Machine Learning 2017 HW6 Report

Matrix Factorization

B03901156 Yu Xuan Huang

1. 請比較有無 normalize(rating)的差別。並說明如何 normalize.

- How to normalize

以 matrix factorization 的 model 為例，首先利用 numpy.mean 及 numpy.std 算出 training data 中 rating 的平均值及標準差；再利用以下公式將 training data 的 rating 做 normalization:

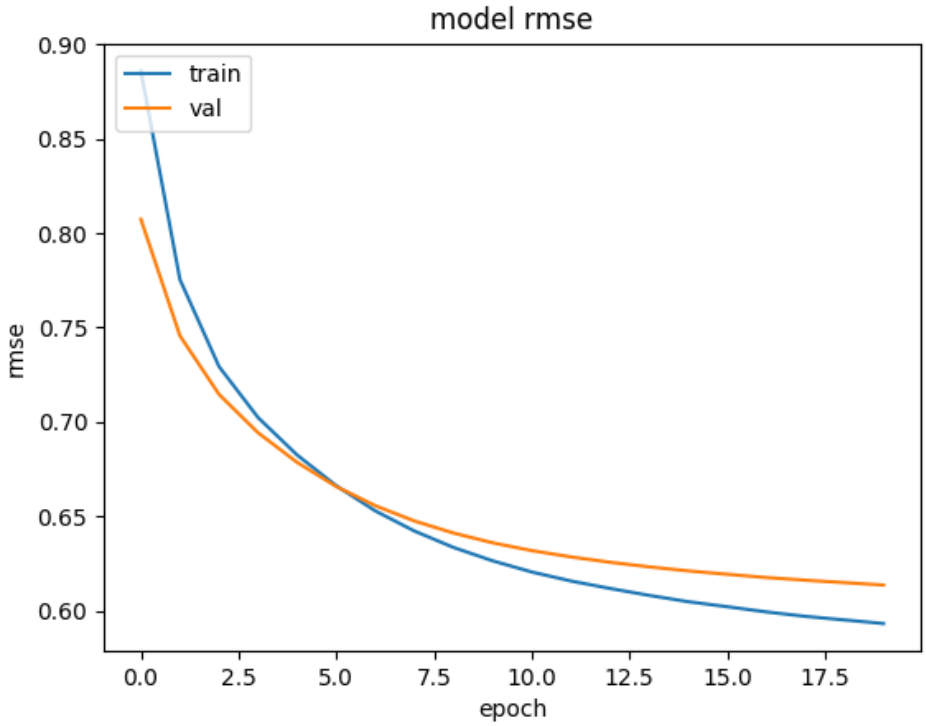
$$rating_{train\_normalized} = (rating_{original} - rating_{train\_mean})/rating_{train\_std}$$

而在最後 predict testing data 時，需將所 predict 出的值以下列公式還原成原先的 rating 值:

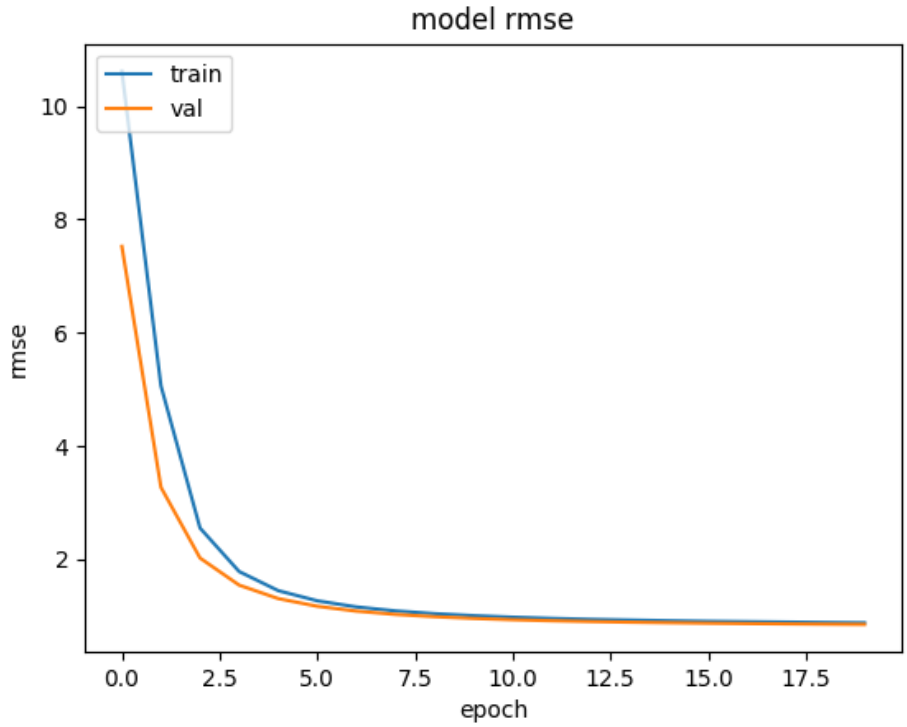
$$rating_{test} = predict * rating_{train\_std}) + rating_{train\_mean}$$

- Difference of two models (model parameters: epochs = 20, validation_split = 0.1, batch_size = 128, latent dimension = 5)

Implementation of normalization: (Kaggle rmse=**0.87801**)



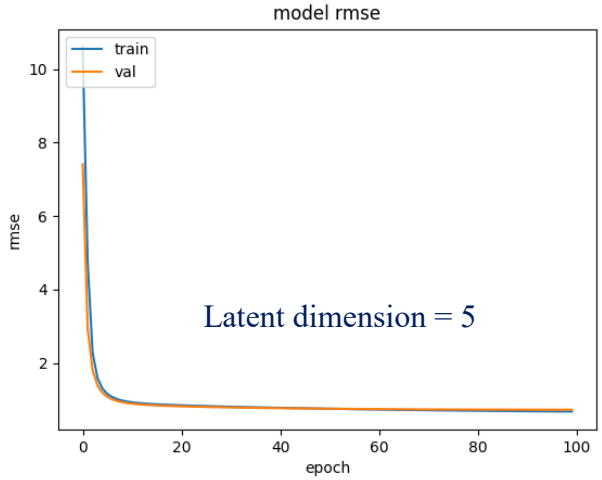
Implementation of not performing normalization: (Kaggle rmse=**0.92957**)

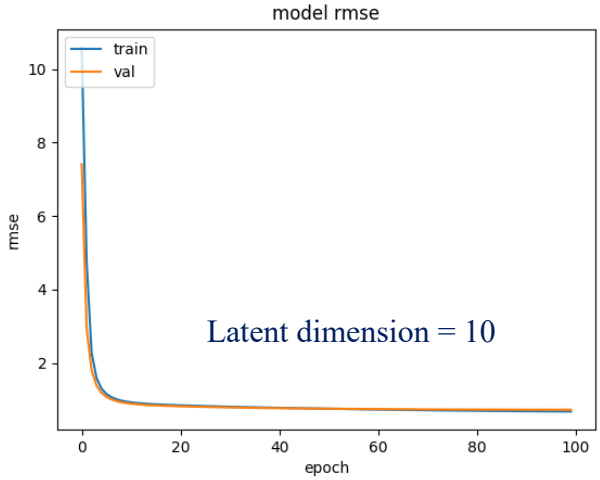


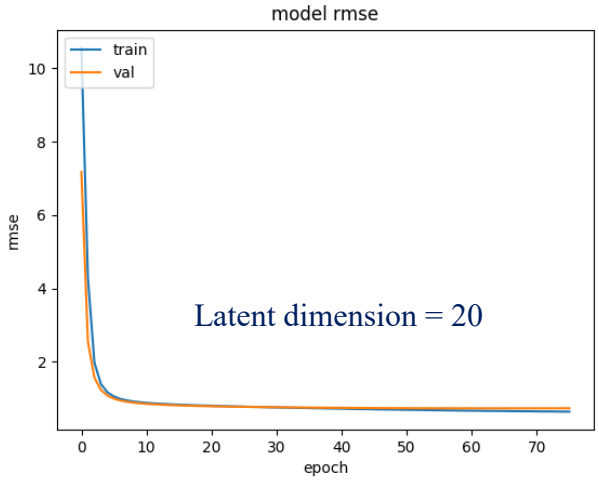
由上列結果觀察可知: rating normalization 可加速 matrix factorization model 的 rmse loss 收斂速率，在相同 epoch 數的 training 下，能獲得較好的 model，kaggle 上分數也較為優越。

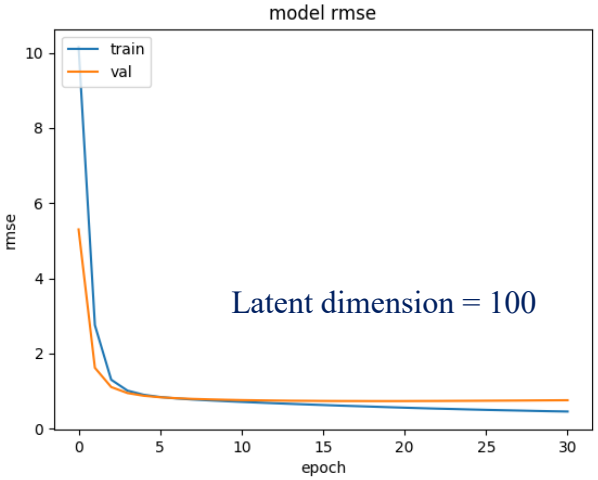
2. 比較不同的 latent dimension 的結果。

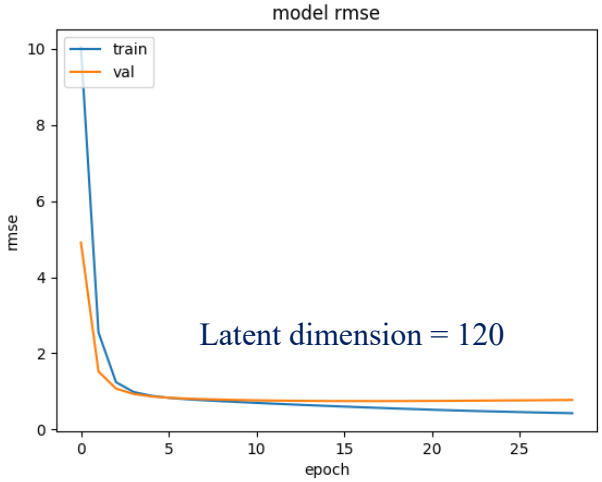
(model parameters: epochs = 100, validation_split = 0.1, batch_size = 128, optimizer = 'adamax')











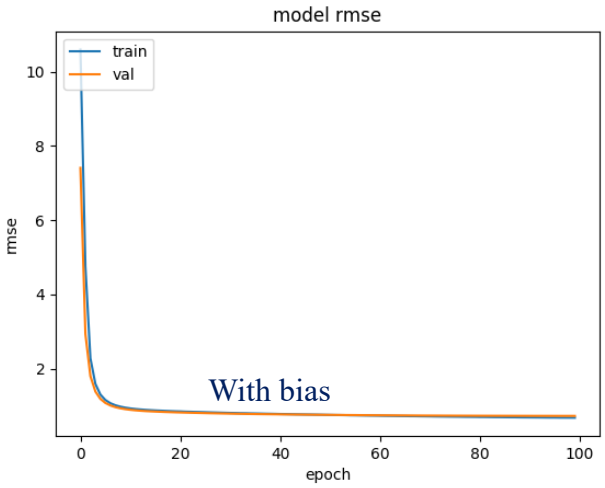
Latent dimension	5	10	20	100	120
RMSE	0.87093	0.85930	0.85752	0.86663	0.86590

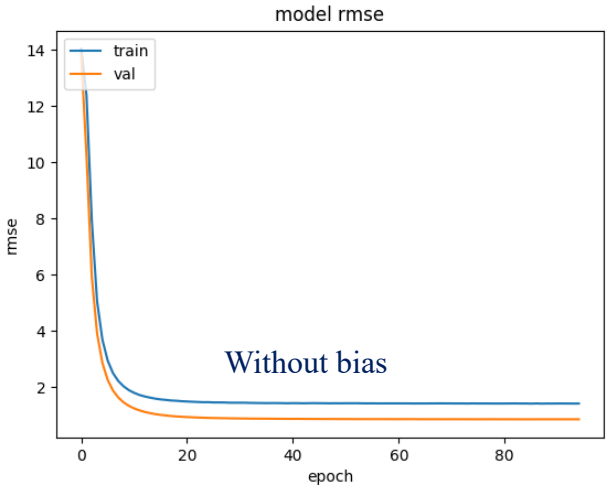
由上述結果觀察可知: latent dimension 越大雖會增加單次 epoch 下 training 所需花費的時間，但其訓練 epoch 數結束較早；而當 latent dimension = 5~20 時，latent dimension 越大所得到的結果(RMSE)較好，但當 latent dimension >=20 時，latent dimension 越大所得結果(RMSE)卻較當 latent dimension = 5~20 時的 model 差。因此從實驗數據表中可推論當 latent dimension

在 5~20 時，training rate 雖會隨著 latent dimension 的加大而降低，但 RMSE 明顯變小，確實對結果能有所改善。

3. 比較有無 bias 的結果。

(model parameters: epochs = 100, validation_split = 0.1, batch_size = 128, latent_dimension = 5, optimizer = 'adamax')





Matrix Factorization Model	RMSE
With bias	0.87093
Without bias	0.89587

因每個 user 可能都會有各自評分的取向，例如傾向於將每部電影都評得較為高分或較為低分；而相同地，每部電影所得評分也可能具有類似的偏差情形，故為了降低這種現象對於結果的影響，在 training 的 model 中加入 bias 項: $r_{i,j} = U_i * V_j + b_i^{user} + b_j^{movie}$.

而從上述實驗結果觀察可知: 加入 bias 所 train 出的 model 確實能有效降低 RMSE，顯示 bias 確實能抑制評分傾向過高或過低對 test 結果可能造成的影響。

4. 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。

Implementation:

```python
def nn_model(n_users, n_items, latent_dim):
    user_input = Input(shape=[1])
    item_input = Input(shape=[1])

    user_vec = Embedding(n_users+100, latent_dim, embeddings_initializer='random_normal')(user_input)
    user_vec = Flatten()(user_vec)

    item_vec = Embedding(n_items+300, latent_dim, embeddings_initializer='random_normal')(item_input)
    item_vec = Flatten()(item_vec)

    merge_vec = Concatenate()([user_vec,item_vec])
    # DNN
    hidden = Dropout(0.1)(merge_vec)
    hidden = Dense(120,activation='relu')(hidden)
    hidden = Dropout(0.1)(hidden)
    # output
    output = Dense(1,activation='linear')(hidden)
    model = keras.models.Model([user_input,item_input],output)
    model.compile(loss='mse',optimizer='adamax')
    return model
```

將 user embedding 及 movie embedding concatenate 在一起後再過 DNN 得出 rating，而 output 則將其視為 regression 問題。

Model structure:

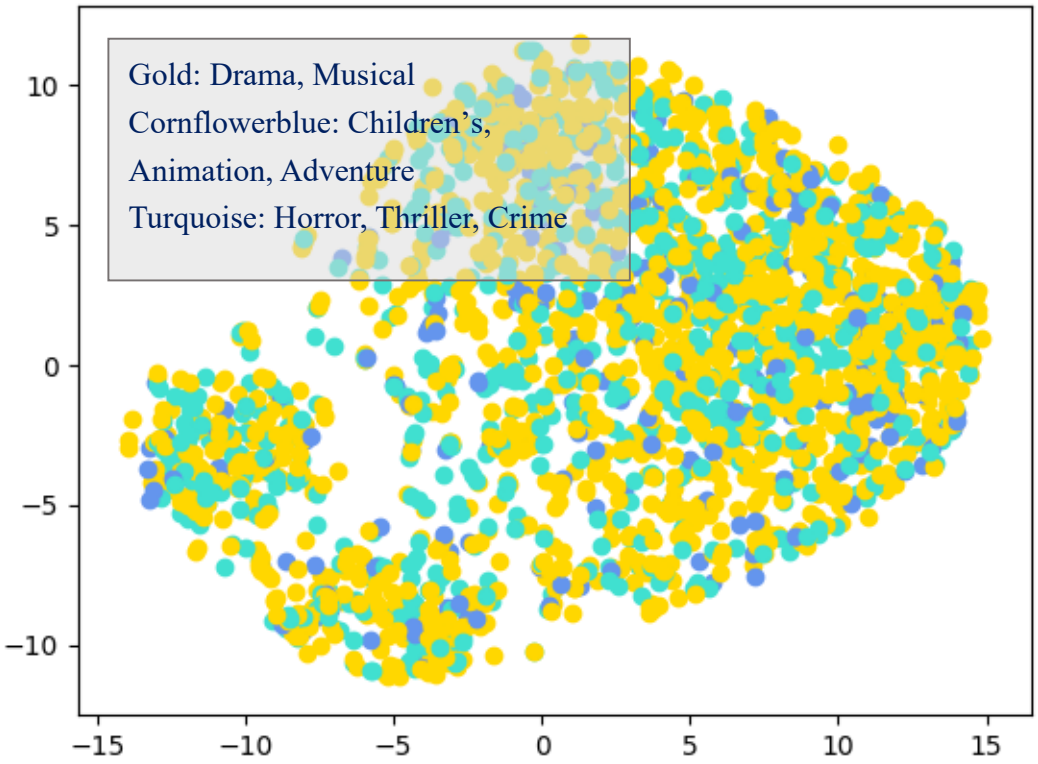
```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
input_1 (InputLayer)         (None, 1)                 0
__________________________________________________________________________________________________
input_2 (InputLayer)         (None, 1)                 0
__________________________________________________________________________________________________
embedding_1 (Embedding)      (None, 1, 120)            736800      input_1[0][0]
__________________________________________________________________________________________________
embedding_2 (Embedding)      (None, 1, 120)            501960      input_2[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)          (None, 120)               0           embedding_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)          (None, 120)               0           embedding_2[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)  (None, 240)               0           flatten_1[0][0]
                                                                   flatten_2[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)          (None, 240)               0           concatenate_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)              (None, 120)               28920       dropout_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)          (None, 120)               0           dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)              (None, 1)                 121         dropout_2[0][0]
==================================================================================================
Total params: 1,267,801
Trainable params: 1,267,801
Non-trainable params: 0
__________________________________________________________________________________________________
```

Model	RMSE
Matrix Factorization model	0.86202
DNN model	0.86663

在上述的 Matrix Factorization model 中其參數為(epochs = 1000, validation_split = 0.1, batch_size = 128, latent_dimension = 5, optimizer = ‘adamax’)，而 DNN model 的參數為(epochs

= 1000, validation_split = 0.1, batch_size = 128, latent_dimension = 120, optimizer = ‘adamax’)；經實驗後得知 Matrix Factorization model 所得 RMSE 較 DNN model 來得低，但其訓練過程較為緩慢，而 DNN model 則有較佳的訓練速度。

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



6. 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。除了可以依照本次作業以評分 rating 作為最直接的喜好(Explicit Feedback)，也可額外考慮將 user 在評分當下對於推薦項目間接的喜好，也就是 implicit feedback；利用 rating 是否大於 4 或是 rating 平均值做為新的 feature 進行 model training，以期望能得到較好的效果。(model parameters: epochs = 1000, validation_split = 0.1, batch_size = 128, latent_dimension = 5, optimizer = ‘adamax’)，結果如下:

Model	**RMSE**
Matrix Factorization model	0.86752
Matrix Factorization model with implicit feedback	0.86590