HW5 report

學號：R06942074 系級： 電信所碩一 姓名：李宇哲

1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.
   首先先把這次 MF 的 model 架構顯示如下：

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 15)	90600	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 15)	55590	input_2[0][0]
reshape_1 (Reshape)	(None, 15)	0	embedding_1[0][0]
reshape_2 (Reshape)	(None, 15)	0	embedding_2[0][0]
dropout_1 (Dropout)	(None, 15)	0	reshape_1[0][0]
dropout_2 (Dropout)	(None, 15)	0	reshape_2[0][0]
embedding_3 (Embedding)	(None, 1, 1)	6040	input_1[0][0]
embedding_4 (Embedding)	(None, 1, 1)	6040	input_2[0][0]
dot_1 (Dot)	(None, 1)	0	dropout_1[0][0] dropout_2[0][0]
reshape_3 (Reshape)	(None, 1)	0	embedding_3[0][0]
reshape_4 (Reshape)	(None, 1)	0	embedding_4[0][0]
add_1 (Add)	(None, 1)	0	dot_1[0][0] reshape_3[0][0] reshape_4[0][0]
lambda_1 (Lambda)	(None, 1)	0	add_1[0][0]

Total params: 158,270
Trainable params: 158,270
Non-trainable params: 0

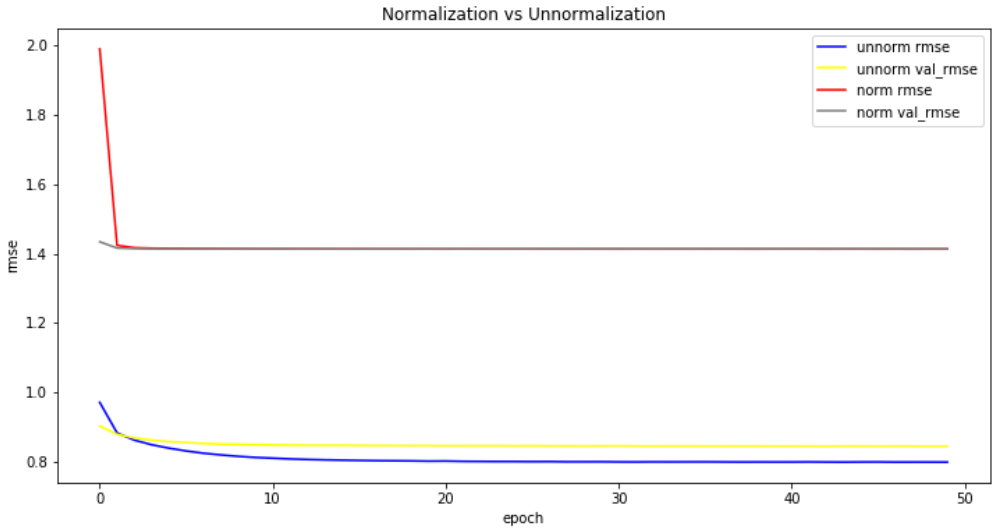
使用的 model hyperparameter:
Latent size: 15, 放 bias, batch size 為 128

我的 normalize 的方法如:

$$x_{new} = \frac{x - \mu}{\sigma}$$

將所有的 rating 減去平均數，然後再除以標準差。不過在 test 的時候需要將 rating 轉換回來

實驗結果如下：

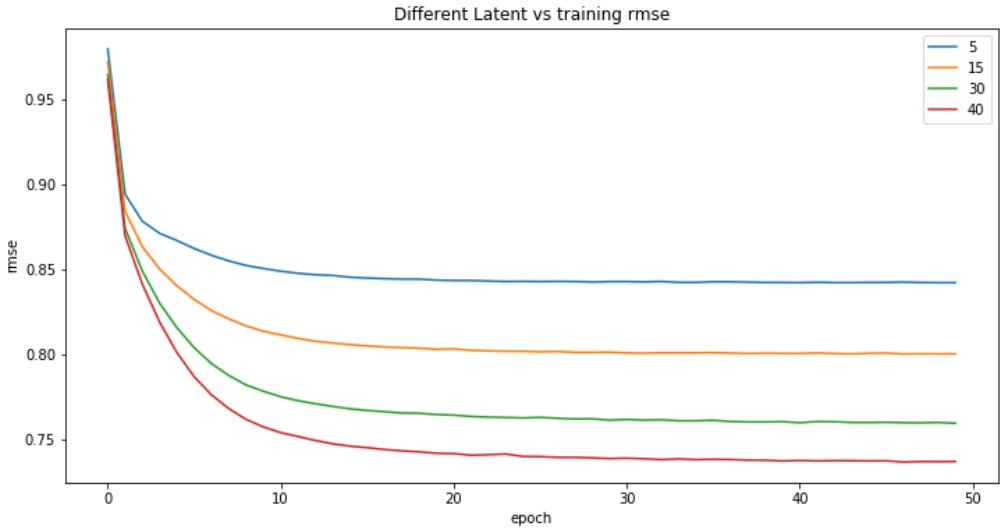


上圖為 normalize 和 unnormalize 的結果，可以發現 normalize 的 traing rmse 和 val rmse 基本上都是大於 1.4，所以可以看出 normalize 後的狀況似乎不佳。

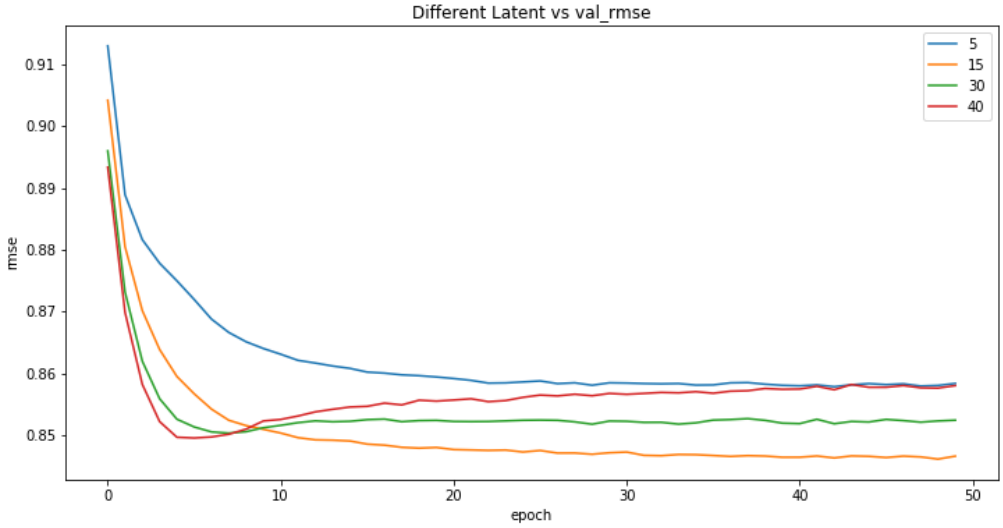
2. (1%)比較不同的 latent dimension 的結果。

使用的 model hyperparameter:
放 bias, batch size 為 128
Latent size: 5, 15, 30, 40

下圖為 training 的 rmse，可以發現 latent size 的 dimension 越大，training rmse 隨著 epoch 增加會比較低。



但是如果我們看另外一張圖的時候。如下圖為 validation 的 rmse，latent 的 dimension 就算在 training 的 rmse 是最低的，但是在 validation 反而不減反增，所以這裡可以推論 latent size 變大雖然可以使 training rmse 變小，但是反而會有 overfitting 的現象。所以 latent 15 反而是有最好的效果。

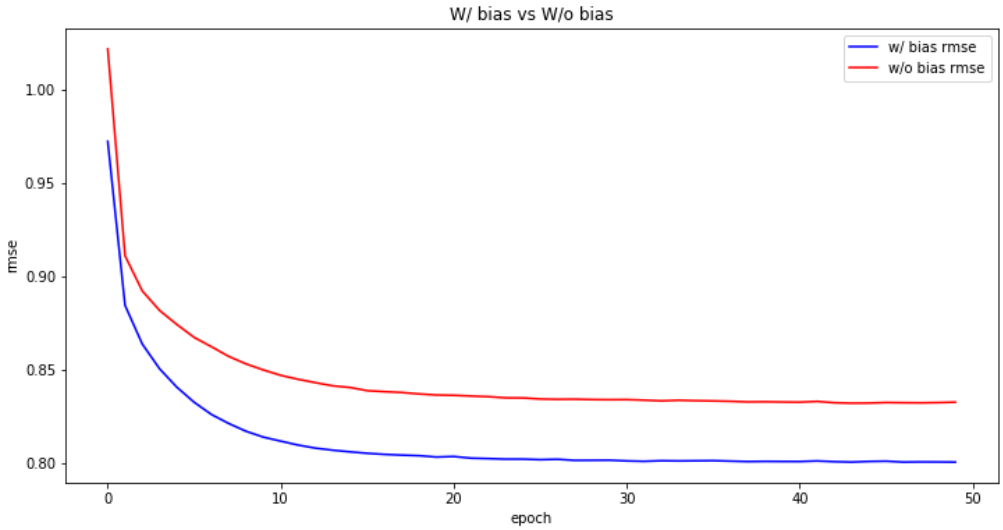


上圖為 validation 的 rmse

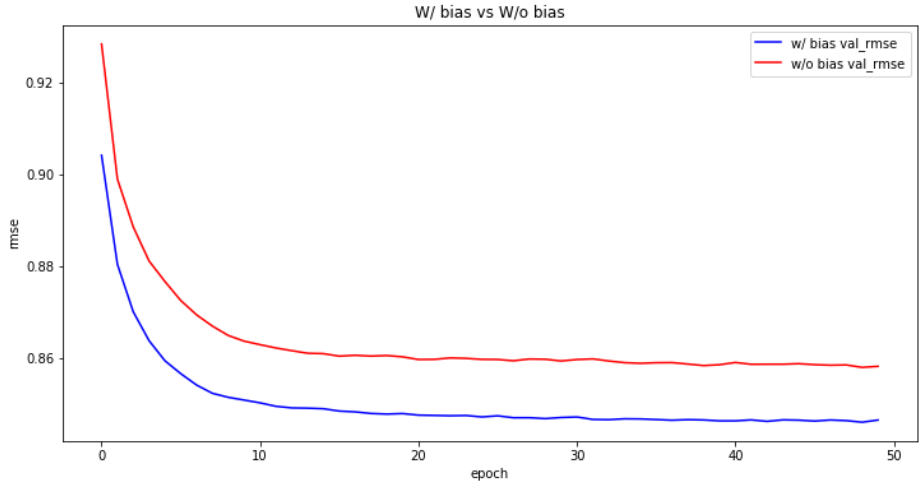
3. (1%)比較有無 bias 的結果。
   放 bias 主要就是在做 UV decomposition 的時候，最後再加上兩個 bias 選項，因為每個 user 可能都會有自己評分的傾向，像是傾向於把每部電影都評得很高分或者很低分；同樣的電影也會也這樣的趨勢。

$$r_{i,j} = U_i \cdot V_j + b_i^{user} + b_j^{movie}$$

Training:



Validation:



可以發現有加上 bias 不論是在 training 還是 testing 都有比較低的 rmse。

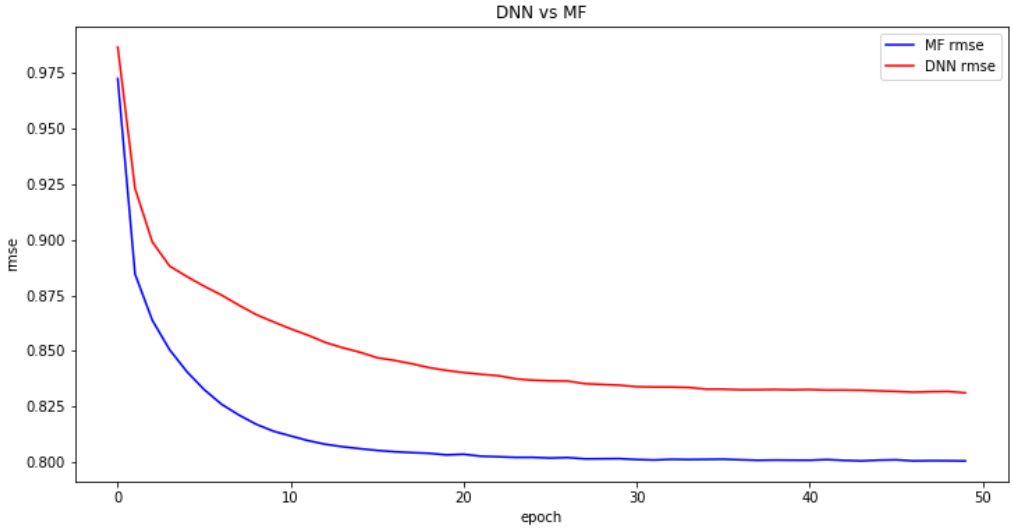
4. (1%)請試著用 DNN 來解決這個問題，並且說明實做的方法(方法不限)。並比較 MF 和 NN 的結果，討論結果的差異。

將 user embedding 以及 movie embedding concatenate 在一起再過 DNN 得出 rating，或者將 user embedding 以及 item embedding 分別通過兩個 DNN 得出 movie vector 以及 item vector 再去做內積得出 rating。
架構如下：

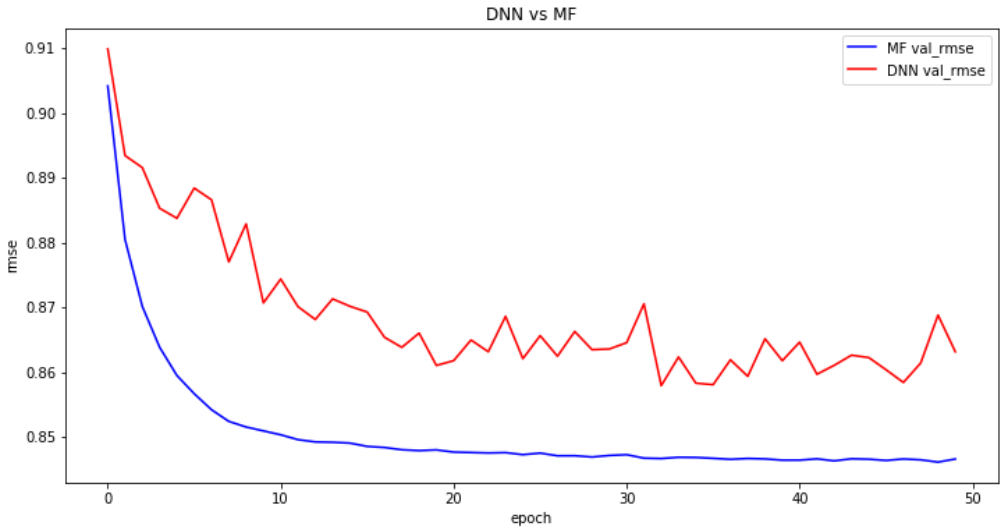
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 15)	90600	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 15)	55590	input_2[0][0]
reshape_1 (Reshape)	(None, 15)	0	embedding_1[0][0]
reshape_2 (Reshape)	(None, 15)	0	embedding_2[0][0]
dropout_1 (Dropout)	(None, 15)	0	reshape_1[0][0]
dropout_2 (Dropout)	(None, 15)	0	reshape_2[0][0]
concatenate_1 (Concatenate)	(None, 30)	0	dropout_1[0][0] dropout_2[0][0]
dense_1 (Dense)	(None, 1024)	31744	concatenate_1[0][0]
dropout_3 (Dropout)	(None, 1024)	0	dense_1[0][0]
dense_2 (Dense)	(None, 512)	524800	dropout_3[0][0]
dropout_4 (Dropout)	(None, 512)	0	dense_2[0][0]
dense_3 (Dense)	(None, 256)	131328	dropout_4[0][0]
dropout_5 (Dropout)	(None, 256)	0	dense_3[0][0]
dense_4 (Dense)	(None, 128)	32896	dropout_5[0][0]
dropout_6 (Dropout)	(None, 128)	0	dense_4[0][0]
dense_5 (Dense)	(None, 1)	129	dropout_6[0][0]

Total params: 867,087
Trainable params: 867,087
Non-trainable params: 0

基本上 DNN 的 parameter 是 MF 的 6 倍左右，但是 performance 卻沒有比較好。
Training:



Validation:



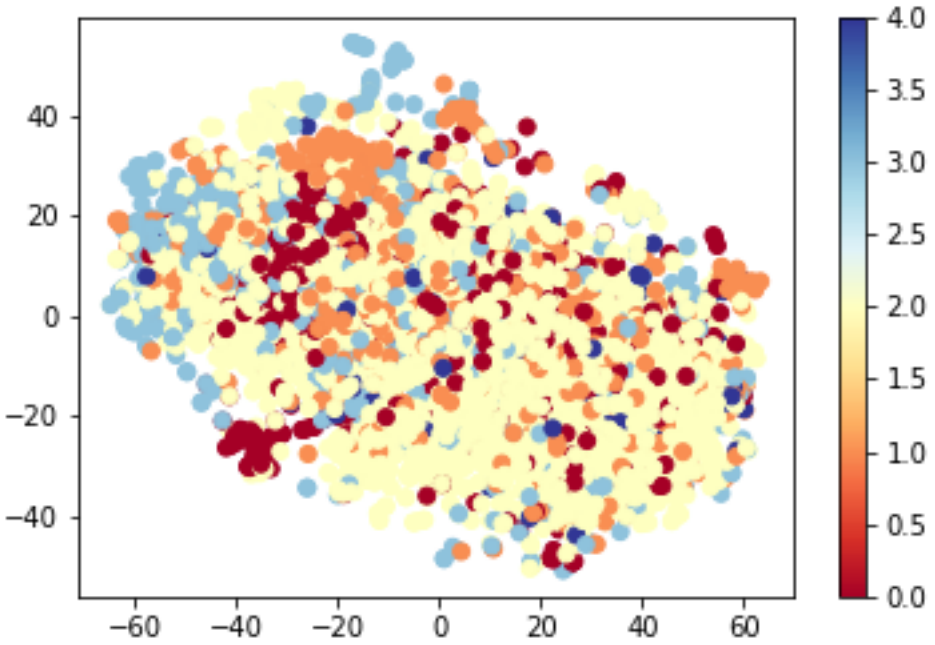
由上面兩張圖可以看出，不論是在 training 還是 testing，DNN 的 rmse 明顯都比較差，而且在 validation 的過程中，DNN 的 rmse 還會有跳動的狀況。

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

所有的種類有：Animation, Adventure, Comedy, Action, Drama, Thriller, Crime, Romance, Children's, Documentary, Sci-Fi, Horror, Western, Mystery, Film-Noir, War, Fantasy, Musical

label	包含的類別

0	Animation, Adventure, Children's, Documentary
1	Thriller, Crime, Horror, Film-Noir
2	Drama, Musical, Comedy, Romance
3	Action, Western, War
4	Mystery, Sci-Fi, Fantasy



上圖為將我選擇的 5 類畫在 tsne 的圖上的狀況，基本上相同的顏色還是有一點群聚的傾象，但是群聚的效果還是不彰。

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。

使用 SVD++:

reference:

http://www.cnblogs.com/Xnice/p/4522671.html

SVD 算法是指在 SVD 的基礎上引入隱式反饋，使用用戶的歷史瀏覽數據，用戶歷史評分數據，電影的歷史瀏覽數據，電影的歷史評分數據等作為新的參數

$$\hat{r}_{ui} = \mu + bi + bu + q_i^T\left(p_u + \frac{1}{\sqrt{\|R_u\|}}\sum_{j\in R_u} y_j\right)$$

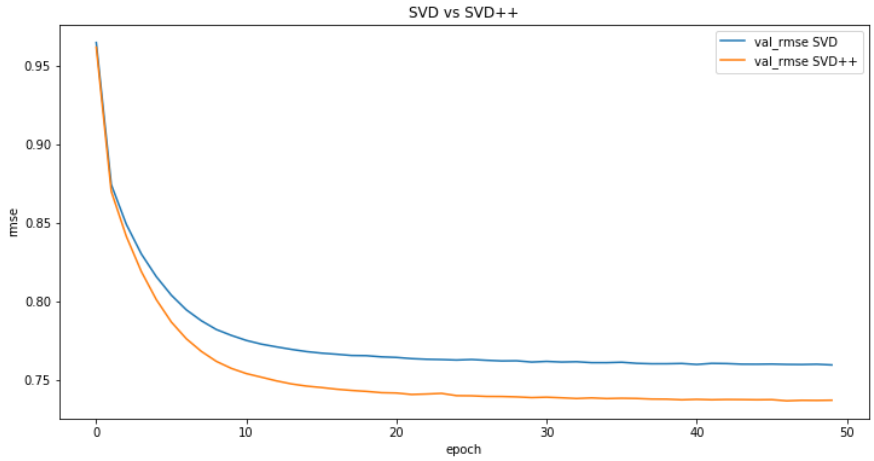
$$\hat{r}_{ui} = \mu + bi + bu + q_i^T(p_u + \frac{1}{\sqrt{\|R_u\|}}\sum_{j\in R_u} y_j)$$

求解公式如下：

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$
$$b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u)$$
$$b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda \cdot b_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$
$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot (p_u + \frac{1}{\sqrt{\|R_u\|}}\sum_{j\in R_u} y_j) - \lambda \cdot q_i)$$
$$y_j \leftarrow y_j + \gamma(e_{ui} \cdot \frac{1}{\sqrt{\|R_u\|}} \cdot q_i - \lambda \cdot q_i)$$

實驗結果可以看出 SVD++ outperform SVD：

Training:



Testing: