學號：R05921033 系級： 電機所碩一 姓名：余政穎

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

Ans:

我使用的 normalize 方法，是將原本的 training data 取其平均與標準差後，將 training data 減去平均後，再除以標準差得到準備丟入模型訓練的資料，有考慮進去 bias term 造成的影響，而預測出來的 test data 結果，會乘回 training data 的標準差再加上 training data 的平均，以符合原本的 Rating 預測範圍，模型的 optimizer 則是選用 adamax。

由於所看到的 error，因為 Rating 被 normalize 過，所以直接看 validation loss 看起來會變小，故我將結果上傳至 Kaggle 觀看 public score，並分別做五種不同較低維度的 dimensions，拿來預測 test data 的模型，為在該次訓練中，validation loss 最低的模型：

dimension	5	8	10	13	15
有 normalize	0.87404	0.87550	0.87854	0.87668	0.87587
無 normalize	0.87551	0.87348	0.87566	0.87601	0.87729

從上述表格可以看到，在各種 latent dimension 的情況下，有無 normalize 對於此次訓練差異不大，推測可能的原因，應該是訓練資料的 Rating 本身也就只有五種不同的結果，而且 Rating 之間的值差異本來就沒有很大，所以把資料 normalize 後依然是五種不同的結果，對於此次訓練並沒有特別的幫助。

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

Ans:

此部分我分別嘗試十種不同的 latent dimension，觀察其在 validation set(為原訓練資料的 0.1，隨機從資料中抽取)以及上傳 Kaggle 的 public score 結果，由於 keras 的 loss 衡量是使用 mse，故 validation loss 會開根號，以符合 Root Mean Square Error，我亦同時記錄最佳 validation loss 出現時，所跑的 epoch 數量；使用的訓練資料為沒有 normalize 過的 rating 以及加上 bias term 的 Matrix Factorization。

dimension	**5**	**8**	**10**	**13**	**15**
Epochs	152	77	62	42	45
Validation	0.88131	0.87390	0.87858	0.87356	0.87676
Kaggle Public	0.87551	0.87348	0.87566	0.87601	0.87729
dimension	**20**	**25**	**30**	**40**	**50**
Epochs	30	25	22	19	16
Validation	0.87515	0.87361	0.87767	0.87584	0.87983
Kaggle Public	0.87627	0.87553	0.87766	0.87954	0.87537

從上表中可以看到，就準確率而言，dimension 在維度高些的時候(如 40、50)，平均而言較差，可能是因為容易 overfit 所導致；不過，若維度太小，所需要花費的訓練時間便比較多，特別是當 dimension 只有 5 的時候，所需要花的 epoch 數達到了 152，是 dimension 為 8 時的兩倍，就效率上來看，選擇 10~25 的 dimension 是比較好的。

3. **(1%)比較有無 bias 的結果。**

Ans:

在這個實驗中，我測試四種不同 bias term 的設定，包含有 user_bias 跟 movie_bias、有 user_bias 但沒有 movie_bias、沒有 user_bias 但有 movie_bias、沒有 user_bias 也沒有 movie_bias 這四種狀況，看看預測出來的 validation loss 跟 Kaggle public score 如何，而此題的 latent dimension 設為 13，沒有 normalize 過，其他設定與第一題相同。

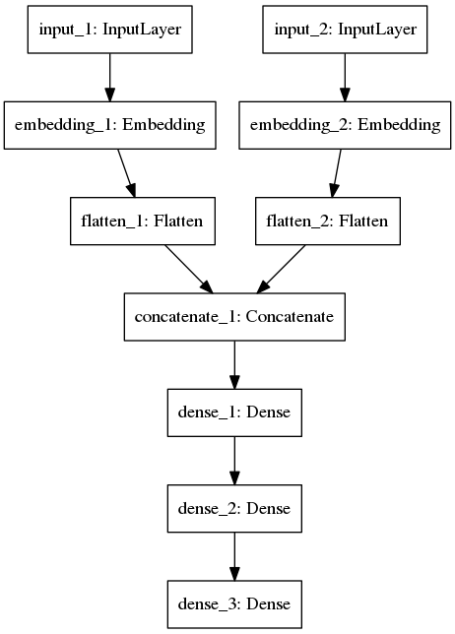
	有 user_bias 有 movie_bias	有 user_bias 無 movie_bias	無 user_bias 有 movie_bias	無 user_bias 無 movie_bias
Validation	0.87356	0.87235	0.87840	0.88623
Kaggle Public	0.87601	0.87658	0.87658	0.88307

從上述結果可以看到，有無 bias term 其實對於訓練結果有些影響，完全沒有 bias term 跟兩個 bias term 都考慮進去的結果相差快要 0.01 左右，在這次的 task 中算是滿大的差異，因為 bias term 所代表的意義是每個 user 的個人偏好以及每個 movie 的知名度等等，因此不同 user 或不同 movie 之間會有差異也是合理的；其中，user bias term 的效果又大於 movie bias term，這部分應是因為 Rating 較容易受到個人主觀或打分偏好影響，所把這部分考慮進去對於訓練的幫助效果較為顯著。

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。**

Ans:

我使用的 DNN 模型，是將 user_vec 跟 item_vec 取 embedding(embedding 維度為 13 維)，Flatten 後再將其 concatenate 形成一個 input 的 feature vector，並分別接大小 150 以及 50 的 relu layer，最後 output 出一個值，直接當成是一個 regression 的 problems，其他的參數設定與先前的 Matrix Factorization 模型相同，模型架構如下圖：

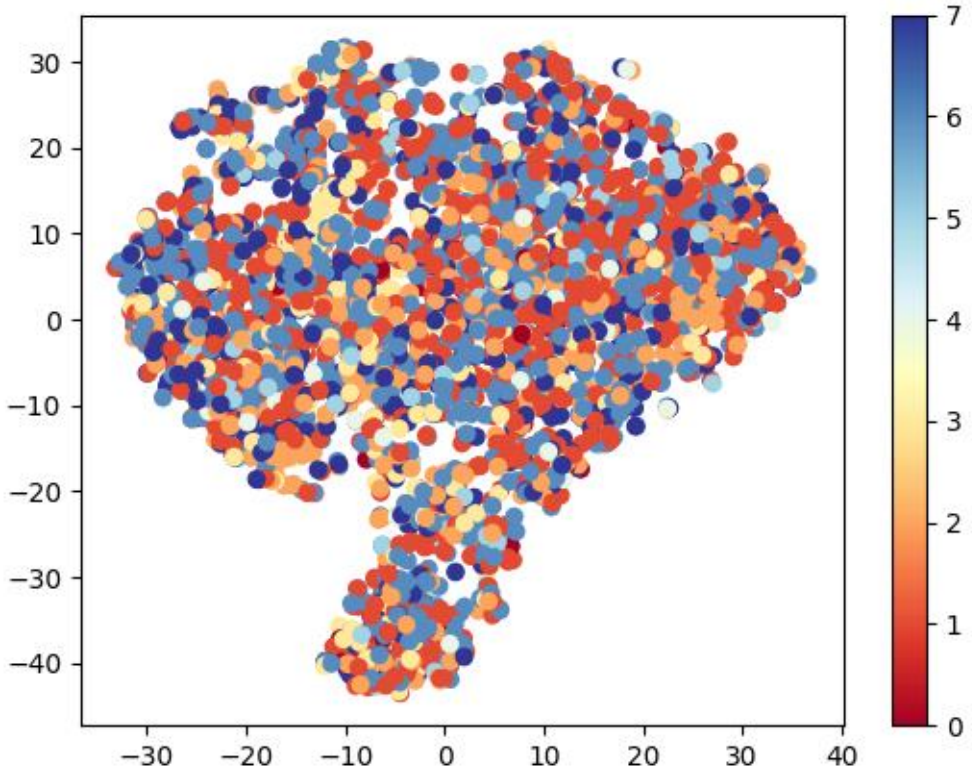


以此模型，我只花了 9 epochs，便可以跑出 validation loss 為 0.87613，上傳 Kaggle 的分數為 0.87794，可以發現，這個模型目前的效果並沒有特別優於 MF，但效果其實也不差，在 DNN 架構其實並不複雜的情況下，我們在很少的 epochs 便可以達到不錯的預測準確率，若是針對 DNN 架構有所調整(比方說增加 layer 數量、加入 dropout 等技巧、加入其他 features 如電影種類、使用者資訊)，有可能表現得比單使用 Matrix Factorization 來的好，這部分未來若真的要做的更加準確，可以考慮使用！

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

Ans:

我先將總共 18 個 category 先預先分類，Drama、Musical 分為第一類，Thriller、Horror、Crime 分為第二類，Adventure、Animation、Children's 為第三類，Fantasy、Sci-Fi、Mystery 為第四類，Documentary、Film-Noir 為第五類，War、Western、Action 分為第六類，Comedy、Romance 為第七類，以減少原始類別太細而分不開的狀況，而對於複數 labels 的 movie，我便隨機從中取一種 category 當成 label，plot 出的結果如下：



可以發現到，基本上最深的藍色，也就是 comedy 跟 romance 比較集中在上面，第三類的黃色(Adventure、Animation、Children's)集中在左半邊，右半邊比較少出現，第五類淺藍

色(Documentary、Film-Noir)則比較出現在上面；不過總體上來看，資料點並沒有被分得很開，這可能是因為我們粗略的把原本 18 種 categories 切成七大類，並沒有切得很準，而且七類也還是有點多，以致看起來仍有些眼花撩亂，另外，亦有可能是我們對於多個 labels 的 movie 是隨機取其中一種 label，而不是選出最像那部電影的分類(事實上也很難選)，故其中所造成的失準，亦是造成圖上似乎沒有切得很開的原因。

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。**

   Ans:

我將 users.csv 中其他欄位的資料，也選取一些拿進去 DNN 的 model，當作 features 之一訓練，實作方式是將 user.csv 的 Gender、Age、Occupation 這三個欄位，跟 user_vec 和 movie_vec 一起丟入 merge_vec 裡面，讓 merge_vec 再多三個維度，前面的部分就照第四題所說的方式去將兩個 embedding 的 vector 做 flatten，之後再三個一起 concatenate，最後丟入一個跟第四題一樣架構的 DNN 模型訓練，總結來說，與第四題的差別是將 concatenate_1 多加一個三維 vector，將這個人的 Gender、Age、Occupation 這三種 features 也考慮進去。

測試出來的結果，花了 11 epochs，便跑出 validation loss 為 0.87263，上傳 Kaggle 的分數為 0.87350，可以看到，即使只是多把一小部份其他資訊加入，也多花了一些時間到達 validation loss 最低之處，但似乎有比第四題的結果好些，表示多將其他資訊加入，應是對訓練有幫助的！