學號：B03705028 系級： 資管三 姓名：陳星宇

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

1.1 如何 normalize：

1.1.1 先算出 train.csv 中 ratings 的 mean 與 std，再將 ratings 先減掉 mean，再除以 std，並將處理完的 ratings 作為 label 丟進 model。

1.1.2 要 predict 時，將 model predict 出的 ratings 先乘上 1.1 的 std，再加上 1.2 的 mean，作為最後送上 Kaggle 的結果。

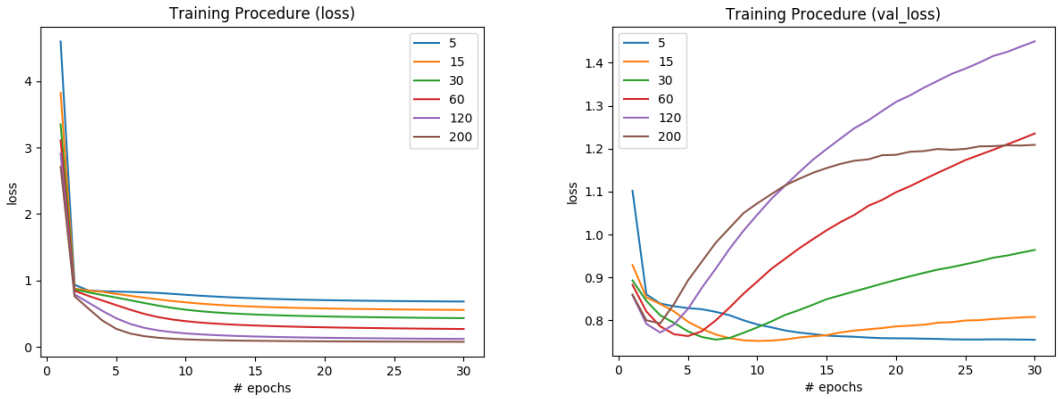
1.2 有無 normalize 的差別 (Kaggle 上的結果)：

without normalization: 0.86035
with normalization: 0.86367

1.3 normalization 的意義

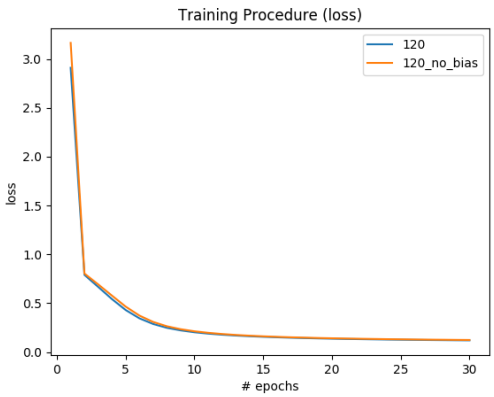
原本 model 是直接預測 rating 的值，但經過 normalization 之後，變成預測 rating 的值偏移平均幾個標準差。然而我們不知道母體的平均與標準差，所以只好用 train.csv 裡的 ratings 估計，我猜想可能是估計值與實際值有差距，才會造成 normalization 之後表現反而略差。

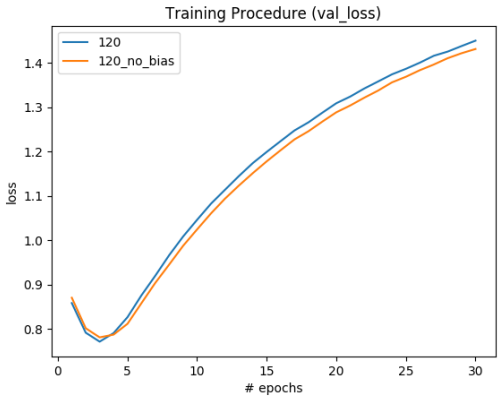
2 (1%)比較不同的 latent dimension 的結果。



Legend 裡的數字代表 latent dimension。Training loss 方面(左圖)，可以發現 latent dimension 越大，最後收斂的 training loss 越低。而 validation loss 方面(右圖)，反而是 latent dimension 越大，圖上的 validation loss 的 minimum 就越大，而且也越容易 overfitting。

3　(1%)比較有無 bias 的結果。





此題以 latent dimension=120 的 MF model 為例。由左圖 (training loss)可以發現，兩者在訓練過程中的表現幾乎相同。再看左圖(validation loss)，可以發現沒有 bias 的 model 甚至表現比有 bias 的 model 略好。

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。

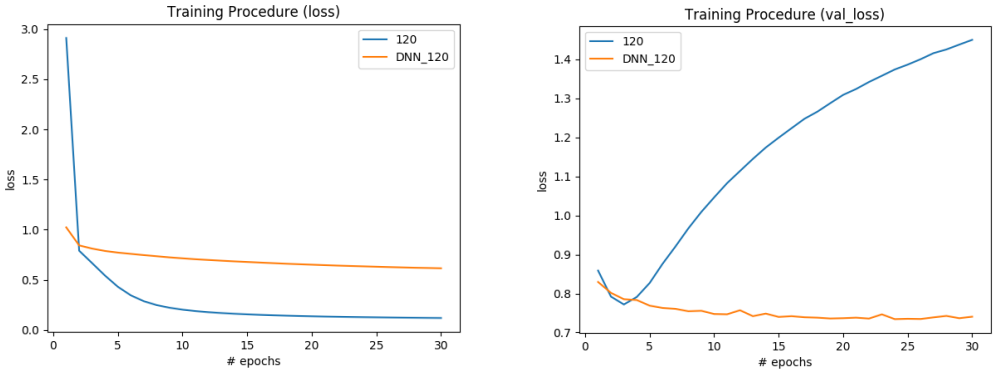
4.1　DNN model 架構：

將 MF without bias 的版本的 dot 改成 concatenate，然後再多接一層 Dense layer，模型架構如下圖。

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 120)	724800	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 120)	465960	input_2[0][0]
reshape_1 (Reshape)	(None, 120)	0	embedding_1[0][0]
reshape_2 (Reshape)	(None, 120)	0	embedding_2[0][0]
concatenate_1 (Concatenate)	(None, 240)	0	reshape_1[0][0] reshape_2[0][0]
dropout_1 (Dropout)	(None, 240)	0	concatenate_1[0][0]
dense_1 (Dense)	(None, 120)	28920	dropout_1[0][0]
p_re_lu_1 (PReLU)	(None, 120)	120	dense_1[0][0]
dropout_2 (Dropout)	(None, 120)	0	p_re_lu_1[0][0]
dense_2 (Dense)	(None, 1)	121	dropout_2[0][0]

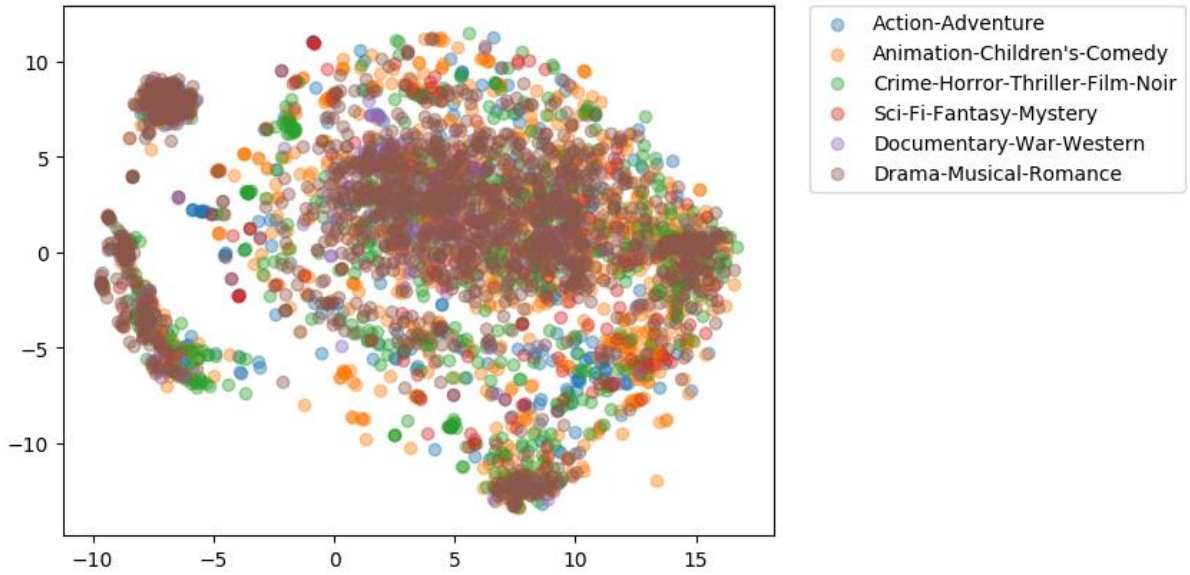
Total params: 1,219,921
Trainable params: 1,219,921
Non-trainable params: 0

4.2　MF 與 DNN 的比較 (latent dimension 皆為 120)：



在 training loss 方面 (左圖)，DNN 明顯比 MD 緩慢；然而，在 validation loss 方面 (右圖)，MF 很快就 overfitting 了，而且 DNN 在訓練過程中的 local minimum of loss 也比 MF 小。

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



此題使用 latent dimension=120 的 MF model 的 embedding matrix weights。我將性質比較類似的 genres 歸類到同一個 category，另外為了顯示點的密集程度，我將每個點的透明度調到 0.4。圖中比較明顯的 cluster 應該是左上角的棕色，還有一些零星的綠色。在繪圖過程中，我想到 embedding matrix 的值除了受到 genre 影響，也受到 rating 影響，如果把電影用 rating 分類，也許可以觀察到 cluster。

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。

6.1 作法

6.1.1 在 users.csv 中，將 Gender 與 Occupation 視為 categorical data，Age 視為 continuous data，忽略 Zip-code。最後每一個 UserID 對應到的 feature vector 為 2+1+21=24 維。

6.1.2 在 movies.csv 中，將 Genres 視為 categorical data，每一個 movie 可以有 multi-label。最後每一個 MovieID 對應到的 feature vector 為 18 維。

6.1.3 在 train.csv 中，使用 6.1.1 與 6.1.2 得到的 ID to feature vector 的 mapping，對於每一筆 training data 再 concatenate UserID 與 MovieID 分別得到的 feature vector。最後每一筆 training data 的 feature vector 為 24+18=42 維。

6.1.4 將 ratings 作為 label，視為解 linear regression 問題。

6.1.5 模型架構。

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1024)	44032
p_re_lu_1 (PReLU)	(None, 1024)	1024
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 512)	524800
p_re_lu_2 (PReLU)	(None, 512)	512
dropout_2 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 256)	131328
p_re_lu_3 (PReLU)	(None, 256)	256
dropout_3 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 128)	32896
p_re_lu_4 (PReLU)	(None, 128)	128
dropout_4 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 64)	8256
p_re_lu_5 (PReLU)	(None, 64)	64
dropout_5 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 1)	65

Total params: 743,361
Trainable params: 743,361
Non-trainable params: 0

6.2 結果

將 6.1 的模型與使用 embedding matrix(latent dimension 為 120)的 DNN 以及 MF 做比較。由於不考慮每 user 之間或是 movie 之間的獨特性與相似性，bonus 的 model 不管是在 training data 或是 validation data 的表現都是最差，而且可發現其曲線形狀與使用 embedding matrix 的 DNN 相似，但表現差了一大截。