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

	public score	private score
無normalize	0.87922	0.87275
normalize	0.87674	0.87086

Normalize的方法： 把rating減掉平均再除以標準差，predict完後再改回來。
由上表可見，normalize後結果較好。另外在training的時候可以看到，normalize後的資料fit得快很多，我想那是因為，實際的rating平均是3點多，跟零相差滿遠的。如果不normalize，光是要學到這個平均值，就需要花許多時間。
ps. batch_size=1024, epoch=10, validation=0.1, embedding_dimension=16

2.(1 %)比較不同的embedding dimension的結果。

embedding dimension	public score	private score
16	0.87922	0.87275
512	0.91701	0.91147
5000	0.94119	0.93591

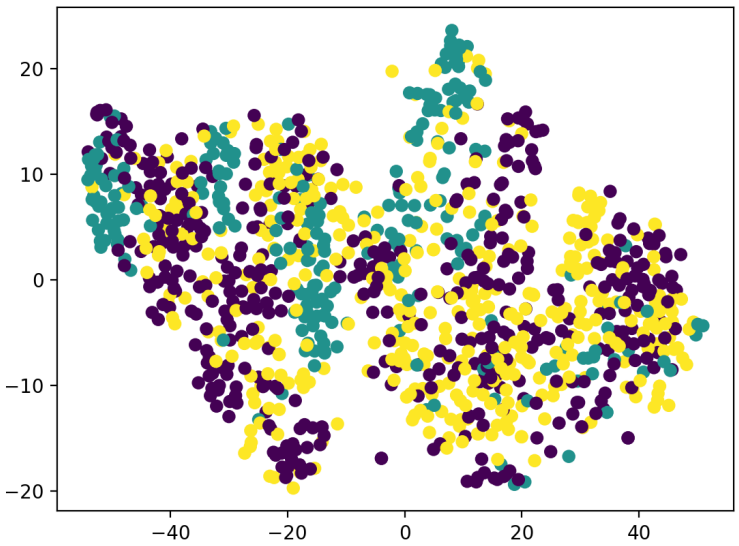
由上表可見，embedding dimension大致上越少越好，那是因為高維度的embedding很容易overfit。
ps.高維度embedding如果搭配regularizer和適當的dropout會好一些，但是依然沒有勝過低維度的。

3.(1 %)比較有無bias的結果。

	public score	private score
無bias	0.90585	0.90010
有bias	0.87922	0.87275

由上表可見，有bias較好，理論上只要不要overfit，越多參數越能夠fit到真正的值，而bias相對於其他參數，本來就比較不容易造成overfit，所以這個結果很符合預期。
ps. batch_size=1024, epoch=10, validation=0.1, embedding_dimension=16

4.(1 %)請試著將movie的embedding用tsne降維後，將movie category當作label來作圖。



我選擇了我認為相差比較多的類別：Thriller、Children's、Romance。
在資料中，如果有某部電影同時是這三個類別的兩個以上，就不採計，也就是說，挑出是Thriller、不是Children's、不是Romance的電影剛做第一類，挑出不是Thriller、是Children's、不是Romance的當第二類...
圖中綠色是Thriller、黃色是Children's、紫色是Romance，經過tsne降維後，並沒有明顯的分群，但是至少個群呈現帶狀分佈，群組內有集中的趨勢。

5.(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。

	public score	private score
無rating以外的feature	0.87922	0.87275
有rating以外的feature	0.87674	0.87542

(1)把user_ID、movie_ID分別embedding
(2)把occupation跟電影種類分別接到dense（跟做embedding意思差不多）
(3)把user_ID、movie_ID、occupation、電影種類embedding後的向量兩兩內積
(4)把兩兩內積的結果，跟age、gender concatenate起來，然後接到dense，最後連到output
ps. batch_size=1024, epoch=10, validation=0.1, embedding_dimension=16

Reference
1. https://github.com/sonyisme/keras-recommendation
2. https://github.com/zhangruiskyline/DeepLearning/blob/master/doc/Recommendation.md
3. https://medium.com/data-from-the-trenches/deep-beers-playing-with-deep-recommendation-engines-using-keras-part-1-1efc4779568f
4. https://github.com/chinchi-hsu/KerasCollaborativeFiltering/blob/master/matrix_factorization.py