1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.
   (collaborator:NO)

	Training MSE	Validation MSE	Public Score	Private Score
無 **Norm**	0.6482	0.7402	0.85774	0.86028
有 **Norm**	0.5057(norm)	0.5987(norm)	0.86756	0.86729

Training data mean = 3.5817 std = 1.1169
因此先將 training data 的 rating 利用算出的 mean 與 std normalize 之後，得到的值去 train，train 完之後 test 時，在乘以 training std 以及將 training mean 加回，得到最終 predict 的結果。
而兩者訓練起來，有 normalize 的模型，收斂的較快，但得到的結果也較差。

2. (1%)比較不同的 latent dimension 的結果。
   (collaborator:NO)
   (皆訓練 100epochs,取 Validation 最低者)

Latent Dim	Training MSE	Validation MSE	Public Score	Private Score
16	0.6956	0.7414	0.86136	0.86256
32	0.6482	0.7402	0.85774	0.86028
48	0.6539	0.7392	0.85812	0.86023
64	0.6553	0.7388	0.85855	0.86241
128	0.6152	0.7461	0.86101	0.86188

維度設在 32 或 48，表現最好。維度低的話，如 16，則尚未收斂；維度調在高的話，雖然收斂的會較快，但也容易 overfitting，導致正確率下降。

3. (1%)比較有無 bias 的結果。
   (collaborator:NO)

	Training MSE	Validation MSE	Public Score	Private Score
有 **bias**	0.6482	0.7402	0.85774	0.86028
無 **bias**	1.1621	0.8510	0.92348	0.92193

   在 train 差不多的 epoch 之下，有 bias 的結果會比無 bias 的好很多，但實作時發現，無 bias 的狀況下收斂得比較慢，但基本上在有 bias 的調整下做出來的結果還是比較好。

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。
   (collaborator:NO)
   模型：

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	(None, 1)	0	
input_1 (InputLayer)	(None, 1)	0	
embedding_3 (Embedding)	(None, 1, 32)	193280	input_2[0][0]
embedding_1 (Embedding)	(None, 1, 32)	126464	input_1[0][0]
flatten_3 (Flatten)	(None, 32)	0	embedding_3[0][0]
flatten_1 (Flatten)	(None, 32)	0	embedding_1[0][0]
dropout_2 (Dropout)	(None, 32)	0	flatten_3[0][0]
dropout_1 (Dropout)	(None, 32)	0	flatten_1[0][0]
concatenate_1 (Concatenate)	(None, 64)	0	dropout_2[0][0] dropout_1[0][0]
dense_1 (Dense)	(None, 16)	1040	concatenate_1[0][0]
dense_2 (Dense)	(None, 8)	136	dense_1[0][0]
embedding_4 (Embedding)	(None, 1, 1)	6040	input_2[0][0]
embedding_2 (Embedding)	(None, 1, 1)	3952	input_1[0][0]
dense_3 (Dense)	(None, 1)	9	dense_2[0][0]
flatten_4 (Flatten)	(None, 1)	0	embedding_4[0][0]
flatten_2 (Flatten)	(None, 1)	0	embedding_2[0][0]
add_1 (Add)	(None, 1)	0	dense_3[0][0] flatten_4[0][0] flatten_2[0][0]

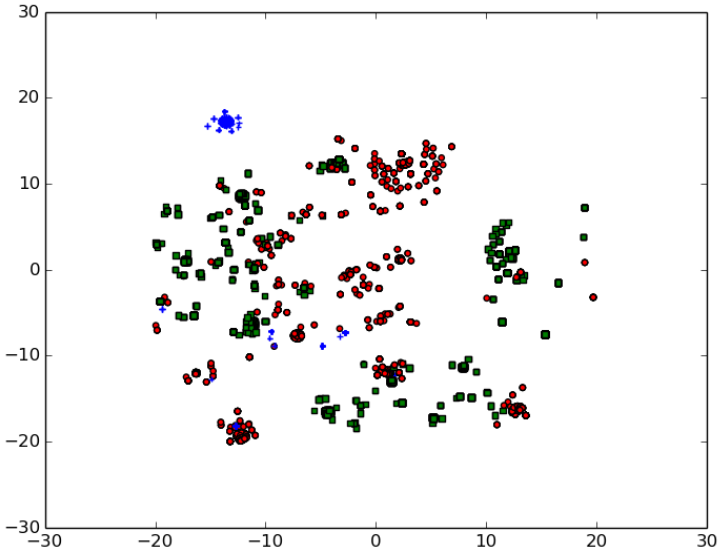
原本的 embedding 為 32 維，將 user&movie concate 起來，變成 64 維之後，再做三層 DNN，降維至 16、8、1 維之後，再加上 user_bias&movie_bias。

	Training MSE	Validation MSE	Public Score	Private Score
MF	0.6482	0.7402	0.85774	0.86028
DNN	0.6732	0.7548	0.86650	0.86854

由於對 user_vec 及 movie_vec 做 dot 有做 similarity 的意義，因此比較符合兩者的相關性。

5. (1%)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。(collaborator:B03902101 楊力權)

   作圖結果：



(紅圈：1-6(Animation、Childrens...)綠正方：7-12(Drama、Action...)藍+：13-18(Documentary...))
我將 18 種 movie category 分成三類，但由於 category 分類的方法是採取 category 在 movie.csv 出現的順序，因此從圖中看不出太大的關連性，但藍色似乎在左上角有特別集中的趨勢。