學號：B03705027 系級： 資管三 姓名：鄭從德

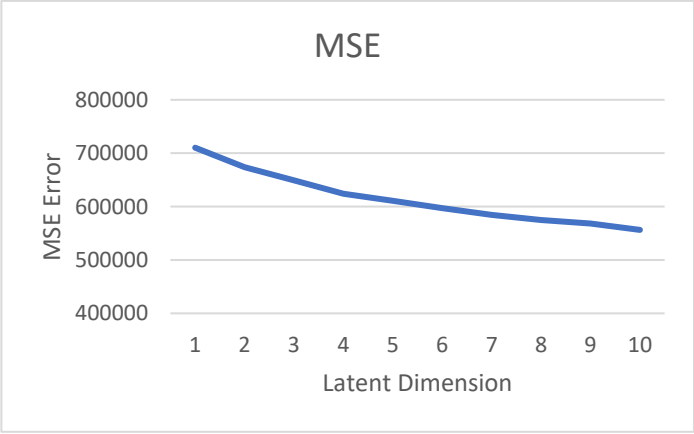
(1) 請比較有無 normalize(rating)的差別。並說明如何 normalize.

我使用的是 standard normalization。把一開始的 rating matrix 進行了 standard normalization 之後拿去 train，在 testing data 預測出來之後每個值再乘上 training data 的 std 並加上 mean，就回成為變形回來的 rating 值。

我原本是用 MF 做出來的結果為 0.8643，用同一個 model 進行了 normalization 之後結果卻變糟了，RMSE 超過 0.92。我覺得在本題中做標準化是不合理的，因為本來 rating 就已經介於 1-5 之間，可以看成已經 normalize 過的結果；而且 rating matrix 有過多的 0，造成全部 matrix 的 mean 幾乎都趨近於零。這本身就是一個非常態的分布，用 std 做標準化的結果並不會太理想。

(2) 比較不同的 latent dimension 的結果。

我實作了 latent dimension 為 1~10 以及 15、20、30 的訓練，同樣因為 CPU 效能有限的問題，我每一種 latent dimension 僅做了 200 個 epoch 的訓練。我將 Latent dimension 對 training mse 以及最後 testing RMSE 做圖得如下結果：



K=	RMSE
1	0.9116
3	0.8742
5	0.8643
7	0.8643
10	0.8696
15	0.8817
20	0.8733

在我實驗最大的情況 latent dimension=30 的時候，training error 可以降到 443152。但是我們如果把這些結果丟到 kaggle 上就會發現，基本上後面的幾個都已經嚴重的 overfit 了。而從右圖的 testing 結果我們可以發現，大約在 latent dimension 為 5 到 7 之間時，效果是最佳的，再來約到 10 以上就會有 overfit 的現象。

(3) 比較有無 bias 的結果。

我實作了 Latent Dimension = 5, 7 時加入 bias 的 MF 分解，結果並沒有太大的幫助。我在 latent dimension=5 時結果為幾乎相同的 0.8649，latent dimension=7 時卻變成高許多的的 0.868。再 training error 方面，有加上 bias 的 model error 降得比較低，可能有造成了 overfit 的現象。為了確定效能沒有提升是否來自 overfit 的發生，我試著降低 training epoch，卻發現 RMSE 上升了，在僅 train 100 個 epoch 的情況下，兩者的 error 都上升到 0.88 左右。

其實 training error 降的比較低是很合理的，因為 bias 一定程度上就是在修正 error。我試著調整 learning rate 但是效果並不理想。因此我認為在我的 model 中加入 bias 並無法有效改善其有效性。

(4) 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。

本題我試驗了把 output 當作分類問題以及單純數字兩種方法。我的作法是將 user ID 以及 movie ID concatenate 起來之後直接丟到 DNN 裡面去 train。我用 categorical 做出來的結果並不理想，最佳的成績上傳 kaggle 也只有 0.87 左右，倒是我把 output 設成單一 rating 分數做出來的結果比較好，RMSE 可以到 0.8519，這也是我最好的成績！

以下是我的 model 架構，我試驗了許多組合，發現其實 Dense 太多層並不會讓效果變好，而 kernel 開太大也會造成反效果。經過實驗，三層分別為 512, 256, 128 的效果最佳，勝過 kernel 數開到 4096 的訓練結果。

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 40)	158120	
embedding_2 (Embedding)	(None, 1, 40)	241640	
flatten_1 (Flatten)	(None, 40)	0	
flatten_2 (Flatten)	(None, 40)	0	
dropout_1 (Dropout)	(None, 40)	0	
dropout_2 (Dropout)	(None, 40)	0	
merge_1 (Merge)	(None, 80)	0	
dense_1 (Dense)	(None, 512)	41472	
dropout_3 (Dropout)	(None, 512)	0	
batch_normalization_1 (BatchNorm	(None, 512)	2048	
dense_2 (Dense)	(None, 256)	131328	
dropout_4 (Dropout)	(None, 256)	0	
batch_normalization_2 (BatchNorm	(None, 256)	1024	
dense_3 (Dense)	(None, 128)	32896	
dropout_5 (Dropout)	(None, 128)	0	
dense_4 (Dense)	(None, 1)	129	

Total params: 608,657.0

其中我的 optimizer 是使用 Adam，loss function 是 MSE。其實發現這個 model 效率這麼好滿驚訝的，因為他並沒有太複雜的架構，但是卻比我使用 MF 以及其他的方法都還要優！

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

這是我做出來最漂亮的一張圖，是利用 MF 分解，latent dimension=7 的 movie matrix 去進行 TSNE 得出的結果。顏色與類別對應如下：

藍色：Thriller, Horror, Crime, Film-Noir

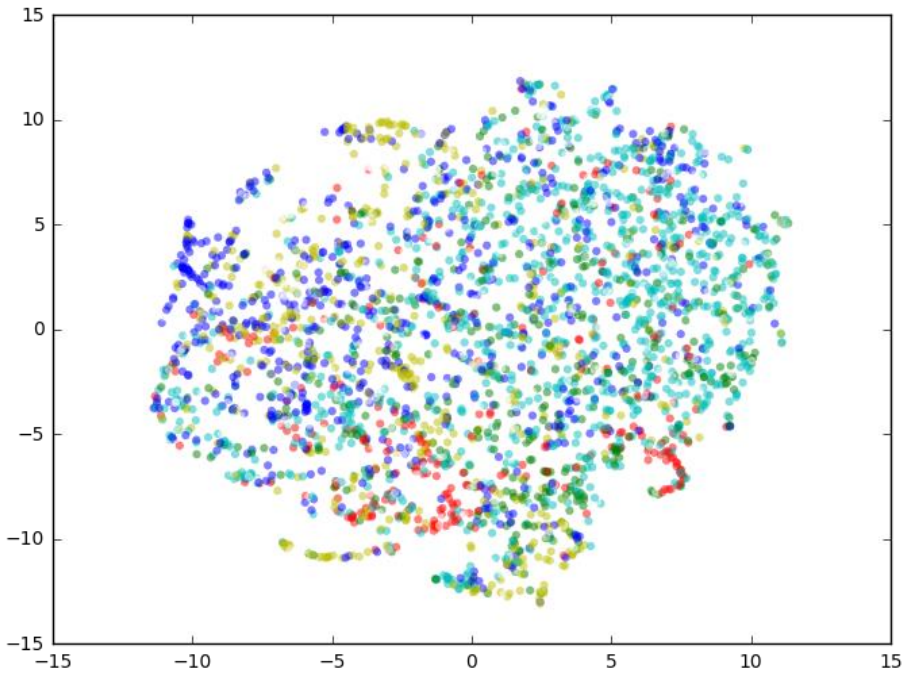
紅色：Animation, Childrens'

黃色：Action, Adventure

綠色：Romance

淺藍：Drama, Musical

其他：白，因為分部過度均勻怕影響視覺而拿掉



我們可以看出，紅色(Children 系列)大多位於右跟中下，藍色(Horror 系列)大多位於左上，算是分得很開的兩類；而綠色(Romance 系列)有一小群更集中右下角，而且跟紅色(Children 系列)有滿明顯的分隔，一樣屬於 horror 的反面。而右邊綠色(Romance)跟淺藍色(Drama)有些混合，也是可以理解的；淺藍色的 drama 分布極廣，但還是多位於右上角那一塊，雖然有與藍色的 horror 混雜，但沒有混到 horror 最多的左上角。最後，黃色的 Action movie 在下半部跟一群綠色(Romance)及一群紅色(應該是 Animation 的成分)十分靠近，也在左上有跟 horror 系列混雜，說明恐怖片也是有很多動作&冒險成分。

雖然我們可以用這張圖找出不同的 category 之間的關係，但我覺得有幾個會造成分類不明確的原因：1.同一個家庭可能會使用同一個 User ID，2.太有名的電影其實並不符合偏好原則。無論如何，這個做圖的結果都比我預期的要好得多。