MLDS: HW1

October 28, 2017

Professor Hung-Yi Lee

電機四 B03901018 楊程皓

Model Description

- RNN
  我的 rnn model 以一個雙層或三層的 LSTM/GRU cell 組成，並以 bidirectional 執行 dynamic rnn (將 cell 複製一遍)。其中加入 dropout = 0.1 (output keep prob = 0.9), 並在 output weight 加入 regularization 避免 overfitting 發生。Training data input 則是先將每個 frame 對應的 label 對齊好 (以 pandas)，然後把每個音檔 (sentence) 歸類好，然後選定一個 sequence 長度，將每個因檔切分為很多個 sequence 長度的 training data, 如音檔一有 200 frames，sequence len = 20, 則有 181 個 sequence (shift 1 frame)，然後再以若干個 sequence 組成 batch 給 rnn model train (我這邊使用 128 作為 batch size)。
- CNN
  我的 rnn+cnn model 為在 rnn 上加一個很簡單的 conv2d cnn 組成，考慮到 fbank, mfcc 皆為對每個 feature 做零次一次兩次微分，且為方便處理 label 對應，不想改變 sequence 長度，故我的 cnn 每個 window 包括一個 frame 的 3 個 features, 且 sliding 為往下滑三個 features，並在產出的 64 layers 做一個 fully connected layer 產出一個跟原 input 同 dimension 的 Tensor 以維持 rnn 的原架構。

How to improve your performance

基本上因為在 rnn model 上的時間遠多於 cnn, 加上 cnn 的架構後 performance 並沒有增加，反而使 model 更複雜，導致需要更多時間做 training，故我後來也主要把時間花在如何改進 rnn 的架構與細節讓 performance 上升。我的最佳 performance model 即為 rnn model, 基本架構描述如上，曾經也參考過 sequence 長度為最長的那個 training/testing data sentence，並把不夠長度的都補零，但這樣的方法我認為會讓 training 很沒效率，故沒有採用，轉而使用固定長度的 sequence 使 rnn 充分利用 training data。另外我在產生完所有 sequence 之後有做個 random shuffle, 再製作 batches，以避免掉入同樣音檔的 bias, 缺點為需要極大量的 memory (我可能需要 20-30 GB) 來做 training。之後我把 lstm cell num units 設為 512 後，在 3 個 epoch 後即開始在 cross validation 觀察到 overfitting, 故加入 regularization, 觀察到 overfitting 有些許減緩，但時間關係已沒有辦法再調整參數取得更高的 performance。因為我的 model 可以對每個重疊的 sequence 都產生 prediction, 我將產出的 logits 對每個 frame 做每次 sequence 重疊的累加，並再取 argmax 作為最終 prediction symbols. 由於觀察到實際的 symbols 大部分都連續三個以上，我的 testing script 只會將連續三個以上同樣的 frame prediction 列入計算 output 以避免被少數錯誤 prediction 影響結果。

Experimental results and settings

RNN 跟 CNN+RNN 的比較上，CNN 加上去的 performance 值降低了約 1.3 左右，可能是因為我的 CNN 做的方式後來想想不太合理，應該要在前後做 padding, 並考慮前後的 frame 後再做 reduce dimension。此比較的其他參數為：feature fbank, lstm num units = 100, 20 epoch, 0.1 dropout.

我有嘗試的有 BasicRNN, GRU, LSTM 三種 cell, 其中 GRU 與 LSTM 大約相當，並比 BasicRNN 好約 1 到 2; bidirection 比單向的好約 2 左右; dropout 並沒有產出明顯的差距; cell 層數有使用過 1, 2, 3 層; 之後主要在調如計算 loss 時使用的只用後半 sequence label (不使用 bidirection 的情況, 畢竟 rnn 在後半段應會比前半段準)，這樣使用跟全部考慮沒有明顯的差距; 在一開始使用 bidirection 時，也有考慮後半 sequence 使用正向 rnn, 前半 sequence 使用反向 rnn，但這樣 train 的結果非常糟糕，後來我就直接把兩個 model 產出的 logits 相加後再計算 prediction 及 loss。將 lstm cell unit num 從 100 調為 512 後，雖然在 3 epoch 之後明顯 overfitting, 但在 cross validation 最高的那個 epoch 仍有目前我試到最好的 public set performance。以上是我分別 apply 不同想法的大致順序與觀察到的結果。