ADL HW1: TIMIT Report

R06725035 陳廷易

Model description

RNN

- LSTM + BLSTM + Dense + LSTM + Dense
- LeakyReLU
- Softmax + categorical cross-entropy + adadelta

在 RNN model 中，嘗試了各式各樣的循環層，從最基礎的 SimpleRNN 到 GRU 再到助教說明所述的 LSTM，經嘗試後發現 LSTM 在這當中是表現比較好的，但訓練時所花的時間也是相對最多的。

而為了能讓 model 學到雙向的資訊，故加入 Bidirectional wrapper。但此舉也會使訓練時間大幅度增加，且過多的 BLSTM 也對 performance 沒有太多的進展，因此在實驗後決定將 BLSTM 加於中間的 hidden layer 對 performance 貢獻最大。

在 Activation 部分，起初嘗試了 sigmoid 及 tanh，但需要進行一大堆複雜運算，造成更新速度慢、訓練效率差且會有 saturation 的情況。因此改採用 relu，大幅改善了學習過慢的問題，讓收斂速度快上不少，但或許因為 learning rate 過大的緣故而發生了 dying relu 的問題，會捨去較多的資訊，而使成效經常不若前者。最後便採用 Leaky-ReLU 修正了數據分布，也保留了負軸的值，不會直接捨去。使得此 model 也是在單一 model 中表現最好的。

而因為是採用 sklearn 的 Label Binarizer 來 encode 的關係，因此最後一層採用 softmax 以對應回去 label，搭配 categorical cross-entropy 以獲得最大效果。

在 Optomizer 的部分，因為對各式優化器不甚熟悉，便依照 documentation 慢慢做實驗嘗試，後來發現 adadelta 在此模型中是表現比較出色的。

RNN+CNN

- Convolution1D + MaxPooling1D + Dense + Flatten + LSTM + Dense
- relu / LeakyReLU / sigmoid
- Softmax + categorical cross-entropy + adadelta

在 CNN 的部分，因為是語音而非圖像因此採用 1D 已足夠，也嘗試過使用 2D，但是對 performance 提升有限且需要花非常大量的訓練時間。CNN Flatten 過後，RNN 使用的是 LSTM 循環層。比較值得一提的是，dropout 在其中也發揮了相當功效，起初因為值設的較低最後結果都不太好，但若有較高的值再加上足夠量的 epoch 就不太容易 overfit 且 performance 也會提升不少。其餘參數的設置原因也與上述 RNN 雷同。

How to improve your performance

- 資料清洗、濾除雜訊、padding 技巧
- 資料前處理、預先 mapping
- 加深模型、增加神經元
- 結合模型、給予權重

在經過無數次的實驗後，終在最後一個禮拜探索出上述的 RNN model 後超越 baseline。然而當中最關鍵的部分並不是訓練出了多偉大的 model，而是重新檢視了資料清洗的流程與細節。

在資料集方面，從 mfcc 改為使用維度較多的 fbank，performance 便提升了些許。為了使我的 model 能夠訓練，需要使用到 padding 方法；但也希望 model 不會受 padding 而影響效果，因此將 padding 原本的前方移至句子後方補

0，同時將 training data 裡面的數值進行 offset，使真實資料與 padding 區隔開來、避免誤判。此外，經過觀察資料狀況也發現，組成句子的 frame 應該是大部分都會連續重複出現的；換言之若某 frame 只有單一字詞者，沒有與前一個或後一個 frame 重複者應為雜訊或轉換錯誤所致，因此也應予以濾除。而 Mapping，我亦將之從原先訓練完再 mapping 移到訓練前就先 mapping，希望能讓 model 對該字母可以有更多的資訊以增加正確率，而在我的情況下也確實使 performance 提升了些許。

在模型方面，幾乎是遵守 the deeper, the better 原則，雖並非絕對，但的確對 performance 能有一定程度的改善；相對地訓練時間也很可能變得相當可觀。為了也讓 model 每一層能學到更多資訊，我也增加了神經元數目，從原先幾十個 nodes 增加到幾百個甚至幾千個，然而增加太多的神經元除了易使訓練速度大幅下降外也相當吃記憶體，且邊際效益會大幅降低。此外，為了防止 overfit，亦使用了 early stopping。

最後便是將各種不同的 model 結合在一起共同 predict，但因各個 model 的 performance 不盡相同，很有可能會有拖累的現象。因此，我依據各個 model 原先各個 model 上傳的 edit distance performance 來給予權重，使最後的預測結果得以再提升些許。

Experimental results and settings

循環層中的 LSTM 實際上對 TIMIT 資料集相當有用，基本上只要資料處理的足夠好，model 有一定的深度效果都會不錯。於前面再加上 CNN 時，若參數沒有選擇妥當(如: filters、kernel size、pool size 等等)，很可能反而使 performance 比原本的 RNN 還差，或是近乎沒有改善幅度。但倘如使用 keras 的 ConvLSTM2D 搭配 Batch Normalization 則提升的效果卻相當顯著，但所需要的記憶體與訓練時間也是相當可觀，且需要對資料進行 reshape 不免會有失真的情況。

傳統基本的 RNN 因為會有難以處理的 vanishing 問題，因此 GRU 或 LSTM 利用閘門的概念來保留住值，使資訊可以記得較久，也就能解決資訊到後來會完全消失的情形，進而提升 performance。

最後在 best model 中共利用了五個 model，給予不同的比重以共同 predict，包含了上述的兩種 RNN、兩種 CNN 與 ConvLSTM2D，才將 edit distance 降低至 13 以內。