

※以下實驗皆是使用10% training data 當作 validation，train到valid loss最低停止。

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

(collaborator: NULL)

latent dimension = 10，有bias

normalize rating	Private Score	Public Score
有		0.86683
無		0.87616

normalize 演算法：

ave = numpy.average(rating)

std = numpy.std(rating)

rating = (rating - ave) / std

因為rating的值其實並不大，以數學理論來說應沒有影響，不過實際跑有normalize rating的分數低了0.01，可能是誤差，也可能是因為normalize後機器fit得更好。

2. (1%)比較不同的latent dimension的結果。

(collaborator: NULL)

有bias，未normalize rating

latent dimension	Private Score	Public Score
8		0.87105
16		0.87490
32		0.88156
64		0.88825
128		0.90128
256		0.91859

可發現latent dimension為個位數具有較佳表現，大於16的話表現會明顯變差。

3. (1%)比較有無bias的結果。

(collaborator: NULL)

latent dimension = 10，未normalize rating

bias	Private Score	Public Score
有		0.87616
無		0.87837

結果稍稍變差，可見每部電影或每個使用者的評分或多或少會有一些bias。

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: b04901146 - 黃禹傑)

輸入的資料同MF。

DNN model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, 10)	60410	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 10)	39530	input_2[0][0]
flatten_1 (Flatten)	(None, 10)	0	embedding_1[0][0]
flatten_2 (Flatten)	(None, 10)	0	embedding_2[0][0]
concatenate_1 (Concatenate)	(None, 20)	0	flatten_1[0][0] flatten_2[0][0]
dense_1 (Dense)	(None, 256)	5376	concatenate_1[0][0]
dropout_1 (Dropout)	(None, 256)	0	dense_1[0][0]
embedding_3 (Embedding)	(None, 1, 1)	6041	input_1[0][0]
embedding_4 (Embedding)	(None, 1, 1)	3953	input_2[0][0]
dense_2 (Dense)	(None, 1)	257	dropout_1[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	dense_2[0][0] flatten_3[0][0] flatten_4[0][0]
Total params: 115,567			
Trainable params: 115,567			
Non-trainable params: 0			

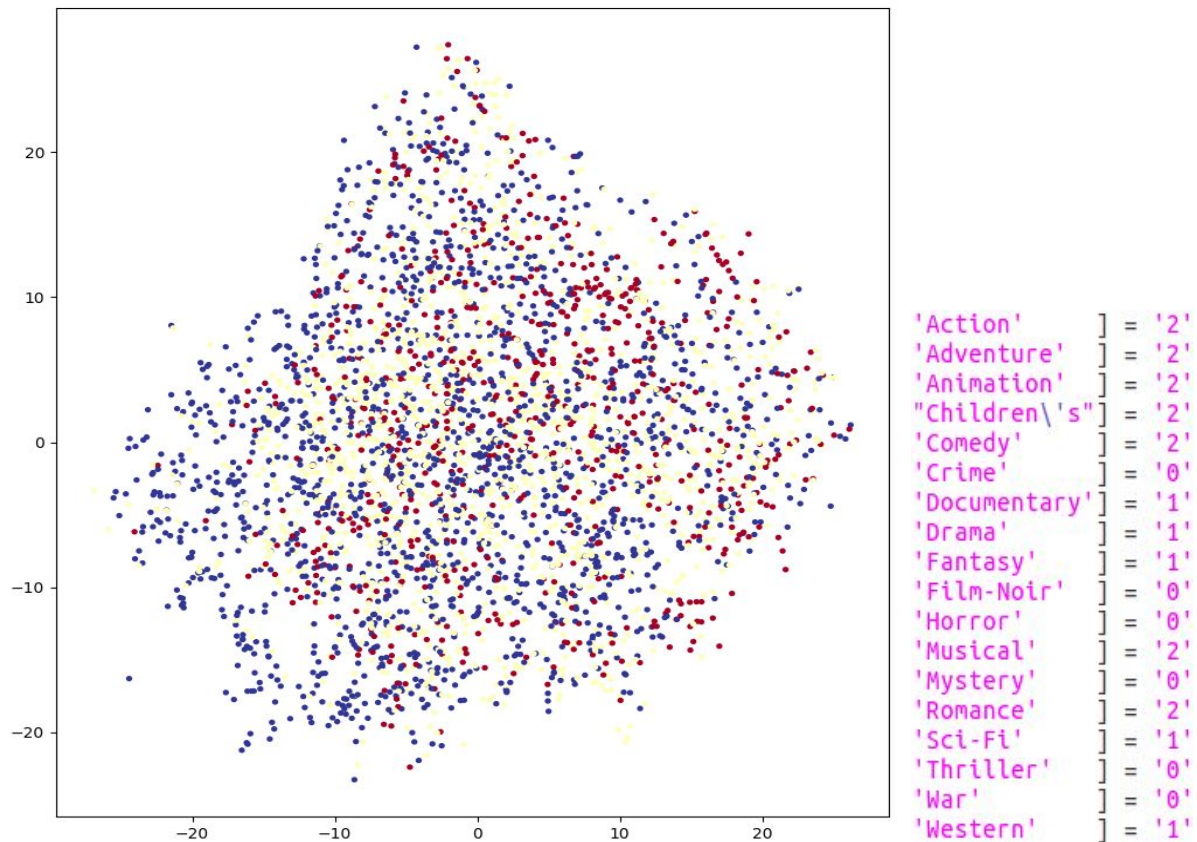
model	Private Score	Public Score
MF		0.87616
DNN		0.86050

結果進步很顯著，推測是因DNN較靈活，MF受到內積運算的限制，有些資訊無法包含。

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

(collaborator: NULL)

分為三種，結果如下，右邊是mapping(紅 0、米 1、藍 2)。



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。

(collaborator: NULL)

將User的Gender、Age和Movie Genres的one-hot encoding整理成20維的向量，加入DNN model的Concatenate層，Keras model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, 10)	60410	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 10)	39530	input_2[0][0]
flatten_1 (Flatten)	(None, 10)	0	embedding_1[0][0]
flatten_2 (Flatten)	(None, 10)	0	embedding_2[0][0]
input_3 (InputLayer)	(None, 20)	0	
concatenate_1 (Concatenate)	(None, 40)	0	flatten_1[0][0] flatten_2[0][0] input_3[0][0]
dense_1 (Dense)	(None, 256)	10496	concatenate_1[0][0]
dropout_1 (Dropout)	(None, 256)	0	dense_1[0][0]
dense_2 (Dense)	(None, 1)	257	dropout_1[0][0]
Total params: 110,693			
Trainable params: 110,693			
Non-trainable params: 0			

Kaggle Score比較：

model	Private Score	Public Score
原始DNN		0.86050
加入其他info之DNN		0.87071

推測是因為這些資訊對於rating並不具有重要的bias或關係，因此加了反而容易overfitting(多達20維的資訊)，結果稍稍變差。