Report: Assignment #4

ITE4005, **Data Science.**
2016025305, **Jihun Kim<**jihunkim@hanyang.ac.kr**>.**

Predict the ratings of movies in test data by using the given training data containing movie ratings of users.

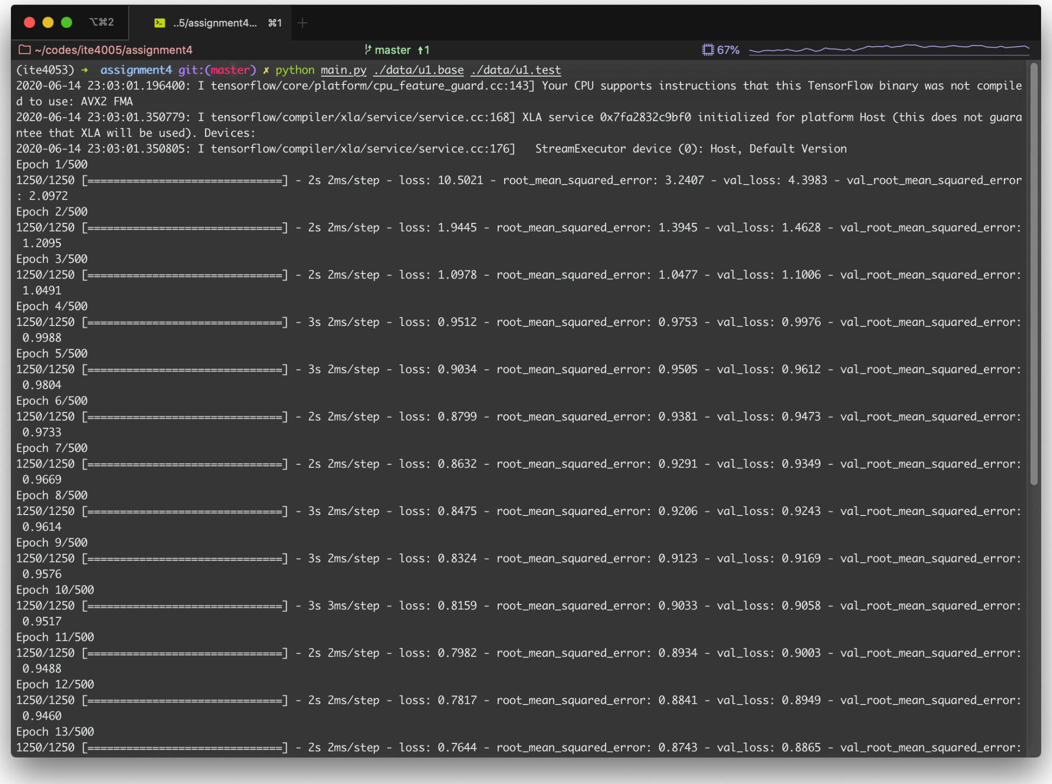
Getting Started

Development Environment

- **OS:** macOS 10.15.5
- **Language:** Python 3.8.3 (TensorFlow 2.2.0, NumPy 1.18.5)

Run

```
$ cd /path/to/repo/assignment4
$ pip install -r requirements.txt
$ python main.py ./data/u1.base ./data/u1.test
```



```
(ite4053) ➜  assignment4 git:(master) ✗ python main.py ./data/u1.base ./data/u1.test
2020-06-14 23:03:01.196400: I tensorflow/core/platform/cpu_feature_guard.cc:143] Your CPU supports instructions that this TensorFlow binary was not compile
d to use: AVX2 FMA
2020-06-14 23:03:01.350779: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x7fa2832c9bf0 initialized for platform Host (this does not guara
ntee that XLA will be used). Devices:
2020-06-14 23:03:01.350805: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
Epoch 1/500
1250/1250 [==============================] - 2s 2ms/step - loss: 10.5021 - root_mean_squared_error: 3.2407 - val_loss: 4.3983 - val_root_mean_squared_error
: 2.0972
Epoch 2/500
1250/1250 [==============================] - 2s 2ms/step - loss: 1.9445 - root_mean_squared_error: 1.3945 - val_loss: 1.4628 - val_root_mean_squared_error:
 1.2095
Epoch 3/500
1250/1250 [==============================] - 2s 2ms/step - loss: 1.0978 - root_mean_squared_error: 1.0477 - val_loss: 1.1006 - val_root_mean_squared_error:
 1.0491
Epoch 4/500
1250/1250 [==============================] - 3s 2ms/step - loss: 0.9512 - root_mean_squared_error: 0.9753 - val_loss: 0.9976 - val_root_mean_squared_error:
 0.9988
Epoch 5/500
1250/1250 [==============================] - 3s 2ms/step - loss: 0.9034 - root_mean_squared_error: 0.9505 - val_loss: 0.9612 - val_root_mean_squared_error:
 0.9804
Epoch 6/500
1250/1250 [==============================] - 2s 2ms/step - loss: 0.8799 - root_mean_squared_error: 0.9381 - val_loss: 0.9473 - val_root_mean_squared_error:
 0.9733
Epoch 7/500
1250/1250 [==============================] - 2s 2ms/step - loss: 0.8632 - root_mean_squared_error: 0.9291 - val_loss: 0.9349 - val_root_mean_squared_error:
 0.9669
Epoch 8/500
1250/1250 [==============================] - 3s 2ms/step - loss: 0.8475 - root_mean_squared_error: 0.9206 - val_loss: 0.9243 - val_root_mean_squared_error:
 0.9614
Epoch 9/500
1250/1250 [==============================] - 3s 2ms/step - loss: 0.8324 - root_mean_squared_error: 0.9123 - val_loss: 0.9169 - val_root_mean_squared_error:
 0.9576
Epoch 10/500
1250/1250 [==============================] - 3s 3ms/step - loss: 0.8159 - root_mean_squared_error: 0.9033 - val_loss: 0.9058 - val_root_mean_squared_error:
 0.9517
Epoch 11/500
1250/1250 [==============================] - 2s 2ms/step - loss: 0.7982 - root_mean_squared_error: 0.8934 - val_loss: 0.9003 - val_root_mean_squared_error:
 0.9488
Epoch 12/500
1250/1250 [==============================] - 2s 2ms/step - loss: 0.7817 - root_mean_squared_error: 0.8841 - val_loss: 0.8949 - val_root_mean_squared_error:
 0.9460
Epoch 13/500
1250/1250 [==============================] - 2s 2ms/step - loss: 0.7644 - root_mean_squared_error: 0.8743 - val_loss: 0.8865 - val_root_mean_squared_error:
```

Implementation

For predicting the ratings, I used simple correlative filtering. Implementation has done with Keras with TensorFlow as backend.

Model Construction

I constructed simple model that performs dot product of two embedded vectors. One vector represents a user, and another vector represents a movie. The dimension of vectors are empirically set for 20. Lower dimensions occurred the increase of loss, while higher dimensions occurred overfitting.

```
def cf_model(n_users, n_movies):
    user_input = layers.Input(shape=(1,))
    user_x = layers.Embedding(n_users, 20, input_length=1, name='user_embed')(user_input)
    item_input = layers.Input(shape=(1,))
    item_x = layers.Embedding(n_movies, 20, input_length=1, name='item_embed')(item_input)
    rating = layers.Dot(axes=-1)([user_x, item_x])
    rating = layers.Flatten()(rating)
    return models.Model([user_input, item_input], rating)
```

Model Compilation

... and I compiled model. Used Adam as optimizer, because it shows great performance in most situations.

```
model.compile(optimizer=optimizers.Adam(1e-3),
              loss=losses.mean_squared_error,
              metrics=[metrics.RootMeanSquaredError()])
```

Training Model

Next step is training, of course. User and movie vector embeddings are trained in training phase. Adopted early stopping method in training phase.

```
model.fit([train_data['user_id'], train_data['item_id']], train_data['rating'],
          validation_data=([test_data['user_id'], test_data['item_id']], test_data['rating']),
          callbacks=callbacks.EarlyStopping(patience=2),
          batch_size=64, epochs=500, verbose=1)
```

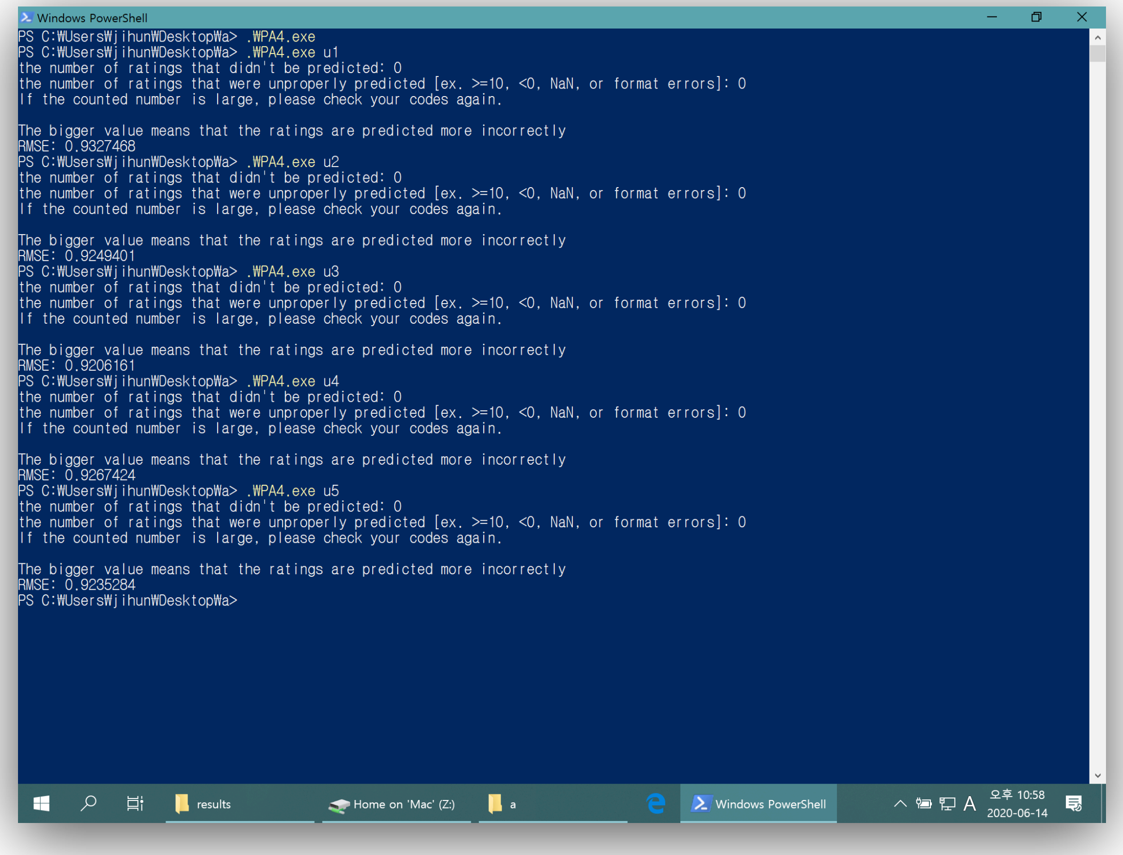
Evaluate on Test Set

For evaluation, I constructed new model which does exactly same thing with previouly constructed model, except it has single additional layer which make network output fit in range [1, 5].

```
rating = layers.Lambda(lambda x : K.minimum(K.maximum(x, 1), 5))(model.output)
final_model = models.Model(model.input, rating)
final_model.compile(loss=losses.mean_squared_error, metrics=[metrics.RootMeanSquaredError()])
final_model.evaluate([test_data['user_id'], test_data['item_id']], test_data['rating'])
    test_data['rating'] = final_model.predict([test_data['user_id'], test_data['item_id']])
```

Results

Results are below. For all train/test sets, my model shows 0.92 ~ 0.94 RMSE with no errors.



```
PS C:₩Users₩jihun₩Desktop₩a> .₩PA4.exe
PS C:₩Users₩jihun₩Desktop₩a> .₩PA4.exe u1
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9327468
PS C:₩Users₩jihun₩Desktop₩a> .₩PA4.exe u2
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9249401
PS C:₩Users₩jihun₩Desktop₩a> .₩PA4.exe u3
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9206161
PS C:₩Users₩jihun₩Desktop₩a> .₩PA4.exe u4
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9267424
PS C:₩Users₩jihun₩Desktop₩a> .₩PA4.exe u5
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9235284
PS C:₩Users₩jihun₩Desktop₩a>
```