學號：R06921082 系級：電機碩一 姓名：陳與賢

1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.
   在其他 training 條件（epoch...等）皆相同的條件下
   無 normalize=> Public: 0.86289 / Private: 0.86598
   使用助教投影片有提到的減掉平均再除標準差 => Public: 0.86112 / Private: 0.86452
   使用 normalize 後分數有提高一點點，但並不顯著。

2. (1%)比較不同的 latent dimension 的結果。
   在其他 training 條件（epoch...等）皆相同的條件下
   dimension = 120 => Public: 0.86289 / Private: 0.86598
   dimension = 240 => Public: 0.86835 / Private: 0.87112
   dimension = 360 => Public: 0.88784 / Private: 0.88059
   可以發現到 dimension 愈高似乎效果愈差，或者是在我的 training 參數下 120 這個數字剛好效果最佳。

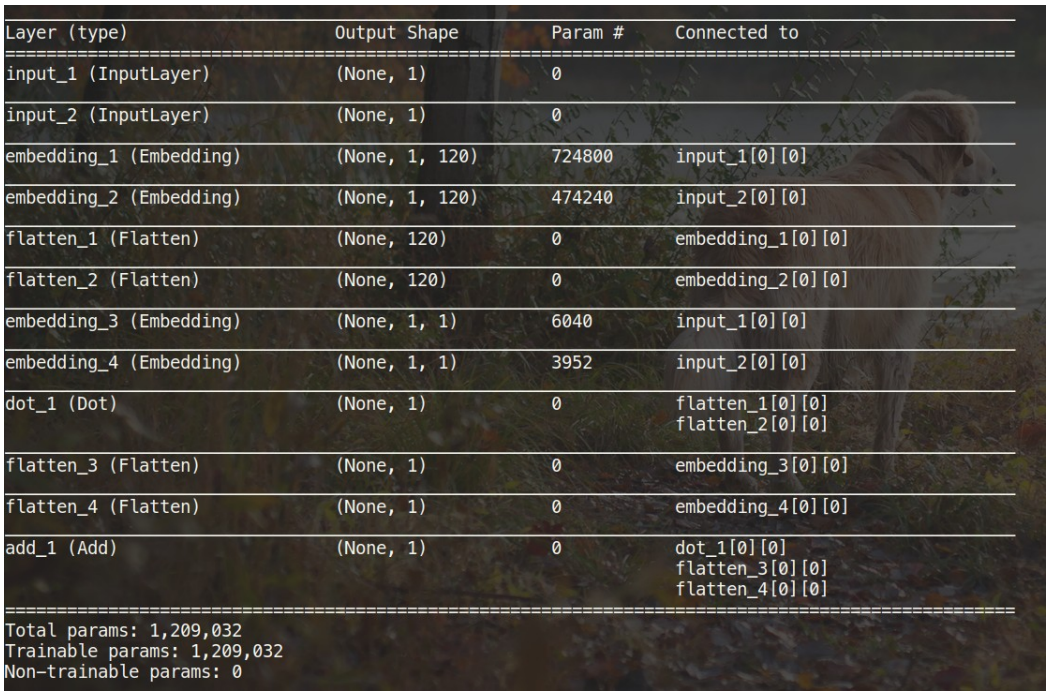
3. (1%)比較有無 bias 的結果。
   在其他 training 條件（epoch...等）皆相同的條件下
   有 bias => Public: 0.86289 / Private: 0.86598
   無 bias => Public: 0.90887 / Private: 0.91045
   可以發現到有加 bias 的影響很大，如果沒有 bias 則分數會較差，我覺得應該是會 underfitting 的關係，因為 MF 是用 gradient descent 的方法。

4. (1%)請試著用 DNN 來解決這個問題，並且說明實做的方法(方法不限)。並比較 MF 和 NN 的結果，討論結果的差異。
   NN model:

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 120)	724800	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 120)	474240	input_2[0][0]
flatten_1 (Flatten)	(None, 120)	0	embedding_1[0][0]
flatten_2 (Flatten)	(None, 120)	0	embedding_2[0][0]
concatenate_1 (Concatenate)	(None, 240)	0	flatten_1[0][0] flatten_2[0][0]
dense_1 (Dense)	(None, 150)	36150	concatenate_1[0][0]
dense_2 (Dense)	(None, 50)	7550	dense_1[0][0]
dense_3 (Dense)	(None, 1)	51	dense_2[0][0]

Total params: 1,242,791
Trainable params: 1,242,791
Non-trainable params: 0

MF model:



Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 120)	724800	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 120)	474240	input_2[0][0]
flatten_1 (Flatten)	(None, 120)	0	embedding_1[0][0]
flatten_2 (Flatten)	(None, 120)	0	embedding_2[0][0]
embedding_3 (Embedding)	(None, 1, 1)	6040	input_1[0][0]
embedding_4 (Embedding)	(None, 1, 1)	3952	input_2[0][0]
dot_1 (Dot)	(None, 1)	0	flatten_1[0][0] flatten_2[0][0]
flatten_3 (Flatten)	(None, 1)	0	embedding_3[0][0]
flatten_4 (Flatten)	(None, 1)	0	embedding_4[0][0]
add_1 (Add)	(None, 1)	0	dot_1[0][0] flatten_3[0][0] flatten_4[0][0]

Total params: 1,209,032
Trainable params: 1,209,032
Non-trainable params: 0

其實兩個 model 都是直接用助教給的 sample code 的架構

NN 的實作方法也就是用助教提到的，將兩個 embedding 連接在一起丟到 DNN 去 train，output 則是視為 regression 的問題來解，最後結果如下：

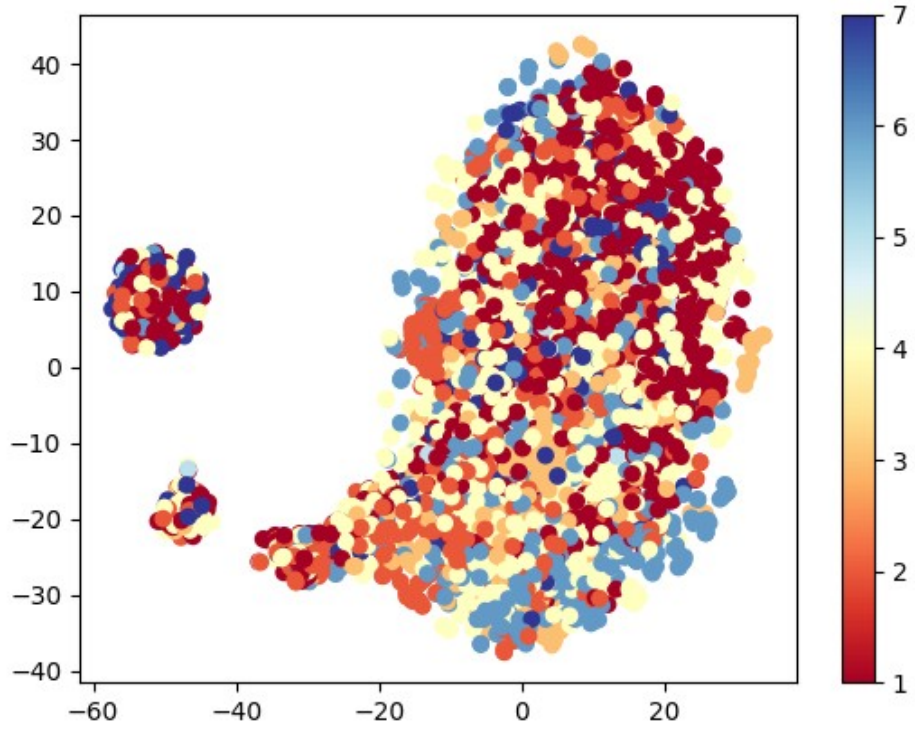
MF: Public: 0.86289 / Private: 0.86598

DNN: Public: 0.86013 / Private: 0.86315

會發現如同老師的 youtube 影片所說，DNN 的效果是可以贏過 MF 的。

5. (1%)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。

   1 => Drama, Musical
   2 => Thriller, Horror, Crime
   3 => Adventure, Animation, Children
   4 => Comedy, Western
   5 => Romance
   6 => War
   7 => No label



6. (BONUS)(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。

   我把性別也加到 DNN 一起 train，做法跟前面一樣，也就是把 gender、user 以及 movie 共 3 個 embedding 一起連接起來，output 視為 regression 的問題來解，最後傳到 kaggle 的分數為 Public: 0.86124 / Private: 0.86195，雖然有進步一點點，不過也不顯著，架構圖如下：

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 180)	1087200	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 180)	711360	input_2[0][0]
input_3 (InputLayer)	(None, 1)	0	
flatten_1 (Flatten)	(None, 180)	0	embedding_1[0][0]
flatten_2 (Flatten)	(None, 180)	0	embedding_2[0][0]
embedding_3 (Embedding)	(None, 1, 180)	360	input_3[0][0]
concatenate_1 (Concatenate)	(None, 360)	0	flatten_1[0][0] flatten_2[0][0]
flatten_3 (Flatten)	(None, 180)	0	embedding_3[0][0]
concatenate_2 (Concatenate)	(None, 540)	0	concatenate_1[0][0] flatten_3[0][0]
dense_1 (Dense)	(None, 150)	81150	concatenate_2[0][0]
dense_2 (Dense)	(None, 50)	7550	dense_1[0][0]
dense_3 (Dense)	(None, 1)	51	dense_2[0][0]

Total params: 1,887,671
Trainable params: 1,887,671
Non-trainable params: 0