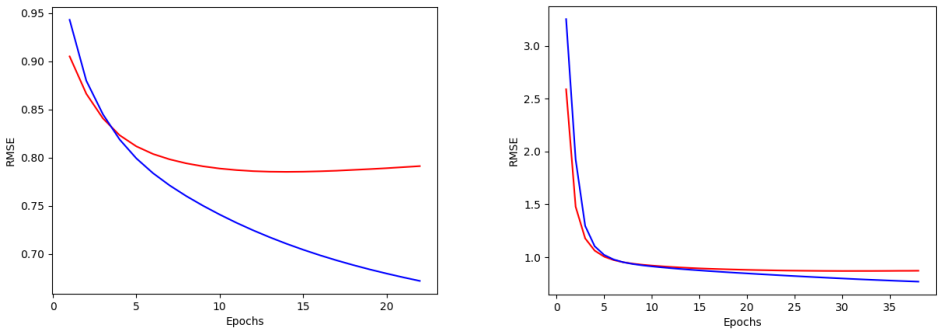
學號：B02404002 系級： 醫技四 姓名：葛竑志

1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.
(1)Normalize 方法為，從 Training data 中的 Rating 取得平均值 Tr_mean 與標準差 Tr_std 後，對 Training data 的 Rating 做 normalization((Rating-Tr_mean)/Tr_std)，將結果送入 model 做 training。在 predict 時，將 predict 的結果乘上 Tr_std 後加上 Tr_mean，以用來接近重建原始資料的分佈。
(2)Model 設計在有 bias 的 MF model 上，除了對 training data 的 rating 做 normalize 以外，其他參數值並不作更動。此外，Normalize 組的 validation data（佔所有資料的 20%，在此不定義為 Training data）使用 Tr_mean 與 Tr_std 做 normalization，是為了模擬在 prediction 時使用 Tr_mean 與 Tr_std 重建資料分布的狀況。
(3)實驗結果：（紅色：Validation set 之 RMSE，藍色：Training set 之 RMSE）

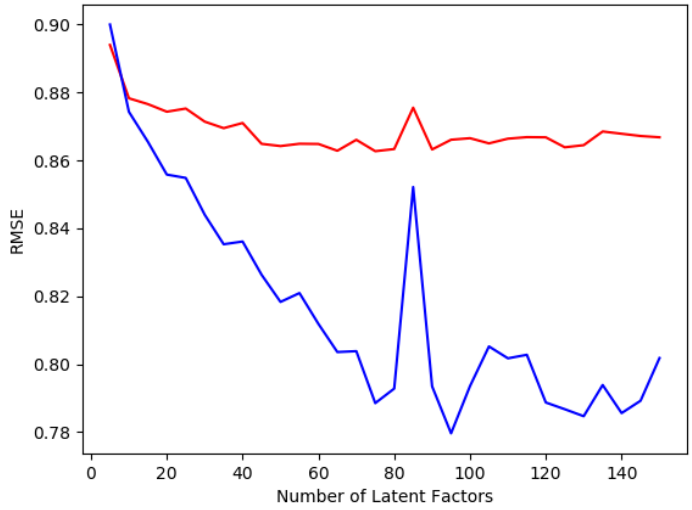



a. 左圖是 normalize 後的結果，可以觀察到 train 的速度很快就收斂，大約在 epoch=15 時 validation RMSE 就達到低點為 0.785。然而，prediction 結果送到 public 上的結果只有 0.88388。
b. 右圖是沒有 normalize 的結果，可以觀察到 train 的速度相對較慢，大約在 epoch=30 時 validation RMSE 達到低點為 0.869，在 public 的結果是 0.87568。
c. 左圖的結果在 validation 上的結果很好，但是最後無法在 public set 上得到一致的結果，很可能是由於 train Rating 與 test Rating 的資料分布情形不太一樣，在套用 train 的分佈於 test set 時，導致結果差很多。即便在從 training set 分出 validation 時也有考慮到這一點而獨立出來，但是從結果來看，也很可能是因 test 的取樣上與 train 的取樣上不太一樣（user 與 item 的分佈情形），而導致 train 與 test 的分佈差很多。

2. (1%)比較不同的 latent dimension 的結果。
(1)Model 設計在無 bias 的 MF Model 上，除了以 latent dimension 作變數，並以 5 為單位調整其值外，其他參數值並不作更動。為了得到最佳結果，每次

training 都有加上 earlystopping 之 patient= 3 的限制。

(2)實驗結果：（紅色：Validation set 之 RMSE，藍色：Training set 之 RMSE）




a. 隨著 latent factor 前期數量增加，無論是 validation 或 training set 的 RMSE 都有出現下降的趨勢，其中 training set 的表現更是明顯，代表 latent factor 數量由 5 開始增加到一定的量前，有助於幫助 loss 下降。

b. 當 latent factor 增加到一定值以上時，便無法使訓練效果獲得更好的提升。可能是因無意義 factor 量過多，導致雜訊過多而無法 train 得很好。

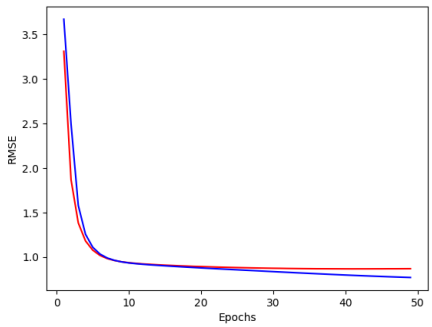
c. 當 latent factor 數量在 70 時，可以達到 validation set 之 RMSE 的最低值，約在 0.863 左右。

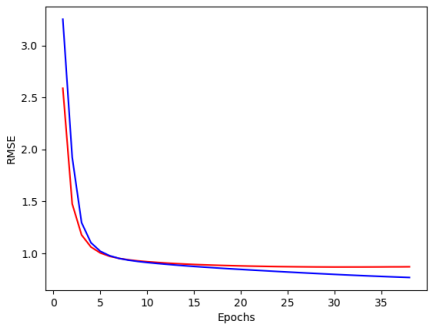
d. latent factor 在 85 時有一個明顯的峰值，明顯與趨勢不同，可能是由於 early stopping 的關係，在結果尚未收斂時就提早結束訓練了，導致結果特別差。

3. **(1%)**比較有無 **bias** 的結果。

**(1)model** 設計在 latent factor= 60 的 MF model 下，除了有無將 bias 層加入的變因以外，其他變數皆無不同。

(2)實驗結果：（紅色：Validation set 之 RMSE，藍色：Training set 之 RMSE）



a.左圖為無 bias 的結果，在 epoch=41 時達到低點在 0.869，在 public 的結果為 0.87720。

b. 右圖為有 bias 的結果，在 epoch=30 時達到低點也在 0.869，在 public 的結果為 0.87568。

c. 有 bias 的較無 bias 的 model 較快收斂，但在同一個 model 下對於預測效果沒有太大影響。很可能是由於 latent factor 已經足以達到 loss minimization 的效果，在整個在整個 flatten layer 上加上 user 與 item bias 的意義並不大。

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。

**(1)DNN** 架構：

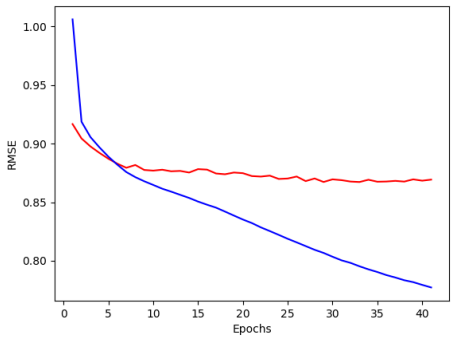
a. Embedding 的 latent factor 數量為 60，與 MF 不同的是在 user 和 item 的 embedding 被 flatten 後用 concatenate 拉長一條 layer。先後加上 Dropout= 0.1, Dense(128, activation=’relu’), Dropout= 0.1, Dense(1)組成 NN model。compile 的 loss= ‘mse’,optimizer= ‘adam’，這點與 MF model 相同。

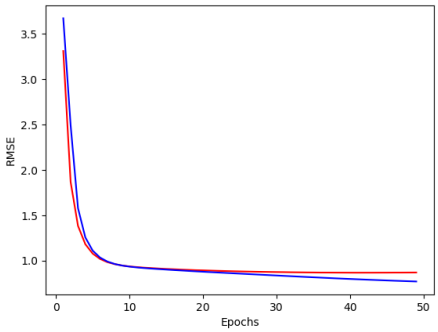
b. model.summary()

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 1) | 0 | |
| input_2 (InputLayer) | (None, 1) | 0 | |
| embedding_1 (Embedding) | (None, 1, 60) | 362460 | |
| embedding_2 (Embedding) | (None, 1, 60) | 237180 | |
| flatten_1 (Flatten) | (None, 60) | 0 | |
| flatten_2 (Flatten) | (None, 60) | 0 | |
| concatenate_1 (Concatenate) | (None, 120) | 0 | |
| dropout_1 (Dropout) | (None, 120) | 0 | |
| dense_1 (Dense) | (None, 128) | 15488 | |
| dropout_2 (Dropout) | (None, 128) | 0 | |
| dense_2 (Dense) | (None, 1) | 129 | |

Total params: 615,257.0
Trainable params: 615,257.0
Non-trainable params: 0.0

(2)實驗結果：（紅色：Validation set 之 RMSE，藍色：Training set 之 RMSE）




a. 左圖為 NN model(without bias)的結果，在 epoch= 34 時達到低點在 0.867，在 public 的結果為 0.86734。

b. 右圖為 MF model(without bias)的結果，在 epoch=41 時達到低點在 0.869，在 public 的結果為 0.87720。

c. NN model 收斂速度較快，在第一個 epoch 時就可以迅速降至 RMSE=1 上下，從最低點的表現而言，可見 NN model 的表現比 MF model 好。不改變 NN model 參數，將 NN model 送入 ensemble 模型時，其表現甚至可以達到 0.84422。

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

**(1)**結果：

採用的模型是經過 20 個 model ensemble 的 MF 模型，以下標數字對應顏色來代表該 movie 所屬 category，對於 multi-labeled movie 則是 random 取其 label。

0: Drama, Musical

1: Comedy

2: Fantasy

3: Romance

4: Sci-Fi

5: Documentary

6: Action

7: War

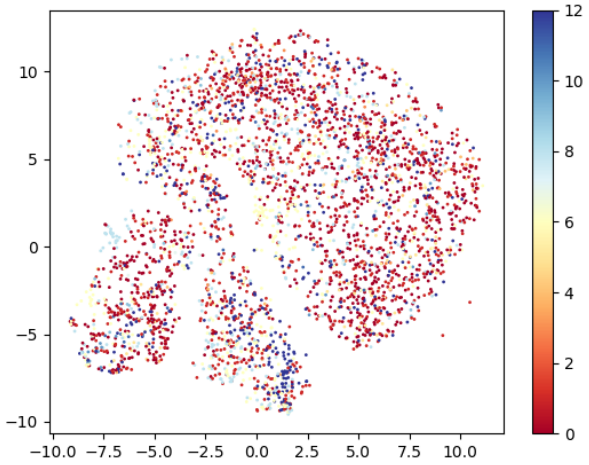
8: Animation, Children’s, Adventure

9: Mystery

10: Film-Noir

11: Western

12: Crime, Thriller, Horror



(2)觀察：

a.在降維的平面上主要分成三群，但是在類別的分佈與此並無明顯關係，可能是代表有類別以外的因素（如電影年代、地區、語言等），導致電影的 latent factor 成如此分佈。

b. Action（黃）與 Crime, Thriller, Horror（深藍）分別在中央與下方位置有相對較為各自群聚的現象，代表 latent factor 有較明顯的趨勢可以使這兩大類別的電影集中。

c. 除了一些各自形成 colony 的 movie 具有相同的顏色分佈外，大致是看不太出 latent factor 是以 movie category 作區分的現象，如同上述 a. 推測的原因。

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。

(1)實作方法：

a. Preprocessing：這裡用到的 feature 為 User 的 Gender（以 0, 1 分別表示, M, F）, Occupation, Age，Zipcode 沒使用是因為個人認為資料太過離散而且直覺上認為地區影響不大，因此沒有採用。

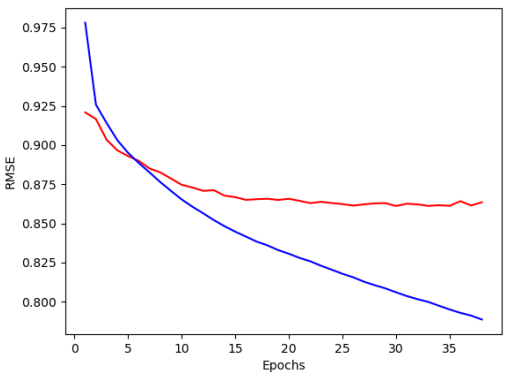
b. Model：使用的是類似第四題描述的 NN model，除了原本以 userID, movieID 作為 input 加上 embedding 的 vector 外，更加入了 Gender, Occupation, Age 作為 input 的 embedding 的 vector，最後將所有 vector concatenate 在一起後，送入 NN 中。

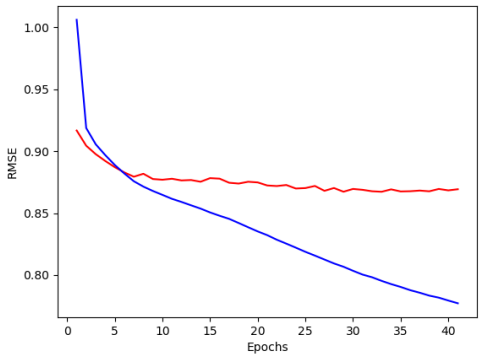
c. model.summary():

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 1) | 0 | |
| input_2 (InputLayer) | (None, 1) | 0 | |
| input_3 (InputLayer) | (None, 1) | 0 | |
| input_4 (InputLayer) | (None, 1) | 0 | |
| input_5 (InputLayer) | (None, 1) | 0 | |
| embedding_1 (Embedding) | (None, 1, 60) | 362460 | |
| embedding_2 (Embedding) | (None, 1, 60) | 237180 | |
| embedding_3 (Embedding) | (None, 1, 60) | 120 | |
| embedding_4 (Embedding) | (None, 1, 60) | 3480 | |
| embedding_5 (Embedding) | (None, 1, 60) | 1260 | |
| flatten_1 (Flatten) | (None, 60) | 0 | |
| flatten_2 (Flatten) | (None, 60) | 0 | |
| flatten_3 (Flatten) | (None, 60) | 0 | |
| flatten_4 (Flatten) | (None, 60) | 0 | |
| flatten_5 (Flatten) | (None, 60) | 0 | |
| concatenate_1 (Concatenate) | (None, 300) | 0 | |
| dropout_1 (Dropout) | (None, 300) | 0 | |
| dense_1 (Dense) | (None, 128) | 38528 | |
| dropout_2 (Dropout) | (None, 128) | 0 | |
| dense_2 (Dense) | (None, 1) | 129 | |

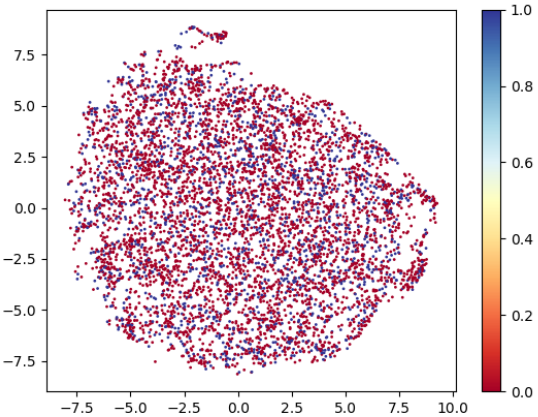
Total params: 643,157.0
Trainable params: 643,157.0
Non-trainable params: 0.0

**(2)**實驗結果：（紅色：Validation set 之 RMSE，藍色：Training set 之 RMSE）
由於這裡用到的 model 與第四題的 NN model 除了加上的額外 layers 以外，其他參數皆相同，因此在此將兩者做比較。

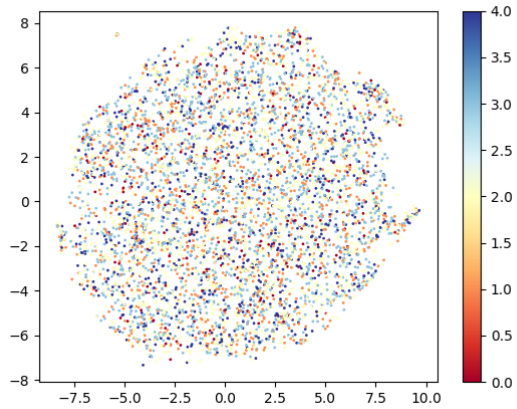


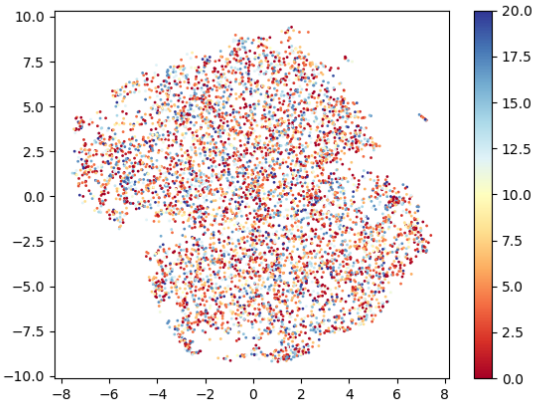

a. 左圖為加入 user feature 後的結果，在 epoch=30 時達到低點在 0.861，public 上結果為 0.86252。
b. 右圖為第四題中所用的 NN model，在 epoch= 34 時達到低點在 0.867，在 public 的結果為 0.86734。
c. 可以看到可能是由於 trainable parameter 較多的關係，新的 model 在初期 train 的速度較慢，但後來收斂的速度與本來的 NN model 差不多。
d. 新 model train 出來的結果明顯較 NN model 佳，若能 finely tune 參數，應該有機會達到更好的效果。
(3) 分析：
a. 以性別作為 tag 在 user embedding 上(0:Male, 1:Female)：



b. 以 Age 作為 tag 在 user embedding 上(0:<18, 1:18~25, 2:26~35, 3:36~49, 4:50~65, 5:>65)：




c. 以 Age 作為 tag 在 user embedding 上(tag 數字即為該 data set 所代表的 occupation)




d. 結果顯示若個別以 age, gender, occupation 作為 tag 時應無法有顯著區分 users 的效果，因為絕大部分都是融合在一起的情況。但在 training 的時候同時加入這三個 vector，結果有較原本 model 的提升，可能是因爲在個別情況較無法觀察到明顯的有助於提升學習效果的現象。