

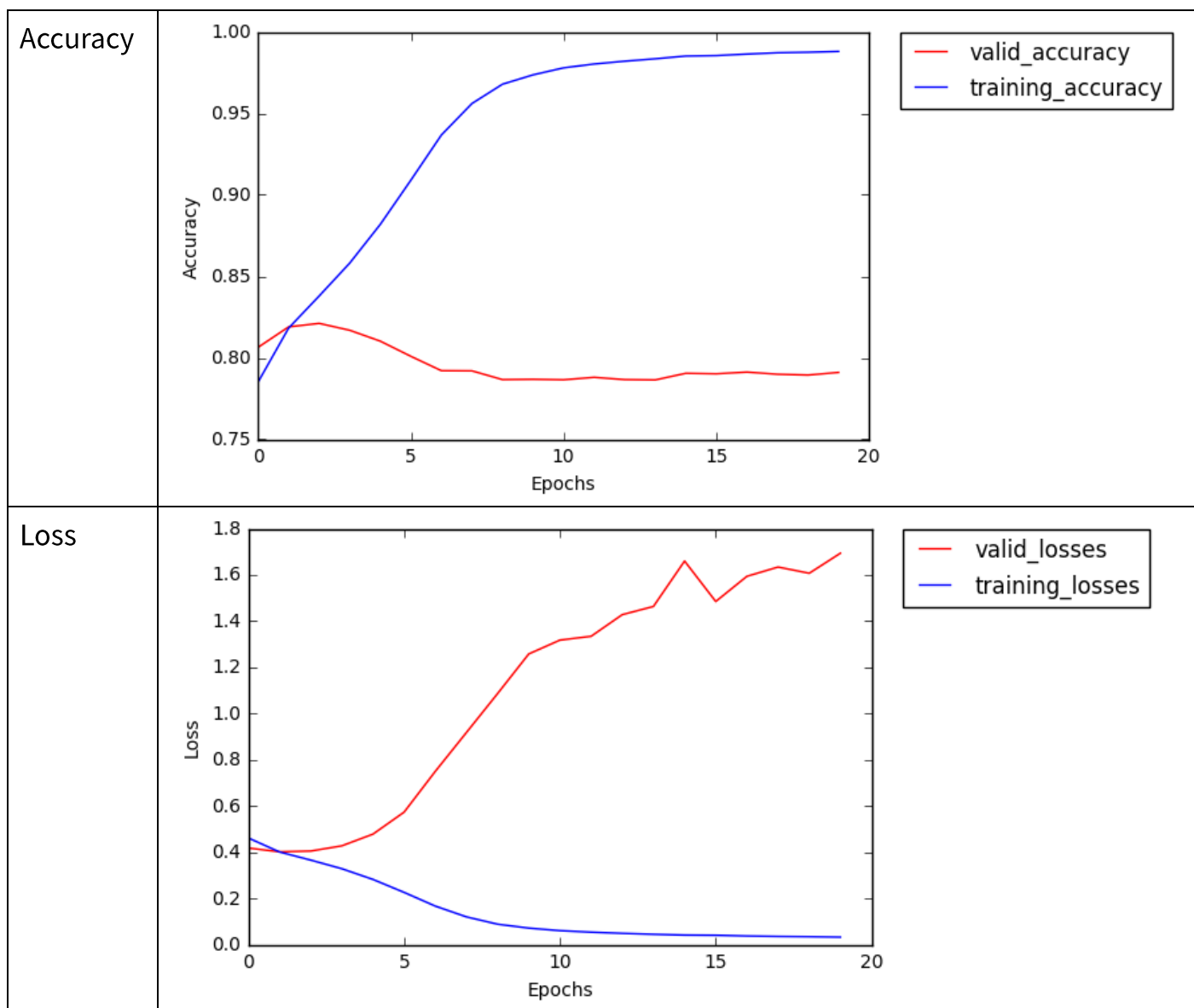
Machine Learning HW4

B04705003 資工三 林子雋

Collaborator：助教提供的 pytorch code、資工三：張立暉提供 gensim word2vec 會讓 performance 變好的意見

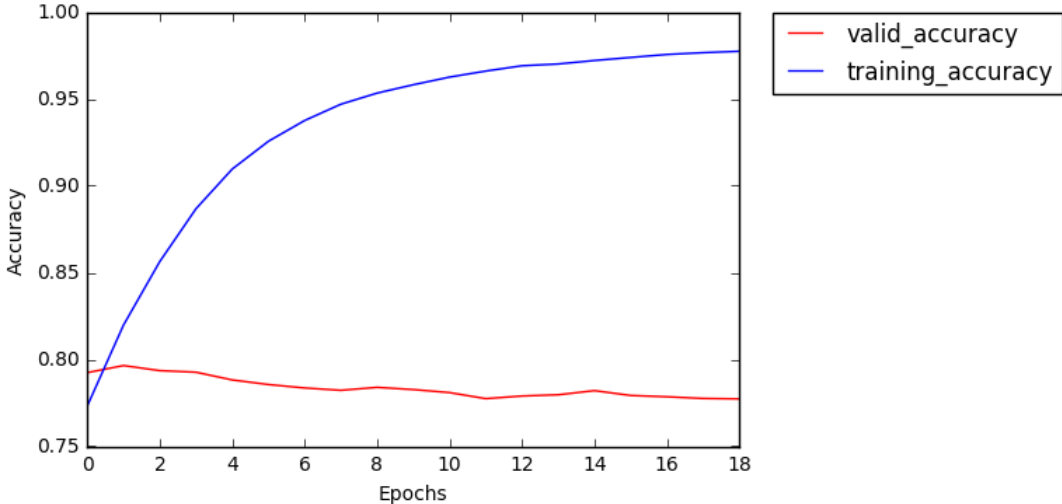
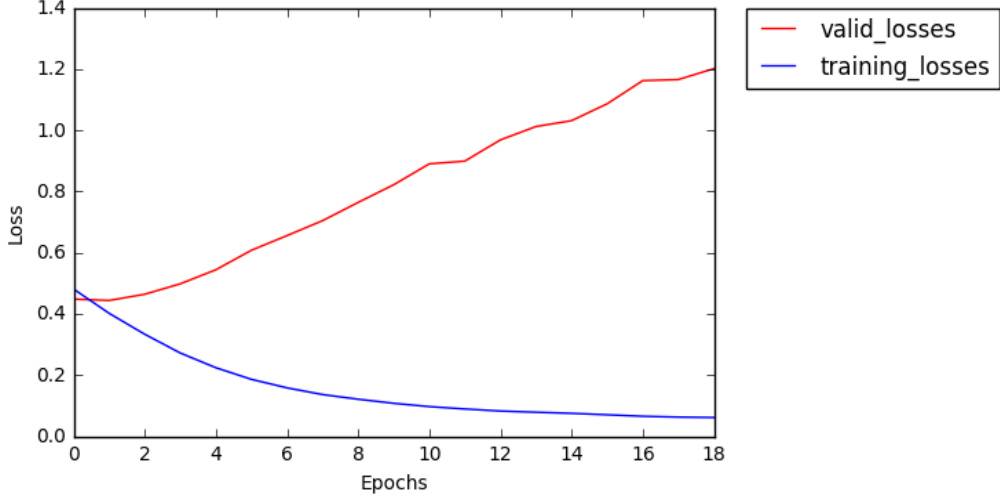
1. (1%) 請說明你實作的 RNN model，其模型架構、訓練過程和準確率為何？

模型架構	<p>模型使用 pytorch 實作：</p> <ol style="list-style-type: none">1. 先讓 encode 好的 sequence 經過 Embedding 取出相對應的 word vector。2. 讓每個 sequence 通過 GRU(雙向)算出 hidden state。3. 讓 hidden state 通過 sequential 的 linear 層，map 到 2 維空間(也就是 label 的維度)
	<pre>DeepJimNetwork ((embedding): Embedding(11464, 100) (gru): GRU(100, 200, dropout=0.4, bidirectional=True) (linear): Sequential ((0): Linear (400 -> 400) (1): Dropout (p = 0.5) (2): Linear (400 -> 2)))</pre>
Epochs	預設 Epochs 是 1000，但是大約 3 個 epoch 就可以關掉了(因為 overfitting 很快)
Batch size	256
Optimizer	<p>使用 RMSProp</p> <p>原因：聽實驗室的學長說用 RMSProp train RNN 系列的會準確率會比較穩定收斂</p>
Loss function	使用 cross entropy loss



2. (1%) 請說明你實作的 BOW model，其模型架構、訓練過程和準確率為何？

模型架構	使用 pytorch 實作 <ol style="list-style-type: none"> 1. 將句子轉成 10000 維的向量 2. 通過 DNN map 到 2 維空間
------	--

	<pre>DNN ((linear): Sequential ((0): Linear (10000 -> 500) (1): ReLU (inplace) (2): Dropout (p = 0.5) (3): Linear (500 -> 500) (4): ReLU (inplace) (5): Dropout (p = 0.5) (6): Linear (500 -> 2)))</pre>																																	
Epochs	1000 個 epoch，但大約第三個 epoch 之後就會 overfit																																	
Batch size	256																																	
Optimizer	RMSProp																																	
Loss function	Cross entropy																																	
Accuracy	 <p>The graph displays training and validation accuracy over 18 epochs. The x-axis represents epochs from 0 to 18, and the y-axis represents accuracy from 0.75 to 1.00. The training accuracy (blue line) starts at approximately 0.78 and rises steadily to about 0.98 by epoch 18. The validation accuracy (red line) starts at approximately 0.79, peaks at epoch 2 with an accuracy of about 0.80, and then gradually declines to approximately 0.78 by epoch 18.</p> <table><tr><th>Epochs</th><th>training_accuracy</th><th>valid_accuracy</th></tr><tr><td>0</td><td>0.78</td><td>0.79</td></tr><tr><td>2</td><td>0.85</td><td>0.80</td></tr><tr><td>4</td><td>0.91</td><td>0.79</td></tr><tr><td>6</td><td>0.94</td><td>0.78</td></tr><tr><td>8</td><td>0.96</td><td>0.78</td></tr><tr><td>10</td><td>0.97</td><td>0.78</td></tr><tr><td>12</td><td>0.97</td><td>0.78</td></tr><tr><td>14</td><td>0.97</td><td>0.78</td></tr><tr><td>16</td><td>0.98</td><td>0.78</td></tr><tr><td>18</td><td>0.98</td><td>0.78</td></tr></table>	Epochs	training_accuracy	valid_accuracy	0	0.78	0.79	2	0.85	0.80	4	0.91	0.79	6	0.94	0.78	8	0.96	0.78	10	0.97	0.78	12	0.97	0.78	14	0.97	0.78	16	0.98	0.78	18	0.98	0.78
Epochs	training_accuracy	valid_accuracy																																
0	0.78	0.79																																
2	0.85	0.80																																
4	0.91	0.79																																
6	0.94	0.78																																
8	0.96	0.78																																
10	0.97	0.78																																
12	0.97	0.78																																
14	0.97	0.78																																
16	0.98	0.78																																
18	0.98	0.78																																
Loss	 <p>The graph displays training and validation loss over 18 epochs. The x-axis represents epochs from 0 to 18, and the y-axis represents loss from 0.0 to 1.4. The training loss (blue line) starts at approximately 0.45 and decreases steadily to about 0.05 by epoch 18. The validation loss (red line) starts at approximately 0.45, remains relatively flat until epoch 4, and then increases steadily to about 1.2 by epoch 18.</p> <table><tr><th>Epochs</th><th>training_losses</th><th>valid_losses</th></tr><tr><td>0</td><td>0.45</td><td>0.45</td></tr><tr><td>2</td><td>0.35</td><td>0.45</td></tr><tr><td>4</td><td>0.25</td><td>0.55</td></tr><tr><td>6</td><td>0.18</td><td>0.65</td></tr><tr><td>8</td><td>0.12</td><td>0.75</td></tr><tr><td>10</td><td>0.08</td><td>0.90</td></tr><tr><td>12</td><td>0.06</td><td>1.00</td></tr><tr><td>14</td><td>0.05</td><td>1.05</td></tr><tr><td>16</td><td>0.05</td><td>1.15</td></tr><tr><td>18</td><td>0.05</td><td>1.20</td></tr></table>	Epochs	training_losses	valid_losses	0	0.45	0.45	2	0.35	0.45	4	0.25	0.55	6	0.18	0.65	8	0.12	0.75	10	0.08	0.90	12	0.06	1.00	14	0.05	1.05	16	0.05	1.15	18	0.05	1.20
Epochs	training_losses	valid_losses																																
0	0.45	0.45																																
2	0.35	0.45																																
4	0.25	0.55																																
6	0.18	0.65																																
8	0.12	0.75																																
10	0.08	0.90																																
12	0.06	1.00																																
14	0.05	1.05																																
16	0.05	1.15																																
18	0.05	1.20																																

3. (1%) 請比較 bag of word 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"這兩句的情緒分數，並討論造成差異的原因。

預測出的機率分布	today is a good day, but it is hot		today is hot, but it is a good day		today is a day, but it is hot		today is a good day, it is hot.	
Bag of Word 方法	正面	負面	正面	負面	正面	負面	正面	負面
	0.5469	0.4531	0.5469	0.4531	0.4239	0.5761	0.8179	0.1821
RNN 方法	正面	負面	正面	負面				
	0.3588	0.6412	0.9955	0.0045				
差異原因	<p>1. 第一句的經過 bag of word 的轉換之後，都會是一模一樣的 vector，所以兩句被判斷成同樣的機率分布是很自然的。另外，我作了拔除 " but " 或 " good " 的實驗，發現到，bag of words 當中，good 對預測結果的影響非常大，推測是因為訓練過程中，good 被標記成正面的 data 比較多，造成 BOW 比較容易因為這樣而誤判。</p> <p>2. 而 RNN 的方法能夠正確分類是因為順序性的資訊有被包含在模型的訓練，因此語氣轉折的句子比較容易被判斷出來。</p>							

4. (1%) 請比較"有無"包含標點符號兩種不同 tokenize 的方式，並討論兩者對準確率的影響，並討論原因。

	「有」標點符號	「無」標點符號
正確率	0.820475	0.808275

討論：

在不調整其他參數，只去調整是否有標點符號這個變因時，會發現到「有」標點符號的情況下準確率會比較高，推測是因為標點符號在 twitter 當中，也是包含重要的語氣資訊(例如： " ! " 符號)

5. (1%) 請描述在你的 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，並討論原因。

	「有」 semi-supervised	「無」 semi-supervised
標記方法	<ol style="list-style-type: none">1. Loop：先用 training data 訓練好一個模型，用這個模型去標信心機率分布有一項大於 0.9 的 data 之後(也就是對某一個 label 的信心大於 0.9)，拿這些新標好的 data 放入 training data 中(並且將新標得 data 從沒有 no label training data 移除)。2. 重複 Loop 三次	0.815175
第一個 iteration 的 testing accuracy	0.815175	
第二個 iteration 的 testing accuracy	0.8129	
第三個 iteration 的	0.8136	

testing accuracy		
<p>討論：</p> <p>很明顯發現使用 semi-supervise 之後，準確率並沒有因此顯著提升，反倒下降，推測是因為新標進來的 label 仍然不夠準確，因此準確率並沒有變得更好。</p>		