**Machine Learning Fall 2020 ——— Homework 4**

學號：B07902037 系級： 資工三 姓名：蔡沛勳

**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\*。**

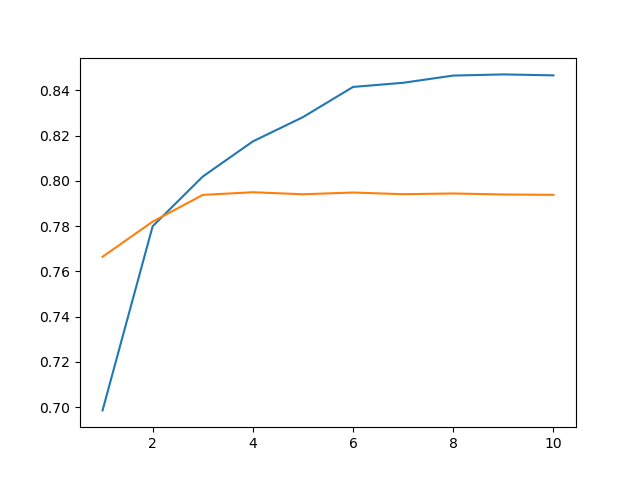
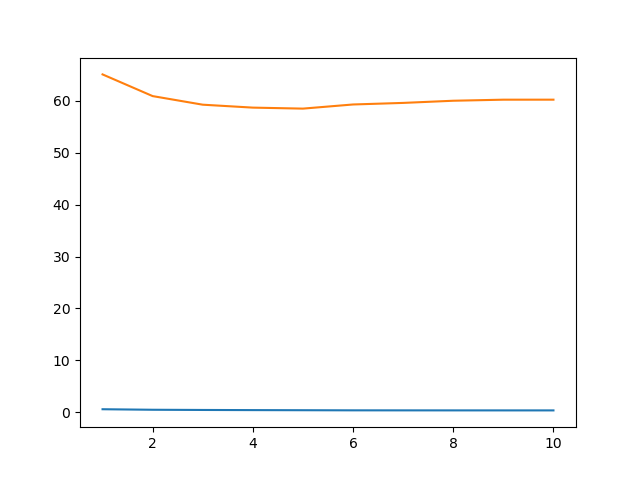
RNN主要透過 torch.nn 中的 lstm 來實作，具體程式如下:

|  |
| --- |
| class RNN(nn.Module):  def \_\_init\_\_(self, embedding, hidden\_size, bidirectional):  super(RNN, self).\_\_init\_\_()  self.embedding = nn.Embedding(embedding.size(0),embedding.size(1))  self.embedding.weight = nn.Parameter(embedding)  self.embedding.weight.requires\_grad = True    self.lstm = nn.LSTM(input\_size = 250,  hidden\_size = hidden\_size,  num\_layers = 5,  batch\_first = True,  dropout = 0.5,  bidirectional = bidirectional  )  self.fc = nn.Sequential(  nn.Dropout(p = 0.5),  nn.Linear(in\_features = 2 \* hidden\_size if bidirectional else hidden\_size,  out\_features = 1),  )  self.sigmoid = nn.Sigmoid()  def forward(self, inputs):  x = self.embedding(inputs)  out, \_ = self.lstm(x, None)  out = out[:,-1,:]  outputs = self.fc(out)  return self.sigmoid(outputs) |

Word embedding 則使用 gensim.models.word2vec 的 Word2Vec來實作，參數如下

|  |
| --- |
| Word2Vec(data, size = 250, iter = 3, window = 5, min\_count = 3, sg = 1) |

此 model 的 training / validation history 如下，其中左圖為accuracy history，右圖為 loss history。 (藍線為training data，橘線為validation data，比例為 0.8 : 0.2)。

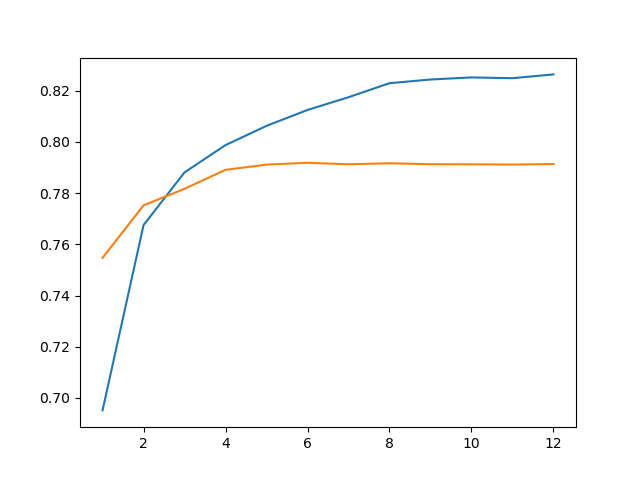
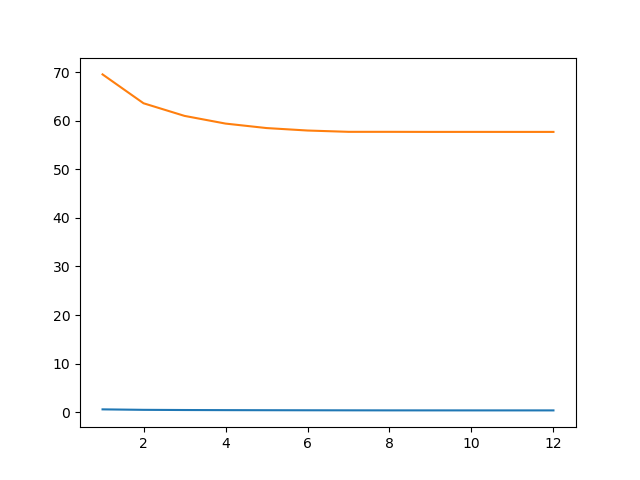
在 kaggle 上的最高分為 0.78340。

**2. (0.5%) 請實作 BOW+DNN 模型，敘述你的模型架構，回報模型的正確率並繪出訓練曲線\*。**

我實作的DNN包含兩層Linear架構，具體程式如下

|  |
| --- |
| class DNN(nn.Module):  def \_\_init\_\_(self, wnum):  super(DNN, self).\_\_init\_\_()  self.wnum = wnum # words number  self.fc = nn.Sequential(  nn.Linear(in\_features = wnum, out\_features = 64),  nn.LeakyReLU(negative\_slope = 0.1),  nn.Dropout(p = 0.5),  nn.Linear(in\_features = 64, out\_features = 1)  )  self.sigmoid = nn.Sigmoid()  def forward(self, inputs):  x = inputs.float()  outputs = self.fc(x)  return self.sigmoid(outputs) |

此 model 的 training / validation history 如下，其中左圖為accuracy history，右圖為 loss history。 (藍線為training data，橘線為validation data，比例為 0.8 : 0.2)

** **

在 kaggle 上的最高分為 0.78480。

預期中BOW+DNN表現應比RNN(LSTM)要差，因為BOW+DNN忽略的語序的影響。而我的結果不如預期的可能原因為RNN沒訓練好或語序對本次測資的影響較低，所以兩者表現相差無幾。

**3. (0.5%) 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。**

preprocess:我對設定Word2Vec的參數min\_count增加到 7，所以可以過濾掉部分錯字或罕見單字。調整windows大小使一些距離較遠但依舊具有關聯性的單字可以被訓練到。

embedding: 調整句子長度使model的判斷資料增加，但也可能導致部分句子中<PAD>太多導致結果變差。

lstm model: biderectional增加反向序列的測資以增進其表現。

**4. (0.5%) 請比較 RNN 與 BOW 兩種不同 model 對於 "Today is hot, but I am happy" 與 "I am happy, but today is hot" 這兩句話的分數（model output），並討論造成差異的原因。**

|  |  |  |
| --- | --- | --- |
|  | today is hot , but i am happy | i am happy , but today is hot |
| RNN | 0.71982 | 0.34323 |
| BOW+DNN | 0.66404 | 0.66404 |

預期的狀況為"Today is hot, but I am happy"會大於0.5而"I am happy, but today is hot"會小於0.5。而對BOW+DNN model兩句話的輸入是相同的，故輸出也會相同。RNN model 則會考量語序而產生出與預期較接近的結果。

**5. (3%) Refer to math problem**

1.

(a).

設10個點依序為，可算出 為

可被正交對角化得到 ，可算出 為

得到 principle axes 依 eigenvalue 由大到小依序為

(b).

由上題得到

代入   得到

(c).

2.

(a).

Symmetric:

positive semi-definite:

share the same eigenvalues:

設為 的 eigenvalue，為對應的 eigenvector，得

兩邊同乘 得

可看出為 的 eigenvalue，為對應的 eigenvector

設為 的 eigenvalue，為對應的 eigenvector，得

兩邊同乘 得

可看出為 的 eigenvalue，為對應的 eigenvector

(b).

已知 為半正定對稱矩陣，故可由 Cholesky decomposition得

設 為

可堆得mean及covariance為

故我們可設，並推得其mean及covariance為

(c).

因為為symmetric，故可被正規對角化。設的eigenvalue為。設為 ，由 得到 為一組正交向量。設 可使得 為一組正交基底。

由得到 為一組對應的eigenvalue及eigenvector。而由得到 為一組對應的eigenvalue及eigenvector。

設 。By Von Neumann’s Inequality，得到

因此得 lower bound 為

因為為symmetric，故可被正規對角化為，其中

，。設 且 。可推得

而 為 的最小值，故 即為本題解。