Machine Learning Spring 2020 - HW4 Report

學號: B07902064 系級: 資工二 姓名: 蔡銘軒

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) 和準確率為何？(盡量是過 public strong baseline 的 model)

   以下是我word2vec/embedding使用的參數：

   ```
   embedding size: 200
   hidden size: 100
   window size: 5
   min count: 5
   iteration = 8
   sg = 1
   sentence length: 40
   ```

   我觀察到word2vec對model的影響很大。

   首先是embedding跟hidden size，我從兩個都是250開始，試著把他們調大，但表現都沒有明顯提升，甚至略為下降，因此我反過來將他們降低，發現表現開始提升，最後決定使用embedding size = 200以及hidden size = 100

   另外iteration是我觀察到影響最大的。原先validation set的正確率在各種調整下大約在78%~80%左右浮動，我試著把iteration設為15，卻發現無論是training set還是validation set的表現都驟降到50%左右且觀察幾個epoch後都沒有提升的跡象。最後我發現設為8~10之間會得到比較正常的結果。

   另外sentence length也有所影響，當我從20提升到40時，validation set上的正確率也提升了1~2%左右。

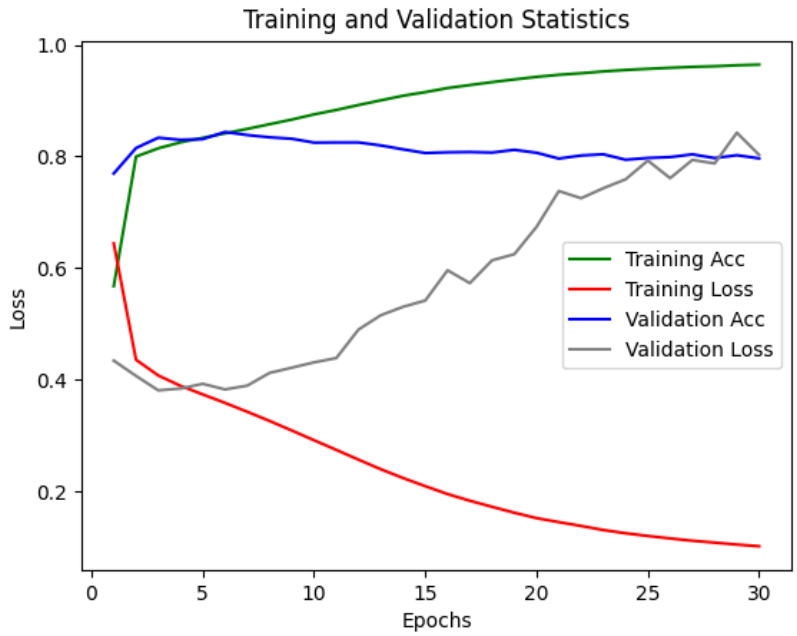
   以下是我的RNN(LSTM)使用的參數：

   ```
   nn.LSTM(input_size = 200, hidden_size = 100, layers = 2, batch_first =
   True)
   nn.Sequential(
     nn.Dropout(p = 0.5),
     nn.Linear(100, 1),
     nn.Sigmoid(),
   )
   ```

   hidden size我發現調到200以上的時候沒有得到更好的結果，因此降低到100，反而表現有些微提升。

   layers方面經過與同學討論/上網查資料後發現通常LSTM疊兩層會有不錯的結果。在疊了兩層後還可以使用LSTM本身的Dropout參數，但我使用後發現表現沒有比較好，因此最後還是沒有使用。

   我還有嘗試過GRU，GRU的訓練時運算速度比LSTM還要快（可能因為參數量較少），表現上也比較早達到最高點，但最終的正確率我認為還是略低於LSTM，因此最後還是使用LSTM。



從訓練過程中可以發現，validation set的Accuracy在很早的時候（約4~5個epoch）就已經達到最高點了，再往後train下去只有training set的表現不斷變好，validation set反而開始變糟，Loss甚至高達0.8。因此實際train的時候我通常只會跑10個epoch然後用validation set上表現最好的那個作為最終的model。

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。

   sentence 1 ⟸ "today is a good day, but it is hot"

   sentence 2 ⟸ "today is hot, but it is a good day"

	RNN	**BOW+DNN**
sentence 1	0.2934	0.5628
sentence 2	0.6401	0.5948

   對RNN的model來說，兩個句子的分數是有差別的。第一句比較負面，第二句比較偏正面。這樣的預測還滿符合理解的。通常"but"代表語氣的轉折，第一句先說"good day"再接"but"，所以重點應該在後半部比較負面的部分；第二句則相反，重點在"good day"。

   對BOW+DNN而言，兩句話分數是差不多的。理論上這兩句話的字詞組成都一樣，而BOW不考慮他們的順序，所以分數應該要一樣。這邊有差異的原因是我處理句子的時候使用空白來切割單字，所以第一句話裡會有一個"day,"的單字，而第二句話則是"hot,"。所以差異來自逗點出現在不同單字後面，造成他們的組成其實不一樣。

   為了驗證BOW不考慮順序，我將兩句話裡的逗點跟單字分開，讓兩句話的單字組成完全相同。得到的分數都是0.6101，符合期待的結果。

3. (1%) 請敍述你如何 improve performance（preprocess、embedding、架構等等），並解釋為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）

先前的Linear regression、Logistic regression、CNN等model，通常在經過validation set篩選出較好的model之後會再把所有training data拿來train一次。但因為RNN的訓練過程比較曲折，隨時可能大起大落，所以需要用validation set來監測目前training的情況來做Early stopping。但這樣就失去了一些可以用的training data，我覺得很可惜。因此我用同樣的model架構，用五個不同的random seed切出不同的validation set分別訓練，最後判斷時讓這5個model做voting，並用多數決的方式決定最終的答案，這樣比較不會讓某些資料沒有使用到。

在Preprocess的部分，我有試著把一些縮寫的地方還原，例如n't→not，等，但沒有得到更好的結果，因此最後還是沒有使用。

基本上我並沒有對model做什麼變動（我有試著做了許多調整但都沒有得到更好的表現），只是使用ensemble的方式讓training data發揮更大的效用。因為單一model缺少部分training data的資訊，而ensemble的model幾乎都有使用到，我認為切validation set來比較正確率不太公平，因此我使用Kaggle上public score來比較兩者的表現：

	Single model	**Ensemble**
Accuracy(Public data set)	0.82347	0.82821

後者提高了大約0.5%的正確率。雖然數字上看起來沒有提升很多，但在這次的作業裡要在public scoreboard上提升0.1%的正確率我認為就很不容易了。而且使用ensemble通常也有助於避免overfitting，我認為在可觀測的範圍內（除了private data set的Accuracy），使用ensemble是有效提昇表現的。

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。

   我首先在labeled data上train出一個在validation set上表現較好的model，對unlabeled data進行預測，然後根據$\min(Prediction, 1 - Prediction)$來取前10萬筆資料加入training set。我進行這樣的操作3次，最後在validation set上得到結果如下：

	Supervised	**Semi-supervised**
Accuracy	0.8390194869479061	0.7989044189453125

我發現validation set上的正確率其實是呈現下降的趨勢。我有實驗讓semi-supervised進行更多輪，到第六、七輪以後Acc甚至降到70%以下。我認為比起製造更多training data的效果，model本身錯誤的預測影響了training比較多。而且隨著每一輪用了預測錯誤的data來訓練，這些錯誤又被放得更大。