# Homework 4 Report - Malicious Comments Identification

資工四 B04902131 黃郁凱

December 21, 2018

1. 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。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。

   (a) RNN

   先將一個句子的每個中文字轉換成 one-hot 的形式，經過 `torch.nn.Embedding()`，會將每個字轉成一個 `Embedding Dimension` 的 vector，接著依序送入 GRU 中，等到 GRU 吃完最後一個字 (<EOS>)，取出其 hidden state 的 vector，flatten 過後送入一連串的 fully connected layer，並 output 一個數值，經過 `sigmoid` 後就是該句子有惡意的機率。模型總參數為 $1.62 \cdot 10^7$ 模型的 public testing accuracy 為 75.2%，private 為 75.1%。

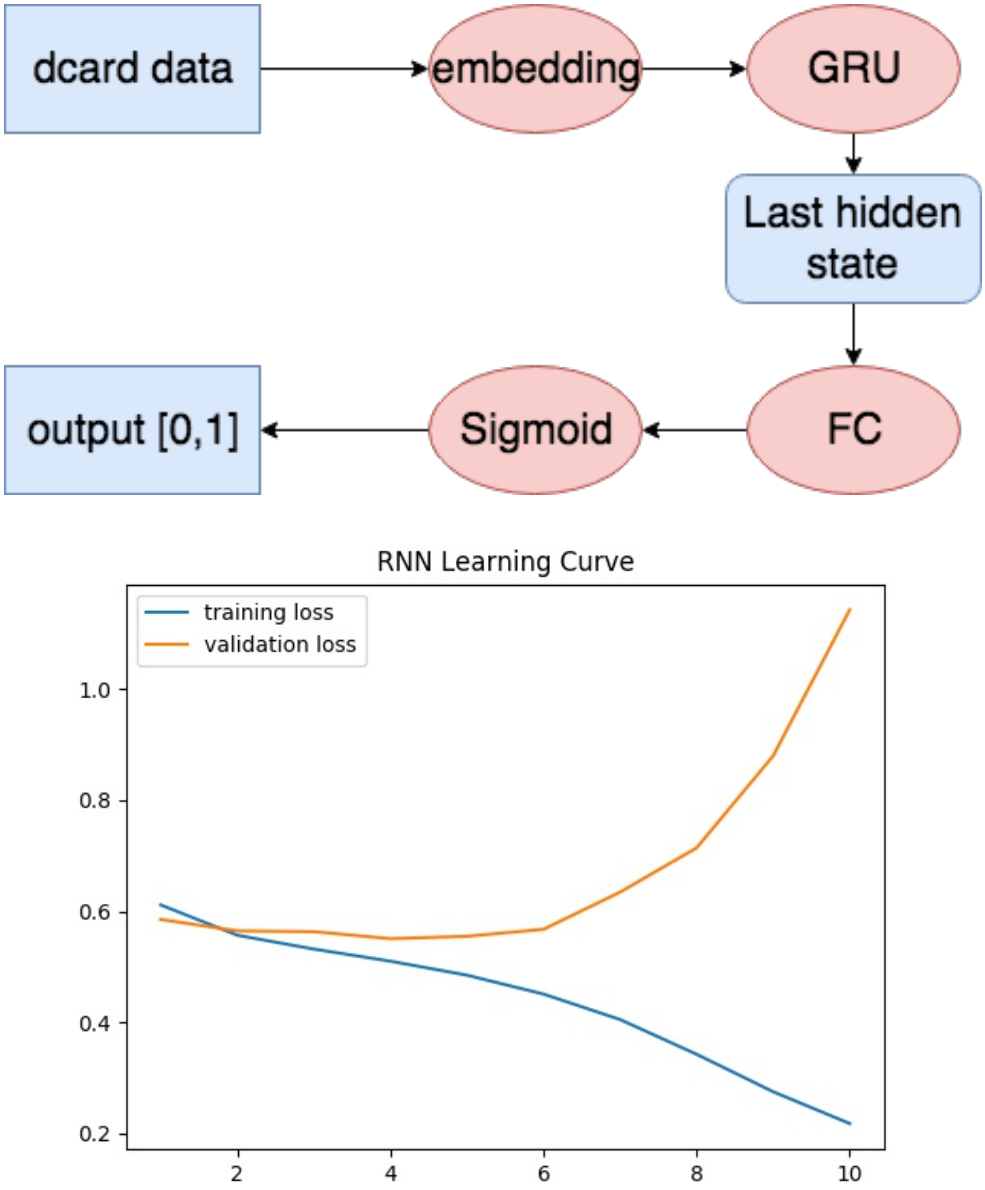



Figure 1: 上圖為 WordEmbedding+RNN 的模型架構，下圖為其 learning curve。

(b) DNN

先將一個句子的每個中文字轉成 one-hot 的形式，將句子內所有的 one-hot vector 相加形成 7000 多維的 integer vector，送入一連串的 fully connected layer，並 output 一個數值，經過 `sigmoid` 後就是該句子有惡意的機率。模型總參數為 $1.64 \cdot 10^7$。模型的 public testing accuracy 為 74.8%，private 為 74.5%。

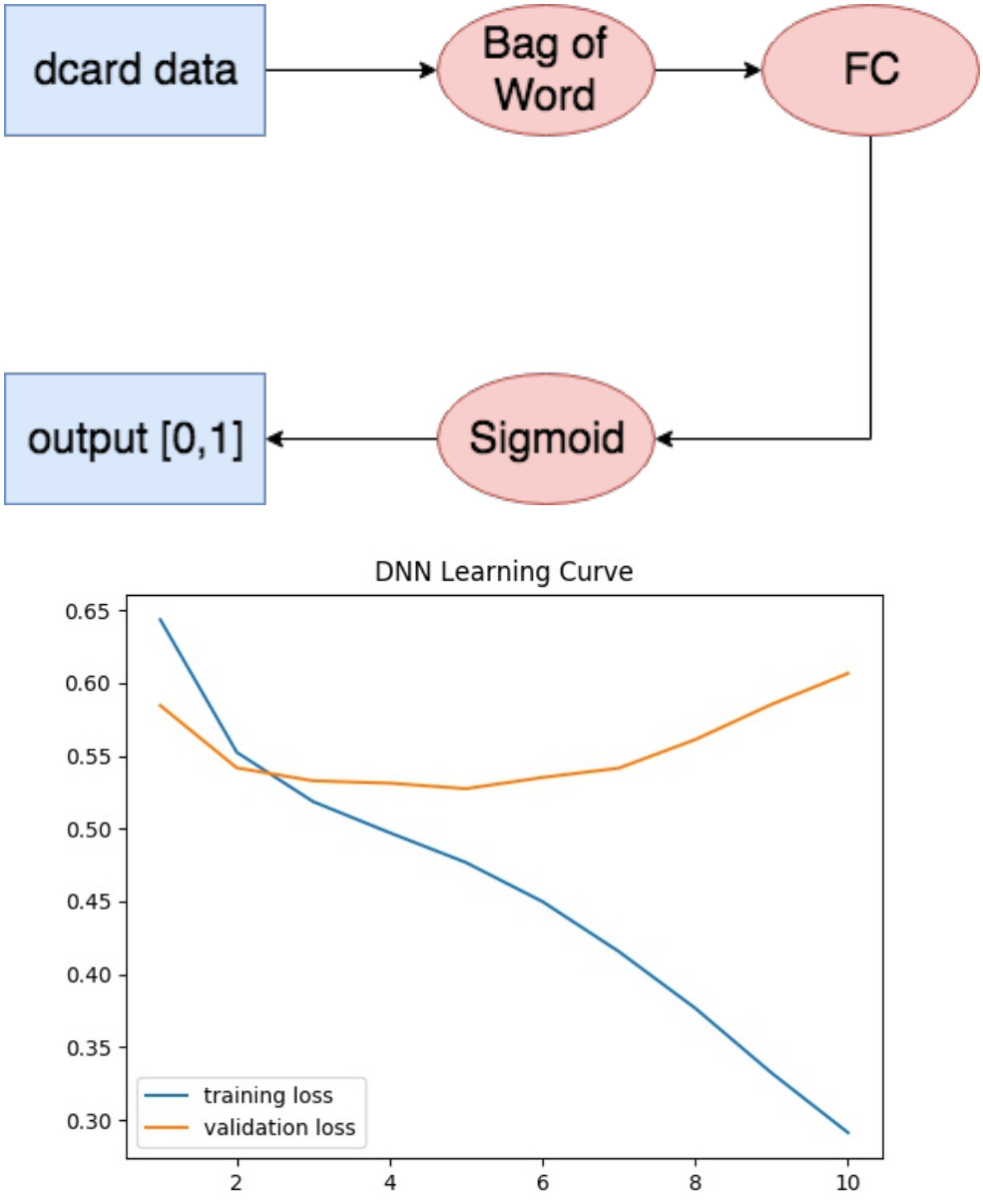



Figure 2: 上圖為 BOW+DNN 的模型架構，下圖為其 learning curve。

2. 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。

   - 使用 mean of word embedding 當作 FC 的 feature input
     因為 word embedding 照理來講可以學到字與字之間關係，因此將所有 word 的 embedding 加起來取平均的向量，可以某個程度上代表這個句子想傳達的意思，例如某個髒話的詞重

複數遍，那個在這個取平均的 vector 就會與那個髒話詞較相近，某種程度上讓 DNN 有更多線索可以判斷是否帶有惡意。正確率從 74.81% 進步到 74.93%。

- 使用 mean of hidden state 當作 FC 的 feature input
  原本只使用了最後一個時間點的 hidden state 當作 FC 的 input，然而取用所有平均的 hidden state 可以傳達更多的訊息，較能代表整個語句的意思；再者，RNN 的一個弱點是經過長時間的 input 會忘掉較先前的段句子傳達的意思，因此會有資訊的損失，在此透過取平均來拿回更多丟失的資訊。正確率從 74.93% 進步到 75.2%。

3. 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。

| 有無斷詞 | public score | private score |
|---|---|---|
| 無 | 75.2 | 75.1 |
| 有 | 73.8 | 73.6 |

Table 1: 我表現最好的 model: RNN，在有無做斷詞下的 testing score 比較結果。

有無斷詞其實效果差不多，正確率不會有太大的影響，而沒有斷詞的效果略為好。造成這樣的原因，可能是斷詞的的方法可能使語意有誤，例如“我們在野生動物園玩”若切成“我們/在野/生動/物/園/玩“，可能使 RNN 判斷語意出現錯誤；再者切詞會使字詞頻率減少，進而影響 model 判斷力，例如“白爛”和“好白爛”會被切成不同的詞，而“好白爛”出現次數相對非常少，RNN model 並無法在少量的 training data 中學會兩者意思是相近的，然而如果每個字切開，那麼 RNN model 先學到“好”有更加的意思，且“白爛”有惡意，所以下次讀到“好白爛”就學到這句是有惡意的句子，相較於 jieba 切詞，將所有字都切開讓 RNN model 有更多自由，可以學到中文真正的意思。

4. 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。

| model | first sentence | second sentence |
|---|---|---|
| RNN | 0 | 1 |
| BOW | 1 | 1 |

RNN model 可以辨別出兩個句子是否惡意，而 BOW 無法。因為 BOW 是統計字詞的出現頻率，語意資訊有時候會丟失，例如此兩句就是例子，同樣在句子中出現白痴兩個字，一者有惡意一者沒有，而 BOW 無法辨別出兩者。RNN model 可以多萃取出字詞順序間的關係，能夠有更多資訊判別語意。

5. 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 $\epsilon_t$, scaling coefficient $\alpha_t$, and the classification function $f_t(x)$. The initial weights $u_1^n$ are set to 1 ($n = 0, 1, \cdots, 9$). 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 | + | - | + | + | + | - | - | + | - | - |

let $sign(x) = \begin{cases} 1 & , \text{if } x \geq 0 \\ -1 & , \text{if } x < 0 \end{cases}$

(a) T = 1

- $u_1 = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$
- $f_1(x) = -sign(x - 5)$
- $\epsilon_1 = \frac{2}{10} = \frac{1}{5}$
- $\alpha_1 = ln\sqrt{\frac{1-\epsilon}{\epsilon}} \approx 0.693$

(b) T = 2

- $u_2 = [0.5, 2, 0.5, 0.5, 0.5, 0.5, 0.5, 2, 0.5, 0.5]$
- $f_2(x) = sign(x - 2)$
- $\epsilon_2 = \frac{0.5+0+0+0+0+0.5+0.5+0+0.5+0.5}{8} = 0.31$
- $\alpha_2 = ln\sqrt{\frac{1-\epsilon}{\epsilon}} \approx 0.394$

(c) T = 3

- $u_3 = [0.742, 1.348, 0.337, 0.337, 0.337, 0.742, 0.742, 1.348, 0.742, 0.742]$
- $f_3(x) = -sign(x - 1)$
- $\epsilon_3 = \frac{0+0+0.337+0.337+0.337+0+0+1.348+0+0}{7.417} = 0.318$
- $\alpha_3 = ln\sqrt{\frac{1-\epsilon}{\epsilon}} \approx 0.381$

Final classifier: $H(x) = sign(\sum_{i=1}^{3} \alpha_i f_i(x)) \approx sign(-0.693sign(x - 5) + 0.394sign(x - 2) - 0.381sign(x - 1))$

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

error = 0.1

6. In this exercise, we will simulate the forward pass of a simple LSTM cell. Figure 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

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

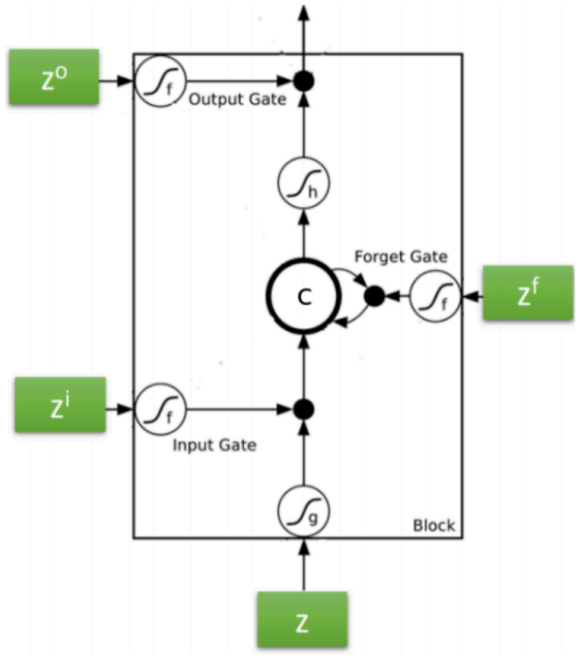
where the value stored in cell memory is updated by

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

Given an input sequence $x^t$ $(t = 1, 2, \cdots, 8)$, 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 |




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

| Time step | $z = g(z)$ | $z_i$ | $z_f$ | $z_o$ | $f(z_i)$ | $cf(z_f)$ | $c' = h(c')$ | $f(z_o)$ | $y$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 90 | 10 | -10 | 1.00 | 0.00 | 3.00 | 0.00 | 0.00 |
| 2 | -2 | 90 | 10 | 90 | 1.00 | 3.00 | 1.00 | 1.00 | 1.00 |
| 3 | 4 | 190 | -90 | 90 | 1.00 | 0.00 | 4.00 | 1.00 | 4.00 |
| 4 | 0 | 90 | 10 | 90 | 1.00 | 4.00 | 4.00 | 1.00 | 4.00 |
| 5 | 2 | 90 | 10 | -10 | 1.00 | 4.00 | 6.00 | 0.00 | 0.00 |
| 6 | -4 | -10 | 110 | 90 | 0.00 | 6.00 | 6.00 | 1.00 | 6.00 |
| 7 | 1 | 190 | -90 | 90 | 1.00 | 0.00 | 1.00 | 1.00 | 1.00 |
| 8 | 2 | 90 | 10 | 90 | 1.00 | 1.00 | 3.00 | 1.00 | 3.00 |