學號：r05922145 系級：資工碩二 姓名：郁錦濤

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

我的normalize的方法是將rating減去它的mean值，再除以它的標準差。所有結果都是在沒有bias的時候獲得的。這種方法在進行predict事需要將model.predict的結果乘以標準差再加上mean值才是最後的結果。通過下表比較，發現在normalize的情況下，performance更好。

有無normalize(rating)	public	private
有	0.87393	0.87127
無	0.88361	0.88465

2. (1%)比較不同的latent dimension的結果。
   (collaborator:)
   答：

我的結果都是在normalize的情況下獲得的，在latent dimension較小的情況下，model在5個epoch的就能達到最低點，隨着latent dimension的增大，結果會overfitting。按照這種情況，latent dimension取到合適就可以，不用太大。

latent dimension	8	16	32	64	128	256
private set	0.86709	0.87127	0.89036	0.92678	0.96984	0.98765

3. (1%)比較有無bias的結果。
   (collaborator:)
   答：

有無bias對於結果還是有一定的影響，有bias的model下performance表現更好。我的結果都是在normalize的情況下獲得的。因爲每個人對於電影rating的評判標準是不一致的，每個人喜歡的電影風格不一致，對於不同的風格的作品rating的尺度也不一樣。對於同一movie的不同user，他的rating尺度也會不一樣，這樣就會造成差異。加上user bias和movie bias確實可以有效消除這個影響。

有無bias	public	private
有	0.86791	0.86882
無	0.87694	0.87489

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:)
答：

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 128)	773120	input_1[0][0]
embedding_4 (Embedding)	(None, 1, 128)	505856	input_2[0][0]
flatten_1 (Flatten)	(None, 128)	0	embedding_1[0][0]
flatten_4 (Flatten)	(None, 128)	0	embedding_4[0][0]
embedding_3 (Embedding)	(None, 1, 23)	138920	input_1[0][0]
embedding_6 (Embedding)	(None, 1, 18)	71136	input_2[0][0]
dropout_1 (Dropout)	(None, 128)	0	flatten_1[0][0]
dropout_2 (Dropout)	(None, 128)	0	flatten_4[0][0]
flatten_3 (Flatten)	(None, 23)	0	embedding_3[0][0]
flatten_6 (Flatten)	(None, 18)	0	embedding_6[0][0]
concatenate_1 (Concatenate)	(None, 297)	0	dropout_1[0][0] dropout_2[0][0] flatten_3[0][0] flatten_6[0][0]
dense_1 (Dense)	(None, 256)	76288	concatenate_1[0][0]
dropout_3 (Dropout)	(None, 256)	0	dense_1[0][0]
batch_normalization_1 (BatchNorm	(None, 256)	1024	dropout_3[0][0]
dense_2 (Dense)	(None, 256)	65792	batch_normalization_1[0][0]
dropout_4 (Dropout)	(None, 256)	0	dense_2[0][0]
batch_normalization_2 (BatchNorm	(None, 256)	1024	dropout_4[0][0]
dense_3 (Dense)	(None, 256)	65792	batch_normalization_2[0][0]
dropout_5 (Dropout)	(None, 256)	0	dense_3[0][0]
batch_normalization_3 (BatchNorm	(None, 256)	1024	dropout_5[0][0]
dense_4 (Dense)	(None, 1)	257	batch_normalization_3[0][0]

我使用的dnn model是將從training data中獲得的user/movie id通過Embeding變成128維的vector，再flatten，然後將user和movie的某些feature也通過Embedding變成vector，然後通過concatenate將四個embedding合在一起過含有3個256unit的Dense，激活函數使用relu，dropout=0.5，最後再通過一個dense(1)輸出一個數字，將此數字作爲rating。

通過這個方法做出來的結果比較好，在private set能有0.85242的performance。最後的best結果也是通過dnn求出來的。而通過matrix fatorization的方法最好才有0.86860多，DNN的model確實比MF的好。但是此種方法訓練過程比較慢，收斂的時間比較長

，訓練的modelb不容易overfitting（MF的mdeol如果不控制epoch非常容易overfitting）。

比較兩種方法，MF可以很快的將loss減小，model比較穩定。DNN model比較慢，但是performance比MF好。

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

我將movie.csv中的genre根據經驗分成5類，分別是

(1)Animation, Children, Comedy

(2)Fantasy, Adventure, Action, Sci-Fi

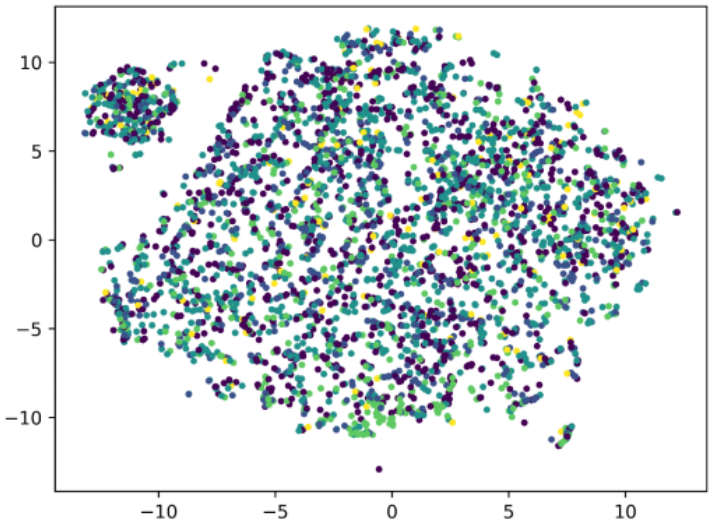
(3)Drama, Musical, Romance

(4)Crime, Thriller, Horror, Film-Noir, Mystery

(5)Western, War, Documentary

每個movie取它的genre中的第一個值爲類別。在將movie embedding用tsne降維後，得到如下圖像。該圖像混亂，無法看出其中的規律。

我認爲因爲movie是multilabel的，所以直接按照我的劃分是不能夠劃分出有規律的圖像。按照我這中劃分的不同類別的電影因爲是multilabel，所以在genre的其他種類中會有相似。



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:)

答：

觀察user.csv和movie.csv，user和movie的很多feature我覺得都會影響最終的結果。user的gender/occupation和movie的genre我覺得是比較重要的feature，所以我在user中抽取了以01表示的gender和以one-hot encoding表示的occupation形成user_info，在movie中的genre提取出來，用one-hot encoding方式表示，形成movie_info。最後將之前的embedding layer的user matrix和movie matrix相互concatente，通過三層dense(256)，最後過一層dense(1)，輸出結果爲rating。

這種方法對performance確實有很大的提高，最後結果能達到0.85204。