學號：B03901030 系級： 電機三 姓名：蕭晨豪

1. (1%)請比較有無normalize(rating)的差別。並說明如何normalize.  
   我normalize的方式是將rating減去其平均值，再除以標準差。

在test時必須將model的output乘上training時計算的標準差再加回平均值。

在沒有normalize時Kaggle上MF最好的結果為0.89061

在normalize後最好的結果變為0.87225(兩者均有加bias)

可看出有顯著進步

1. (1%)比較不同的latent dimension的結果。

在有normalize的情況下，幾乎都在前十個epoch就會到達最低的val\_loss，在latent dimension很高的時候，幾乎在前一兩個epoch就會有最低的val\_loss了，後面的epoch只會overfit，但前一兩個epoch由於才剛開始train因此loss還是很高，造成必須取低一點的latent dimension才會有較好的結果，我最好的結果是取latent dimension = 16

1. (1%)比較有無bias的結果。  
   由於電影的rating可能受個別使用者的影響(例如A給大部分的電影較高分，B卻均給較低分)，因此加上user和movie bias可以有效的消除此影響，較準確地得到針對個別使用者以及個別電影的rating。

在沒有加bias時Kaggle上MF最好的結果為0.89697

在加上bias後變為0.89061(兩者均沒有normalize)

可看出有些許進步

1. (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MF最好的結果為加上bias和normalization，並且latent dimension為16，其RMSE為0.87225

DNN的model前面和MF一樣先將user和movie的ID 通過embedding layer變成一個64維的vector，之後concatenate兩個embedding layer後通過三次有256 unit的dense，中間均使用relu做activation以及加上0.5的dropout，最後用dense(1)去output一個單一的數字做為rating。

此方法做出來的DNN參數量為MF的一半，但訓練過程反而較慢，並且收斂的時間較久，在kaggle上最好的結果為0.87135，較MF稍好一些

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

我取電影檔案中的第一個label作為該電影的genre(原本是multilabel)  
由於電影種類太多，我手動將其分成五類:

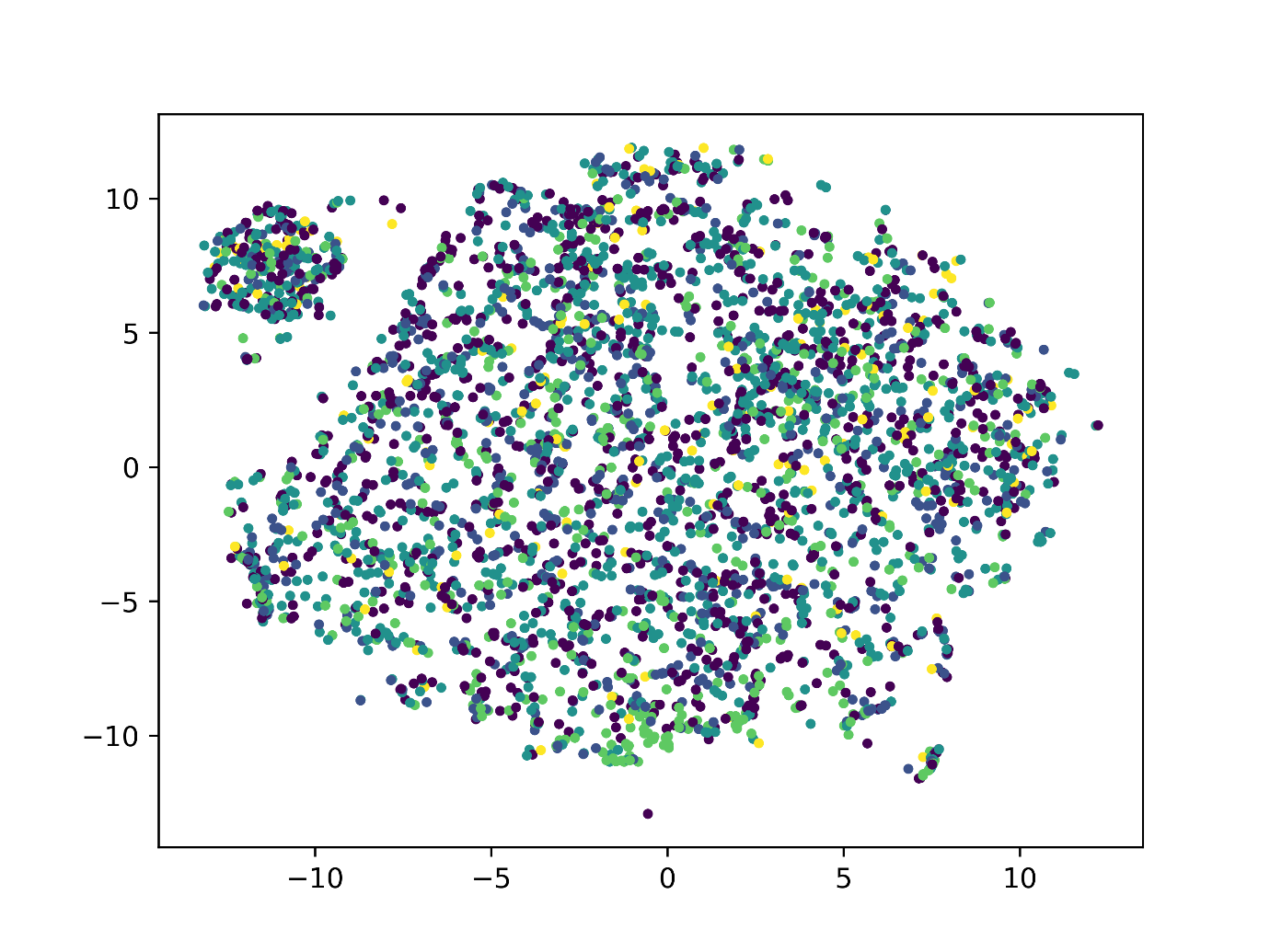
(1)Animation, Children, Comedy

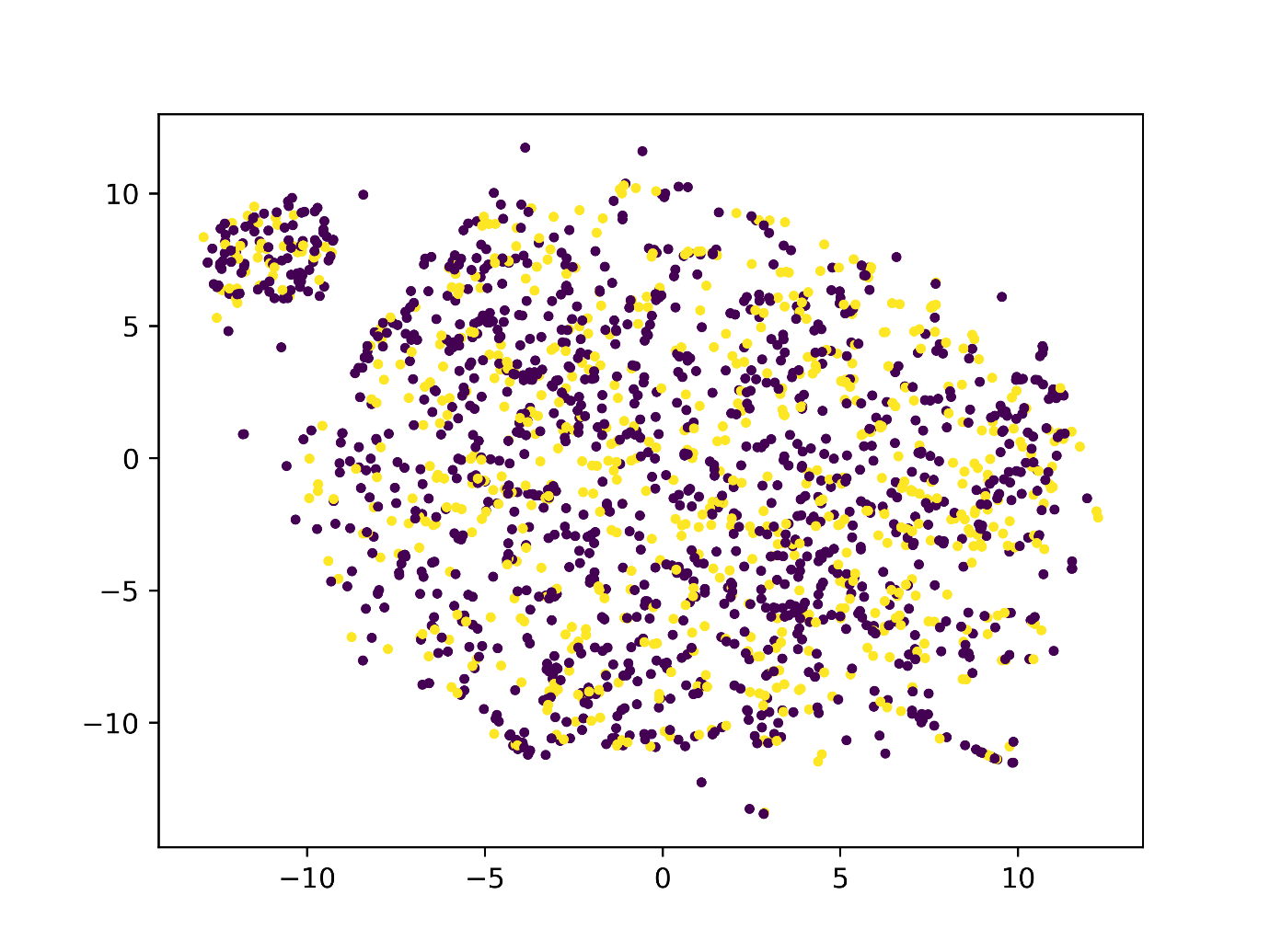
(2)Fantasy, Adventure, Action, Sci-Fi

(3)Drama, Musical, Romance

(4)Crime, Thriller, Horror, Film-Noir, Mystery

(5)Western, War, Documentary

將movie embedding layer的weight用tsne降維後的結果表示如下:

可從上圖看出分布非常混亂，完全沒有規則可言，事實上我就算只拿只屬於(1)和(2)的電影來作圖依舊如此，結果如下:

即使只看兩群依然沒有任何分開的傾向。

可以看到不論是哪張圖，均有一小群的電影群聚在左上角，可以把他們當作在高維空間(embedding space)中比較相近的電影，可惜這些電影並不能用顏色也就是genre來區分。

我猜測會造成此結果除了train的沒有很好以外，也有可能是由於電影的genre是multilabel的，我只取第一個可能不是一個太好的做法，另外即使是被分到不同種類的電影之間也會有一些劇情或是類型的相似性，因此不太容易分開。

1. (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user information和movie information拿來使用:

將user的age和gender還有occupation作為feature(age為數字，gender為0/1，occupation為one-hot encoding)

以及movie的genre作為feature(one-hot encoding)，將這兩個information與DNN中的user和movie的embedding layer concatenate起來，之後與第4題所述架構一樣，通過三層dense(256)，加上relu和dropout。

有顯著進步，為我目前在kaggle上最好的結果，RMSE為0.8484