學號：B04902097 系級： 資工二 姓名：陳家棋

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

normalize 方法：取 training data rating 的 mean 和 std，將 training data rating 減掉 mean 後除以 std 拿去 train model，最後 predict 出來的值乘上 std 再加 mean。(mean、std 皆為取自 training rating)

```
26
27  movie_input = Input( shape=[1] )
28  movie_vec = Embedding( n_movies + 1, 100 )( movie_input )
29  movie_vec = Flatten()( movie_vec )
30  movie_vec = Dropout(0.5)( movie_vec )
31  # movie_vec = BatchNormalization()( movie_vec )
32
33
34  user_input = Input( shape=[1] )
35  user_vec = Embedding( n_users + 1, 100 )( user_input )
36  user_vec = Flatten()( user_vec )
37  user_vec = Dropout(0.5)( user_vec )
38  # user_vec = BatchNormalization()( user_vec )
39
40
41
42  out = Dot(1)( [movie_vec, user_vec] )
43
44
45  model = Model( [movie_input, user_input], out)
46  model.compile( loss = 'mse', optimizer='adam' )
47
```

| | 有 BatchNormalization | | 無 BatchNormalization | |
|---|---|---|---|---|
| | 無 normalize | 有 normalize | 無 normalize | 有 normalize |
| kaggle score | 0.84720 | 0.84786 | 0.88208 | 0.84877 |

我發現在沒加 BatchNormalization 前，對 rating 做 normalize 的處理會提高準確度，但加了 BatchNormalization 後則差異不大。故做 normalization 能提高準確度。

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

| dimension | 50 | 100 | 150 | 200 | 250 | 300 |
|---|---|---|---|---|---|---|
| kaggle score | 0.85221 | 0.84720 | 0.84666 | 0.84795 | 0.85039 | 0.85085 |

我發現約150的時候效果最佳。

**3. (1%) 比較有無 bias 的結果。**

我做了兩種 bias。第一種是多加一個 embedding 到 1 維，第二種是拿用來做內積的 100 維 embedding dense 到 1 維。兩者效果看似差不多。有無 bias 對 kaggle score 來說差異不大。(無 bias : 0.84666)

| | embedding | dense |
|---|---|---|
| kaggle score | 0.84647 | 0.84673 |

```
26
27  movie_input = Input( shape=[1] )
28  movie_vec = Embedding( n_movies + 1, dim )( movie_input )
29  movie_vec = Flatten()( movie_vec )
30  movie_vec = Dropout(0.5)( movie_vec )
31  movie_vec = BatchNormalization()( movie_vec )
32
33  # movie_bias = Embedding( n_movies + 1, 1 )( movie_input )
34  # movie_bias = Flatten()( movie_bias )
35  movie_bias = Dense(1, activation='elu')( movie_vec )
36
37
38  user_input = Input( shape=[1] )
39  user_vec = Embedding( n_users + 1, dim )( user_input )
40  user_vec = Flatten()( user_vec )
41  user_vec = Dropout(0.5)( user_vec )
42  user_vec = BatchNormalization()( user_vec )
43
44  # user_bias = Embedding( n_users + 1, 1 )( user_input )
45  # user_bias = Flatten()( user_bias )
46  user_bias = Dense(1, activation='elu')( user_vec )
47
48  out = Dot(1)( [movie_vec, user_vec] )
49
50  out = Add()( [out, movie_bias, user_bias] )
51
52  model = Model( [movie_input, user_input], out)
53  model.compile( loss = 'mse', optimizer='adam' )
54
```

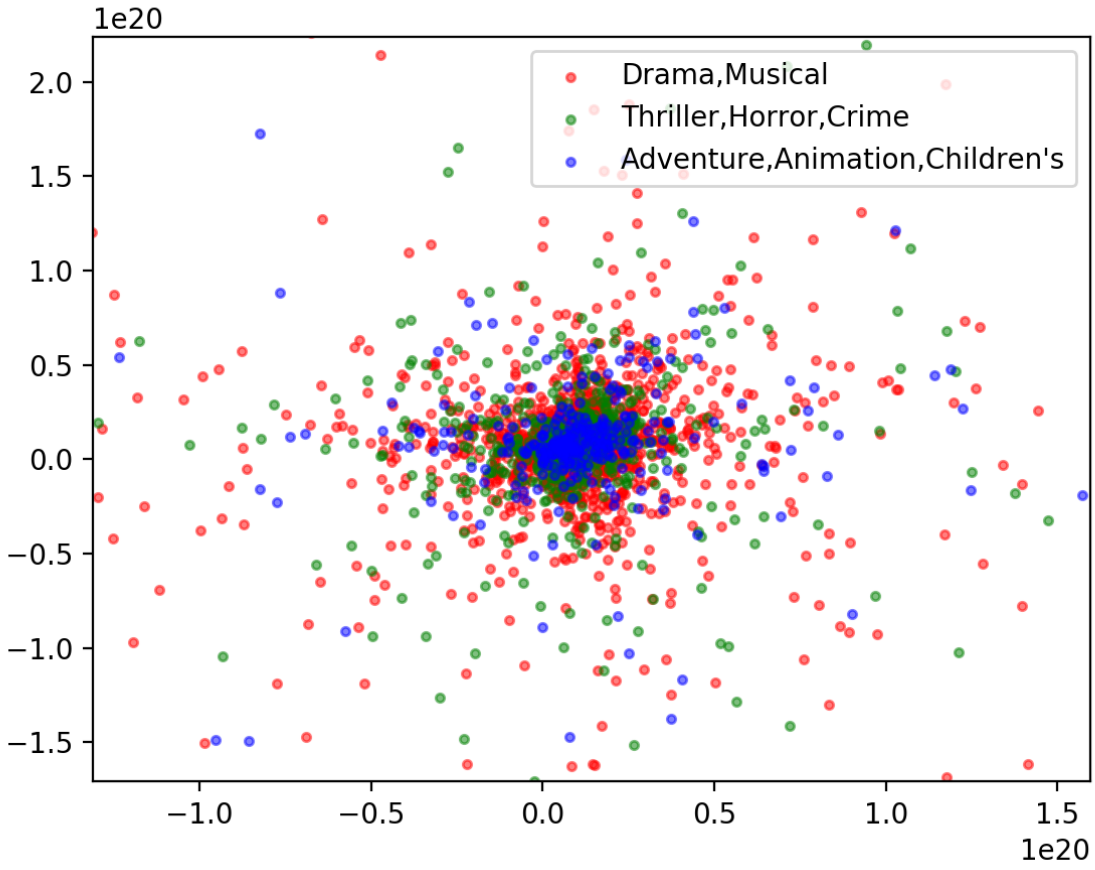
**4. (1%) 請試著用 DNN 來解決這個問題，並且說明實做的方法(方法不限)。並比較 MF 和 NN 的結果，討論結果的差異。**

我做出來 MF 的效果比 DNN 來的要好，我認為這樣的差距應該只是參數沒調好造成，DNN 應該可以有相當於 MF 的表現。

| | MF | DNN |
|---|---|---|
| kaggle score | 0.84720 | 0.86196 |

```
25
26
27   movie_input = Input( shape=[1] )
28   movie_vec = Embedding( n_movies + 1, dim )( movie_input )
29   movie_vec = Flatten()( movie_vec )
30   movie_vec = Dropout(0.5)( movie_vec )
31   movie_vec = BatchNormalization()( movie_vec )
32
33
34
35   user_input = Input( shape=[1] )
36   user_vec = Embedding( n_users + 1, dim )( user_input )
37   user_vec = Flatten()( user_vec )
38   user_vec = Dropout(0.5)( user_vec )
39   user_vec = BatchNormalization()( user_vec )
40
41
42   out = Concatenate()( [movie_vec, user_vec] )
43   out = Dense(32, activation='elu')( out )
44   out = Dropout(0.2)( out )
45   out = Dense(32, activation='elu')( out )
46   out = Dropout(0.2)( out )
47   out = Dense(32, activation='elu')( out )
48   out = Dropout(0.2)( out )
49   out = Dense(1)( out )
50
51
52
53   model = Model( [movie_input, user_input], out)
54   model.compile( loss = 'mse', optimizer='adam' )
55
```

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




**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。**

我將 users.csv 中的 Gender、Age、Occuaption 做 1 of n encoding，再將 movies.csv 中的 Genres 做 1 of n encoding，training 時與 embedding 後的結果 concatenate 起來做 DNN。最後上傳 kaggle 分數只有 0.97 左右。

```
37
38
39  movie_input = Input( shape=[1] )
40  movie_vec = Embedding( n_movies + 1, dim )( movie_input )
41  movie_vec = Flatten()( movie_vec )
42  movie_vec = Dropout(0.5)( movie_vec )
43  movie_vec = BatchNormalization()( movie_vec )
44
45
46
47  user_input = Input( shape=[1] )
48  user_vec = Embedding( n_users + 1, dim )( user_input )
49  user_vec = Flatten()( user_vec )
50  user_vec = Dropout(0.5)( user_vec )
51  user_vec = BatchNormalization()( user_vec )
52
53  usr_bias = Input(shape=[30])
54  mov_bias = Input(shape=[18])
55
56  out = Concatenate()( [movie_vec, user_vec, usr_bias, mov_bias] )
57  out = Dense(64, activation='elu')( out )
58  out = Dropout(0.2)( out )
59  out = Dense(32, activation='elu')( out )
60  out = Dropout(0.2)( out )
61  out = Dense(32, activation='elu')( out )
62  out = Dropout(0.2)( out )
63  out = Dense(1)( out )
64
65
66
67  model = Model( [movie_input, user_input, usr_bias], out)
68  model.compile( loss = 'mse', optimizer='adam' )
69
```