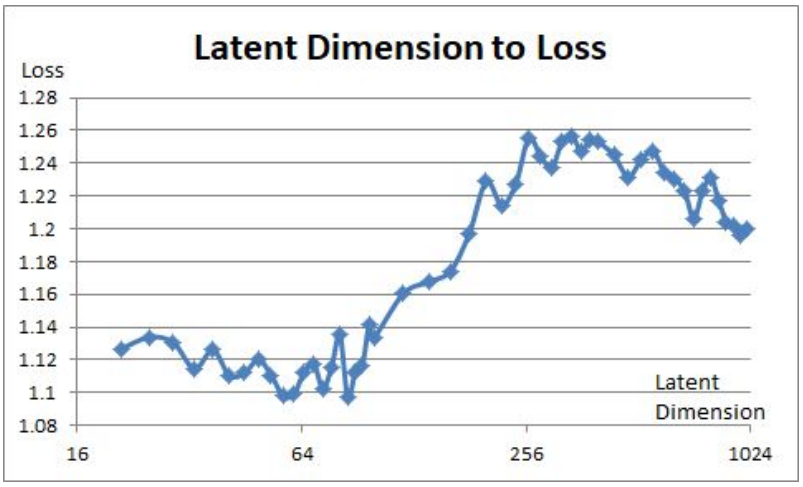
1. (1%)請比較有無normalize(rating)的差別。並說明如何normalize.
   我嘗試normalize的方式為將所有分數都減掉平均後除以一個標準差。做為參考，我求得的平均為3.78，標準差為1.11。以Matrix Factorization的實作方法為依據，未標準化之前的Loss(MSE)為1.1268，標準化之後的Loss為0.8788。由此可見，此normalize方法對於結果有著相當顯著的正面影響。

2. (1%)比較不同的latent dimension的結果。
   取51個資料點後作圖結果如右。我們可以看出latent dimension對於Loss有著不小影響。不過約30~100的範圍內，有著差不多良好的表現。（本實驗所有資料點都只有跑一次，但是重複training的過程每次都會跑出略有差異的結果。因只能下個約略的結論）。Latent Dimension 我把他理解為個人喜好的電影特徵的向量，以及電影本身特徵的向量。譬如我特別喜歡恐怖類的電影，而某電影正好是恐怖類的，那麼我們這兩個數值都高，而點積出來結果自然也有較高的數值。



3. (1%)比較有無bias的結果。
   無bias的結果1.2381
   有bias的結果1.0299

   可以看出很明顯的差異。bias的用意是針對個別user和個別movie提供一個offset，可以有效針對個人用戶以及個別電影收到的給分做調整。因此對於預測的結果有幫助也是合情合理。

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。
   NN: 0.8927
   MF: 0.8788

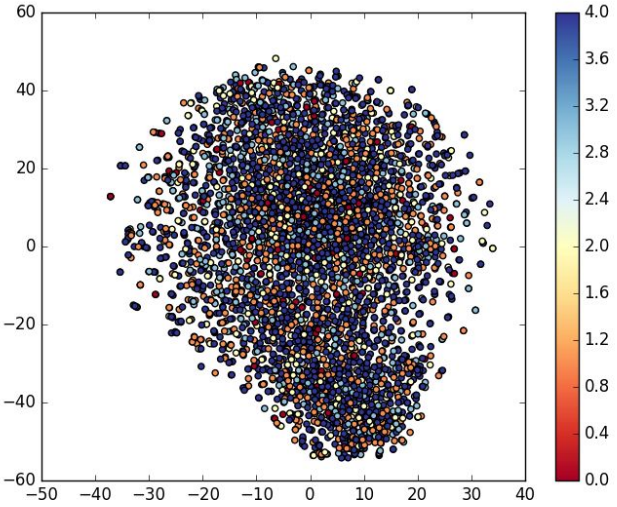
   DNN的實作方法前半部與MF相似，皆為針對每個用戶以及每部電影建立專屬的向量，但DNN的後半部為將用戶以及電影的兩個向量並列後一起丟入神經網路中給他找關聯。DNN出來的

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 50)	302000	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 50)	185300	input_2[0][0]
flatten_1 (Flatten)	(None, 50)	0	embedding_1[0][0]
flatten_2 (Flatten)	(None, 50)	0	embedding_2[0][0]
concatenate_1 (Concatenate)	(None, 100)	0	flatten_1[0][0] flatten_2[0][0]
dense_1 (Dense)	(None, 150)	15150	concatenate_1[0][0]
dense_2 (Dense)	(None, 50)	7550	dense_1[0][0]
dense_3 (Dense)	(None, 1)	51	dense_2[0][0]

Total params: 510,051
Trainable params: 510,051
Non-trainable params: 0

結果雖然略遜於MF(見題初數據)，但仍然相當有競爭力。當然我們知道DNN有很廣泛的用途，但也許在這種類型的題目上，MF就是有著更優越的表現。

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



我的分類方法為只要類型有出現'Documentary'，便設定為0。只要有出現Horror、Thriller、Film-Noir、Mystery便設定為1。只要有出現Animation、Children、Adventure、Fantasy便歸類為2。只要有出現Action、Crime、War、Western、Sci-Fi便設定為3。其餘的包括Drama、Romance、Comedy、Musical便歸類為4。而training後取出weight做圖得到的成果如右。很可惜再輸出的圖裡並沒有找到太明顯的區別。也許這可以間接達到一個結論，觀眾針對於電影的喜好不能直接依傳統電影分類作為依據。

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。

在作業剛派下來的時候，曾經有常試過用簡報教的方法以外的作法切入這次的題目。首先就是直接針對用戶給分習慣取得用戶給分平均，再根據電影所收到的分數取得電影評分平均，最後根據這兩個數字平均，當作是某用戶給予某電影的評分。可惜這個演算法所得到的kaggle分數Loss為1.26，完全不值得採用。