Homework 6 - Matrix Factorization

學號：b05901033 系級： 電機二 姓名：莊永松

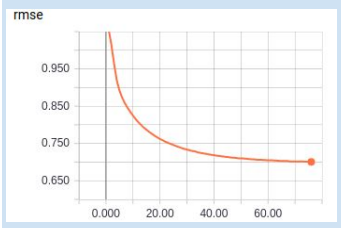
1. (0.8%)請比較有無normalize的差別。並說明如何normalize. (no Collaborators)

如何normalize

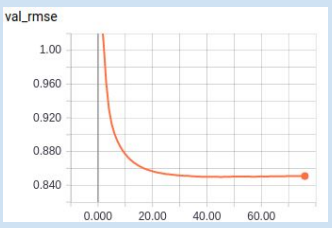
normalize 的方法是把 training data 中的每一個 rating 都減去 mean 再除以 std。訓練完 model 後，預測 testing data 時再把預測的結果都乘以 std 再加上 mean 來還原。

訓練過程

有做 normalize

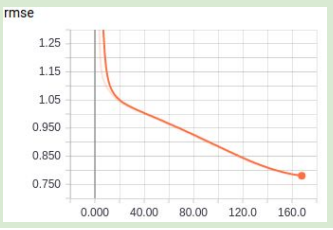


training set rmse

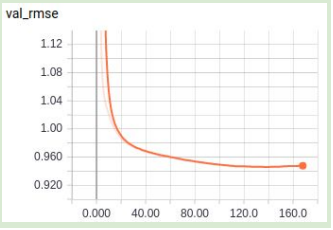


validation set rmse

沒做 normalize



training set rmse



validation set rmse

可以觀察到沒做normalize的訓練過程不斷overfit，training rmse一直減小，但validation rmse早早收斂在0.94附近，表現差強人意。有做normalize的訓練過程很快收斂，雖然也有些微overfit，但狀況不太嚴重。

在kaggle上成績如下：	public score	private score
有做 normalization	0.84795	0.84089
未做 normalization	0.85725	0.85188

2. (0.8 %)比較不同的embedding dimension的結果。(no Collaborators)

比較 embedding dimension 設 16, 32, 64, 128, 256, 512, 1024 的表現：

(會在val_rmse最小時存下model，epoch是指存下model時跑了多少epoch)

Dimension	16	32	64	128	256	512	1024
val_rmse	0.87332	0.86268	0.85556	0.85104	0.84973	0.85106	0.85296
epoch	160	160	87	65	45	45	54
kaggle public	0.87277	0.86281	0.85503	0.85190	0.85024	0.85055	0.85389
kaggle private	0.86548	0.85559	0.84813	0.84439	0.84331	0.84507	0.84822

實驗結果：dimension=256時的收斂速度最快(45 epoch)，效果也最好(0.84973)。

3. (0.8 %)比較有無bias的結果。(no Collaborators)

有bias時，在epoch=45收斂，最小val_rmse=0.84973。

有bias時，也是在epoch=45收斂，但最小val_rmse=0.86057，明顯較差。

在kaggle上成績如下：	public score	private score
有做 bias	0.84795	0.84089
未做 bias	0.86066	0.85388

4. (0.8 %)請試著將movie的embedding用tsne降維後，將movie category當作label來作圖。(no Collab.)

為了讓圖畫出來好看一點，我選擇較為相近的category來作圖：

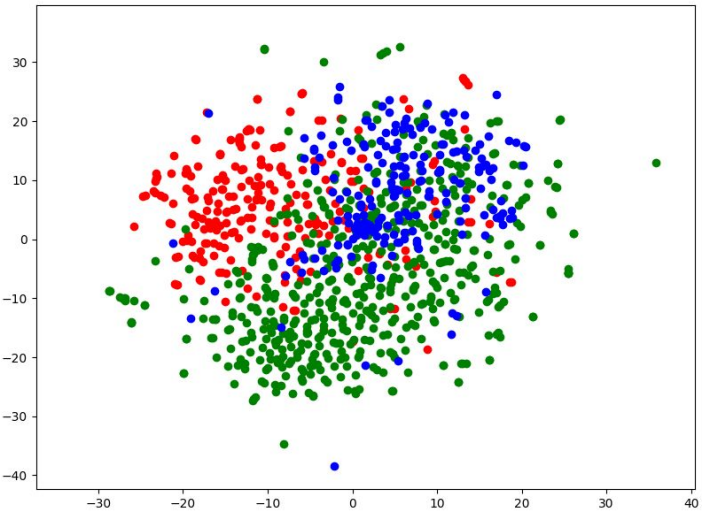
紅色(1)Animation+Children's

綠色(2) Action+Crime

藍色(3) Documentary+War

並且只有在該筆data沒有重複出現在這三種的狀況才拿來作圖，因此取出約1000多筆data。

embedding vector部分抽取未加額外feature的model的embedding層來做t-SNE成兩維(我有試過



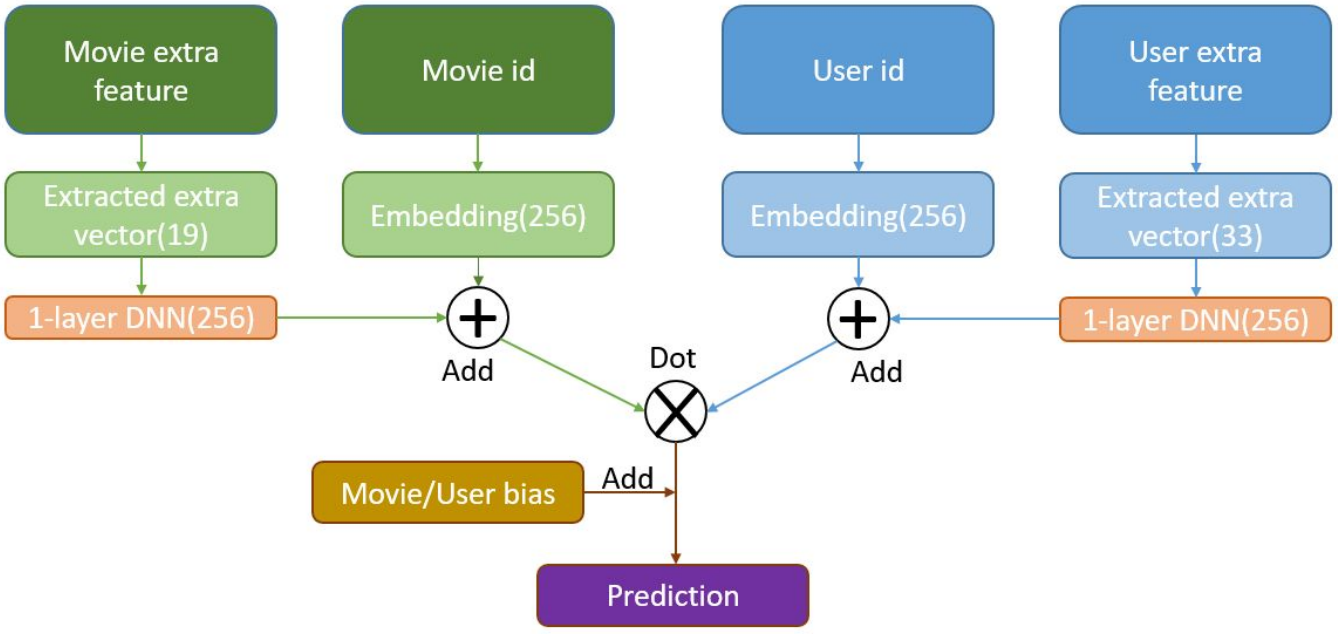
用有加入額外feature的model來作但效果不好)，結果如上圖，雖然還是有部分重疊了(藍色比較嚴重)，但整體效果還算不錯，看得出有分成三群。

5. (0.8 %)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。(no Collab.)

feature extract 的方法

user: 我將每個ID所對應的額外 data 弄成一個33維的向量，其中向量第0個位置是性別(M:1,F:0)，第1個位置是經標準化之後的age，第2~22個位置是經過one-hot encoding的occupation(編號0~20，共21種)，第23~32個位置是ZIP-code的one-hot encoding，但只取每串號碼第一位數字(查了一下，美國的ZIP-code如右圖分布[1]，只取第一位應該蠻有代表性的，像是東岸跟西岸人民性格也許會有一些不同)

movie: 將每部電影所對應 genres 弄成一個one-hot encoding的18維向量，並且以L1-normalize處理(使這18維數字的總合為1)，最後多加一維電影年份，變19維。(原本沒加電影年份，加了之後，準確率進步0.004左右，感覺上老人常看老片，年輕人常看新片，hen合理~~)



1 layer linear DNN: 經過上面兩招得到了user跟movie的extracted extra vector之後，我把這兩個vector各自通過一層linear的DNN(也就是做一次trainable的矩陣乘法，模型簡單也避免overfitting)，轉成跟latent dimension一樣的extra vector後，user的extra vector直接用Add()加回去原本user ID Embedding得到的vector；movie的extra vector一樣直接加回去原本movie ID Embedding得到的vector，加完之後一樣去做MF，架構及訓練參數就跟前面一樣，跑了約79 epoch左右時，val_rmse收斂到最小值為0.8415，

在kaggle上成績如下：	public score	private score
有用 Extra Feature	0.84178	0.83435
未用 Extra Feature	0.84795	0.84089

準確率總共進步了0.006左右，效果不錯。

最後我將三個表現較好的model做ensemble後，在kaggle上public score為**0.84037**，private score為**0.83290**。

Ref:

[1] https://www.unitedstateszipcodes.org/