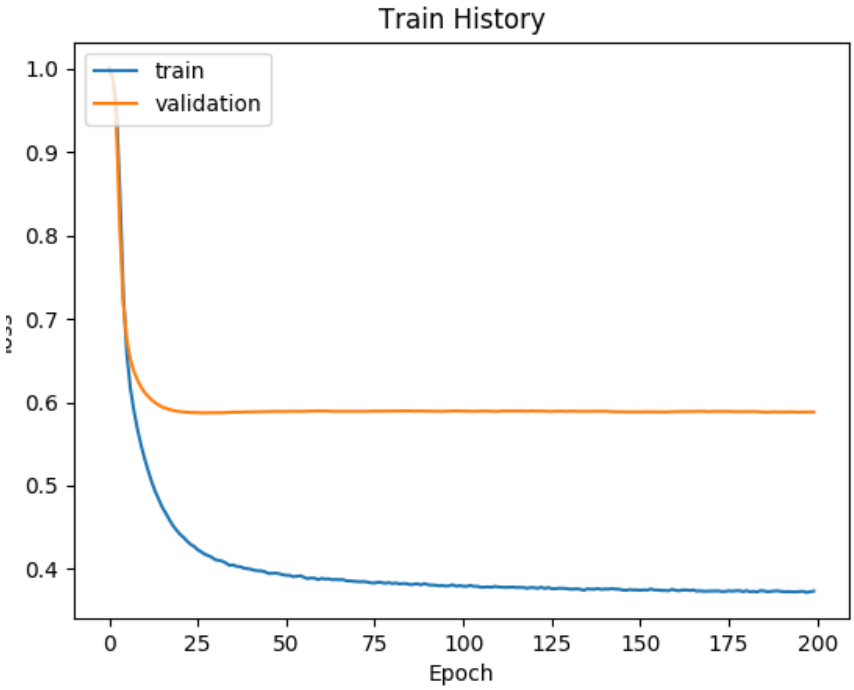
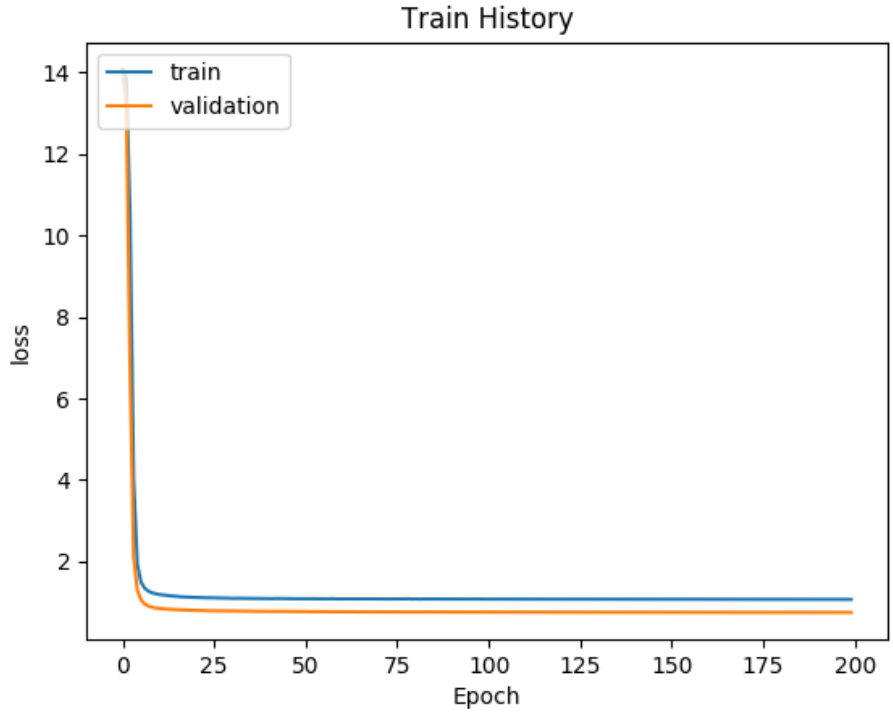
學號：B04901056 系級： 電機三 姓名：張承洋

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

With normorlization（圖 1）：



Without normorlization（圖 2）：



我的 normalization 方法是將 rating 減掉平均再除以標準差，同時把平均跟標準差的值記錄下來，在做預測的時候將結果乘回標準差再加回平均。
從上面兩張圖可以看出，做了 normalization 之後無論是 training 還是 validation 的 loss 都有顯著的下降（尤其 training loss 表現更好），可見把 bias 的因素去除

之後 model 能有更好的表現。

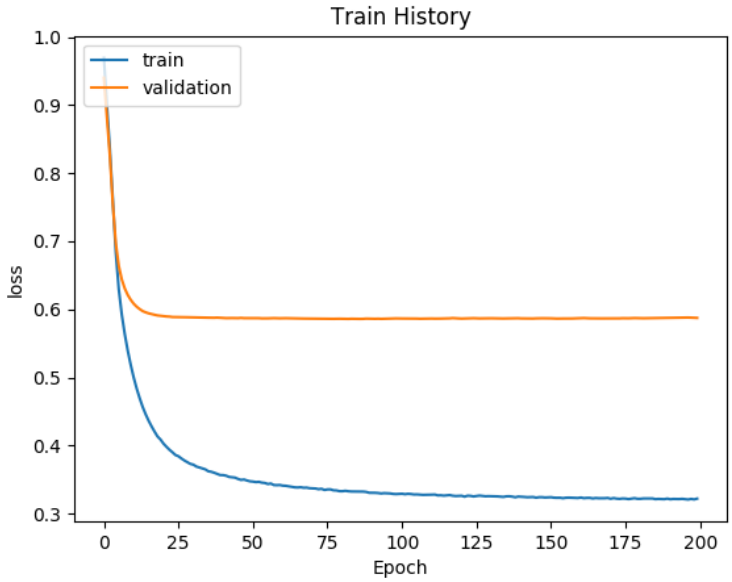
2. (1%)比較不同的 latent dimension 的結果。

   由下表可知，當 latent dimension 到達一定值後（大約 256），loss 便不再下降，我想是因為這樣的維度已經足夠紀錄 input data 的資訊，當 latent dimension 繼續上升時發現 loss 反而有些微上升，我猜是給的 information 不足以應付這麼多的參數所導致。

Latent dimension	32	64	128	256	512	1024
Validation_loss	0.6042	0.5884	0.5880	0.5880	0.6010	0.6074

3. (1%)比較有無 bias 的結果。

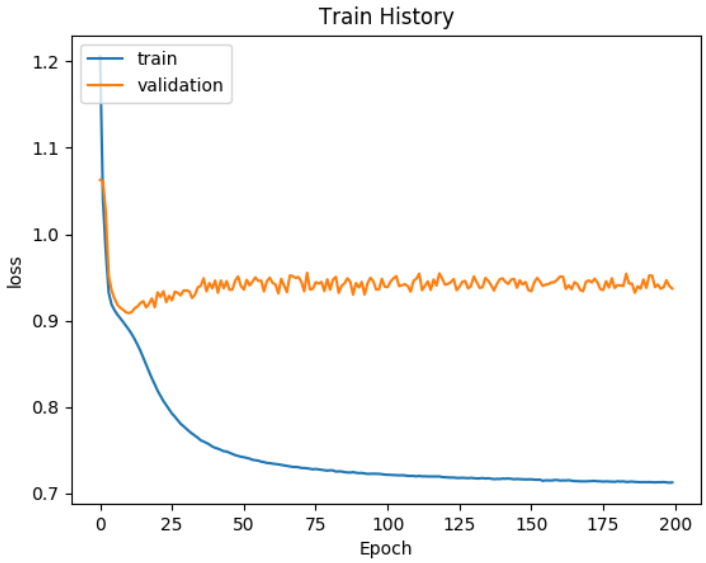
With bias（圖 3）：



（Without bias 如圖 1）加了 bias 之後，validation loss 有些微的進步，training loss 則有較明顯的下降，推測是因為每個人對於電影的評價都有主觀的高（低）標準，考慮 bias 之後能讓 training 的過程去除掉這個因素，獲得更好的表現。

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。

DNN（圖 4）：

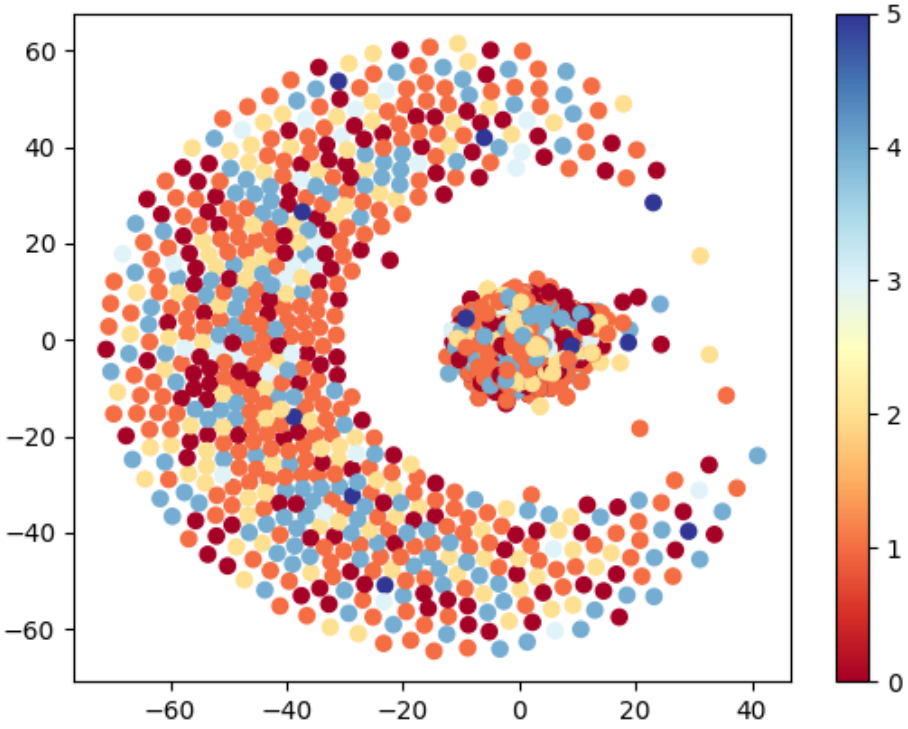



```
concat = keras.layers.Concatenate()([User_reshape, Movie_reshape])
dnn = keras.layers.Dense(256, activation='relu')(concat)
dnn = keras.layers.Dropout(DROPOUT_RATE)(dnn)
dnn = keras.layers.BatchNormalization()(dnn)
dnn = keras.layers.Dense(256, activation='relu')(dnn)
dnn = keras.layers.Dropout(DROPOUT_RATE)(dnn)
dnn = keras.layers.BatchNormalization()(dnn)
dnn = keras.layers.Dense(256, activation='relu')(dnn)
dnn = keras.layers.Dropout(DROPOUT_RATE)(dnn)
dnn = keras.layers.BatchNormalization()(dnn)
output = keras.layers.Dense(1, activation='relu')(dnn)
model = keras.models.Model(inputs=[User_input, Movie_input], outputs = output)

model.compile( loss='mean_squared_error', optimizer='adam' )
model.summary()
train_history = model.fit([user, movie ], ranking, batch_size=10000, epochs=200 , validation_split=0.1 )
```

我實作 DNN 的方法是將 user_id、movie_id 的 embedding 的結果 reshape 再 concatenate 起來，後面接 3 層 DNN 來訓練。我覺得這次的作業主題比較簡單，所以 MF 就可以有很好的表現，在我的測試中 DNN 的表現比 MF 還要差，而且由上圖可以觀察出 validation loss 還有振盪的情況發生，但我認為只要微調參數或加上其他 feature 讓 DNN 訓練，能得到比單純 MF 還要好的表現。

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

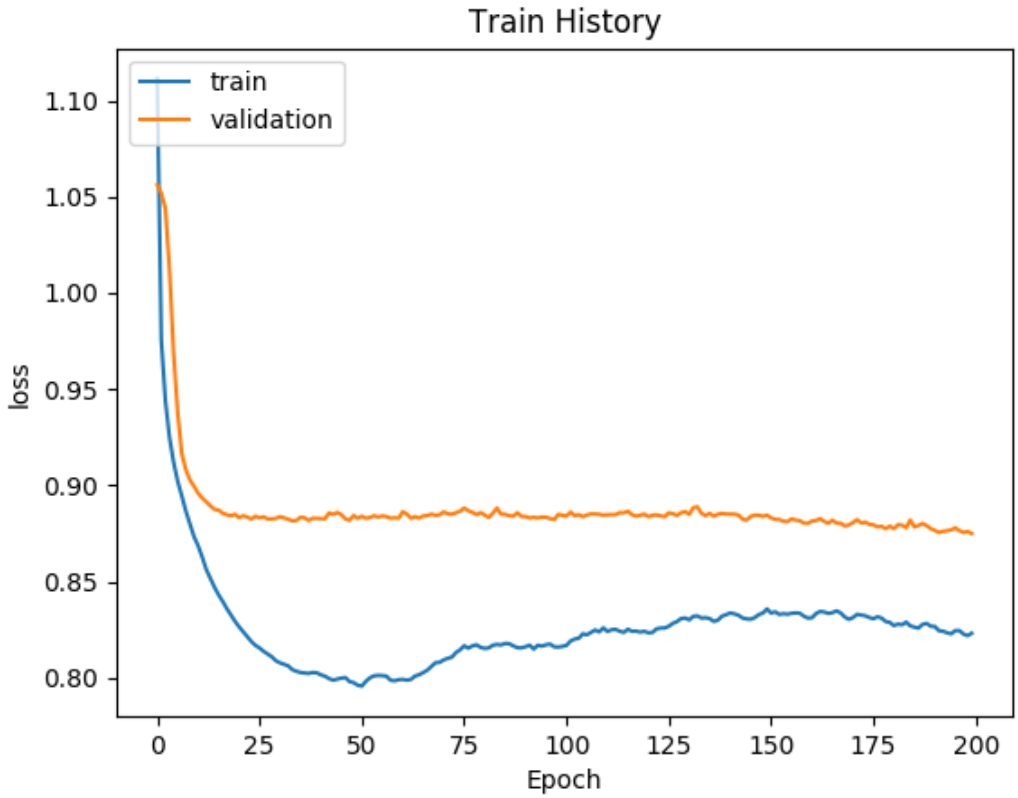



```
1 = ['Action', 'Adventure', 'Western']
2 = ['Animation', "Children's", 'Comedy', 'Romance']
3 = ['Crime', 'Thriller', 'Film-Noir', 'Horror', 'Mystery']
4 = ['Documentary', 'War']
5 = ['Drama', 'Musical']
6 = ['Fantasy', 'Sci-Fi']
```

上圖示我對於電影的主要分類，從結果可以看出右中那一塊圓圈中代表無法分類的電影，左半邊像月亮的區塊勉强能分類，但效果不明顯，我覺得可能的原因是電影的分類本來就很主觀，而且光從這個 label 就要得到好的分佈效果，同時要將資訊壓縮在二維空間，有點不太實際。

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。

   我使用了 user 的性別、年齡以及電影的分類作為 embedding layer 的 weight，再跟原本 embedding layer 的結果 concatenate，再接 3 層 DNN 去訓練，得到的結果如下圖所示，表現比單純 MF 差而稍微優於 DNN，且 training loss 及 validation loss 都有振盪的情況。


Train History
train
validation
loss
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0
25
50
75
100
125
150
175
200
Epoch