Homework 4 Report

資工所碩二 R06922132 何羿辰

EE5184 - Machine Learning

Problem 1. (0.5%) 請說明你實作之 RNN 模型架構及使用的 word embedding 方法，回報模型的正確率並繪出訓練曲線 *。(0.5%) 請實作 BOW+DNN 模型，敘述你的模型架構，回報正確率並繪出訓練曲線。

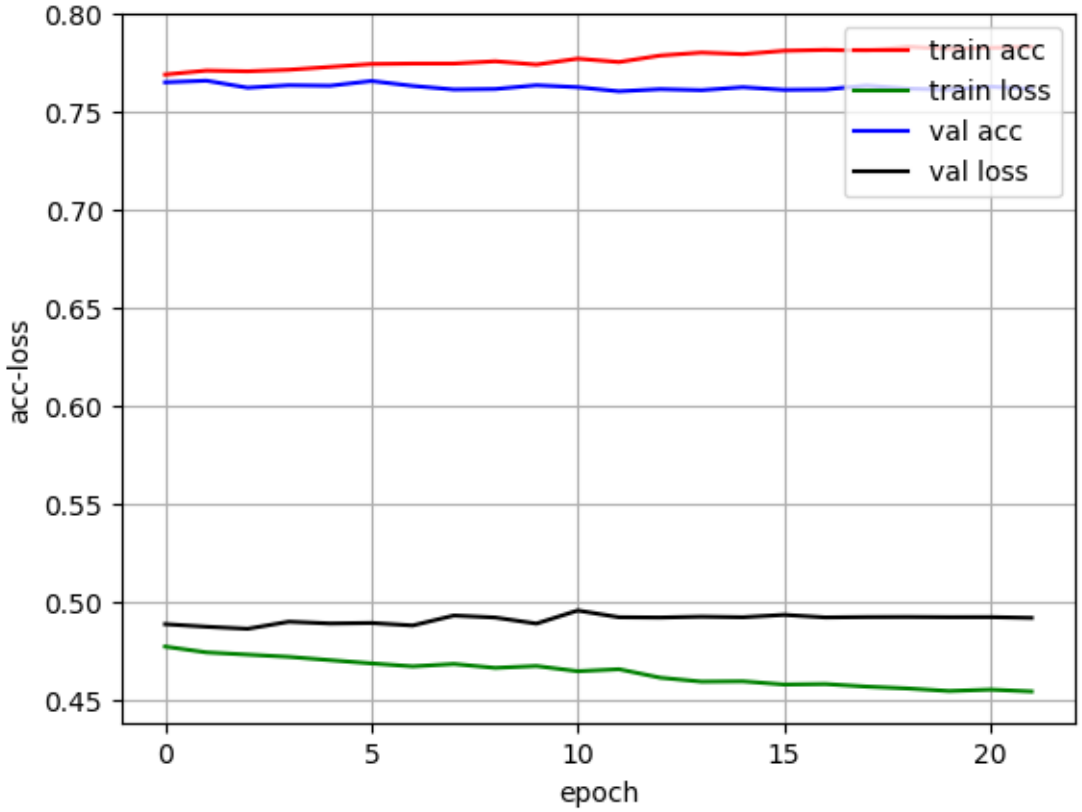
* 訓練曲線 (Training curve)：顯示訓練過程的 loss 或 accuracy 變化。橫軸為 step 或 epoch，縱軸為 loss 或 accuracy。

(a) RNN model:

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 100, 500)	18923000
bidirectional_1 (Bidirection	(None, 100, 128)	289280
bidirectional_2 (Bidirection	(None, 64)	41216
dense_1 (Dense)	(None, 1)	65
activation_1 (Activation)	(None, 1)	0

Total params: 19,253,561
Trainable params: 330,561
Non-trainable params: 18,923,000

RNN 架構我使用兩層 bidirectional LSTM，而 w2v 則使用了 train+test data 來訓練 model，最後得到的正確率是 0.75517，訓練曲線如下:



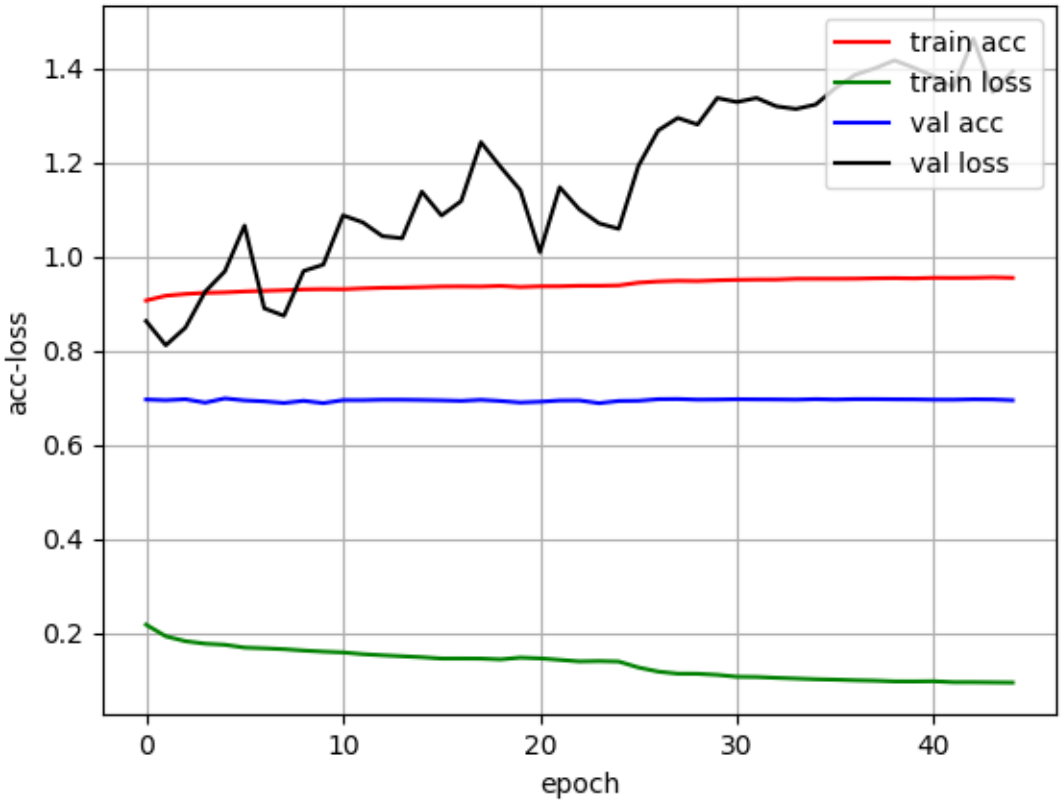
從結果可以看到 train acc 穩定上升，但 val acc 卻沒太大變動

(b) BOW+DNN model:

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 512)	512512
activation_1 (Activation)	(None, 512)	0
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
activation_2 (Activation)	(None, 256)	0
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
activation_3 (Activation)	(None, 128)	0
dropout_3 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8256
activation_4 (Activation)	(None, 64)	0
dropout_4 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 64)	4160
activation_5 (Activation)	(None, 64)	0
dropout_5 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 1)	65
activation_6 (Activation)	(None, 1)	0

Total params: 689,217
Trainable params: 689,217
Non-trainable params: 0

DNN 架構用 5 層的 dense+drop 完成，BOW 則使用 keras 的 tokenizer 完成 text to matrix 的部分，最後得到的正確率是 0.70075，訓練曲線如下:



從圖中可以明顯看到有 overfit 的現象，val 的 loss 趨勢與 train 的背道而馳，這個部分我嘗試過更改 drop rate 與 batch size 以及 dense 層數，都無法得到太有效的改善。

Problem 2. (1%) 請敘述你如何 improve performance（preprocess, embedding, 架構等），並解釋為何這些做法可以使模型進步。

我主要專注在於改善 RNN 的架構，在前處理方面，除了使用 jieba 斷詞以外，我根據 data 的特性設了兩個條件式，

i. 因為資料來自 Dcard，網友彼此稱呼對方的名稱會使用帳號名稱，例如 b123111，為了收斂 w2v 的效果，所以我在把資料丟進去 w2v 之前會把這些 b+數字的詞一律改成’某人’

ii. 因為每個網友打字習慣不同，在句子與句子之間的空格有的用一格，有的用兩格，這些我一律改成只空一格

iii. 此外還有觀察到句子尾端常有類似洗版現象，不斷重複的表情符號或單詞，這些雖然應該被拿掉，但我單純設定一個合理的 padding 來把過長的句子截掉，但我預期這部分如果能準確拿掉的話應該可以提升更多準確度。

然後在 w2v 的部分，我認為 padding 與不在 w2v model 裡的詞(not found)兩者的性質不一樣，因此我給這兩者另外各開了一個 vector(0 與 1)，其他字從 2 開始往後增加。

在架構的部分，有試過單純 LSTM 與雙向 LSTM，最後發現雙向 LSTM 較好，而 LSTM 之後的 Dense，也有試過先接個 64 的 input 再接到 Dense(1)，最後發現 LSTM 處理完後直接接 Dense(1)的效果還是最好。

在參數的部分，batch size 也試過許多種，最後發現最佳還是落在 1024~2048 之間，而 epoch 因為我有使用 early stopping 所以我只是抓個大概 250 左右，通常一百多 epoch 就會提早中斷。

Problem 3. (1%) 請比較不做斷詞 (e.g., 以字為單位) 與有做斷詞，兩種方法實作出來的效果差異，並解釋為何有此差別。

有做斷詞準確率為 0.75517，而沒做斷詞的準確率為 0.74995，明顯有做斷詞會比較好，原因應該是因為在中文的文法裡一個意思是由好幾個中文字組成，例如 ”雞蛋” 和 “雞” “蛋” 在中文裡代表的意思不完全一樣，若沒有先做斷詞的話，則 DNN 無法學到中文語意與惡性留言之間的關係，自然準確率就會大幅下降。

Problem 4. (1%) 請比較 RNN 與 BOW 兩種不同 model 對於" 在說別人白痴之前，先想想自己" 與" 在說別人之前先想想自己，白痴" 這兩句話的分數（model output），並討論造成差異的原因。

W2v+RNN model:

“在說別人白痴之前，先想想自己”這句話預測的分數為 0.433(非惡意)，而”在說別人之前先想想自己，白痴”這句話預測的分數為 0.734(惡意)，由此可見雖然兩句斷詞後是由同樣的詞所組成，但由於詞的順序不同，LSTM 預測出來的結果也會不同，這個例子精確的偵測出了第二句是惡意而第一句非惡意。

BOW+DNN model:

兩者的分數都是 0.999(惡意)，原因是因為 BOW 完的結果兩者會得到一樣的 text matrix，當然把一樣的 text matrix 丟到 DNN 裡預測會得到一樣的答案

Problem 5. (1%) In this exercise, we will train a binary classifier with AdaBoost algorithm on the data shown in the table. Please use decision stump as the base classifier. Perform AdaBoost algorithm for T = 3 iterations. For each iteration (t = 1, 2, 3), write down the weights u_t^n used for training, the weighted error rate ϵ_t, scaling coefficient α_t, and the classification function $f_t(x)$. The initial weights u_1^n are set to 1 (n = 0, 1, ..., 9). Please refer to the course slides for the definitions of the above notations. Finally, combine the three classifiers and write down the final classifier.

x	0	1	2	3	4	5	6	7	8	9
y	+	-	+	+	+	-	-	+	-	-

根據講義 W7a p23~p26 做法，考慮十個點中的九個間距的切法(假設都切在 x+0.5 的地方)，作出如下 table

Iter	weights										ε	d	α	*f(x)*
	0	1	2	3	4	5	6	7	8	9				
	+	-	+	+	+	-	-	+	-	-				
1	1	1	1	1	1	1	1	1	1	1	0.200	2.000	0.693	sign(4.5-x)
2	0.5	2	0.5	0.5	0.5	0.5	0.5	2	0.5	0.5	0.375	1.291	0.255	sign(7.5-x)
3	0.387	2.582	0.387	0.387	0.387	0.645	0.645	1.549	0.387	0.387	0.350	1.362	0.309	sign(0.5-x)
sum value	0.94	0.41	0.70	0.70	0.70	-0.76	-0.76	-0.66	-0.94	-0.94				
final output	+	+	+	+	+	-	-	-	-	-				

Table 裡黃色部分為每個回合預測錯誤的 unit，最後的 **output 為 [+++++-----]**，各回合的ε_t α_t

$f_t(x)$ 如 table 裡所示，而最終 classifier 為:

$$\text{sign}\left(0.693 * f_1(x) + 0.255 * f_2(x) + 0.309 * f_3(x)\right)$$
$$= \mathbf{sign}(\mathbf{0.693} * (\mathbf{4.5} - \mathbf{x}) + \mathbf{0.255} * (\mathbf{7.5} - \mathbf{x}) + \mathbf{0.309} * (\mathbf{0.5} - \mathbf{x}))$$

Problem 6. (1%) In this exercise, we will simulate the forward pass of a simple LSTM cell. Figure.1 shows a single LSTM cell, where z is the cell input, z_i, z_f, z_o are the control inputs of the gates, c is the cell memory, and f, g, h are activation functions. Given an input x, the cell input and the control inputs can be calculated by

$$z = w \cdot x + b$$

$$z_i = w_i \cdot x + b_i$$

$$z_f = w_f \cdot x + b_f$$

$$z_o = w_o \cdot x + b_o$$

Where w, w_i, w_f, w_o are weights and b, b_i, b_f, b_o are biases. The final output can be calculated by

$$y = f(z_o)h(c')$$

where the value stored in cell memory is updated by

$$c' = f(z_i)g(z) + cf(z_f)$$

Given an input sequence x^t (t = 1, 2, ..., 8), please derive the output sequence y^t. The input sequence, the weights, and the activation functions are provided below. The initial value in cell memory is 0. Please note that your calculation process is required to receive full credit.

$w = [0,\ 0,\ 0,\ 1]$, $b = 0$
$w_i = [100,\ 100,\ 0,\ 0]$, $b_i = -10$
$w_f = [-100, -100,\ 0,\ 0]$, $b_f = 110$
$w_o = [0,\ 0,\ 100,\ 0]$, $b_o = -10$

t	1	2	3	4	5	6	7	8
x^t	0	1	1	0	0	0	1	1
	1	0	1	1	1	0	1	0
	0	1	1	1	0	1	1	1
	3	-2	4	0	2	-4	1	2

$$f(z) = \frac{1}{1+e^{-z}} \qquad g(z) = z \qquad h(z) = z$$

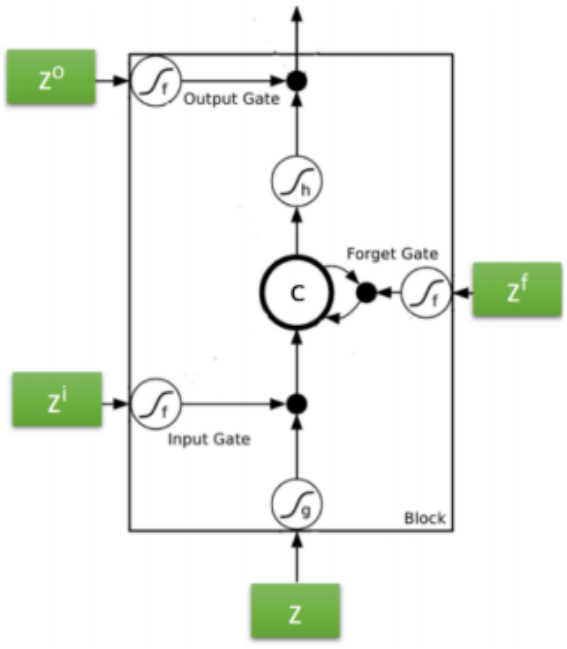


Figure 1: The LSTM cell

根據題目所給公式，做出如下 table:

t	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
x^t	0	1	1	0	0	0	1	1
	1	0	1	1	1	0	1	0
	0	1	1	1	0	1	1	1
	3	-2	4	0	2	-4	1	2
z	**3**	**-2**	**4**	**0**	**2**	**-4**	**1**	**2**
z_i	**90**	**90**	**190**	**90**	**90**	**-10**	**190**	**90**
z_f	**10**	**10**	**-90**	**10**	**10**	**110**	**-90**	**10**
z_o	**-10**	**90**	**90**	**90**	**-10**	**90**	**90**	**90**
$f(z_i)$	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**
g(z)	**3**	**-2**	**4**	**0**	**2**	**-4**	**1**	**2**
$f(z_f)$	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**
c'	**3**	**1**	**4**	**4**	**6**	**6**	**1**	**3**
$f(z_o)$	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**1**
h(c')	**3**	**1**	**4**	**4**	**6**	**6**	**1**	**3**
y^t	**0**	**1**	**4**	**4**	**0**	**6**	**1**	**3**

可得出 **output sequence 為 [0 1 4 4 0 6 1 3]**