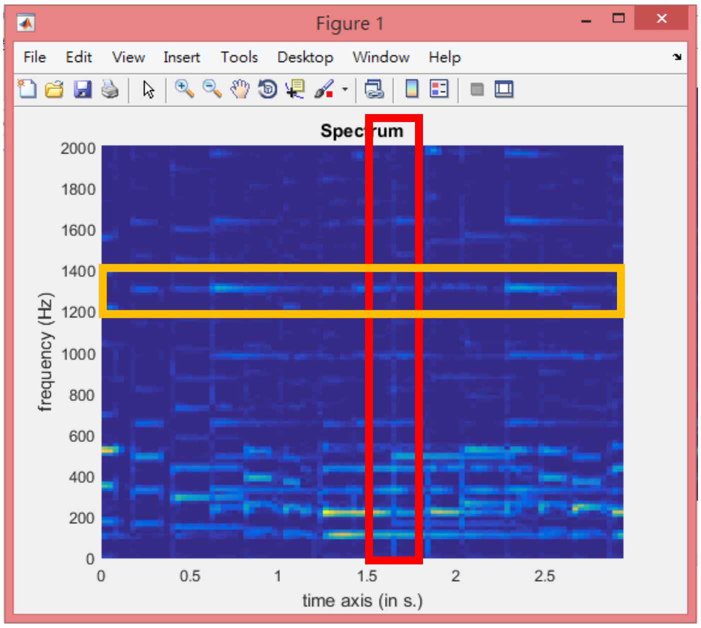
Music Information Retrieval HW1 Report

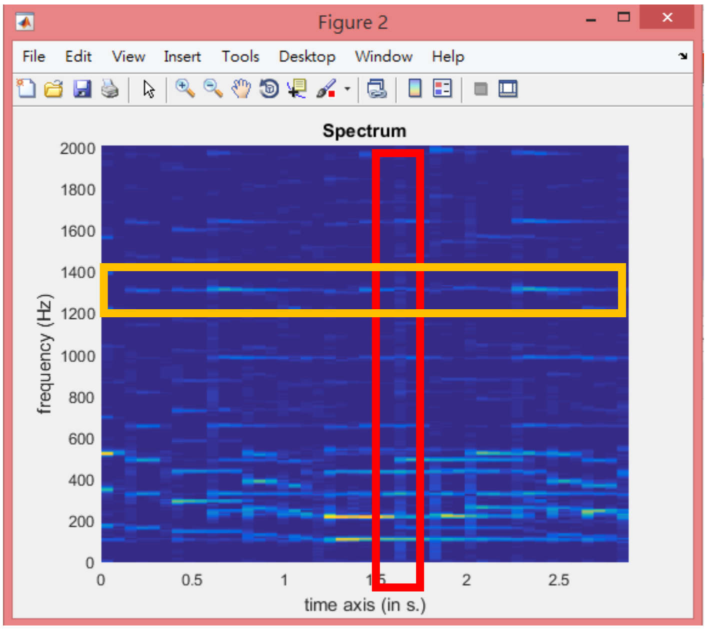
102062209 邱政凱

Q1、

第一題的部分，主要是讓我們比較不同的 Window Size 跟 Hop Size 對最後用 Short Time Fourier Transform 做出來的 Spectrum 之間的差異。這邊比較要注意的是要使用的單位是 Samples 而非 sec，所以用 mirframe 函式切割音檔的時候要特別用'sp'的參數註明。



Window Size = 1024, Hop = 512 (samples)



Window Siz3 = 2048, Hop = 1024 (samples)

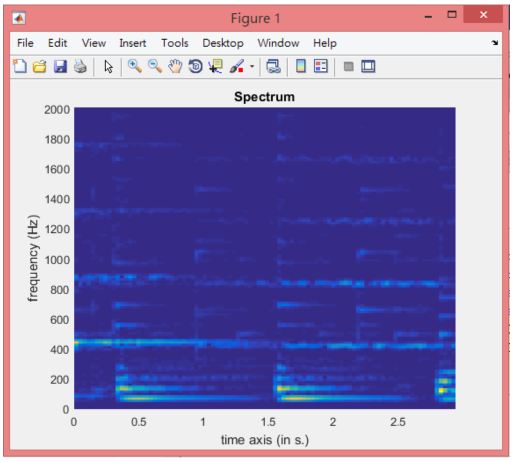
為了更集中頻率的顯示範圍以便觀察，我取 2000 頻率當作 spectrum 的上限，從上面的比較可以看出，Window 跟 Hop 較小的(上圖)，在時間上的解析度上會比較清楚(紅框處)，也就是 spectrum 比較不會在橫軸上模糊掉。

而在 Window 跟 Hop 較大的(下圖)中，在頻率上的解析度會比較高(黃框處)，也就是在縱軸上比較不會模糊掉。這也就是課堂上提到過的，STFT 經常必須面對到的 Heisenberg Uncertainty 問題，頻域與時域的解析度無法兩全其美。

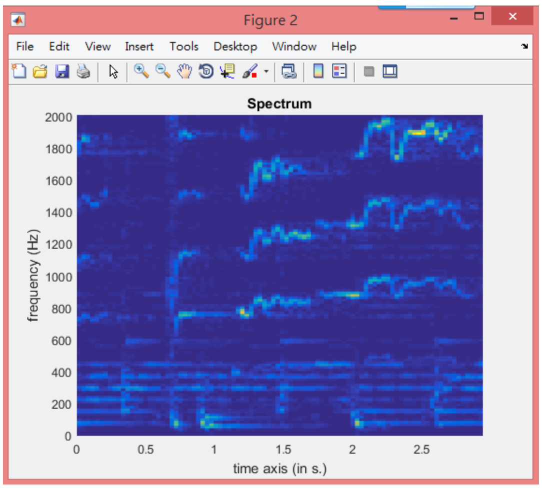
然後，也可以從頻率的分布狀況看出，因為人類對 pitch 的定義是 log scale 的，所以如果縱軸是 linear scale 而非 log scale 的狀況下，很明顯的大部分的 pitch 都會落在 spectrum 的較下方(下面的線性頻率包含到的 pitch 數遠高於上面的頻率)。

Q2、

這題要我們比較不同樂器的 spectrum 並討論。



Violin(3)



Voice(6)

　　可以看出，小提琴的頻譜相當的乾淨俐落，拉出來的音色在頻率上幾乎呈現直線。而人聲則很明顯的在頻率上不停地抖動，顯示人類比較沒有辦法精準控制準確的音程，不過換句話說這也是人聲如果作為一種 instrument 的角度來看它的獨特之處(下面的直線頻率應該是背景配樂)。

要想分辨出這兩種音色的不同，我一開始覺得 irregularity 會是個不錯的 Feature，因為人聲在時間在頻率上的震動幅度很大，而小提琴的音色頻率相對穩定，因此人聲應該會得出較大的 irregularity 值，不過隨機跑過幾對小提琴&人聲組合的 irregularity 平均之後，其實發現人聲的 irregularity 不會系統性地比小提琴來的大；

於是，為了更系統性地觀測兩種音色在哪種 Feature 上的差異較大，我把小提琴跟人聲各自 200 首的數個特徵(Centroid、Spread、Skewness、Kurtosis、Entropy、Flatness、Brightness、Roughness…)取出來後取平均以得到更完整個觀察。

2x52 double

		19	20	21	22	23	24	25	26	27	28	29	30	31
1	2456	0.2673	0.2832	0.3066	0.3222	0.3134	0.2911	0.2661	0.2606	1.5761e+03	168.0005	0.2467	0.0613	1.5894e
2	2731	0.2493	0.2486	0.2282	0.2226	0.2140	0.2220	0.2158	0.2224	1.8160e+03	260.1341	0.3618	0.0963	1.7096e
3														
4														
5														
6														
7														
8														

比對後發現 Roughness 和 Roll off 應該會在實際上比較可以分辨這兩種音色的 Feature。

Q3、

這題要求我們跑過一遍 Feature Extraction、data split、Feature Normalization、以及 training classifier 的過程。

根據老師給的 code，跑出來的 Accuracy 大約會落在 75%~80%(重複執行多次 data split，用不同的 validation set 平均出來的結果)。為了更清楚觀察到整個分類的詳細結果，我把 Confusion Table、Recall、Precision 和 F Score 都用程式算了出來。

```
Accuracy = 78.125% (125/160) (classification)
        guitar  violin  piano  voice
guitar    35      5       3      3
violin    5       23      5      3
piano     1       0       27     5
voice     2       1       2      40
Recall(guitar) = 0.760870 || Recall(violin) = 0.638889 || Recall(piano) = 0.818182 || Recall(voice) = 0.888889
Precision(guitar) = 0.813953 || Precision(violin) = 0.793103 || Precision(piano) = 0.729730 || Precision(voice) = 0.784314
FScore(guitar) = 0.786517 || FScore(violin) = 0.707692 || FScore(piano) = 0.771429 || FScore(voice) = 0.833333
fx >>
```

```
Accuracy = 76.875% (123/160) (classification)
        guitar  violin  piano  voice
guitar    30       7       1       1
violin    6       32       1       2
piano     1        6      26       1
voice     0        6       5      35
Recall(guitar) = 0.769231 || Recall(violin) = 0.780488 || Recall(piano) = 0.764706 || Recall(voice) = 0.760870
Precision(guitar) = 0.810811 || Precision(violin) = 0.627451 || Precision(piano) = 0.787879 || Precision(voice) = 0.897436
FScore(guitar) = 0.789474 || FScore(violin) = 0.695652 || FScore(piano) = 0.776119 || FScore(voice) = 0.823529
fx >>
```

```
Accuracy = 79.375% (127/160) (classification)
        guitar  violin  piano  voice
guitar    22       2       2       2
violin    4       34       1       1
piano     3        1      34       1
voice     4        3       9      37
Recall(guitar) = 0.785714 || Recall(violin) = 0.850000 || Recall(piano) = 0.871795 || Recall(voice) = 0.698113
Precision(guitar) = 0.666667 || Precision(violin) = 0.850000 || Precision(piano) = 0.739130 || Precision(voice) = 0.902439
FScore(guitar) = 0.721311 || FScore(violin) = 0.850000 || FScore(piano) = 0.800000 || FScore(voice) = 0.787234
fx >>
```

以上三個是用不同的 random permutation 把所有 data 做隨機分堆，做出不同的 training 和 validation set 之後 train 出來的 classifier 的測試結果。從這邊可以看出不同的 training set 最後做出來的 classifier 並沒有一個一致的趨勢，也就是說不會系統性的對某種音色有較好的辨別度。有可能是因為 training set 用隨機排序取到比較多筆資料的音色拿去 train classifier 之後算出來的 classifier 對該音色就會有更高的分辨度。

另外也可以發現，其實不管怎麼抽出 Validation set，每次跑出來的 Accuracy 都沒有太大的差別。由此可見在訓練樣本數多的狀況下其實，在相同的特徵跟算法下，train 出來的 classifier 都有一定程度的 robustness(在參數抓的好的狀況下)。

最後，也可以看到 Recall 跟 Precision 之間的差異，有些 classifier 對某些音色有較低的 Recall，Precision 卻相對偏高，反之亦然。也許可以解釋成如果這個 classifier 對於某個 class 的猜測比較保守嚴格，那麼的確可以預想會有較高的 Precision，但是也因為猜測保守，所以很多屬於該 class 的 input 就沒辦法正確歸類，導致偏低的 Recall。

老師的 Feature 只用了最常用在音色辨識使用的 MFCC 特徵。這是種把 FFT 做完後的頻譜取 Log scale 之後再用 DCT 轉去倒譜(Cepstrum)去分析的技術。藉由丟棄 DCT 轉換後高 Rank 的部分(通常取到前 13 個 Rank)，可以有效抓取到發聲系統的特徵(老師的 MFCC 取了前 20 個 Rank 做平均)。

不過這邊只取了 MFCC 這個特徵，卻已經可以對樂器的分辨程度到達 3/4 以上的精確度，不難認同 MFCC 真的是音色辨識領域最為出色的特徵之一。

Q4、

這題要我們測試 Feature Normalization 的功效。

```
Accuracy = 76.875% (123/160) (classification)
        guitar  violin  piano  voice
guitar    19      5      9      2
violin    4       31     2      1
piano     1       2      42     2
voice     2       4      3      31
Recall(guitar) = 0.542857 || Recall(violin) = 0.815789 || Recall(piano) = 0.893617 || Recall(voice) = 0.775000
Precision(guitar) = 0.730769 || Precision(violin) = 0.738095 || Precision(piano) = 0.750000 || Precision(voice) = 0.861111
FScore(guitar) = 0.622951 || FScore(violin) = 0.775000 || FScore(piano) = 0.815534 || FScore(voice) = 0.815789
fx >>
```

Before Normalization

```
Accuracy = 79.375% (127/160) (classification)
        guitar  violin  piano  voice
guitar    21      5      6      3
violin    5       30     2      1
piano     1       4      40     2
voice     0       1      3      36
Recall(guitar) = 0.600000 || Recall(violin) = 0.789474 || Recall(piano) = 0.851064 || Recall(voice) = 0.900000
Precision(guitar) = 0.777778 || Precision(violin) = 0.750000 || Precision(piano) = 0.784314 || Precision(voice) = 0.857143
FScore(guitar) = 0.677419 || FScore(violin) = 0.769231 || FScore(piano) = 0.816327 || FScore(voice) = 0.878049
fx >>
```

After Normalization

可以看到 Normalization 前後 Accuracy 確實有些許的差異，然而卻不是非常大的差異。我認為應該是因為這邊只用了 MFCC 當特徵的關係。MFCC 除了 Rank1 的係數外其他回傳的值大部分都介於-1~1 之間，其實 scale 差距不大，因此 Normalization 的效果並沒有非常顯著。

然而，如果我們來比較加入其他 Feature 之後 Normalization 的影響，就會發現其實 Normalization 是非常重要的。

```
Accuracy = 22.5% (36/160) (classification)
        guitar  violin  piano  voice
guitar    1       40     0      0
violin    0       35     0      0
piano     0       40     0      0
voice     0       44     0      0
Recall(guitar) = 0.024390 || Recall(violin) = 1.000000 || Recall(piano) = 0.000000 || Recall(voice) = 0.000000
Precision(guitar) = 1.000000 || Precision(violin) = 0.220126 || Precision(piano) = NaN || Precision(voice) = NaN
FScore(guitar) = 0.047619 || FScore(violin) = 0.360825 || FScore(piano) = NaN || FScore(voice) = NaN
fx >>
```

52 Features without Normalization

```
Accuracy = 88.125% (141/160) (classification)
        guitar  violin  piano  voice
guitar    34      1      4      2
violin    3       30     1      1
piano     0       2      36     2
voice     0       3      0      41
Recall(guitar) = 0.829268 || Recall(violin) = 0.857143 || Recall(piano) = 0.900000 || Recall(voice) = 0.931818
Precision(guitar) = 0.918919 || Precision(violin) = 0.833333 || Precision(piano) = 0.878049 || Precision(voice) = 0.891304
FScore(guitar) = 0.871795 || FScore(violin) = 0.845070 || FScore(piano) = 0.888889 || FScore(voice) = 0.911111
fx >>
```

52 Features with Normalization

我把 extract_MFCC 函式中抽出的 Rank 從 20 改成 13，並且加入額外的 13 個 Feature，然後把各自的平均和標準差拿來當作特徵。可以發現，不同特徵之

間的 scale 很不一樣，有些是-1~1 之間的 Range，有些卻可以上到幾百甚至幾千，沒有做 Normalization 的後果可想而知一定是會讓用這些根本就是不同 scale 的 feature train 出來的 classifier 毫無標準可言。

Q5、

這一題是叫我們測試 LIBSVM 中的兩個參數對於 Train 出來的 model 的影響為何。分別是 C，也就是對於放寬在特徵空間中分隔每個 class 的 hyper plane 的重疊條件ξ_i(誤差項)所做的懲罰權重(相對於我們要 minimize 的 objective function 而言)，以及 g，也就是用來把資料映射到高維空間中以方便我們分割的 kernel function 的參數。

```
Accuracy = 78.125% (125/160) (classification)
        guitar  violin  piano  voice
guitar    30      3      6      3
violin    6       32     1      3
piano     5       4      30     2
voice     0       2      0      33
Recall(guitar) = 0.714286 || Recall(violin) = 0.761905 || Recall(piano) = 0.731707 || Recall(voice) = 0.942857
Precision(guitar) = 0.731707 || Precision(violin) = 0.780488 || Precision(piano) = 0.810811 || Precision(voice) = 0.804878
FScore(guitar) = 0.722892 || FScore(violin) = 0.771084 || FScore(piano) = 0.769231 || FScore(voice) = 0.868421
fx >>
```

C= 1 、g= 1

```
Accuracy = 76.875% (123/160) (classification)
        guitar  violin  piano  voice
guitar    30      2      5      5
violin    5       33     2      2
piano     5       3      31     2
voice     2       2      2      29
Recall(guitar) = 0.714286 || Recall(violin) = 0.785714 || Recall(piano) = 0.756098 || Recall(voice) = 0.828571
Precision(guitar) = 0.714286 || Precision(violin) = 0.825000 || Precision(piano) = 0.775000 || Precision(voice) = 0.763158
FScore(guitar) = 0.714286 || FScore(violin) = 0.804878 || FScore(piano) = 0.765432 || FScore(voice) = 0.794521
fx >>
```

C=100 、 g= 1

```
Accuracy = 76.875% (123/160) (classification)
        guitar  violin  piano  voice
guitar    30      2      5      5
violin    5       33     2      2
piano     5       3      31     2
voice     2       2      2      29
Recall(guitar) = 0.714286 || Recall(violin) = 0.785714 || Recall(piano) = 0.756098 || Recall(voice) = 0.828571
Precision(guitar) = 0.714286 || Precision(violin) = 0.825000 || Precision(piano) = 0.775000 || Precision(voice) = 0.763158
FScore(guitar) = 0.714286 || FScore(violin) = 0.804878 || FScore(piano) = 0.765432 || FScore(voice) = 0.794521
fx >>
```

C=10000、g=1

```
Accuracy = 78.125% (125/160) (classification)
        guitar  violin  piano  voice
guitar    30       3       6       3
violin    6       32       1       3
piano     5       4       30       2
voice     0       2       0       33
Recall(guitar) = 0.714286 || Recall(violin) = 0.761905 || Recall(piano) = 0.731707 || Recall(voice) = 0.942857
Precision(guitar) = 0.731707 || Precision(violin) = 0.780488 || Precision(piano) = 0.810811 || Precision(voice) = 0.804878
FScore(guitar) = 0.722892 || FScore(violin) = 0.771084 || FScore(piano) = 0.769231 || FScore(voice) = 0.868421
fx >>
```

C=0.01 ,g=1

```
Accuracy = 66.875% (107/160) (classification)
        guitar  violin  piano  voice
guitar    25       6       7       4
violin    8       32       1       1
piano     5       14       22       0
voice     3       4       0       28
Recall(guitar) = 0.595238 || Recall(violin) = 0.761905 || Recall(piano) = 0.536585 || Recall(voice) = 0.800000
Precision(guitar) = 0.609756 || Precision(violin) = 0.571429 || Precision(piano) = 0.733333 || Precision(voice) = 0.848485
FScore(guitar) = 0.602410 || FScore(violin) = 0.653061 || FScore(piano) = 0.619718 || FScore(voice) = 0.823529
fx >>
```

C=0.0001,g=1

```
Accuracy = 21.875% (35/160) (classification)
        guitar  violin  piano  voice
guitar     0       0       0       42
violin     0       0       0       42
piano      0       0       0       41
voice      0       0       0       35
Recall(guitar) = 0.000000 || Recall(violin) = 0.000000 || Recall(piano) = 0.000000 || Recall(voice) = 1.000000
Precision(guitar) = NaN || Precision(violin) = NaN || Precision(piano) = NaN || Precision(voice) = 0.218750
FScore(guitar) = NaN || FScore(violin) = NaN || FScore(piano) = NaN || FScore(voice) = 0.358974
fx >>
```

C=0.00001,g=1

上面我藉由固定 g，並改變 C 來觀察 C 對於結果的影響。可以看出，C 有一個最適的值(1)，往上增加會導致 Accuracy 微幅減少，並且會有一個減少的上限(如上所示是 76.875)。而減少的話會導致 Accuracy 劇烈下降(甚至下降到超過隨機預測的期望精確度 25%)。

這可以想像成如果對於誤差項的 Penalty 根本就微不足道的話，那麼就算這個 classifier 分類的很糟，各個 class 的特徵空間都混在一起，也不會對 objective function 造成什麼影響，所以 train 出來的 classifier 很糟是預期中的事。

至於 C 太大的話可能會造成 Overfitting 的問題，因為你幾乎不允許 training model 中有錯誤，為了可以完全符合 training set 的分類，弄出來的 classifier 對於不再這個 training set 之中的資料敏感性就會變很低，因此也會降低之後 validation 時的精確度。

```
Accuracy = 53.125% (85/160) (classification)
        guitar  violin  piano  voice
guitar    12       17       10       3
violin    3       34       2       3
piano     2       20       19       0
voice     2       13       0       20
Recall(guitar) = 0.285714 || Recall(violin) = 0.809524 || Recall(piano) = 0.463415 || Recall(voice) = 0.571429
Precision(guitar) = 0.631579 || Precision(violin) = 0.404762 || Precision(piano) = 0.612903 || Precision(voice) = 0.769231
FScore(guitar) = 0.393443 || FScore(violin) = 0.539683 || FScore(piano) = 0.527778 || FScore(voice) = 0.655738
fx >>
```

C=1,g=1000

```
Accuracy = 21.875% (35/160) (classification)
        guitar  violin  piano  voice
guitar     0       0       0      42
violin     0       0       0      42
piano      0       0       0      41
voice      0       0       0      35
Recall(guitar) = 0.000000 || Recall(violin) = 0.000000 || Recall(piano) = 0.000000 || Recall(voice) = 1.000000
Precision(guitar) = NaN || Precision(violin) = NaN || Precision(piano) = NaN || Precision(voice) = 0.218750
FScore(guitar) = NaN || FScore(violin) = NaN || FScore(piano) = NaN || FScore(voice) = 0.358974
fx >>
```

C=1,g=10000

```
Accuracy = 76.875% (123/160) (classification)
        guitar  violin  piano  voice
guitar    33      1       5      3
violin    6       31      3      2
piano     5       6       29     1
voice     3       1       1      30
Recall(guitar) = 0.785714 || Recall(violin) = 0.738095 || Recall(piano) = 0.707317 || Recall(voice) = 0.857143
Precision(guitar) = 0.702128 || Precision(violin) = 0.794872 || Precision(piano) = 0.763158 || Precision(voice) = 0.833333
FScore(guitar) = 0.741573 || FScore(violin) = 0.765432 || FScore(piano) = 0.734177 || FScore(voice) = 0.845070
fx >>
```

C=1,g=0.001

至於 g 的影響，我們可以看到，1 是最適合的值。降低 g 的話也會導致 Accuracy 降低(並且有下限)，提升 g 的話則也會導致整個 Classifier 的 Accuracy 持續降低。

有此可知，Parameter tuning 在 Training 的過程中是非常重要的，因此，可以看出來老師用兩個 for 迴圈去 tune 出 best model 是個非常實際的方法，可以幫助我們快速逼近最適的參數。

Q6、

這題是重頭戲，也就是要我們發揮創意改善 classifier 來實驗不同的算法和特徵。一開始我直接把在 mirtool box user manual 中的 timbre 類 feature 全部加進去(frame level feature)，導致 feature 數變成上百(因為 3 秒鐘的音檔可以切成 92 個 1024 samples 的 frame)。用這樣子上百個 feature 去 train 出來的 model 的 Accuracy 卻是越來越低，而且從頭到尾都沒有增加過。(這邊少了 Pooling 的步驟，我後來才意識到)

一開始我想到的是，是否過多的 feature 反而會讓 model 變得越來越糟，於是我改用 Late Fusion。我把 mirtool box 裡 Timbre 類的特徵都獨自抽出個別訓練出的 Classifier，全部都跑過之後，個別去 predict validation set 的結果，並且根據 validation 的 accuracy 當作該 feature 的權重，最後去做 major voting，某個 feature 的 classifier 若認為某個音檔是小提琴，則把該音檔可能是小提琴的機率加上該 feature 的 classifier 的 accuracy。這也就是所謂 Decision level 的 fusion。

讓所有 feature level classifier 都對某個音檔做投票之後取出機率最高的音色

當作最後的 prediction。然而實際上效果卻沒有想像中的好。

```
    for i=1:length(Ypred),
        cumPred(i,Ypred(i)) =   cumPred(i,Ypred(i)) + accuracy;
        cumPred(i,Ypred2(i)) =   cumPred(i,Ypred2(i)) + accuracy2;
        cumPred(i,Ypred3(i)) =   cumPred(i,Ypred3(i)) + accuracy3;
        cumPred(i,Ypred4(i)) =   cumPred(i,Ypred4(i)) + accuracy4;
        cumPred(i,Ypred5(i)) =   cumPred(i,Ypred5(i)) + accuracy5;
        cumPred(i,Ypred6(i)) =   cumPred(i,Ypred6(i)) + accuracy6;
        cumPred(i,Ypred7(i)) =   cumPred(i,Ypred7(i)) + accuracy7;
        cumPred(i,Ypred8(i)) =   cumPred(i,Ypred8(i)) + accuracy8;
        cumPred(i,Ypred9(i)) =   cumPred(i,Ypred9(i)) + accuracy9;
        cumPred(i,Ypred10(i)) =   cumPred(i,Ypred10(i)) + accuracy10;
        cumPred(i,Ypred11(i)) =   cumPred(i,Ypred11(i)) + accuracy11;
        m = 0;
        for j=1:11,
            if cumPred(i,j) > m ,
                m = cumPred(i,j);
                finalPred(i) = j ;
            end
        end

    end
    count = 0.0;
    for i=1:length(Ypred),
        if finalPred(i) == Yvalidation(i),
            count = count + 1;
        end
```

就算使用 Late Fusion 技術，Accuracy 還是沒有上升的跡象，讓我百思不得其解了一陣子。是否有某些 feature 太像了，所以如果該 feature 對於這些音色的分辨來說不是好的 feature，那麼就會重疊性的把錯誤 predict 的權重加到最後的 Major voting 上。

最後，終於在講義上看到了 Pooling 的 Approach，是採用 frame level feature 的平均數和標準差來當作特徵，而不是把每個 frame 的 feature 都塞進去。

恍然大悟之後便回到基本的算法，也就是 Early Fusion，把所有特徵都放到同一個 vector 中去 train 出並且直接用這個長向量 train 出單獨的 classifier。把 Timbre 類的特徵重新抽出，然後用 mean 和 std 函式把 frame level feature integrate 成 clip level 的 feature。重新跑過之後發現 Accuracy 還是沒什麼上升，納悶之下就跑去把 Xtrain 這個 variable 點開來看看裡面的值，赫然發現裡面有很多 NaN 存在。這些 NaN 應該是再抽出 Feature 時因為函是無法處理的狀況而造成。

因此，我多寫了一些式子來把 NaN 都換成該 feature 的平均值。

```
%% Transfer the NaNs into mean


for i = 1 : size(X,1),
    for j = 1 : size(X,2),
        if isnan(X(i,j)),
            X(i,j) =  mean(X(:,j),'omitnan');
        end
    end
end
%%
for i = 1 : size(Xtest,1),
    for j = 1 : size(Xtest,2),
        if isnan(Xtest(i,j)),
            Xtest(i,j) =  mean(Xtest(:,j),'omitnan');
        end
    end
end
```

經過一番波折之後，終於讓 Accuracy 從 75%~80%提升到了 80%~90%。

```
Accuracy = 88.125% (141/160) (classification)
          guitar  violin  piano  voice
guitar     34       1       4       2
violin     3       30       1       1
piano      0        2      36       2
voice      0        3       0      41
Recall(guitar) = 0.829268 || Recall(violin) = 0.857143 || Recall(piano) = 0.900000 || Recall(voice) = 0.931818
Precision(guitar) = 0.918919 || Precision(violin) = 0.833333 || Precision(piano) = 0.878049 || Precision(voice) = 0.891304
FScore(guitar) = 0.871795 || FScore(violin) = 0.845070 || FScore(piano) = 0.888889 || FScore(voice) = 0.911111
fx >>
```

雖然可以想見，實際上在用 test set 測試時，結果應該不會像 validation 那麼好。就像講義上所說的，Cross-dataset generalizability，就算 train 出了 validation accuracy 逼近 90 的 model 不一定有辦法 fit 所有的 test set。