科學計算期末專題

AI-cup 歌聲轉譜競賽

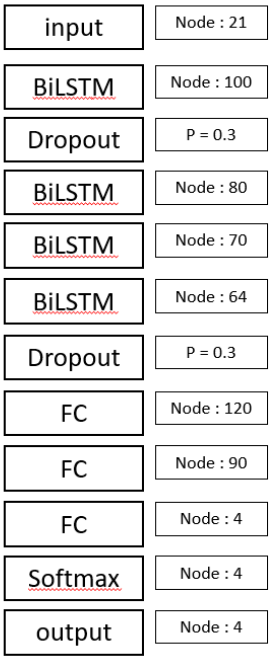
莊立楷 留崇恆 吳逸鈞 -【耶耶笙歌轉譜】

Department of Mechanical Engineering
National Taiwan University

Abstract- **在這個報告中我們將簡述我們在比賽中用來預測的模型、特徵使用、後處理等等，以及我們任何嘗試過的方法，雖然最後在 public set 上只有 0.316 的分數，但我們已在現有結果上盡力了。**

【模型】

在這個比賽中最重要的就是音符的起始點 on 準確度，故我們的決策結果是使用機器學習的方法訓練出可以預測 on 與 off 時間點的模型，而音符的音高則單純使用主辦給的「vocal_pitch」特徵做標註。而我們所使用的機器學習模型為 LSTM，input 為一個當前時間點的前後 10 個時間點的特徵(故包括自己共 21 個時間點)，而 output 則為當前時間點的分類(分類共 4 種：on、off、both、Nothing，分別對應起始、結束、同為結束和起始、以上皆非)，故一首歌的一個時間點就算一筆資料，整首歌約莫會有 5000~8000 筆資料，batch_size 設定為 300 筆，學習率常在 0.0001~0.001 之間調整。我們最終使用的模型架構如下圖：



【使用的資料特徵】

對於每個時間點，起初我們僅選用一些與 on 和 off 較有關係的特徵如 energy, spectral flux 等等下去訓練，發現模型很容易 overfit。後來才使用更多不同的 feature，我們秉持著一個精神：「就算是沒用的特徵，model 也要學到它是沒用的」，所以使用了如下的 9 個特徵，在丟進網路前都有做 normalization，使平均為 0，標準差為 1：energy, energy_entropy,

spectral_centroid, spectral_flux, spectral_rolloff, ectral_entropy, vocal_pitch, zcr，以及 chroma_1 到 12 的總和。另外，為了減少運算量，我們不讓一首歌的所有資料下去進行訓練，而是設定門檻：對於 energy < 0.00004 且 vocal_pitch 為 0 的點，我們不丟進去訓練，在這樣門檻的設定下，資料量大約可減少 30%左右。

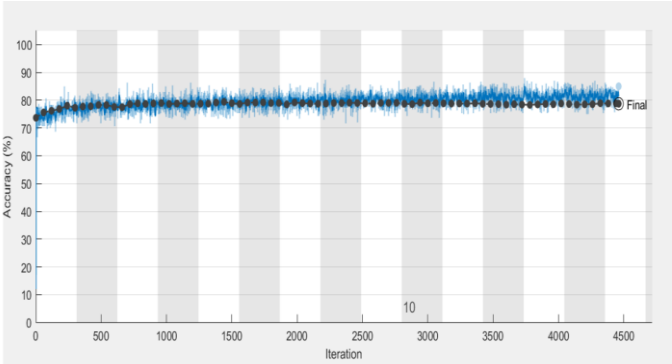
【資料預處理-K-means 分群】

起初我們預想應讓這 500 首歌分調性，先依據簡單的特徵讓歌曲分群，對於每個群皆訓練一個專屬的 model，對於一筆新輸入的資料先做一次小分類到專屬的群中，再使用 LSTM 做預測。然而我們發現這樣的效果不是很顯著，故最後沒有採用，而只是訓練一個大模型，讓 1~450 首歌進去訓練。

【訓練與驗證】

由於在這個任務中，分類「Nothing」佔了 90%以上，造成數據相當不平衡，起初直接使用原資料下去訓練，我們雖然可以看到準確率達 95%，但模型預測的結果為「幾乎全部預測為 Nothing」，於是我們嘗試丟棄一些分類為 Nothing 的資料，使資料稍微平衡一點，而這次的結果則為「預測了太多的 on 以及 off」，也就是模型太過敏感。至此我們才瞭解到，「捨棄 Nothing 資料的機率」在一定程度上影響著模型的好壞，而後我們多加調整這個參數，最後發現「捨棄 Nothing 的機率為 0.85」時，模型不會預測太多音，同時該預測的音也有預測到(在 vocal_pitch 上觀察)。

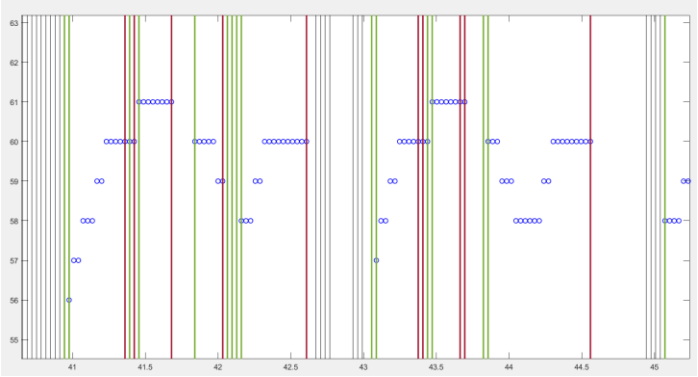
然而，當我們使用 10 首歌下去訓練時，training acc 很快可以達到 100%，而 validation acc 則停滯在 80%左右。這顯示我們的 model 有能力 overfit training data，而後我們才陸續加入更多資料以及 dropout 層。最後我們使用 1~450 首歌進行訓練，留下 451~500 首歌當作 validation set，下去訓練了一整天，整個訓練過程如下圖：



其中藍色線為 training acc，黑色線為 validation acc，可以發現到由於資料的不平衡，準確率一直處在 70~80%左右，而在迭代次數過一定數量後驗證集的準確率開始下降，模型 overfit 了。我們一直在調整各式參數(模型的 node, 選用的 feature, 丟棄 Nothing 的機率, 學習率等等)，使想要模型在更高準確率的地方 overfit 以提高整體準確率，但不論怎麼調適準確率皆在 81%左右停滯下來，我們也只好讓訓練 early stop，使用當前的模型去預測 on 與 off。稍微觀察我們預測的結果(在 validation set 上)會發現，其實預測的還不差，on 與 off 皆會切在 groundtruth 的左右兩格以內，只要我們進行適當的後處理，整體預測都還算準。

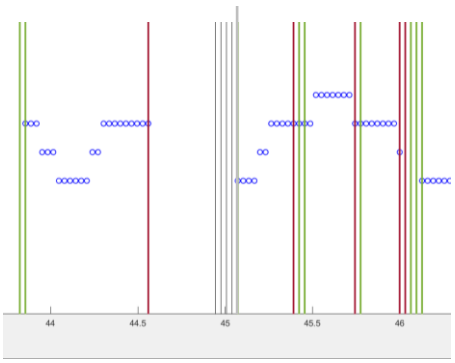
【資料後處理-補足模型的不足】

將機器學習預測結果用 matlab 作圖後，我們發現了一些問題，於是我們決定在不破壞預測結果的最大限度下將預測資料進行修正以確保輸出資料是正確的格式。

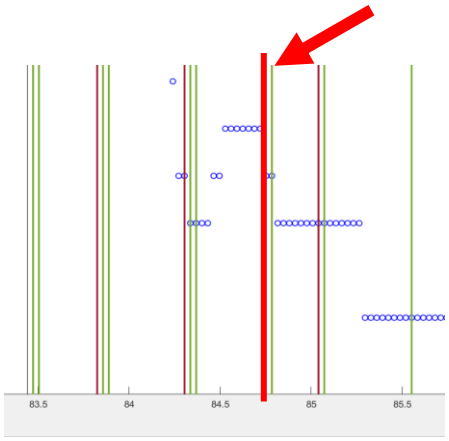


上圖一為某一首歌之預測結果，針對每個時間點會有的預測結果為 on：圖中綠線、off：圖中紅線、no predict(沒過門檻的點)：圖中黑線、Nothing：圖中空白處，而上圖的藍色圓圈為此首歌之 vocal pitch。

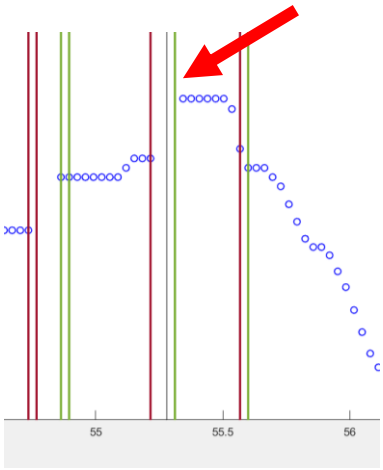
從上圖可以看出第一個問題是會有連續的 on 被預測出或是連續的 off，很顯然的這並不合理，因此我們將有出現連續線的地方取平均線，意即若有連續三條綠線，則將第一條與第三條綠線刪除，保留中間第二條綠線。同理紅線亦是如此。



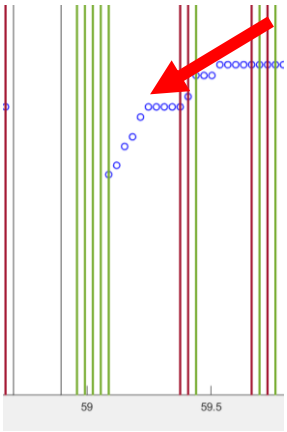
問題二為某些地方會有連續預測黑線，而卻沒有預測出綠線(on)或紅線(off)，因此我們決定若有出現連續黑線處，則在連續黑線後面補綠線，連續黑線前補紅線，這樣可以確保不會有音符被忽略掉。



問題三是在間隔較遠的兩個 on 之間並無預測 off，為確保每個 on 都有對應到一個 off，我們決定若發生此問題，則在第二個 on 的前面加上 off(如上圖所示)，如此一來可以保留原先預測出之 on 位置，亦可使最後輸出時不會出現有起始而無結束之狀況。



問題四是所預測出的 on 或 off 沒有對應到 vocal pitch，針對此問題我們的解決辦法是將 on 往後移至有 vocal pitch 的地方或是將 off 往前移至有 vocal pitch 的地方。



問題五為 vocal pitch 出現滑音。由於我們不清楚主辦方標記解答的方法是甚麼，也不確定主辦方之解答判定，我們在討論後決定若是有滑音出現，則將此音符的 on 往後移動到 pitch 是穩定的地方(上圖紅色箭頭處)。

經過多次的嘗試後，我們發現問題二對結果的影響並不大，而問題四與問題五若加以處理反而會得到較不理想得結果，因此我們決定使用第一與第三個處理。

此外，我們將訓練出的模型拿去預測原先預留的 451~500 首，比對 groundtruth 發現我們的預測都較為提早，因此我們大膽的將所有的 on 時間都往後移，結果發現分數提高不少，經多次嘗試後，得出最好的偏移量是將 on 時間加 0.048 秒、off 時間加 0.016 秒。

而 pitch 的判斷我們則是取每個 on,off 區間中的眾數做為此音符的 pitch，我們試過取平均、取前四分之一到二分之一之平均、或中位數，但結果都不是很理想，且透過評分系統可以得知我們上傳的檔案在 on 是對的情況下 pitch 的正確率約為 70%，於是我們決定使用 vocal pitch 的眾數做為 pitch 判斷的依據。

【結論】

由於模型訓練下去就要花半天，要對 1500 首歌進行預測又要 2~3 天，故我們犯錯機會非常少，需要妥善利用 validation set 去改進模型。然而我覺得整個過程最難的是模型的架構設定，因我們總是在不盡理想的準確率階段就看到 overfit，對於這種狀況只能先加入多一點 dropout 下去訓練，加入的結果常常是訓練不起來或訓練進步速度慢，加大學習率又會有震盪發生。故直到比賽最後我都不知道我們模型是否有能力能再更上一階，只讓準確率停滯在 80% 左右就結束拿去做後處理。不過我也想這可能是模型泛化的極限了，因為觀察歌曲的 feature 後會發現常常看不出甚麼規律，甚至同樣是 on 時，但 energy 可能在峰或谷，我們認為機器學習到這個階段應會困惑，所以整體 80%已是它的最佳表現了。

【組員分工】

- **莊立楷：**

LSTM 模型架設與訓練。

- **留崇恆：**

資料後處理、預測結果的改進和完善

- **吳逸鈞：**

資料前處理、程式輔助

【Reference】

[1] Matlab-Deep learning Toolbox
https://www.mathworks.com/solutions/deep-learning.html