學號：R06942077 系級： 電信碩一 姓名：洪健鈞

1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.
   在 latent dimension 為 240 的情況下:
   無 normalize 的 public score 為 1.18984, private score 為 1.18677
   有 normalize 的 public score 為 0.88111, private score 為 0.87868

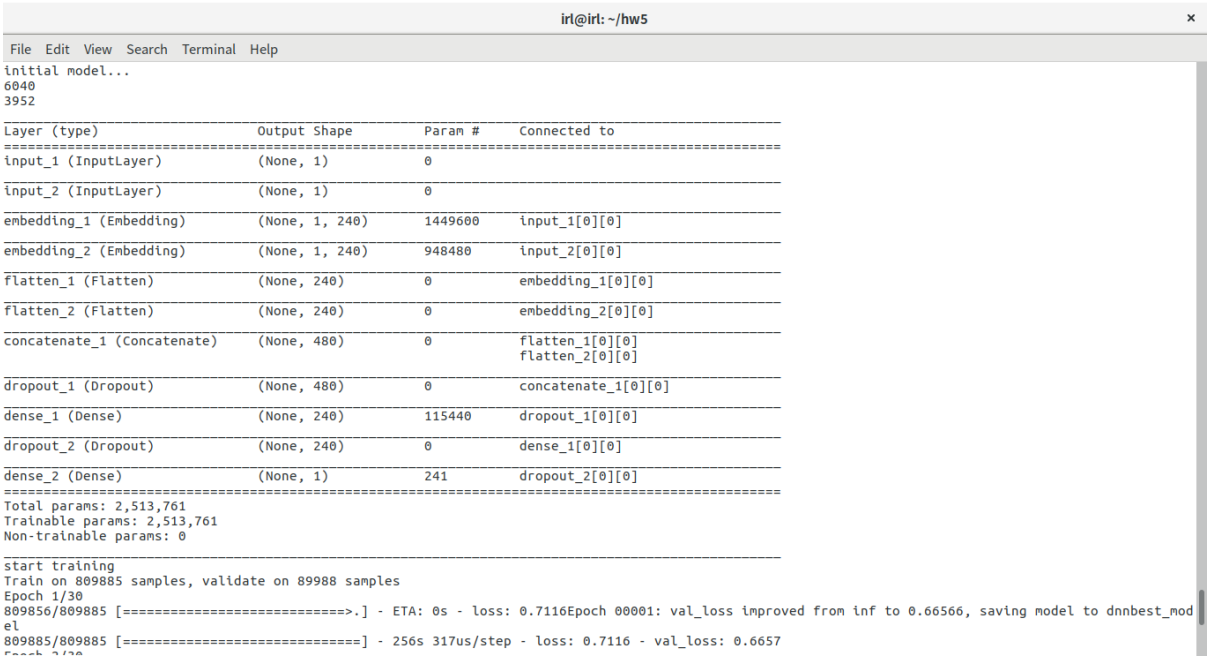
   我 normalize rating 的方式為減掉平均值(mean),在除以標準差(std)
   Normalized_rating = (rating- mean) / std

2. (1%)比較不同的 latent dimension 的結果。
   在無 normalize 的情況下
   Dimension 120 public score 為 0.87820, private score 為 0.87834
   Dimension 240 public score 為 0.88104, private score 為 0.87966
   Dimension 480 public score 為 0.89013, private score 為 0.88911

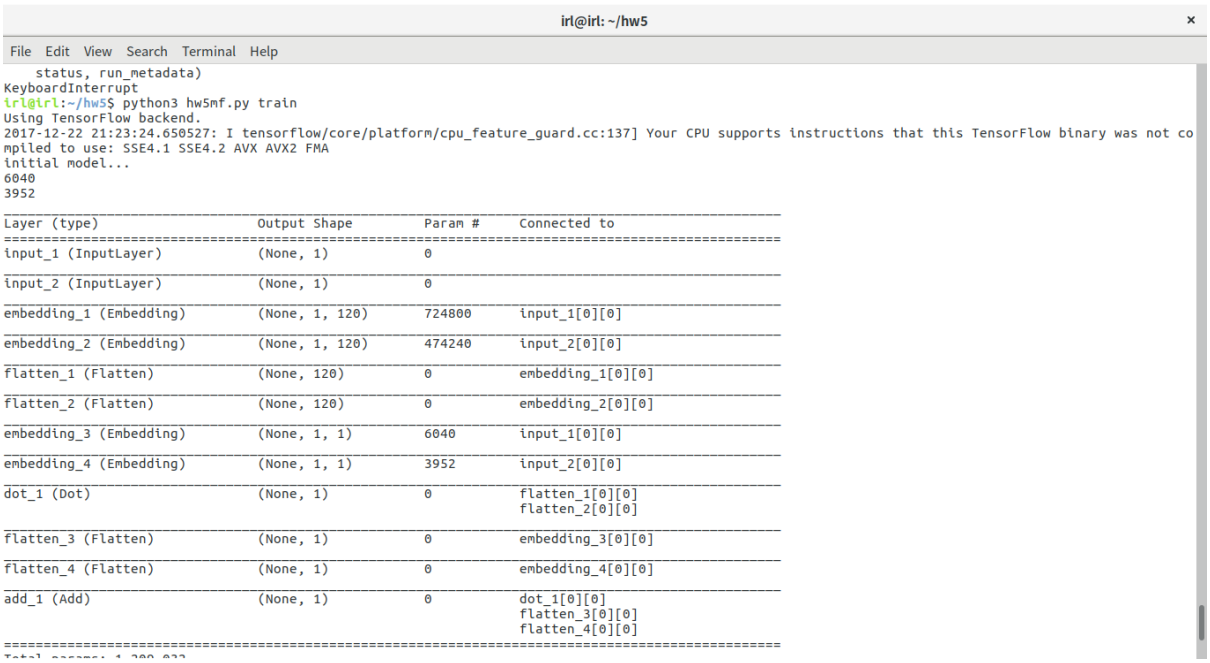
   Dimension 較低, 預測結果較準確。

3. (1%)比較有無 bias 的結果。
   有加 bias 項的情況(latent dimension=120), public score 為 0.87820, private score 為 0.87834
   然而沒加 bias 項, public score 為 1.19992, private score 為 1.19835
   每個人可能都會有自己評分的傾向，像是傾向於把每部電影都評得很高分或者很低分； 同理, movie 也會有這個現象, 所以加入 bias 來修正使用者造成的偏差, 可以使 model 更為準確。

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。

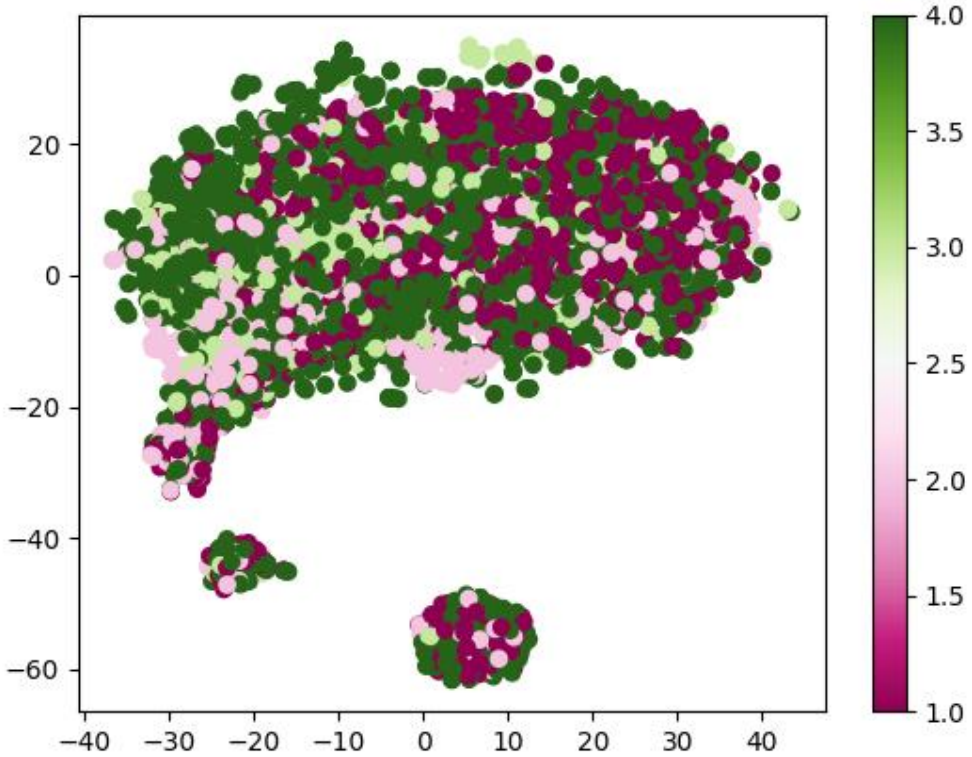


我的 DNN model 架構如圖所示, 將 user embedding 以及 movie embedding concatenate 在一起, 然後加入 dropout 防止 overfitting, 再經由 DNN 得出 rating。
Latent dimension 設 240, dropout rate 設 0.3, 跑 30 個 epoch。並設定 earlystop。
準確率為: public score= 0.85358, private score= 0.85676 通過 strong baseline.
MF 使用內積來變成純量, NN 則是用 dense 。
這個 model 出來的準確率也比已經 normalized 過的 MF model 還高。



上圖為我的 MF model

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



Label1: Drama, Musical
Label2 : Thriller, Horror, Crime, Film-Noir
Label3: Adventure, Animation,Children
Label4: Others

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 dimension 設定為 480, 跑 30 個 epoch, 切 0.1 的 validation set, dropout rate 設定為 0.3, 多加入使用者資訊 “性別”, 及”年齡”, 總共使用 4 個 embedding , concatenate 在一起訓練。並用 relu 及 linear activation。
Optimizer 用 adamax。結果 public score 為 0.85445 private 為 0.85771

irl@irl: ~/hw5

File Edit View Search Terminal Help

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
input_1 (InputLayer)         (None, 1)                 0
__________________________________________________________________________________________________
input_2 (InputLayer)         (None, 1)                 0
__________________________________________________________________________________________________
embedding_1 (Embedding)      (None, 1, 480)            2899200     input_1[0][0]
__________________________________________________________________________________________________
embedding_2 (Embedding)      (None, 1, 480)            1896960     input_2[0][0]
__________________________________________________________________________________________________
input_3 (InputLayer)         (None, 1)                 0
__________________________________________________________________________________________________
flatten_1 (Flatten)          (None, 480)               0           embedding_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)          (None, 480)               0           embedding_2[0][0]
__________________________________________________________________________________________________
embedding_3 (Embedding)      (None, 1, 480)            960         input_3[0][0]
__________________________________________________________________________________________________
input_4 (InputLayer)         (None, 1)                 0
__________________________________________________________________________________________________
concatenate_1 (Concatenate)  (None, 960)               0           flatten_1[0][0]
                                                                   flatten_2[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)          (None, 480)               0           embedding_3[0][0]
__________________________________________________________________________________________________
embedding_4 (Embedding)      (None, 1, 480)            5760        input_4[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)  (None, 1440)              0           concatenate_1[0][0]
                                                                   flatten_3[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)          (None, 480)               0           embedding_4[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)  (None, 1920)              0           concatenate_2[0][0]
                                                                   flatten_4[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)          (None, 1920)              0           concatenate_3[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)              (None, 480)               922080      dropout_1[0][0]
__________________________________________________________________________________________________
```

~/hw5/hw5Q6.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

hw5Q6.py | hw5.py | hw5dnn.py | hw5mf.py | hw5_best.sh

```
76      r_hat = Add()([r_hat, user_bias, item_bias])
77      model = keras.models.Model([user_input, item_input], r_hat)
78      model.compile(loss='mse', optimizer='adamax') #opt=sgd
79      return model
80
81
82  # set the UV decomposition model
83  print('initial model...')
84  def dnnmodel(users, items, age, dim, dropout_rate):
85      user = Input(shape=[1])
86      item = Input(shape=[1])
87      gender = Input(shape=[1])
88      age_input = Input(shape=[1])
89
90      user_vec = Embedding(users, dim, embeddings_initializer='random_normal')(user)
91      user_vec = Flatten()(user_vec)
92      item_vec = Embedding(items, dim, embeddings_initializer='random_normal')(item)
93      item_vec = Flatten()(item_vec)
94      gender_vec = Embedding(2, dim, embeddings_initializer='random_normal')(gender)
95      gender_vec = Flatten()(gender_vec)
96      age_vec = Embedding(age, dim, embeddings_initializer='random_normal')(age_input)
97      age_vec = Flatten()(age_vec)
98
99      model = Concatenate()([user_vec,item_vec])
100     model = Concatenate()([model, gender_vec])
101     model = Concatenate()([model, age_vec])
102
103     model = Dropout(dropout_rate)(model)
104     model = Dense(dim, activation='relu')(model)
105     model = Dropout(dropout_rate)(model)
106     model = Dense(1, activation='linear')(model)
107     model =keras.models.Model([user, item, gender, age_input], model)
108     model.compile(loss = 'mse', optimizer = 'adamax')
109     return model
110
111 model = dnnmodel(max_user, max_movie, max_age, dimension, dropout_rate)
112 #model = mfmodel(max_user, max_movie)
```

sort | Find | Find Prev | Find All

ASCII, 2 matches | Tab Size: 4 | Python