Machine Learning - Homework 4

資工四 B05902023 李澤諺

April 30, 2020

1. (1%) 請說明你實作的 RNN 的模型架構、word embedding 方法、訓練過程 (learning curve) 和準確率爲何?(盡量是過 public strong baseline 的 model)

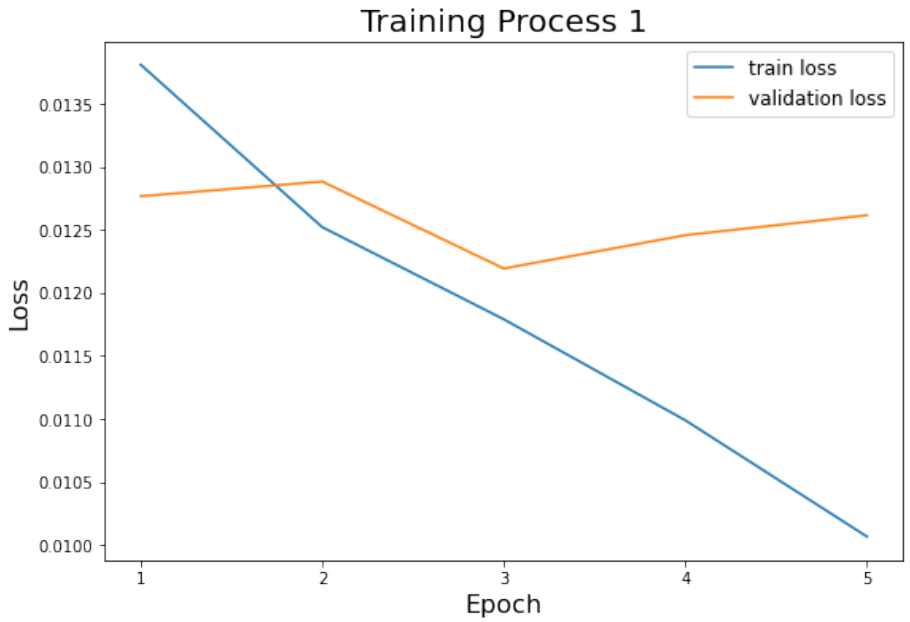
在 word embedding 中，首先，我將 training_label.txt、training_nolabel.txt、testing_data.txt 中所有的句子使用空白進行斷詞，再將包含非數字或非英文字母的詞移除，最後將所有的句子 trim 或 pad 使得其長度爲 32，以此進行 preprocessing，接著，我將所有的句子輸入 gensim 的 Word2Vec model 中進行訓練，其參數如下：

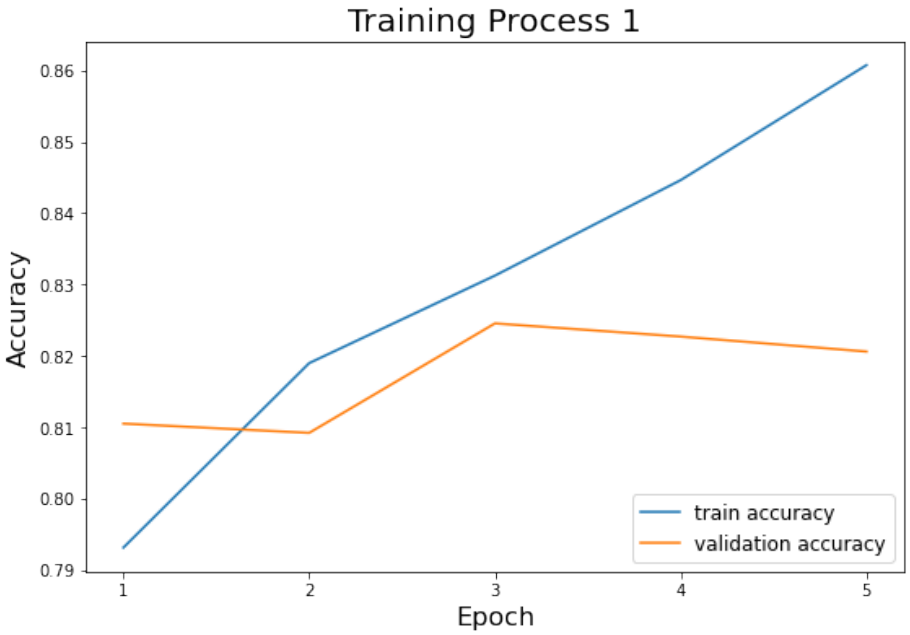
Word2Vec(sentences , size = 256 , sg = 1 , window = 5 , min_count = 5 , iter = 10 , workers = 12)

接著，以下爲我於本次作業中使用 PyTorch 所實作的 RNN 架構：

embedding	Embedding(embedding.size(0) , embedding.size(1) , padding_idx = padding_index)
recurrent	LSTM(embedding.size(1) , 150 , batch_first = True , bias = True , num_layers = 2 , dropout = 0.3 , bidirectional = True)
linear	Dropout(0.5)
	Linear(900 , 1 , bias = True)
	Sigmoid()

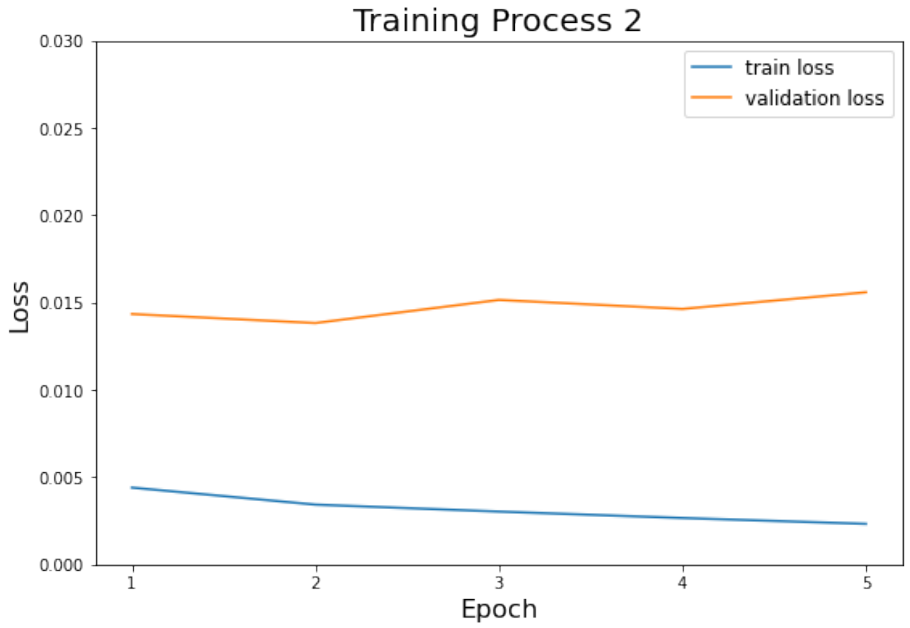
在以上的 RNN 架構之中，我將 embedding 的值固定使其不做更新，此外，在 LSTM 的 output 要傳給 linear 之前，我將 LSTM 的 output 對 time series 取最大值、最小值以及平均值，並將這三個值 concatenate 之後才傳給 linear。接著，我從 training_label.txt 中選出 20000 個句子作爲 validation data，剩下的句子則作爲 training data，並且，我使用了 Adam 訓練 RNN，其中 learning rate 爲 0.001，batch size 爲 32，以此訓練了 5 個 epoch，所得到的 loss 和 accuracy 如下圖所示：

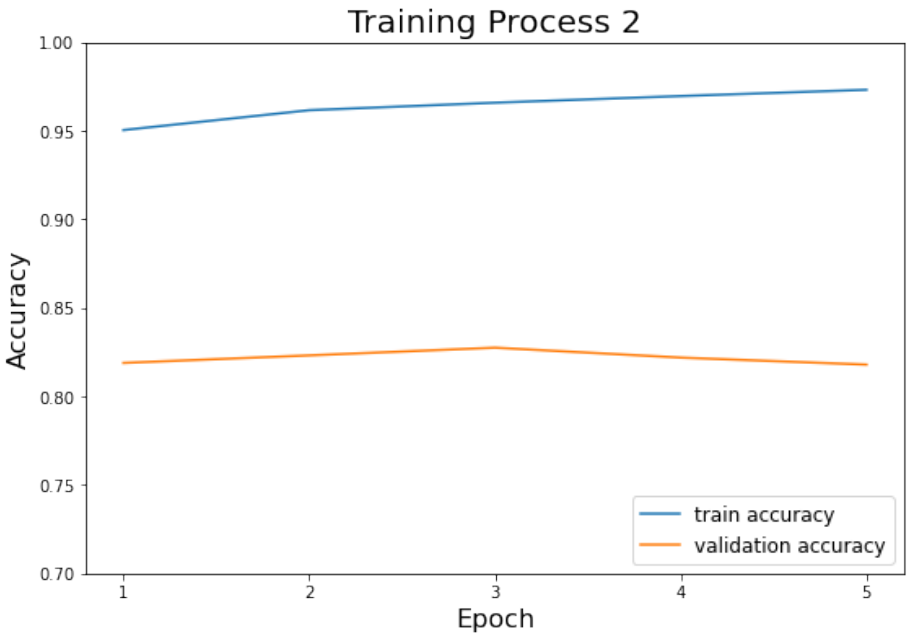




接著，我將 training_nolabel.txt 中所有的句子輸入 RNN 之中進行預測，若 RNN 對某一筆句子所輸出的 probability 大於 0.9，就將其 label 標示爲 1 並加入 training data 之中，而若 RNN 對某一筆句子所輸出的 probability 小於 0.1，就將其 label 標示爲 0 並加入 training data 之中，如此進行 self-training 之後，再重新

訓練 RNN，與第一次訓練時相同，我使用了 Adam 訓練 RNN，其中 learning rate 爲 0.001，batch size 爲 32，以此訓練了 5 個 epoch，所得到的 loss 和 accuracy 如下圖所示：





在以上的訓練過程中，我會不斷儲存到目前爲止 validation accuracy 最高的 RNN，以下爲我最後所得到的 RNN 其所得到的 accuracy：

Train	Validation	Test	
		Public	Private
0.96577	0.82750	0.82653	0.82459

2. (2%) 請比較 BOW+DNN 與 RNN 兩種不同 model 對於"today is a good day, but it is hot" 與"today is hot, but it is a good day" 這兩句的分數 (過 softmax 後的數值)，並討論造成差異的原因。

以下爲我使用 PyTorch 所實作的 DNN 架構：

linear	Linear(input_dim , 1024 , bias = True)
	BatchNorm1d(1024)
	ReLU()
	Dropout(0.5)
	Linear(1024 , 1 , bias = True)
	Sigmoid()

由於 BOW 所需要的記憶體空間過大，因此我從 training_label.txt 中僅選出了 50000 個句子作爲 training data，並另外選出了 20000 個句子作爲 validation data，再將這些句子皆轉換爲 BOW，以此進行訓練。以下爲 BOW+DNN 和 RNN 分別對"today is a good day, but it is hot" 和"today is hot, but it is a good day" 這兩句話所輸出的分數：

	"today is a good day, but it is hot"	"today is hot, but it is a good day"
BOW+DNN	0.7062	0.7062
RNN	0.0186	0.9962

由此可以看出 RNN 可以分辨出這兩句話在語意上的不同，但是 BOW+DNN 卻無法做到，原因在於 RNN 會考慮到字詞的順序是如何影響語意，但是 BOW+DNN 則不考慮字詞順序，因此對 BOW+DNN 來說這兩句話是一樣的，故 BOW+DNN 無法分辨出這兩句話語意的不同。

3. (1%) 請敘述你如何 improve performance (preprocess、embedding、架構等等)，並解釋爲何這些做法可以使模型進步，並列出準確率與 improve 前的差異。(semi-supervised 的部分請在下題回答)

以下爲助教提供的 sample code 所得到的 accuracy：

Train	Validation	Test	
		Public	Private
0.81849	0.80389	0.80648	0.80832

首先，我做了以下簡單的修改：

(1) 由於我認爲包含非數字或非英文字母的詞，可能是無法顯示的表情符號等等，其幾乎不會影響到句子的語意，但是其在 RNN 的訓練過程中可能會變成一種 noise，因此將其去除。

(2) 由於當 feature 的 dimension 變大時，可能可以提供更多的資訊，因此我將 word embedding 的 dimension 增加爲 256。

(3) 我將 LSTM 改爲 bidirectional，並將其 layer 數目改爲 2 層，以從資料中得到更多的資訊，以及增加 model 的 complexity。

(4) 根據老師上課的內容，當 batch size 較小時可能可以提高 model 的 accuracy，因此我將 batch size 降低爲 32。

經過以上簡單的修改之後，可以使得 model 的 accuracy 稍微上升，如下表所示：

Train	Validation	Test	
		Public	Private
0.84484	0.80750	0.80729	0.80920

接著，以下爲我認爲可以使得 model 的 accuracy 進步較多的重要修改：

(1) 我將所有句子的最大長度從 20 增加爲 32，由於在資料中有 99.9% 的句子其長度不超過 32，因此將句子的最大長度增加爲 32 時，可以保留大部分句子中的資訊，進而提高 model 的 accuracy。

(2) 在 LSTM 的 output 要傳給 linear 之前，我將 LSTM 的 output 對 time series 取最大值、最小值以及平均值，並將這三個值 concatenate 之後才傳給 linear。RNN 輸出的 sequence 其各個 dimension 都代表了一種 feature，因此各個 dimension 的最大值和最小值就很有可能代表了這句話之中哪個詞最能表現出該 feature，以此加上 sequence 的平均值再傳給 linear，可能可以做出更正確的分類。

經過以上的修改之後，model 的 accuracy 如下表所示：

Train	Validation	Test	
		Public	Private
0.84522	0.82395	0.82126	0.82335

4. (2%) 請描述你的 semi-supervised 方法是如何標記 label，並比較有無 semi-supervised training 對準確率的影響並試著探討原因。(因爲 semi-supervise learning 在 labeled training data 數量較少時，比較能夠發揮作用，所以在實作本題時，建議把有 label 的 training data 從 20 萬筆減少到 2 萬筆以下，在這樣的實驗設定下，比較容易觀察到 semi-supervise learning 所帶來的幫助)

我將 training_nolabel.txt 中所有的句子輸入 RNN 之中進行預測，若 RNN 對某一筆句子所輸出的 probability 大於 0.9，就將其 label 標示爲 1 並加入 training data 之中，而若 RNN 對某一筆句子所輸出的 probability 小於 0.1，就將其 label 標示爲 0 並加入 training data 之中，如此進行 self-training。爲了比較有無使用 semi-supervised training 的影響，我從 training_label.txt 中僅選出 10000 個句子進行訓練，在不使用 semi-supervised training 的情況下，所得到的 accuracy 如下表所示：

Train	Validation	Test	
		Public	Private
0.84730	0.78195	0.78315	0.78379

而在使用 semi-supervised training 之後，所得到的 accuracy 如下表所示：

Train	Validation	Test	
		Public	Private
0.99738	0.78395	0.78590	0.78592

由此可以看出 semi-supervised training 可以些許提高 model 的 performance，我認爲其原因爲，訓練 NN 時需要有大量的資料才能得到好的 performance，因此在 training data 只有 10000 筆時，model 的 performance 很難達到最好，但是在加上大量 unlabeled data 所提供的資訊後，便可以幫助 model 提高 performance，此外，由於 NN 很容易被 noise 影響，因此在產生 pseudo-label 時，需要對該筆資料有很高的 confidence，才能減少 label 錯誤的情況。