

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

Normalize 後的結果和未 Normalize 並沒有顯著差異，在 Kaggle 上 Normalize 的分數為 0.86094，未 Normalize 的分數為 0.86029。Normalize 的方式為原始 Rating 減去所有 Rating 的平均後再除以 Standard Deviation。推測會有這樣的結果是因為 Rating 的值本身只有 1~5 並沒有特別大的幅度變動。

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

以下試了三種 Latent Dimension，結果如下(以 kaggle public score 比較)：

Latent 300: 0.85609

Latent 200: 0.86301

Latent 120: 0.87128

以 Latent dimension 為 300 時有最佳結果。

3. (1%)比較有無 **bias** 的結果。

在 Latent Dimension 為 200 的 model 上增加了 User_Bias，效果只好了一些些，在 Kaggle 上只有 0.86205，比起原本沒有增加 Bias 減少了 0.0095 的誤差。可能原因是 user 中可能有特別有些人標準高，給每部片都評分較嚴格，也有些人標準寬鬆，給每部片評分都較高。

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。

我的 DNN model 架構如下：

1. input_layer (input = user)

embedding_layer

flatten()

2. input_layer (input = movie)

embedding_layer

flatten()

Concatenate 1 and 2

Dense_layer (200, activation = relu)

Dropout(0.45)

Dense_layer (50, activation = relu)

Dropout(0.25)

Output = Dense(1)

使用此 model 基本上和MF 一樣可以輕鬆過 strong baseline，比較不一樣的是MF 的 model 大概訓練到第 70 個 epoch 的時候才 early stopping，DNN 只要 50 個左右。另外觀察到DNN在前幾個 epoch 時 loss 就小於 1，MF 的前三個 epoch 的 loss 值都大於 5，由此可以發現DNN的 training 速度較快。

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



把所有的電影稍微分類後，例如 War 跟 Horror 並為同類，Children 與 Animation 與 Fantasy 並為同類後進行降維，如上圖。基本上真的看不出什麼差異。簡單來說所有的電影大概就像哈密瓜的哈味一樣，有些人喜歡，有些人不喜歡，很難用二分法或三分法說明什麼樣年紀或什麼樣性別就會喜歡什麼樣的電影，這樣的分類太果簡單，因此廣告或電影推薦系統才要客製化，根據每個人的經驗來推斷。

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。