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

在我的 MF 中，有做 normalize ($Y_{train} = (Y_{train} - Y_{train\_mean}) / Y_{train\_std}$)的話，在 Kaggle 上的 RMSE 反而更糟，但若是我把 rating 做放大的效果($Y_{train} = (Y_{train} - 1) * 10$)或($Y_{train} = (Y_{train} - 1) * 100$)，反而是把 rating 放大越多倍的話，Kaggle 上的 RMSE 越有好的成績。就這樣的數據看來，我覺得 MF 是不需要做 normalize 的，而若是用 DNN 來解決這次作業的話，才有可能靠 normalize 來提高 Kaggle 分數。而 MF 只是經過兩個 embedding layer，再通過矩陣相乘與加法，在 embedding layer 出來的東西已經是被 normalize 過了，通過矩陣運算後再去計算 loss (MSE、RMSE)，若把 rating 做 normalize 後，loss 會都是非常小的數字；反而是把 rating 放大後，loss 也可以跟著被放大，則可以看出更多細微變化，讓整體 RMSE 更小。

	無 normalize	有 normalize	減 1 乘 10	減 1 乘 100
Rating range	1 ~ 5	大約-3 ~ 1.5	0 ~ 40	0 ~ 400
Normalized	No	No	No	No
Latent dimension	64	64	64	64
Embedding dropout	0.2	0.2	0.2	0.2
Epochs (Early stop)	26	6	76	283
Best validation RMSE	0.86360	0.78770	8.5293	84.8005
Public score (RMSE)	0.86158	0.87793	0.85240	0.84821
Private score (RMSE)	0.86425	0.87982	0.85384	0.84756

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

我發現 latent dimension 越小，則需要計算越多個 epoch 才會 early stop，而 latent dimension 越大則越少 epoch。但是 dimension 太大又會讓 RMSE 在 training set 與 validation 上都壞掉，所以 latent dimension 不能太大也不能太小，在我的 model 中，dimension 為 32 是最棒的！可以用這個 dimension 再回去把 validation 拿掉，全部重新 train 一次，能有更好的結果！

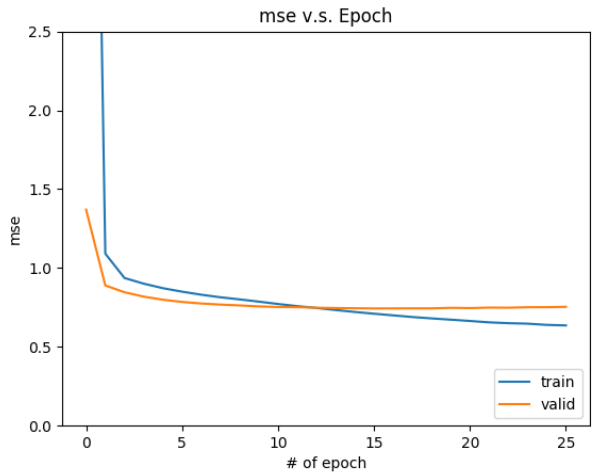
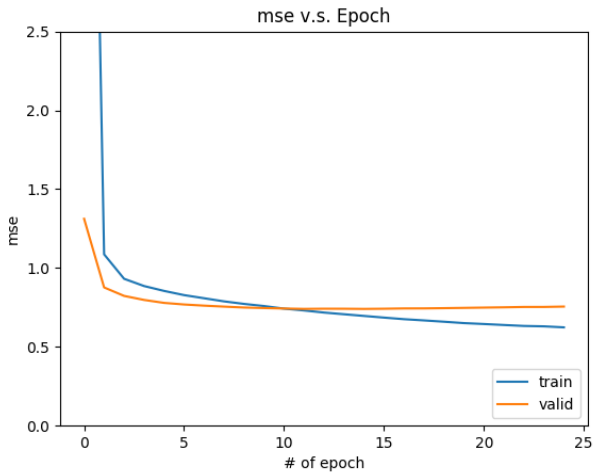
Latent dimension	16	32	64	100	128	256
Normalized	No	No	No	No	No	No
Embedding dropout	0.2	0.2	0.2	0.2	0.2	0.2
Epochs (Early stop)	162	111	26	15	12	6
Best validation RMSE	0.87790	0.86040	0.86360	0.86890	0.87500	0.89110
Public score (RMSE)	0.87778	0.86093	0.86425	0.86928	0.87391	0.88695
Private score (RMSE)	0.87673	0.86017	0.86158	0.86734	0.87442	0.88871

3. (1%)比較有無 bias 的結果。

(collaborator:)

加上 bias 項是有必要的，每個人評分一個東西，都有一定傾向，例如有些人喜歡打高分，有些人喜歡打低分。而每個電影也是類似，有些電影爛到大家都給低分，有些電影好到大家都給高分。所以加上 bias 項會讓 RMSE 下降，讓 model 預測更準確。

	無 bias	有 bias
Normalized	No	No
Latent dimension	100	100
Embedding dropout	0.2	0.2
Epochs (Early stop)	25	24
Best validation RMSE	0.8619	0.8603
Public score (RMSE)	0.85641	0.85556
Private score (RMSE)	0.85651	0.85608

無 bias	有 bias
	

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。

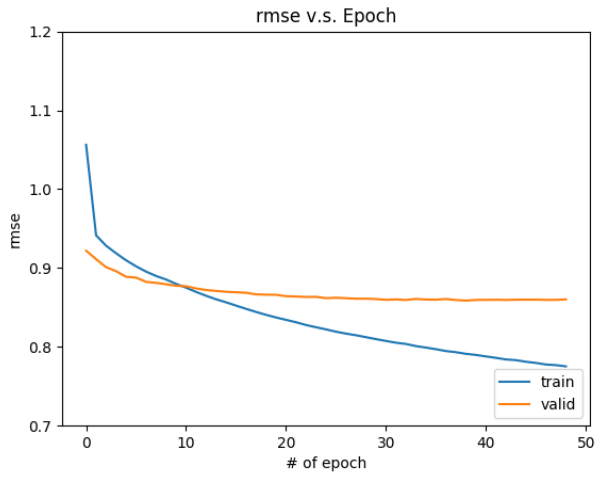
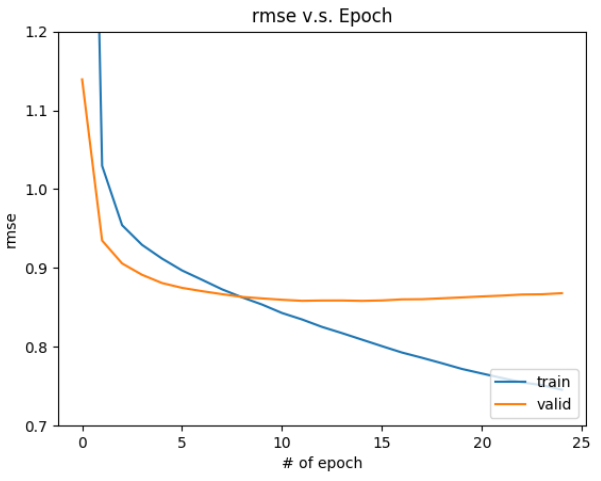
(collaborator:)

我的 DNN 的前半段跟 MF 一樣，只是當 MF 要做 DOT 的時候，DNN 把兩組 Vector 並起來，整個丟到 Fully connected layer，而其因為要訓練的參數比較多，所以要比較多的 EPOCH 後才會收斂，但其實給他很多個 EPOCH 後，就可以非常 overfit 在 training set 上。而總體分數 DNN 是比 MF 是好一點的，但其實沒有差很多。

我也有額外試過把 user embedding 與 movie embedding 分別通過 DNN 後再去 DOT 得出 rating 的方法，但是其效能跟我上一個 DNN 差不多，所以我就沒畫圖與數據了。

```
112     print('Building model...')
113     userInput = Input(shape=(1, ), name='userInput')
114     userEmbedding = Embedding(userNumber, EMBEDDING_DIM)(userInput)
115     userEmbedding = Dropout(0.2)(userEmbedding)
116
117     movieInput = Input(shape=(1, ), name='movieInput')
118     movieEmbedding = Embedding(movieNumber, EMBEDDING_DIM)(movieInput)
119     movieEmbedding = Dropout(0.2)(movieEmbedding)
120
121     userEmbedding = Flatten()(userEmbedding)
122     movieEmbedding = Flatten()(movieEmbedding)
123     predict = Concatenate()([userEmbedding, movieEmbedding])
124     predict = Dense(512, activation='relu')(predict)
125     predict = Dropout(0.5)(predict)
126     predict = Dense(1, activation='relu')(predict)
127
128     model = Model(inputs=[userInput, movieInput], outputs=[predict])
```

	DNN	MF
Normalized	No	No
Latent dimension	100	100
Embedding dropout	0.2	0.2
Epochs (Early stop)	49	24
Public score (RMSE)	0.85487	0.85556
Private score (RMSE)	0.85564	0.85608

DNN	MF
	

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

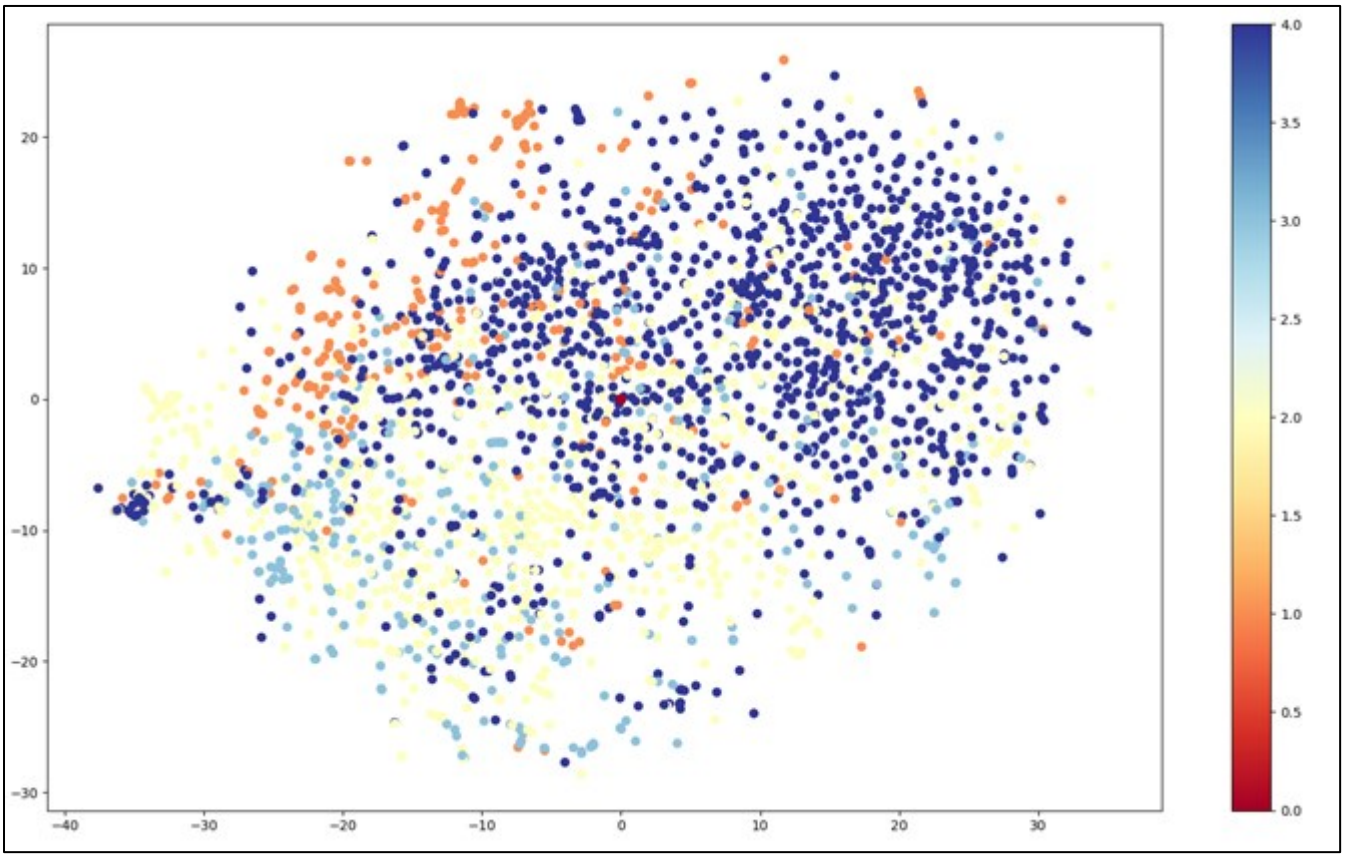
橘色(值為 1)是 Animation、Children's、Fantasy，集中在左上角。

淡黃色(值為 2)是 Thriller、Horror、Crime，平均散佈在下方偏左。

淡藍色(值為 3)是 Action、Adventure，散布很廣，但有稍微聚集在左下角。

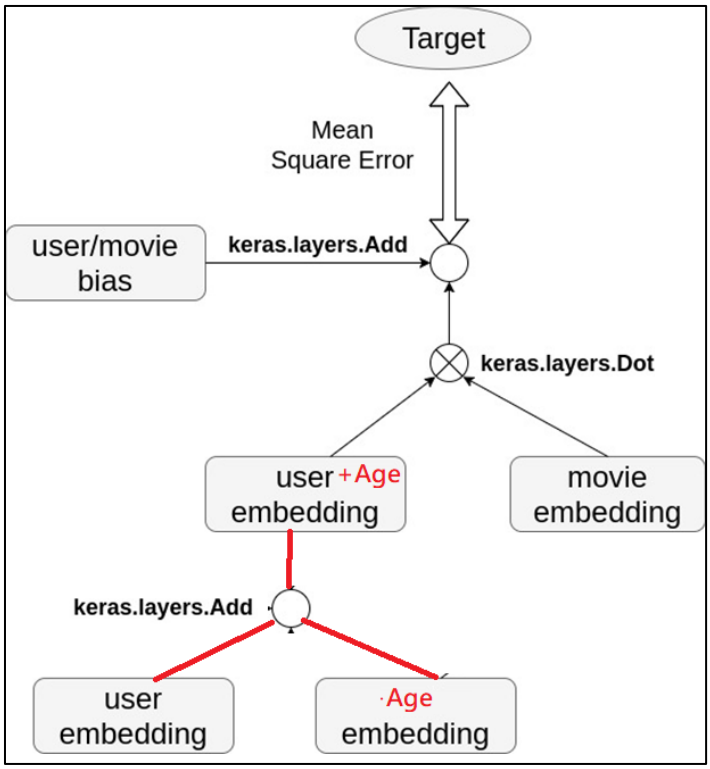
深藍色(值為 4)是 Drama、Musical，數量超多，主要分布在右上角。

而其他類的電影則被我以紅色(值為 0)畫在(0, 0)的位置上，避免讓畫面很亂。



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。(collaborator:)

除了基本的 user 與 movie 以外，我從 user.csv 中額外拿了每個 user 的 age 去做 embedding，再把 age embedding 與 user embedding 相加後，其他就和助教給的架構圖一樣了。而這個 model 在 training set 上在 25 個 epoch 就比基本的 MF 還要 overfit 了，但 validation set 上與 Kaggle 上的分數就沒那麼理想了。或許 user age 並不是一個好的 feature，也或是需要將 movie 的類別也 embedding 後拿進來一起 train 才會有更好的結果。



	基本的 biased MF	增加 Age 的 biased MF
Normalized	No	No
Latent dimension	100	100
Embedding dropout	0.2	0.2
Epochs (Early stop)	25	28
Public score (RMSE)	0.85556	0.86162
Private score (RMSE)	0.85608	0.86063

基本的 biased MF	增加 Age 的 biased MF