

# ADLxMLDS2017 - HW1

電信碩一 宋易霖 r06942076

## 1 Problem Define

根據觀察到的一串 phone sequence 的特徵來預測該 phone sequence。

## 2 Dataset

fbank (69個feature), mfcc (39個feature)。分別共有3696筆data，其中最長的sequence為777。

## 3 Model Description

### 3.1 RNN

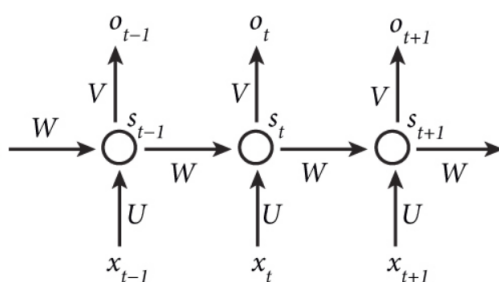


Figure 1: rnn model (reference: <http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>)

RNN的結構如Figure 1所示，其中  $x_1, x_2, \dots, x_n$  ( $n$  為sequence長度) 是每個phone的feature，經過以下運算得到每個hidden state

$$s_t = \tanh(Ux_t + b_U + Ws_{t-1} + b_W)$$

得到hidden state後，接上fully connected layer和softmax來預測最有可能的phone是甚麼。

$$\text{predict}_t = \text{softmax}(Vs_t + b_V)$$

### 3.2 CNN + RNN

CNN主要用來對 $X$ 取出一些區域性的特徵，再將這些特徵當作新的 $X$ 餵給RNN。而kernel的寬度儘量不會跨太多的frame，因為後面還會再用RNN取出時間性的feature，在這邊就專心對每個frame各自的feature取出區域特徵即可。

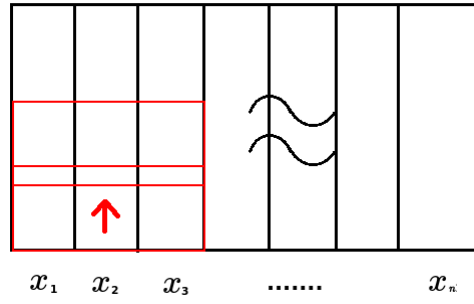


Figure 2: CNN block

## 4 How to Improve Your Performance

### 4.1 LSTM

因為傳統的RNN在sequence很長的時候容易遺忘較前面的資訊，因此利用LSTM能夠有較好的表現。其中每個Variable的求法如下：

記住前面部份的比例  $f_t$ :

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

新進的資訊  $i_t * \tilde{C}_t$  :

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

hidden state  $h_t$ :

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

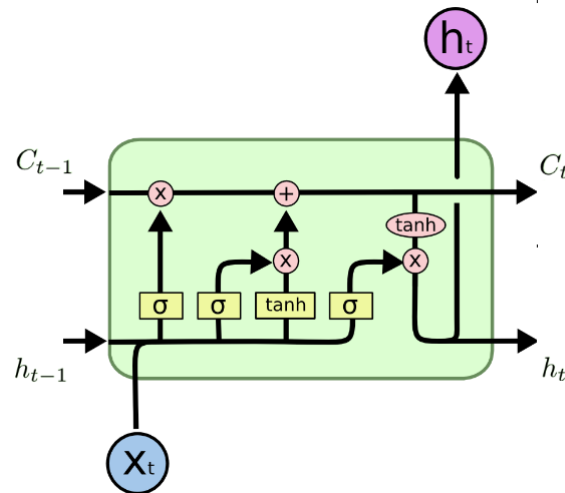


Figure 3: LSTM block (reference: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

## 4.2 Normalization

在rnn和lstm的model中，會將前一層output和現在的input接起來在繼續運算(參考3.1的式子)，而前一層的output因為經過了tanh函數數值介於-1和1之間，因此我把input都normalize到-1和1之間數值之間就會比較相近，訓練結果也較好。詳細normalize作法如下

for i in features of  $X$ : do

$$\begin{aligned} middle_i &= \frac{\max(X_i) + \min(X_i)}{2} \\ range_i &= \frac{\max(X_i) - \min(X_i)}{2} \\ X_{i,normalized} &= \frac{X - middle_i}{range_i} \end{aligned}$$

## 4.3 Smoothing

在預測出來的phone sequence中(尚未去除重複以前)，連續同樣的phone中間不太可能出現一個不一樣的，所以在去除掉重複的phone以前，會先做一次smoothing。方法是用一個sliding window掃過處理前phone sequence中每個phone，在這個window裏面出現最多的phone替換掉現在window中間那個phone，如果有相同出現次數的phone的話，以靠左邊的為主。參考Figure 4

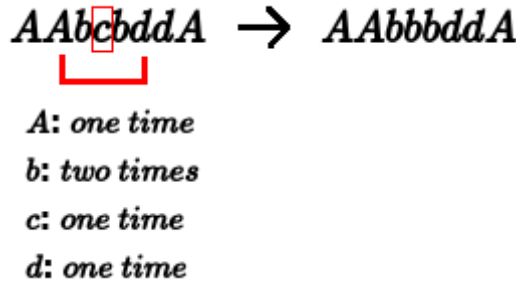


Figure 4: smoothing process

## 5 Experimental Results and Settings

model type	edit distance		
	training set	validation set	test set
RNN (both, w_s = 5, norm=[-1, 1])	18.70	21.20	0.0
RNN + CNN (both, w_s = 5, norm=[-1, 1])	30.04	30.86	0.0

Table 1: RNN及RNN + CNN的edit distance比較, model type欄位的格式為：model (用的feature: [fbank, mfcc, both]其中一個, window size, 標準化方法: [standardization(std), [-1, 1]]其中一個)

以上的實驗中，RNN的hidden layer為3層，每層有64個node和dropout=0.5，並且是bidirectional的。CNN則是有兩層，兩層的kernel size分別是 (3, 3), (1, 3)。依照3.2的想法，kernel不要跨很多frame，這邊是第1層跨3個，第2層就只有取一個frame而已了(因為前一層已經取了前後各一frame的資訊，這邊如果kernel size大於一的話，就會取到原本

前後各二個frame的資訊了)

從Table 1 可以看出，加上了CNN後performance差了不少，可能是因為data本身並沒有區域的特性，因此做了CNN後反而讓進入RNN的input變得更混亂了。

model type	edit distance		
	training set	validation set	test set
a. RNN (both, w_s = 5, norm=[-1, 1])	18.70	21.20	0.0
b. LSTM (both, w_s = 5, norm = std)	12.63	16.96	0.0
c. LSTM (both, w_s = 5, norm = [-1, 1])	9.90	14.78	0.0
d. LSTM (fbank, w_s = 5, norm = [-1, 1])	10.49	14.66	0.0
e. LSTM (both, no smoothing, norm = [-1, 1])	11.62	16.81	0.0
f. LSTM (both, w_s = 7, norm = [-1, 1])	9.78	14.46	0.0
g. LSTM (both, w_s = 9, norm = [-1, 1])	9.62	14.27	0.0
h. LSTM (both, w_s = 11, norm = [-1, 1])	9.80	14.05	0.0

Table 2: 其他model edit distance比較, model type欄位的格式為: model (用的feature: [fbank, mfcc, both]其中一個, window size, 標準化方法: [standardization(std), [-1, 1]]其中一個)

LSTM的設定都和RNN一樣: hidden layer三層, 每層64個node和dropout=0.5, 並且為bidirectional。

1. 從a, c兩個model可以看出LSTM比起RNN的表現好上不少, 在validation set上edit distance少了4.3。
2. 從b, c兩個model可以發現用4.2提到的normalization比用standardization在validation set上edit distance少了2.2。
3. 從c, e兩個model可以發現加上smoothing可以使得model在validation set上edit distance減少了2。
4. 在c, d兩個model比較feature的使用: 單用fbank和用fbank+mfcc差不了太多, 在public test上也差不多。但最後保持著用多點feature準沒錯的心態下還是繼續兩個feature都用了。
5. 在e-h中試著增加window size, 發現在window size為9的狀況下在training set和validation set下是最好的。