學號：R05921069 系級： 電機碩一 姓名：黃武昱

(1%)請問 softmax 適不適合作為本次作業的 output layer? 寫出你最後選擇的 output layer 並說明理由。

這次的作業並不適合使用 softmax, 因為 softmax 是一個輸入資料對應一個輸出資料的分類方法， 但是, 在本次作業是一個輸入資料會對應到多個輸出資料的分類, softmax 在這方面就不行使用了, 這次作業最適合使用的是 sigmoid 這個函數, 因為他在最後一層會把輸出的所有可能結果以機率來顯示， 故可以得到多個輸出的效果.

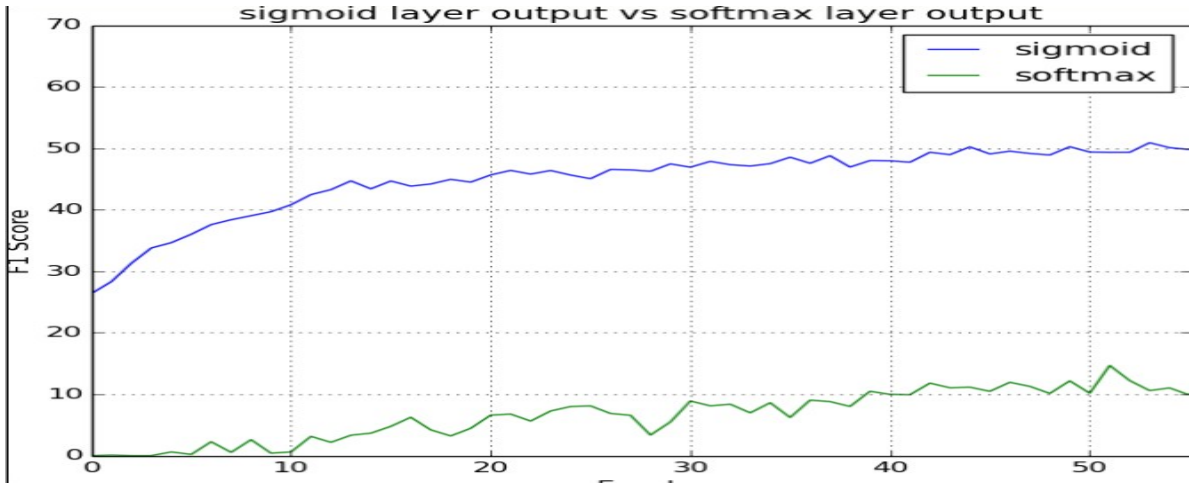
(1%)請設計實驗驗證上述推論。

我使用的參數跟模型是：
split_ratio = 0.1
embedding_dim = 100
nb_epoch = 1000
batch_size = 64
max_article_length = 190
range_value = 0.4

```
model.add(Embedding(num_words,
                    embedding_dim,
                    weights=[embedding_matrix],
                    input_length=max_article_length,
                    trainable=False))

model.add(GRU(128,activation='tanh', dropout=0.4))
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.4))
```

只有在輸出層的方面改變 activation 的函數




由上圖表可得知, 當使用 sigmoid 時的 f1 score 遠比使用 softmax 來的高.

(1%)請試著分析 tags 的分布情況(數量)。

```
SCIENCE-FICTION: 959
SPECULATIVE-FICTION: 1448
FICTION: 1672
NOVEL: 992
FANTASY: 773
CHILDREN'S-LITERATURE: 777
HUMOUR: 18
SATIRE: 35
HISTORICAL-FICTION: 137
HISTORY: 40
MYSTERY: 642
SUSPENSE: 318
ADVENTURE-NOVEL: 109
SPY-FICTION: 75
AUTOBIOGRAPHY: 51
HORROR: 192
THRILLER: 243
ROMANCE-NOVEL: 157
COMEDY: 59
NOVELLA: 29
WAR-NOVEL: 31
DYSTOPIA: 30
COMIC-NOVEL: 37
DETECTIVE-FICTION: 178
HISTORICAL-NOVEL: 222
BIOGRAPHY: 42
MEMOIR: 35
NON-FICTION: 102
CRIME-FICTION: 368
AUTOBIOGRAPHICAL-NOVEL: 31
ALTERNATE-HISTORY: 72
TECHNO-THRILLER: 18
UTOPIAN-AND-DYSTOPIAN-FICTION: 11
YOUNG-ADULT-LITERATURE: 288
SHORT-STORY: 41
GOTHIC-FICTION: 12
APOCALYPTIC-AND-POST-APOCALYPTIC-FICTION: 14
HIGH-FANTASY: 15
```

(1%)本次作業中使用何種方式得到 word embedding?請簡單描述做法。

Word embedding 的意思是：給出一個文檔, 文檔就是一個單詞序列, 希望對文檔中每個不同的單詞都得到一個對應的向量表示.

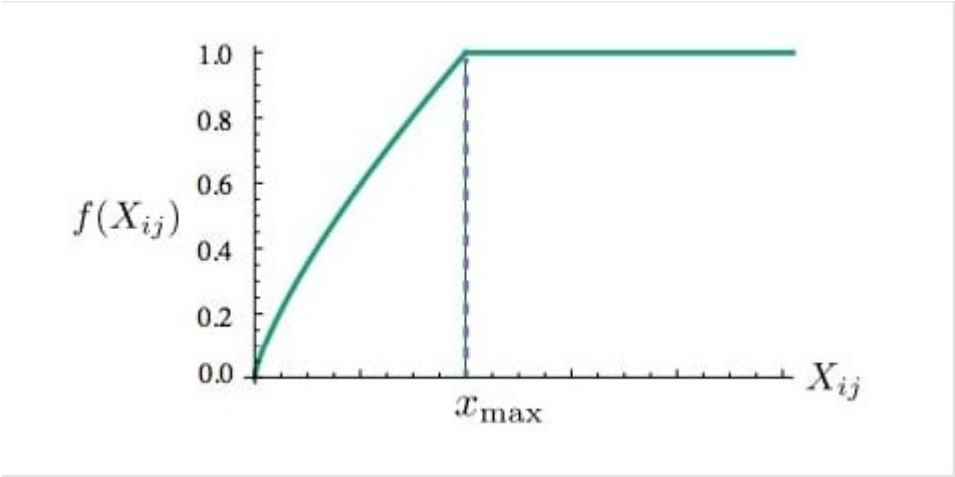
這次的作業是使用 Glove 的 100 維度的 Word embedding 來做處理, Glove 的作法是判別語意距離相近的詞, 共出現的次數, 語意距離遠的詞貢獻次數少, 來做機率的處理, Glove 定義了一個 weighted square loss：

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{ij}))^2$$

其加權函數 f 為：

$$f(x) = \begin{cases} (x/x_{max})^\alpha & if\ x < x_{max}, \\ 1 & otherwise \end{cases}$$

其圖像為：




意思是，在共現超閾值後，其 loss 的權重维持在 1.0 的不變的水平。當然，这也横生了两个參數：$x_{max}$ 和 $\alpha$。

GloVe 给的参考值分别为 $x_{max}=100, \alpha=0.75$。

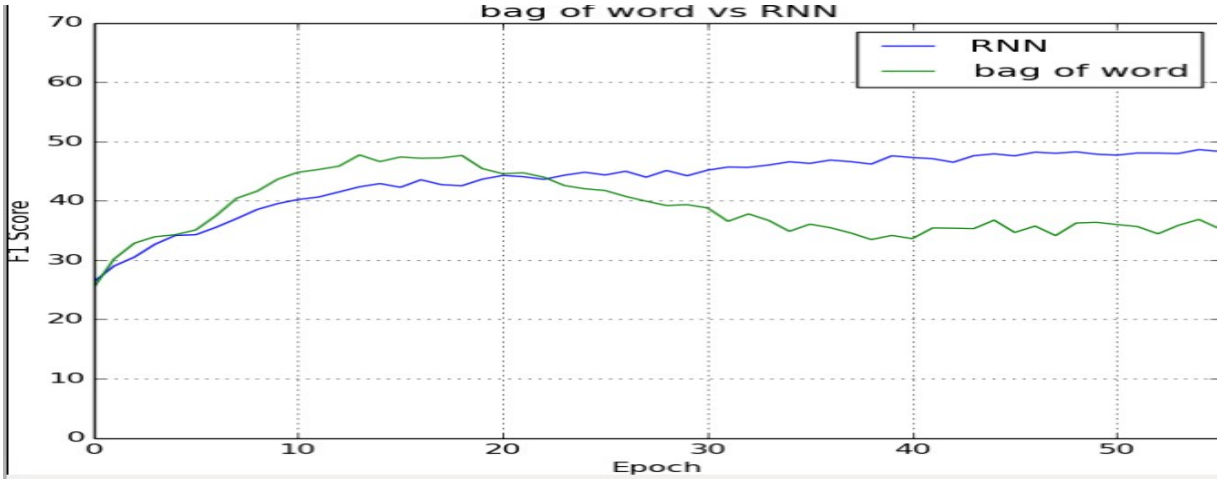
(1%)試比較 bag of word 和 RNN 何者在本次作業中效果較好。

```
model.add(GRU(128,activation='tanh', dropout=0.4))
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.4))
```

上圖為建構 RNN 的網路模型

```
model = Sequential()
model.add(Dense(128,input_shape=X_train.shape[1:],activation='tanh'))
model.add(Dropout(0.4))
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.4))
```

上圖為建構 bag of word 的網路模型




由上圖的 f1_score 比較表可得知, 雖然 bag of word 在一開始時的上升速度略高於 RNN, 但在後續 RNN 還是很持續穩定的上升, 但 bag of word 卻是持續的下降.