學號：B03902090 系級： 資工三 姓名：邵楚荏

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

Normalize:

將 training set 的所有 rating 算出平均以及標準差，並將所有資料減去平均之後再除上標準差。

$X_n = \frac{X - \mu_x}{\sigma_x}$ (X_n為 normalize 之後的 rating)

在 training 的時候會發現，有 normalize 的 data 會收斂的比較快，原本無 normalize 的資料大概要 30~35 個 epoch 才會訓練到收斂，如果加了 normalize 之後，大概 15~20 個 epoch 就可以收斂。另外在 testing set 上的表現，沒有 normalize 過的資料在 Kaggle 上的 loss 為 0.85600，normalize 過後 loss 為 0.85178，由此可知，有無 normalize 在這個 task 上 testing set 的 performance 差不多。

2. (1%)比較不同的 latent dimension 的結果。

這題我試了四種不同的 latent dimension，分別為 50, 100, 150, 200，其中在 Kaggle 上的 testing loss 如下表格:

Latent dimension	Testing loss(On Kaggle)
50	0.86624
100	0.85491
150	0.85165
200	0.85145

從上面的表格可以發現，似乎 latent dimension 越高，testing loss 越低，但基本上經過實驗，維度大於 150 之後的 validation loss，或者是 testing loss，都差不多，但維度大於 250 之後，就比較難讓 training loss 下降，在 validation 跟 testing set 上的表現，也不如 150~250 維那麼好。

3. (1%)比較有無 bias 的結果。

加了 bias 之後，因為多了兩個參數，所以在 training 收斂的速度，會比沒有加 bias 來的慢一點點，原先大概 30~35 個 epoch 可以達到收斂，加了 bias 之後大約 40~45 個 epoch 達到收斂，另外，加了 bias，validation 的 performance 跟原本的相比來的好上一點點，在 Kaggle 上的 loss 也從原本的

0.85178 下降到 0.85165，在訓練的過程以及 validation 的表現上觀察到，加了 bias，可以讓 overfitting 的情況好一點點。此外，經過幾次實驗之後發現，加了兩項的 bias 可以讓 testing set 以及 validation set 的 performance 最好。

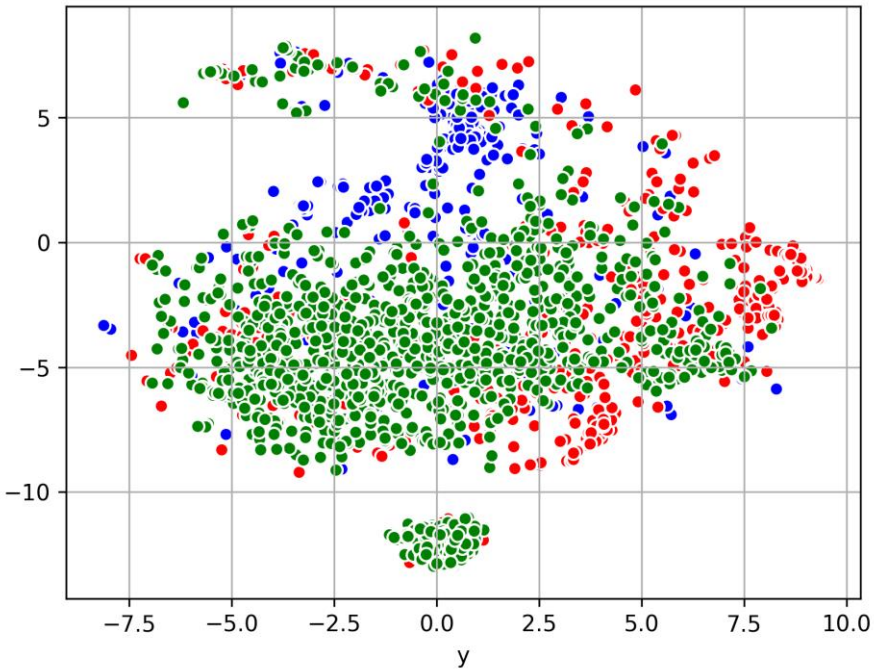
4. (1%)請試著用 DNN 來解決這個問題，並且說明實做的方法(方法不限)。並比較 MF 和 NN 的結果，討論結果的差異。

```
class DeepModel(Sequential):
    def __init__(self, n_users, m_items, emb_dim, **kwargs):
        P = Sequential()
        P.add(Embedding(n_users, emb_dim, input_length=1))
        P.add(Reshape((emb_dim, )))
        Q = Sequential()
        Q.add(Embedding(m_items, emb_dim, input_length=1))
        Q.add(Reshape((emb_dim, )))
        super(DeepModel, self).__init__(**kwargs)
        self.add(Merge([P, Q], mode='concat'))
        self.add(Dropout(0.2))
        self.add(Dense(150, activation='relu'))
        self.add(Dropout(0.2))
        self.add(Dense(1, activation='relu'))
~
```

上圖是我實作 DNN 的 model 的架構，P 和 Q 代表 user 和 movie 通過 embedding 後的 vector，原本的 MF 是將兩個 vector 做向量的內積，在 DNN 實作，我將兩個向量先做"concatenate"，之後再加了兩層的 Neural Network，分別為 150 維度以及最後一層 1 個 neuron，分別在每層後面都加上了 dropout，其中 model 最後一層 output 一個值當作最後所預測出來的 rating。

訓練過程中，DNN 的收斂速度自然會比較慢，但加了 dropout 之後，發現即使訓練了 50~60 個 epoch，也不容易在 validation 上 overfitting，另外在收斂的時候 validation loss 和 MF 的最好的情況差了 0.01 左右，在 Kaggle 上的 testing loss 為 0.86205，比 MF 的 0.85165 稍微高一點點。

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



藍色點 movie label 為"Adventure, Animation, Children's"

紅色點 movie label 為"Thriller, Horror, Crime"

綠色點 movie label 為"Drama, Musical"

可以看見，data 中有些類別是比較像的，把它們分別歸成一類之後去做 tsne 降維，可以明顯看出紅色、綠色、藍色點被區分為大致三個區域。

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。

原本的 MF 利用 user 和 movie 通過 embedding 得到的 vector 座內積，加上 bias 之後去預測 rating，在這題，我試著將'user.csv'這份檔案裏頭的 gender 資訊，將 Male encode 成 1，Female encode 成-1，另外將 Age 經過 Normalize 之後也變成一項 feature，所以整個 rating prediction 的式子變成:

$$r_{i,j} = U_i \cdot V_j + b_i^{user} + b_j^{movie} + W_{Age} * Age + W_{gender} * Gender$$

在 training 的過程中，收斂的速度比原本的 MF 更慢一點，當然這是因為參數變的更多，至於在 validation loss 上，原本的 loss 大概在 0.85~0.86，加了這幾個 feature 可以降到 0.849 左右，但在 Kaggle 上的 testing loss，

還是跟原本的 MF 差不多。