Report

電機四 陳緯哲 B03901109

1. (1 %)請比較有無 normalize 的差別。並說明如何 normalize.

	有 normalize	無 Normalize
Public	0.86740	0.93007
private	0.86127	0.92696

我在 Dot 層設定 Normalize 參數為 True，以達到 Normalize 的效果，該效果在 keras 的說明文檔中為[Whether to L2-normalize samples along the dot product axis before taking the dot product. If set to True, then the output of the dot product is the cosine proximity between the two samples.]，也就是說先對兩個向量進行 normalize，再進行 dot；最後發現有進行 normalize 的表現較沒有進行的還要好。

2. (1 %)比較不同的 embedding dimension 的結果。

	Dimension=64	Dimension=128
Public	0.86740	0.86881
private	0.86127	0.86233

Dimension 設定 64 或是 128 並沒有太大差異，在 public set 與 private set 間的表現也有些許差異，兩者的差別或許不是太大。

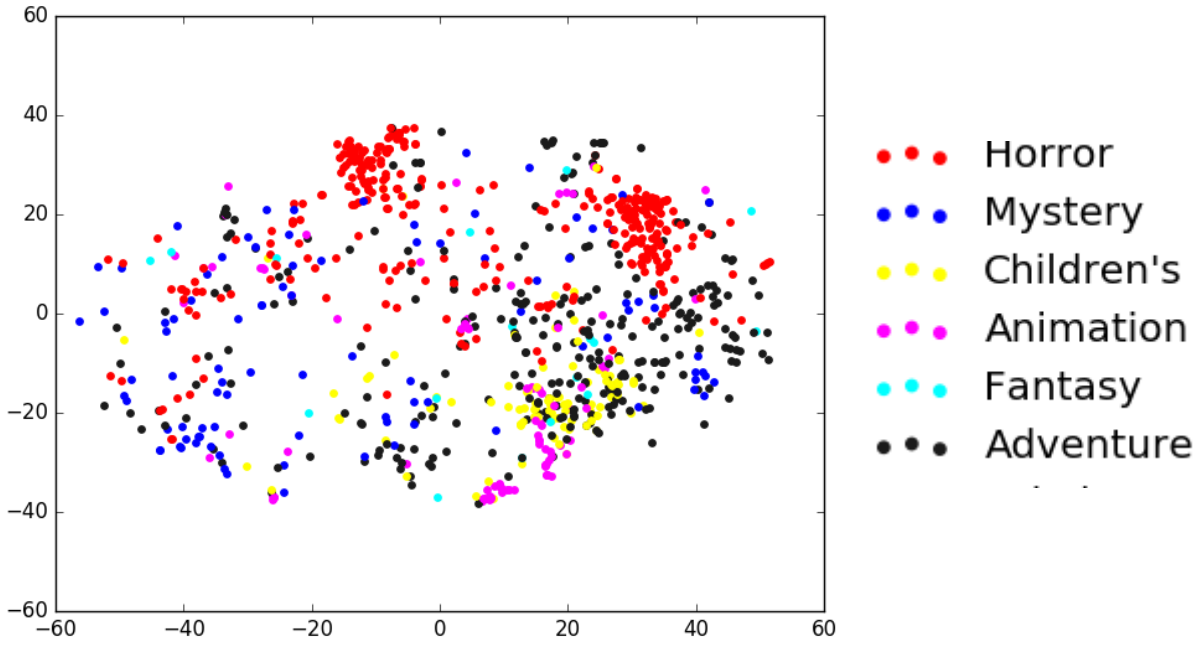
3. (1 %)比較有無 bias 的結果。

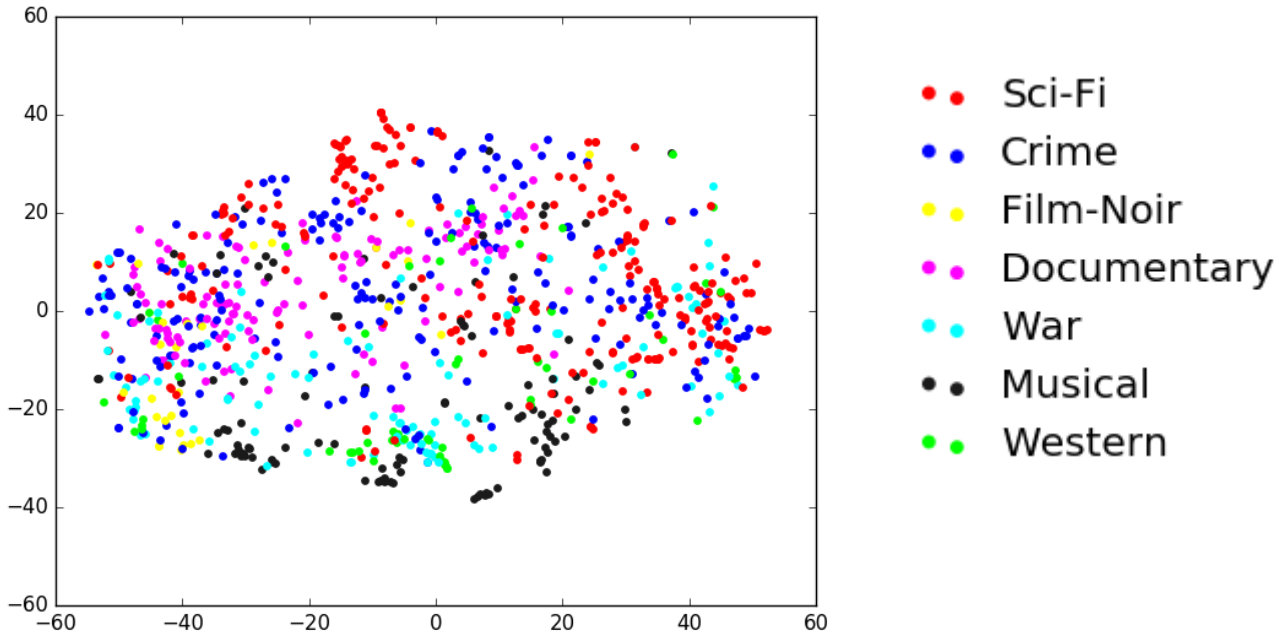
	有 bias	無 bias
Public	0.86740	2.80838
private	0.86127	2.82431

如同課堂上所說的，加上 bias 之後對於預測的表現有了大幅改善。

4. (1 %)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。

因為 label 的種類太多了，因此以下分為兩張圖來進行表示，而在進行 label 的過程中，也發現數量十分龐大的 3 個 label 種類，其分佈過於平均，沒辦法看出什麼趨勢，反而還會影響判讀，因此將這三個 lable：Comedy、Thrillered、Romance 拿掉，便得到以下兩張圖，雖然分群上並不是分得十分乾淨，但是可以觀察到每一個分群都有自己的趨勢與分佈。





5. (1 %)試著使用除了 rating 以外的 feature, 並說明你的作法和結果，結果好壞不會影響評分。

除了使用 rating 之外，我也將電影的種類利用 One hot encoding 作為 feature，因為每部電影可能身兼兩種以上的類別，因此利用 one hot encoding 可以有效率的表達 feature，首先讓每部電影都有一個 dimension 為 18 的 vector(電影一共有 18 個種類)，若該部電影屬於某個種類，便將對應的 feature 設為 1，否則設為 0，再將得到的 feature 利用 DNN 訓練成 64-dimension 的 feature，並與 user 的 feature 進行 dot，再跟其他 dot 完的結果相加，即為預測的 rating。
架構如下：

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
input_3 (InputLayer)	(None, 18)	0	
embedding_1 (Embedding)	(None, 1, 64)	386560	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 64)	252928	input_2[0][0]
dense_1 (Dense)	(None, 64)	1216	input_3[0][0]
flatten_1 (Flatten)	(None, 64)	0	embedding_1[0][0]
flatten_2 (Flatten)	(None, 64)	0	embedding_2[0][0]
embedding_4 (Embedding)	(None, 1, 1)	3952	input_2[0][0]
embedding_3 (Embedding)	(None, 1, 1)	6040	input_1[0][0]
batch_normalization_1 (BatchNor	(None, 64)	256	dense_1[0][0]
dot_1 (Dot)	(None, 1)	0	flatten_1[0][0] flatten_2[0][0]
flatten_4 (Flatten)	(None, 1)	0	embedding_4[0][0]
flatten_3 (Flatten)	(None, 1)	0	embedding_3[0][0]
dot_2 (Dot)	(None, 1)	0	flatten_1[0][0] batch_normalization_1[0][0]
add_1 (Add)	(None, 1)	0	dot_1[0][0] flatten_4[0][0] flatten_3[0][0]

此訓練模型的表現為 0.86913(private)、0.87578(Public)。