(1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.
 我是將 label 除以 5,來做 normalize,再以同樣的 batch size(9096) train 差不多的 epoch 後,得到的結果如下表

	有 normalize	無 normalize
Public	0.87543	0.86858
Private	0.87414	0.86747

可發現有 normalize 後好了不少

2. (1%)比較不同的 latent dimension 的結果。

	512	256	128	64
Public	0.85236	0.85931	0.87543	0.90957
Private	0.85223	0.85965	0.87414	0.90743

基本上得到的結果是 latent 越大會有越好的結果,但花的時間也相對越多,可惜是在寫 report 的時候才發現這點。

3. (1%)比較有無 bias 的結果。

下列都是以 latent 128, batch size 9096 train 約 100 epoch 的結果

	有 bias	無 bias
Public	0.86114	0.87543
Private	0.86077	0.87414

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。

	MF	DNN
Public	0.87543	0.88002
Private	0.87414	0.88125

Structure:

	Private∂	₽	6 6	
concatenate_1 (Concatenate)	(None, 256)	0	dropout_1[0][0]	
			dropout_2[0][0]	
dense_1 (Dense)	(None, 128) ovie 的 ember	id32896 sne 1421	concatenate_1[0][0]	
dense_2 (Dense)	(None, 64)	8256	dense_1[0][0]	
dense_3 (Dense)	(None, 32) (BONUS)(1%)試著使用除了r	2080 ating 以外的 featu	dense_2[0][0] re. 並說明你的作法和結果,	
dense_4 (Dense)	(None、1) 影響評分。我多	⊶ 33 movie categ	odense_3[0][0] _{Hy}	
Total params: 1,322,497	array(總共 18 種 category),若這部電影是屬於 A 這個 category,則他的			
Trainable params: 1,322,497 Non-trainable params: 0	category array 會在 A 處為 1,其他地方為 0,例如: Movie A 是 Comedy			
Non-traffiable paralis. 0	Drama,則應該會做出一個。			

因為我覺得評分 1~5 分之間的關係是線性的而非分類問題,因此我把它當作 regression 問題解決,在 batch size 9096 時,這個 model 大約會在十幾個 epoch 後開始收斂,再來就會 overfit 了。取 validation 最好的結果上傳 kaggle 可以得到上面的成績。

我認為 concatenate 起來過 NN 的那個 input 感覺不太能夠將 user 跟 movie 區分開來,雖然 NN 本來就有點像黑魔術,但是直覺來看覺得這樣不太 work,雖然我沒怎麼 tune 就有還行的結果了,兩個無法真的說出有多大的 差異。

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

我多拿了 movie category,做成一個 18 維的 array(總共 18 種 category),若這 部電影是屬於 A 這個 category,則他的 category array 會在 A 處為 1,其他 地方為 0,例如: Movie A 是 Comedy | Drama,則應該會做出一個[0 0 0 1 0 1 0 0 0 ... 0 0],兩個 1 分別代表 Comedy 和 Drama。將這些 Array 送進 DNN,使輸出與 movie 及 user 的 embedding output,做 dot,再將這兩個 dot 的結果與 movie 與 user 的 embedding output dot 的結果 concatenate 起來,再 送進一個 DNN,去 predict 結果。

Structure:

Layer (type) × test_user, x_test_	Output S	Shapest(opts.te	Param #path)	Connected to
movie_input (InputLayer)	s(None,()1	L)	0	
user_input (InputLayer)	(None, 1	i) [x_test_user,	0 test_movie	
embedding_4 (Embedding)	(None, 1	1, 128) W	505984	movie_input[0][0]
embedding_3 (Embedding)erow(["Te	(None,) 1	L, r128)g"])	773248	user_input[0][0]
flatten_4 (Flatten)	(None, 1	L28)	0	embedding_4[0][0]
movie_feat (InputLayer)	(None, 1	L8)	0	
flatten_3 (Flatten) _user, x_trai	in(None, 1	L28)ain = readf	Ole(opts.trai	embedding_3[0][0]
dropout_4 (Dropout)	(None, 1	L28)	0	flatten_4[0][0]
dense_4 (Dense)	(None, 1	L28) feat[int(row)]	2432	movie_feat[0][0]
dropout_3 (Dropout)=_feat = np.ar	(None, 1	[28] eat, <i>dtype=f</i>	Oat)	flatten_3[0][0] ted, otherwise one or more
<pre>dot_4 (Dot) print(x_movie_feat[1 161 model = feat_input_N</pre>	(None, 1 MF_Model()	l)	0	dropout_4[0][0] on properly. dense_4[0][0]
<pre>dot_5 (Dot) callbacks = keras.ca model.fit({'user_inp</pre>	(None, 1	$\mathbf{p}^{lelCheckpoint}$ ain_user, 'movi		dropout_3[0][0] dropout_4[0][0]
dot_6 (Dot) x_test_user, x_test_ movie_feat = readmov	(None, rl vie('movie	$\mathbf{Q}_{dtest(opts.te}$	0 t_data_path)	dropout_3[0][0] dense_4[0][0]
concatenate_2 (Concatenate) x test_mo x test_mo x_movie_feat.app	(None, B ovie: pend(movie		0	dot_4[0][0] dot_5[0][0] dot_6[0][0]
dense_5 (Dense)	(None, 1	[6]	64	concatenate_2[0][0]
dense_6 (Dense) del load_model(m	(None, 1	<pre>custom_object t user x test n</pre>	17 rmse : rm	dense_5[0][0]
Total params: 1,281,745 Trainable params: 1,281,745 Non-trainable params: 0	sult,1,5) sub_simple (submissio	1.csv",'w') n)		

	有加上 movie feature	無加上 movie feature
Public	0.85066	0.86114
Private	0.84995	0.86077