Machine Learning HW6 Report

電機四 B03901015 梅希聖

1. (1 %)請比較有無 normalize 的差別。並說明如何 normalize.
   若要在 matrix factorization 實作 normalize，唯一可實現的部分是對 rating 做 normalize（對 user/movie ID 做 normalize 應無太大意義），比較兩者 kaggle 表現的分數差異，結果如下：

	private	public
with normalize	0.86026	0.86801
without normalize	0.86867	0.87688

   經過 normalize 後，表現並沒有比較好。
   關於訓練過程的比較，經過 normalize 後之 model 收斂速度極快，大約在的 3 個 epoch 時，val_loss 即達到最低，而無 normalize 則要在第 15 個 epoch 左右才會收斂，原因應是 keras 預設之 learning rate，在 rating 數值 range 壓縮後（原先是 1~5，轉換成 mean=0、variance=1）相對變大，因此導致收斂速度加快。

2. (1 %)比較不同的 embedding dimension 的結果。
   比較 dimension=64, 128, 196 之差異。

dimension	private	public
64	0.87137	0.87790
128	0.86026	0.86801
196	0.85941	0.86526

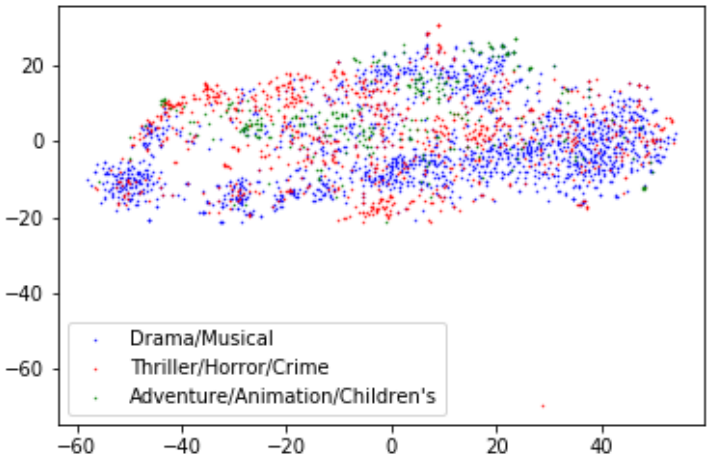
   關於訓練過程：
   d=196 時，model 在第 8 個 epoch 即達最低點。
   d=128 在第 12 個 epoch 達最低點。
   d=64 則在第 29 個 epoch 才到達最低點。
   愈高之 dimension，收斂速度愈快，原因應是 dimension 愈高，給予 model 的資訊量就會愈多，導致學習的速度愈快。

3. (1 %)比較有無 bias 的結果。
   使用相同的模型架構與訓練方式，將 user/movie bias 項移除後比較結果，在 kaggle 上之 loss 分數如下：

	private	public
with bias	0.86026	0.86801
without bias	0.86017	0.86707

   兩者之分數並無太大差異，但有 bias 之 model 收斂速度較快，在第 12 個 epoch 即收斂，無 bias 之 model 則要到第 21 個 epoch 才收斂，應是模型因為多了可訓練的參數，使得收斂速度變快。

4. (1 %)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。
   依助教在投影片上之提示，將 movie 分成三類：Drama/Musical, Thriller/Horror/Crime, Adventure/Animation/Children's。
   原本之 embedding layer 之 output 有 128 維，降至二維後作圖如下：



   結果顯示，並沒有發現不同類別明顯的分界線，可能是因為維度太低，embedding layer 之 output 無法在二維平面上有明顯區隔，或是在人為分類的這三大類別當中，模型並不認為他們有表現出共同的特徵。

5. (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。

```
genre_type = ["Animation", "Children's", "Comedy", "Adventure", "Fantasy",
"Romance", "Drama", "Crime", "Thriller", "Horror", "Sci-Fi",
"Documentary", "War", "Mystery", "Musical", "Western", "Action"]
```

   從 movies.csv 取出每部電影的 genre，轉換成 one-hot encoding 後，與 user/movie 之 embedding output 相連後進入 DNN 模型訓練。

```
user_input = Input(shape=(1,))
user_embedding = Embedding(input_dim=n_users+1, output_dim=d)(user_input)
user_embedding = Flatten()(user_embedding)

movie_input = Input(shape=(1,))
movie_embedding = Embedding(input_dim=n_movies+1, output_dim=d)(movie_input)
movie_embedding = Flatten()(movie_embedding)

movie_genres_input = Input(shape=(17,))

concat = concatenate([user_embedding, movie_embedding, movie_genres_input], name='Concat')
```

   DNN 之模型架構如下：

```
dense_1 = Dense(256,name='FullyConnected-1', activation='relu')(concat)
dropout_1 = Dropout(0.2,name='Dropout1')(dense_1)
dense_2 = Dense(128,name='FullyConnected-2', activation='relu')(dropout_1)
dropout_2 = Dropout(0.2,name='Dropout2')(dense_2)
dense_3 = Dense(64,name='FullyConnected-3', activation='relu')(dropout_2)
dropout_3 = Dropout(0.2,name='Dropout3')(dense_3)

result = Dense(1, activation='relu',name='Activation')(dropout_3)
model = Model([user_input, movie_input, movie_genres_input], result)
model.compile(loss='mse', optimizer='rmsprop')
```

結果如下：

	private	public
MF	0.86026	0.86801
DNN	0.87133	0.87658
DNN + movie feature	0.85060	0.86152

結果顯示，加入了電影類型之特徵，有助於提升預測結果之準確度，結果與預期相符。