學號：R05921078 系級： 電機碩 1 姓名：洪立達

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

Normalization：(rating – rating.mean)/(rating.max – rating.min)

使用 normalize 來預測在 kaggle 上的分數為：0.90960

不使用 normalize 的分數為：0.86689

推測原因是因為原始 rating 分數範圍為 1~5，做 normalize 後分數與分數之間的差距變得不明顯了，因此增加預測結果的誤差。

為了驗證此論點，我將 rating*N 來放大 rating 間的差距：

N=10 的時候，分數為：0.85642

N=100 的時候，分數為：0.84962

N=1000 的時候，分數為：0.84818

很明顯的，放大 rating 間的差距，可以使預測結果間的差異更加明顯，得到的準度也越高。不過大於一定的程度後，效果就差不多了。

2. **(1%)比較不同的 latent dimension 的結果。**

Latent dimension	Kaggle 分數
32	0.87331
64	0.87019
128	0.87059
256	0.87011
512	0.87629
1024	0.87043
2048	0.89227

由圖表可以發現 latent dimension 在 1024 以內的結果都是差不多的，而高於 1024 後便開始明顯變差。綜合來講，latent dimension 不要取太大或是太小比好，最終我取的 latent dimension 為 256。

3. **(1%)比較有無 bias 的結果。**

bias	Kaggle 分數
User bias + movie bias	0.87011
User bias only	0.87015
Movie bias only	0.87049
No bias	0.87169

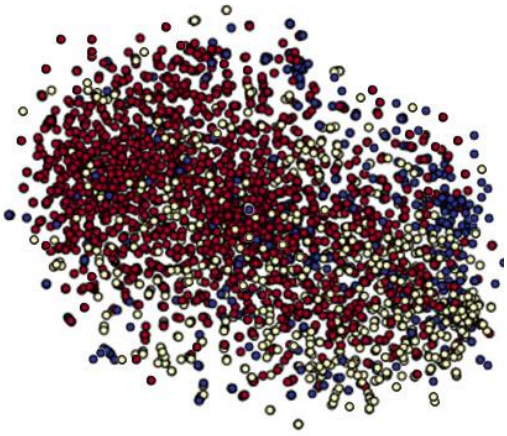
加 bias 對於結果是有影響的，其中又以加入 user bias 的影響程度比較大，因為 user bias 表示一個 user 在打分數的基本分。基於此概念，我覺得可以嘗試拿每個 user 的歷史最低 rating - alpha 當做 user bias(alpha 為一個可調整的變數，讓 bias 略小於最低的 rating 分數)。測試的結果分數為：0.86992 (alpha=0.5)，感覺有幫助，但是效果不是很明顯。

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。**

   DNN 所使用的方法是各別將 user id 及 movie id 做 embedding 後再 concatenate 起來，接著再透過兩層的 Dense 後再接 output layer，測試的分數是 0.88301。

   另外有嘗試結合 MF 和 DNN：做法是在 MF 做完 DOT 之後再接兩層的 dense，結果超爆差，只有 0.93519。

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



使用三個大分類

1.Drama,Musical

2. Thriller,Horror,Crime

3. Adventure,Animation,Children's

三者無明顯分界，不過紅色感覺比較集中在左上和中間，白色集中在右下，而藍色集中在右上。

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。**

   我試著加入 user age 來當作 feature。

   做法(1)：

   將 user id 及 user age 分別 embedding 後 concatenate 起來，再跟經過 embedding 的 movie id 做 DOT，測試的分數為：0.87235

   做法(2)：

   把 user age 也當成一個 bias 與 user bias 跟 movie bias 相加，測試的分數為：0.86945

   做法(3)：

   結合(1),(2)的方法，測試的分數為：0.87447

   結論：單獨拿來當 bias 比較有效！