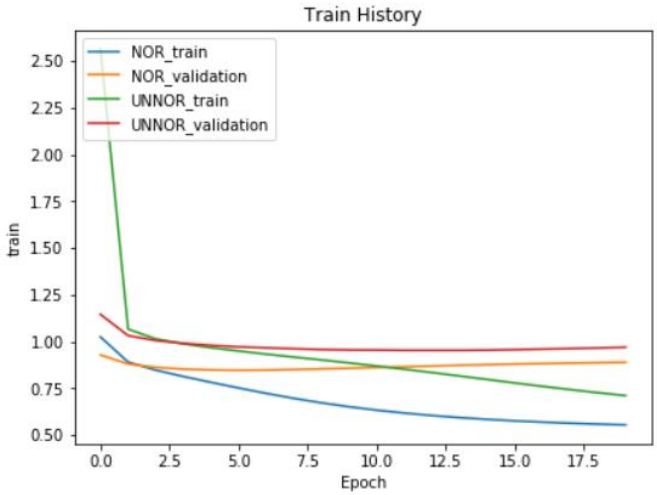
學號：B03505052 系級： 工海四 姓名：李吉昌
collaborator: b03303032 劉祐瑄/b03505040 林後維

```
user_id = Input(shape=(1,))
user_model = Sequential()
user_model.add(Embedding(user_num, embed_dim, input_length=1, embeddings_regularizer = l2(reg)))
user_model.add(Flatten())

movie_id = Input(shape=(1,))
movie_model = Sequential()
movie_model.add(Embedding(movie_num, embed_dim, input_length=1, embeddings_regularizer = l2(reg)))
movie_model.add(Flatten())

user_vec = user_model(user_id)
movie_vec = movie_model(movie_id)

user_bias = Embedding(user_num, 1, embeddings_regularizer = l2(reg))(user_id)
user_bias = Flatten()(user_bias)

movie_bias = Embedding(movie_num, 1, embeddings_regularizer = l2(reg))(movie_id)
movie_bias = Flatten()(movie_bias)

score = Add()([Dot(axes=-1)([user_vec, movie_vec]), user_bias, movie_bias])

MF_model = Model(inputs = [user_id, movie_id], outputs = score)
MF_model.compile(loss= 'mse', optimizer= 'adam', metrics=[rmse])
```

以下第一題和第三題的 dim 皆取 100，來自同一筆 train 和 valid 的 dataset

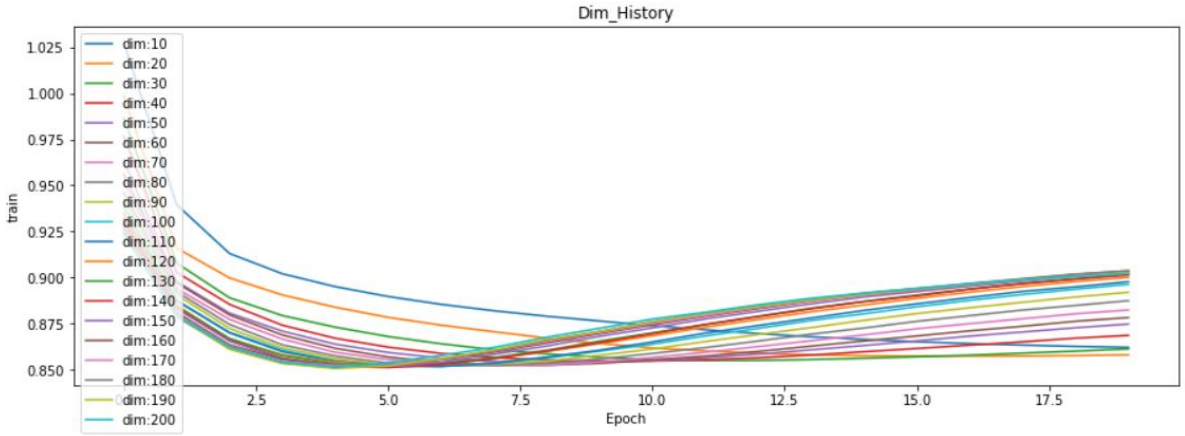
1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.



Standard normalize 將資料減掉平均除以標準差
由圖可見，不論 train 還是 valid 的 loss 曲線皆為 normalize 後的較低，
由此可知量級大小對 curve 影響很大。
但 score 上表現剛好一樣。
NOR :
private : 0.84505 | public : 0.85397

UNNOR :
private : 0.84505 | public : 0.85397

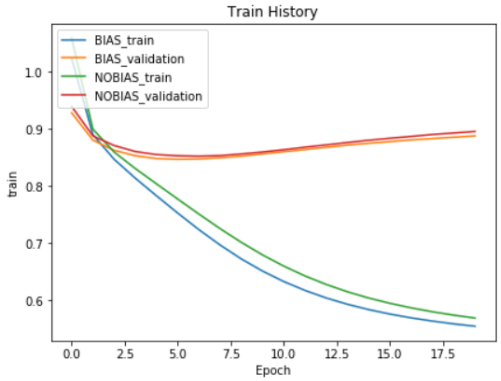
2. (1%)比較不同的 embedding dimension 的結果。



上為不同維度在同一筆 dataset 下的 valid loss 曲線，其 dim 越小的時候越早能夠

達到最低點，但相對的較容易 overffiting，另外能達到最低的最低點約莫落在 130 的維度。

3. (1%)比較有無 bias 的結果。

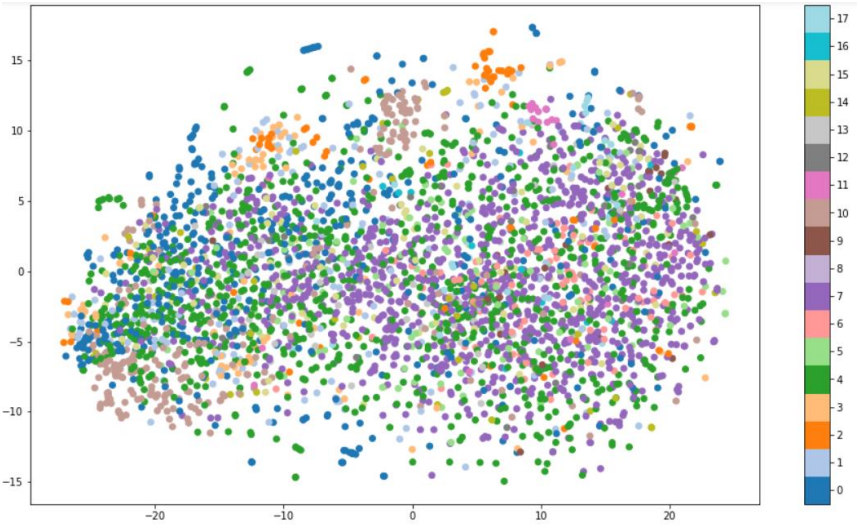


由圖可知，在有 bias 的 model 下結果會比較好。

BIAS :
private : 0.84505 | public : 0.85397

NOBIAS :
private : 0.85010 | public : 0.85818

4. (1%)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。



	class	genres
0	0	Action
1	1	Adventure
2	2	Animation
3	3	Children's
4	4	Comedy
5	5	Crime
6	6	Documentary
7	7	Drama
8	8	Fantasy
9	9	Film-Noir
10	10	Horror
11	11	Musical
12	12	Mystery
13	13	Romance
14	14	Sci-Fi
15	15	Thriller
16	16	War
17	17	Western

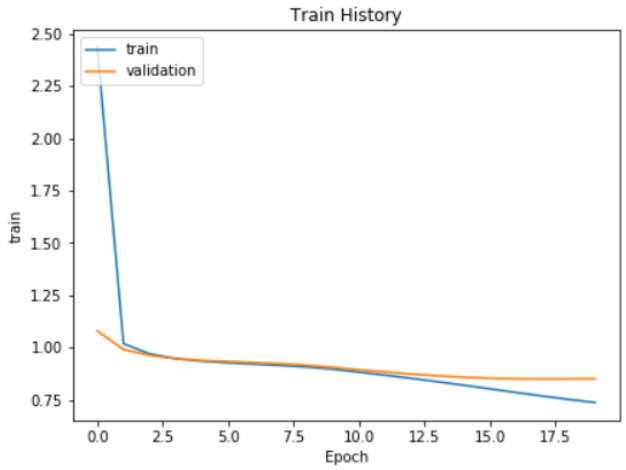
movie.csv 中取電影的 Genres，其中電影有可能會屬於兩種以上的 Genres，label 取第一種，由圖可見，藍色和紫色還有淺咖啡較為容易區分。

5. (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。

user 取的資料有 Gender、Age、Occupation，將 gender 做 one-hot encode，並將 Age、Occupation 做 normal scaling。
movie 取的資料為將十八種 Genres 做 one-hot encode。
後得出 user_matrix(? , 4)以及 movie_matrix(? , 18)

```
user_id = Input(shape=(1,))
user_model = Sequential()
user_model.add(Embedding(user_num, embed_dim, input_length=1, embeddings_regularizer = l2(reg)))
user_model.add(Flatten())

movie_id = Input(shape=(1,))
movie_model = Sequential()
movie_model.add(Embedding(movie_num, embed_dim, input_length=1, embeddings_regularizer = l2(reg)))
movie_model.add(Flatten())

user_vec = user_model(user_id)
movie_vec = movie_model(movie_id)

user_pros = Sequential()
user_pros.add(Embedding(user_num, u_matrix.shape[1], input_length=1, weights=[u_matrix], trainable=False, embeddings_regularizer
user_pros.add(Flatten())
user_pros.add(Dense(units = 1, activation=swish))
u_pros = user_pros(user_id)
user_vec = Add()([user_vec, u_pros])

movie_pros = Sequential()
movie_pros.add(Embedding(movie_num, m_matrix.shape[1], input_length=1, weights=[m_matrix], trainable=False, embeddings_regulariz
movie_pros.add(Flatten())
movie_pros.add(Dense(units = 1, activation=swish))
m_pros = movie_pros(movie_id)
movie_vec = Add()([movie_vec, m_pros])

user_bias = Embedding(user_num, 1, embeddings_regularizer = l2(reg))(user_id)
user_bias = Flatten()(user_bias)

movie_bias = Embedding(movie_num, 1, embeddings_regularizer = l2(reg))(movie_id)
movie_bias = Flatten()(movie_bias)


score = Add()([Dot(axes=-1)([user_vec, movie_vec]), user_bias, movie_bias])

P_model = Model(inputs = [user_id, movie_id], outputs = score)
P_model.compile(loss= 'mse', optimizer= 'adam', metrics=[rmse])
```

其架構為，將 user_matrix 以及 movie_matrix 作為 untrainable 的 embed 後作 DNN，壓成一個常數分別與 user vec 和 movie vec 作相加的動作，大致上架構與 MF 的 model 相同，較為不同的地方是，自己參考了網路上較新的 activation function : swish
其函數為 x * sigmoid(x)



從結果來看，曲線並沒有像 MF 一樣平滑且在 valid loss 的表現上也退步了，試想其原因可能是自己沒有找到比較適合的 model 架構能夠發揮其 feature。
Private : 0.84732 | Public : 0.85604

另外 kaggle 上的分數是 MF 模型架構跟 DNN 模型架構與十種不同的 train data set 做 ensemble 得來。

各種模型架構紀錄於 train.py 中。