學號：B03505027　系級：電機三　姓名：劉亦浚

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

Train 了兩個 MF model 來做比較（latent dim 為 5，一個標準化，一個則無）。當有做 rating normalize 時，在 train model 時很快就可以收斂，大約 15-20 個 epoch；若無 normalize，則收斂較慢，需 100 個 epoch 左右，在 Kaggle 分數方面，有標準化的為 0.86741；無標準化的為 0.87448，所以對 rating 做標準化有較快的收斂速度還有些許的準確度提升。

normalize 的過程為，取 training data rating 的 mean 及 std，再把 training data rating 做標準化並且拿來當作 training 的 label。在 prediction 的時候則將 $pred_{test} = pred^{*}_{test} \times train_{std} + train_{mean}$ 當作最後的 answer。

2. (1%)比較不同的 latent dimension 的結果。

Train 了七個 MF model 來做比較，latent dim 分別為 5、10、15、20、200、350、400，kaggle 分數為 0.86800(5)、0.87048(10)、0.87155(15)、0.87144(20)、0.85530(200)、0.85275(350)、0.85385(400)。一開始使用較低的 dim 就可以得到不錯的分數，後來發現將 dim 提高並搭配 Dropout 會有更好的效果，若較高的 dim 沒有使用 Dropout 的話就會 overfitting，在 val_rmse 表現非常差。

dim	5	10	15	20	200	350	400
score	0.86800	0.87048	0.87155	0.87144	0.85530	0.85275	0.85385

3. (1%)比較有無 bias 的結果。

Train 了三組有無 bias 的 MF model（共六個 model）來做比較，其 latent dim 分別為 5、200、350。以下皆為 kaggle 分數。

Latent dim	5	200	350
With bias	0.86800	0.85521	0.85275
Without bias	0.86741	0.85530	0.85313

從以上分數可看出，此次作業有無 bias 並不太會影響分數的好壞，兩者分數並沒有太大的差距，但有加 bias 的時候，其收斂速度較快，以 dim 為 200 舉例，約 65-70 個 epoch；反之沒有加 bias 時，收斂速度較慢，約 90-95 個 epoch。

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。**

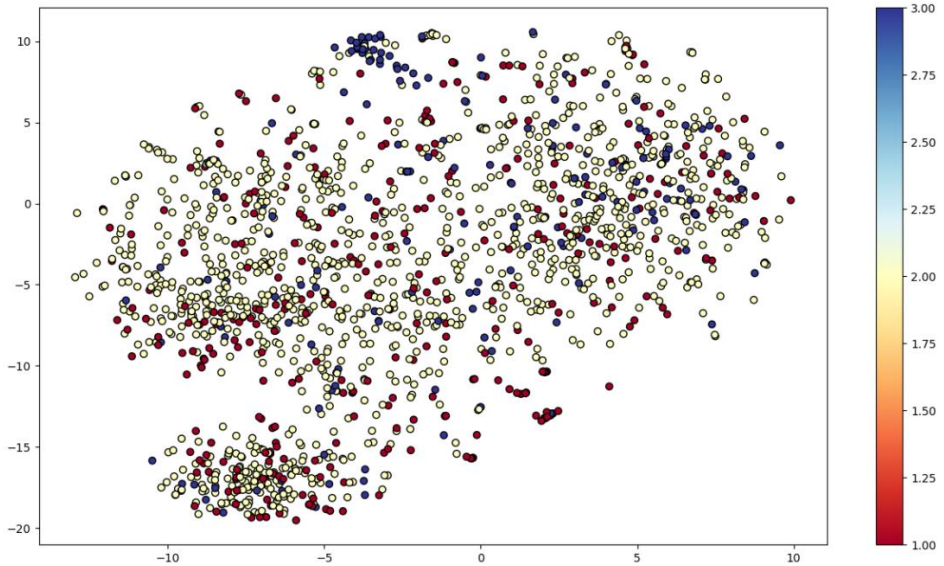
```
#model
user_input = Input(shape=[1])
user_vec = Embedding(max_userid, embedding_dim, embeddings_initializer='random_normal')(user_input)
user_vec = Flatten()(user_vec)

movie_input = Input(shape=[1])
movie_vec = Embedding(max_movieid, embedding_dim, embeddings_initializer='random_normal')(movie_input)
movie_vec = Flatten()(movie_vec)

concat_vec = Concatenate()([user_vec, movie_vec])
dnn = Dense(256, activation='relu')(concat_vec)
dnn = Dropout(0.5)(dnn)
dnn = Dense(128, activation='relu')(dnn)
dnn = Dropout(0.5)(dnn)
dnn = Dense(64, activation='relu')(dnn)
dnn = Dropout(0.5)(dnn)
result = Dense(1, activation='linear')(dnn)

model = Model(inputs=[user_input, movie_input], outputs=result)
model.summary()

model.compile(loss='mse', optimizer='adam', metrics=[rmse])
```

實作方法為：將通過 user embedding 與 movie embedding 的 vector concatenate 在一起再通過 DNN（三層 relu，output linear）得到最後的 rating。

MF 與 NN 的比較：以 latent dim 為 200 做對照，kaggle 的分數分別為 0.87721(NN)、0.85521(MF)，使用 MF 會有較好的準確率；但在收斂速度方面，NN 有較好的表現，約在 25-30 個 epoch 收斂，MF 則需要 90-95 個 epoch。

結果差異：MF 會有較好的準確率應該是在 train MF 的時候是兩個 vector 互相做內積，有考慮到 user 與 movie 背後的相關強度，rating 高可能表示 user 的興趣與 movie 的內容有較大的相關；而 NN 只有將兩者的 vector 當作 feature 送入 DNN 使之趨近 rating，並無探討背後的關係。在收斂方面，NN 有較多的參數可能增加其收斂的速度。

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



紅色：Thriller、Horror
米色：Drama
藍色：Children's、Animation
採用的 MF model 的 latent dim 為 200，透過 TSNE 降至 2 維。
其中紅色的部分，較多分布在左下的位置；藍色的部分則是分布在右上偏中的位置，一個是恐怖驚悚片；一個是兒童動畫片，兩者風格與內容截然不同，因此有所區隔也是很合理的。在米色 Drama 的部分，分布的很廣也很平均，可見 Drama 是一個很大的分類，多數的電影都包含在此類裡面，較難從此分類看出其區隔性。

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。

```
#model
user_input = Input(shape=[1])
user_vec = Embedding(max_userid, embedding_dim, embeddings_initializer='random_normal')(user_input)
user_vec = Flatten()(user_vec)

movie_input = Input(shape=[1])
movie_vec = Embedding(max_movieid, embedding_dim, embeddings_initializer='random_normal')(movie_input)
movie_vec = Flatten()(movie_vec)

age_input = Input(shape=[1])
age_vec = Embedding(max_age, embedding_dim, embeddings_initializer='random_normal')(age_input)
age_vec = Flatten()(age_vec)

occ_input = Input(shape=[1])
occ_vec = Embedding(max_occ, embedding_dim, embeddings_initializer='random_normal')(occ_input)
occ_vec = Flatten()(occ_vec)

concat_vec = Concatenate()([user_vec, movie_vec, age_vec, occ_vec])
dnn = Dense(256, activation='relu')(concat_vec)
dnn = Dropout(0.5)(dnn)
dnn = Dense(128, activation='relu')(dnn)
dnn = Dropout(0.5)(dnn)
dnn = Dense(64, activation='relu')(dnn)
dnn = Dropout(0.5)(dnn)
result = Dense(1, activation='linear')(dnn)

model = Model(inputs=[user_input, movie_input, age_input, occ_input], outputs=result)
model.summary()

model.compile(loss='mse', optimizer='adam', metrics=[rmse])
```

多參考了 user.csv 中的 Age 與 Occupation，也將他們通過 embedding 轉成相對應 vector，再將 user、movie、age、occupation 的 vector 全部 concatenate 在一起，然後丟入 DNN train。Kaggle 的分數為 0.87144，比第四題只有使用 user、movie 的分數（0.87721）進步了一點點，可見 age、occupation 對每個人的 rating 還是有點影響，但其影響並不是太大。