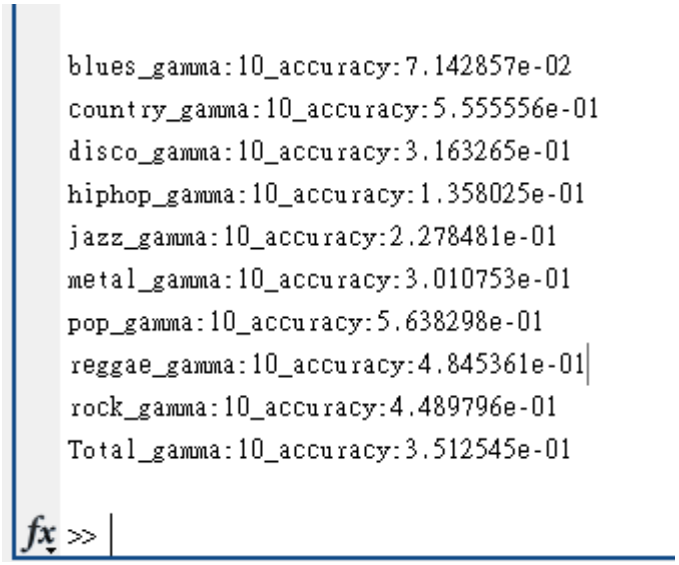
重作第一題，可以發現整體的 Accuracy 有了明顯的增加(27% - > 34 %)，而在各個曲風分向上的準確率，準確的比率大致上跟之前 Binary template matching 的結果差不多，比較容易有高準確率和低準確率的曲風大致吻合，有些上升明顯，有些則微幅上升。

不過比較讓我注意到的是，嘻哈(Hip Hop)類的準確率居然降低了，這也說

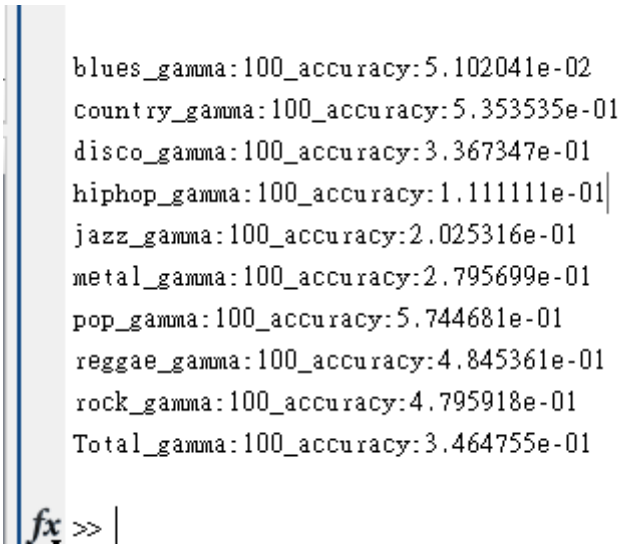
明了 Krumhansl Schmukler 並不一定全然是準確率的改進，對某些曲風來說可能反而會造成反效果。



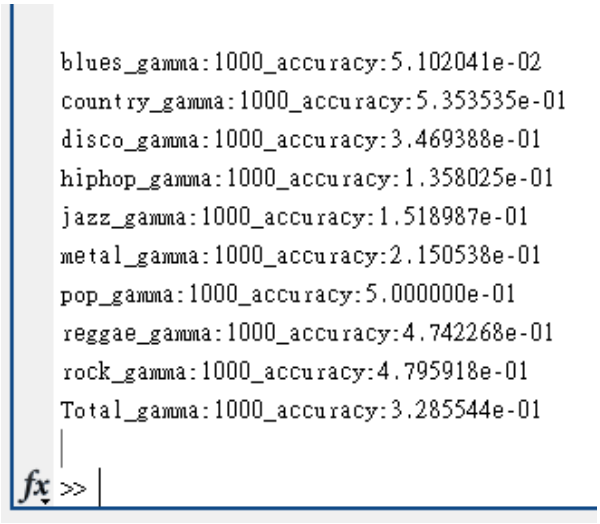
Factor of compression = 1



Factor of compression = 10

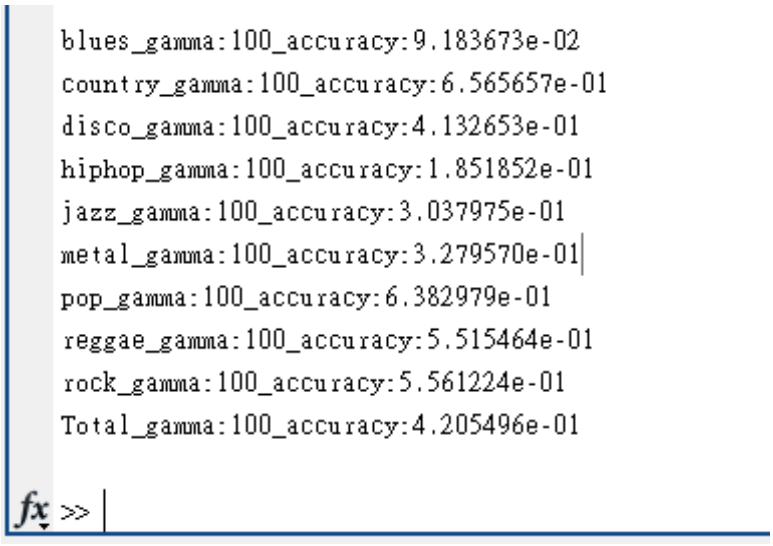


Factor of compression = 100



Factor of Compression = 1000

Factor of compression 的影響大致上跟用 Binary template matching 差不多。只不過我們可以發現在這種演算法下，Gamma 值越小最後的總和準確率是越大的，隨著 gamma 值的上升準確率是逐漸下降的(當然這很可能是因為我們只用了四組參數)。而不同曲風在不同的 Gamma 下也都會有不同的趨勢，這點跟在 Binary template matching 中的結果是一致的。



重作第三題的結果，可以發現總和準確率已經衝破四成了，來到這次作業的新高點。不過儘管如此，四成多的準確率卻離可以商業、工業化到實際應用中有非常非常一大段的距離，因此，Key-Finding 這個領域在 MIR 之中也許還是個有非常多地方仍待改進的處女地。

Extra:

這兩種演算法其實都只是非常單純的對於 Chroma Feature 的總和去做比對而已，這個總和的值裡面包含的資訊也許還是太少了。整個曲子的結構完全沒有考慮進去，只單純計算每個音出現的次數，這其實是有點 Greedy 而不夠全面的演算法。

我認為還是得用到 Machine Learning 的技巧用 training model 等方式從龐大的資料庫中學習出 Key detection 的 Pattern 才會是比較好的做法。

關於 GTZAN，我想到可能的問題則有 (1)它已經太廣泛地被使用了，太多研究者都用 GTZAN 來當作 DATASET，這可能會造成整個領域的研究都會因為太常使用同樣的 Dataset 而讓一些微妙的錯誤累積。(2)它的音樂種類儘管很多但也許不夠全面，主要都還是以美國的音樂為主，而如果是利用 GTZAN 做出來的研究成果可能會變成比較針對美國的音樂。