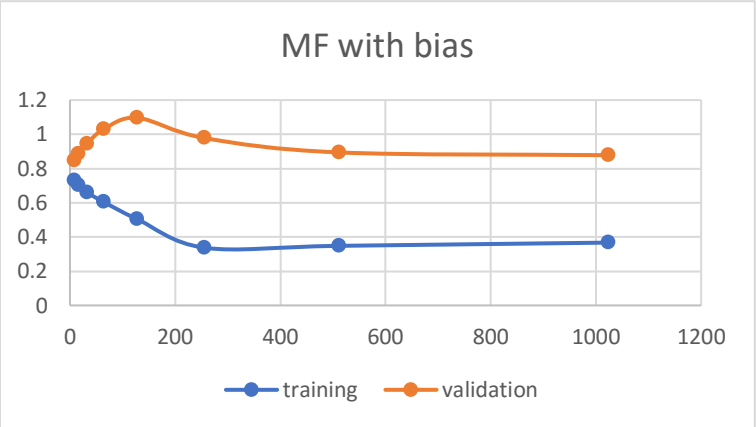
學號：b03901096　系級： 電機三　姓名：周晁德

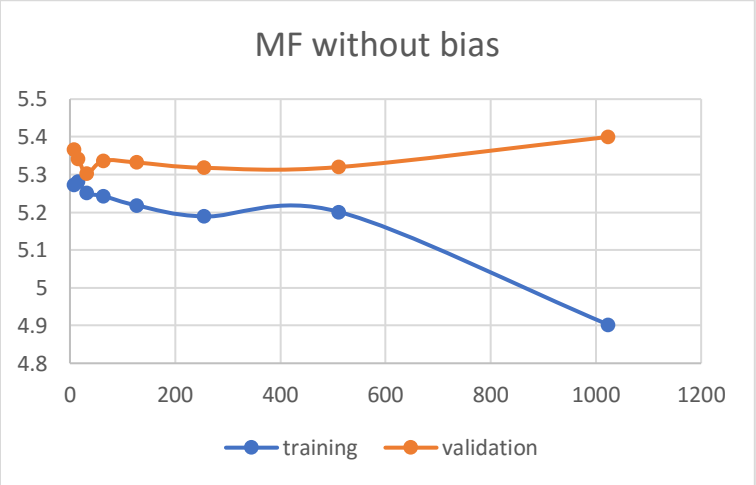
表（一）

MF without bias (loss : mse)								
latent dimension	8	16	32	64	128	256	512	1024
training	0.729	0.702	0.661	0.6047	0.504	0.337	0.3482	0.367
validation	0.848	0.886	0.944	1.0292	1.0963	0.978	0.894	0.878
MF with bias (loss : mse)								
latent dimension	8	16	32	64	128	256	512	1024
training	5.273	5.281	5.251	5.243	5.218	5.189	5.201	4.902
validation	5.367	5.342	5.303	5.337	5.332	5.318	5.3201	5.4
MF with bias and normalization								
latent dimension	8	16	32	64	128	256	512	1024
Kaggle	0.9137	0.916893	0.91745	0.91952	0.92342	0.92494	0.93075	0.9304
DNN (loss : mse)								
latent dimension	8	16	32	64	128	256	512	1024
training	0.696	0.758	0.713	0.764	0.753	0.743	0.746	0.758
validation	0.771	0.759	0.774	0.769	0.77	0.766	0.756	0.755

圖（一）



圖（二）



1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.
   normalization 的方式是把 training data(包含 training set 以及自己切出來的 validation set)的每筆 label 減去平均再除以標準差，由於這個成績比較難自己數據化表現，因此我使用 Kaggle 的分數來討論！
   由表（一）我們可以發現 MF 沒有 bias 的情況下，不無論是 training set 或 validation set loss 都非常大，因此我試過很多次都無法通過 simple baseline，MF 有加 bias 的情況下，也只有 latent dimension = 8 ，可以通個握 simple baseline，分數為 0.92612，然而當我對 label 做 normalization，然後再做$pred_{test} = pred^{*}_{test} \times train_{std} + train_{mean}$，由結果可知只要 latent dimension不要太大(<256)都可以過 simple baseline。
   此外有對 label 做 normalization，train 的時候收斂比較快， validation set 的 loss 在 10 個 epochs 以內就到最低點，如果沒做 normalization，train 的時候大概要 20 幾個 epochs loss 才會到最低點！

2. (1%)比較不同的 latent dimension 的結果。
   因為我一開始是用 MF 有 bias 的方式想過 simple base line，但 latent dimension 從 64 以 2 的倍數增加到 1024 都無法過 simple base line，而助較有提示 latent dimension 不用太大！再加上 latent dimension 如果太小 loss 直接 = np.nan， 所以這題我從 latent dimension = 8 開始測試！
   由表（一）以及圖（一）我們可以發現，隨著 latent dimension 的增加，training set 的 loss 會漸減，大約在 latent dimension = 256 之後幾乎就不太會有改變，但對於 validation set 來說隨著 latent dimension 的增加 loss 先增後減，但之後隨 latent dimension 增加減少很慢，也因此我一開始助教公布 code 前，我是從 latent dimension = 64，開始增加，因此無法用 MF 加 bias 的方式衝過 simple baseline，但這次寫報告時用 latent dimension = 8 ，就過 simple baseline 了，成績為 rmse = 0.92612。

3. (1%)比較有無 bias 的結果。
   由表（一）可以很明顯的發現不論是 training set loss 或 validation set loss，有 bias 的 loss 都比沒 bias 的 loss 低很多，此外由圖（一）我們可以發現在有加 bias 的情況下，隨著 latent dimension 的增加，training set 和 validation set 的 loss 後來都有去進穩定的趨勢，但由圖（二）我們可以發現，在沒加 bias 的情況下，隨著 latent dimension 的增加，training set 的 loss 一直下降，而 validation set 的 loss 一直上升，推測原因可能是在沒加 bias 的情況下，overfitting 當 latent dimension 大的時候 overfitting 比較嚴重吧！

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。

我的做法是將 user embedding 以及 movie embedding concatenate 在一起再過 DNN 得出 rating，而 DNN 有五個 layer 第一層 layer neuron 數目最多，之後以 2 或 4 的倍數漸減，最後一層只有一個 neuron，DNN 中每個 neuron 的 activation function 是 relu，使用 DNN 就可以很輕鬆地通過 strong baseline！

由表（一），以及我在 Kaggle 的分數 DNN 可以輕鬆過 strong baseline，而 MF 只可過 simple baseline，可以發現 DNN 的表現最好，而且不論 latent dimension 變化對於 training set 和 validation set 上 loss 的影響不大，推測可能原因是因為 concatenate 之後的 DNN，可以再把 embedding的 output 再做比較複雜的轉換，而不是單純做 dot。

我也試過把把 DNN 最後一層用 softmax，來決定 output 的分數是 1~5 中的哪一個，但結果並沒有最後一層只有一個 neuron，直接 predict 分數來得好！！

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

   我先取每個 movie 的第一個類別，所以總共有 18個類別，在對這 18 個類別分成五類：

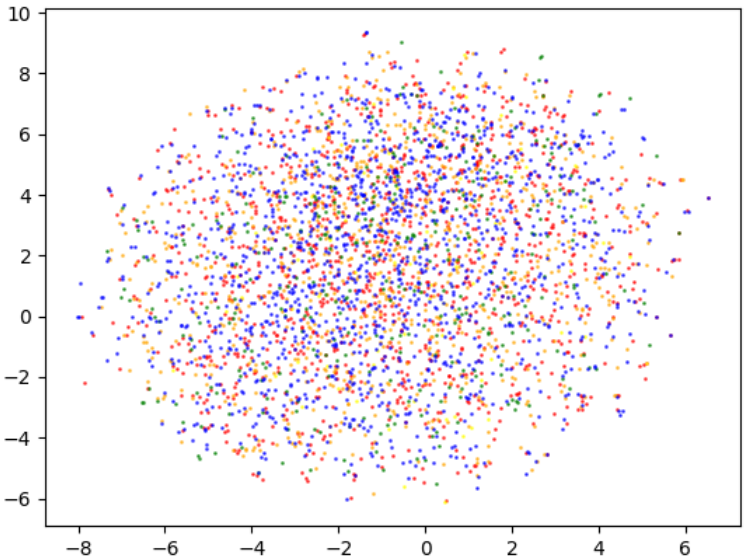
   Animation + Children’ s+ Comedy→ red

   Adventure + Action +war +Crime + Film-noir→ $orange$

   Sci-fi +Mystery +Fantasy→ $yellow$

   Thriller + horror → $green$

   Drama + Musical + Documentary + Romance +Western→ $blue$

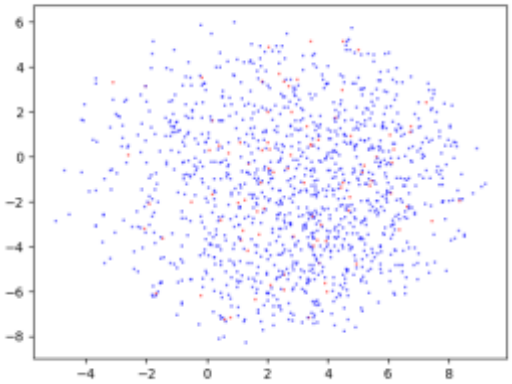


顯然這些都點都混在一起了，可能原因為：

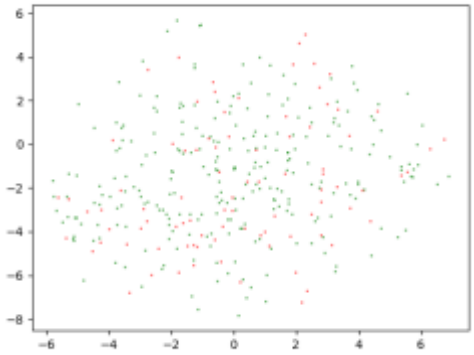
1. 我選擇電影分類方式不太好
2. 原始 embedding 不是做得很好，導致降維之後分不太開

因此我改成選兩種我覺得類型差異比較大的電影做圖：

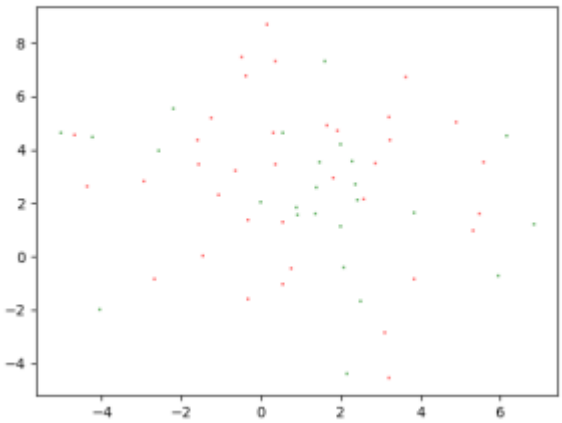
(1) animation : red
drama : blue



(2) Children’ s : red
Horror : green

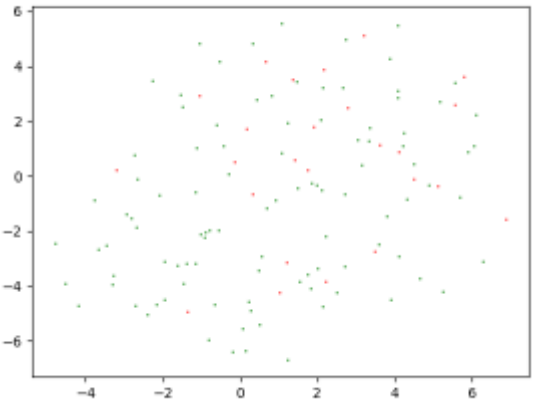


(3) Mystery : red
Musical : green



有這張圖發現 Mystery 和 Musical 算是比較分得開的 Musical 大部分在右下角，Mystery 大部分在左下角！

(4) Musical : red
Thriller : green



Musical 大部分在右上角，Thriller 雖然分佈在整個平面，但在坐下角的部分比較集中一些！

由以上幾張作圖可知，其實 embedding 還是有將一些電影種類分開的，但是並不是分的好。

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。

	latent dim	8	16	32	64
age+occupat+gen	training	0.719	0.691	0.689	0.705
	validation	0.771	0.773	0.764	0.768

我使用第四題的 DNN model， 在 concatenate 的時候，除了原本 user_embed 和 movie_embed 的結果之外也加入 user 的 age, occupation, gender，但所得到的結果並沒有明顯比較好（與表（一）比較），可能原因是因為使用 DNN，我們其實就是希望 DNN 可以自己學到那些重要的 feature，而對於這次作業來說最重要的 feature 應該就是 embedding 後的結果，而 DNN 很有可能最後也是以此判斷分數，因此最後結果跟原來（表（一））的結果差異不大！