COMP 4332 Project 3

Rating prediction

Yuen Man Him
Tsang Hing Ki

Introduction

This project is making a rating system to predict the e-book reviews ratings. We are given three sets of data: training, validation, and testing (which is the prediction) and another set of data about the products. We are also given one of the algorithms – the Collaborative Filtering Model, which we will talk about more in the next part.

Algorithms

In this project, we deploy two distinct algorithms to enhance our recommendation system:

1. **Collaborative Filtering Model**: Initially, we implemented the collaborative filtering approach as guided by existing tutorials. This model uses both reviewers' and products' data, embedding these entities to input into a multi-layer perceptron (MLP) model. The results from this model were quite satisfactory.

2. **NeuMF Model**: To further improve the system's accuracy, specifically the Root Mean Squared Error (RMSE), we integrated the Neural Matrix Factorization (NeuMF) model: This innovative model also utilizes reviewers' and products' data but processes them through dual pathways:
    - **GMF Layer**: Generalized Matrix Factorization (GMF) that performs element-wise multiplication of embeddings.

    - **MLP Layer**: A multi-layer perceptron that captures complex patterns and interactions between features.

   The outputs from these two pathways are concatenated to make the final prediction, providing a blended approach that harnesses both shallow and deep aspects of user-item interactions (based on https://arxiv.org/pdf/1708.05031).

Training Pipeline

Data Loader:

The datasets, consisting of reviews and product metadata, are initially stored in CSV and JSON formats, and loaded into memory using Pandas. The reviews data contains interactions between users (reviewers) and items (products) which are then transformed into a user-item matrix format. In this matrix:

- Each row represents a unique user.
- Each column represents a unique item.

- Matrix values represent the ratings given by users to items.

Neural Collaborative Filtering Models:

We train two primary configurations of our Neural Collaborative Filtering model:

- **Basic NCF Model**: Utilizes the dot product of embeddings to predict user ratings, simulating traditional matrix factorization.
- **Advanced NeuMF Model**: Combines Generalized Matrix Factorization (GMF) and MLP to benefit from both shallow and deep aspects of user-item interactions. The GMF component uses simple element-wise multiplication of embeddings, while the MLP component employs a series of dense layers to learn interactions at a higher level of complexity.

Hyperparameter Tuning

After using these two algorithms, we try to tune the models to get the best predictions. Here we show one of the iterations:

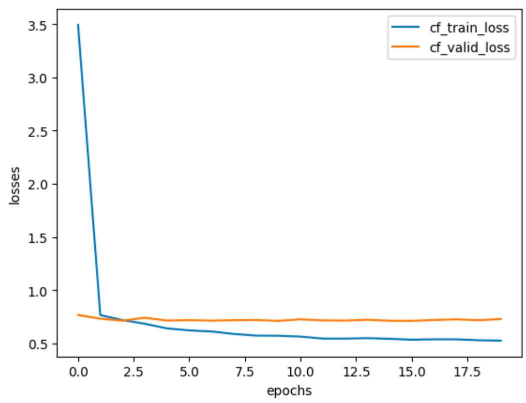

Fig.1 loss of cf model

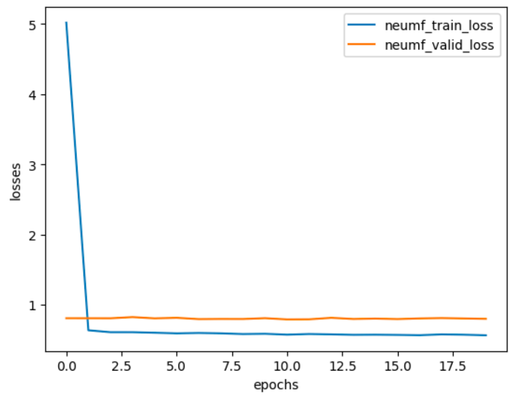

Fig. 2 loss of neuMF model

In this same iteration, the RMSEs are:

Models	Train RMSE	Validation RMSE
CF model	0.7187089770880871	0.8521158953880907
NeuMF model	0.6629437474106505	0.8386133326074092

We found that around 3 epochs are already enough and the loss of models won't change much after that. Once again, we try to automatic this process like we did in project 2. However, some random errors keep crashing our program so the process of finding the optimal model is not that smoothly.

```
1/1647 ━━━━━━━━━━━━━━━━━━━━ 27:24 999ms/step - loss: 21.2014

2024-05-26 14:45:46.605898: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: INVALID_ARGUMENT: Shapes of all inputs must match: values[0].shape = [0] != values[1].shape = []
	 [[{{function_node __inference_one_step_on_data_7588702}}{{node packed}}]]

Output exceeds the size limit. Open the full output data in a text editor
---------------------------------------------------------------------------
InvalidArgumentError                      Traceback (most recent call last)
Cell In[198], line 9
      5 model = build_NeuMF_model(n_reviewers, n_products, gmf_dim, gmf_regularizer, mlp_layers, mlp_regularizer_layers, mlp_dropout)
      7 model.compile(optimizer='adam', loss='mse')
----> 9 history = model.fit(
     10         x=[train_reviewers, train_products],
     11         y=train_ratings,
     12         batch_size=32,
     13         validation_data=([valid_reviewers, valid_products], valid_ratings),
     14         epochs=3,
     15         verbose=1,
     16         callbacks=[ModelCheckpoint('models/model.keras')])
     17 model = tf.keras.models.load_model('models/model.keras')
     18 # y_pred = model.predict([train_reviewers, train_products])
     19 # print("Train set RMSE: ", rmse(y_pred, train_ratings))

File ~/anaconda3/envs/COMP4332_Project/lib/python3.11/site-packages/keras/src/utils/traceback_utils.py:122, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    119     filtered_tb = _process_traceback_frames(e.__traceback__)
    120     # To get the full stack trace, call:
    121     # `keras.config.disable_traceback_filtering()`
--> 122     raise e.with_traceback(filtered_tb) from None
    123 finally:
    124     del filtered_tb

...

  File "/Users/him/anaconda3/envs/COMP4332_Project/lib/python3.11/site-packages/keras/src/backend/tensorflow/core.py", line 114, in convert_to_tensor

Shapes of all inputs must match: values[0].shape = [0] != values[1].shape = []
	 [[{{node packed}}]] [Op:__inference_one_step_on_iterator_7588787]
```

```
Epoch 1/3

/Users/him/anaconda3/envs/COMP4332_Project/lib/python3.11/site-packages/keras/src/layers/core/embedding.py:81: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

1/2196 ━━━━━━━━━━━━━━━━━━━━ 26:06 713ms/step - loss: 21.1051

The Kernel crashed while executing code in the the current cell or a previous cell. Please review the code in the cell(s) to identify a possible cause of the failure. Click here for more info. View Jupyter log for further details.

Canceled future for execute_request message before replies were done

There are more than 500 outputs, show more (open the raw output data in a text editor) ...
```

Fig. 3, 4 some of the errors we met and we could not solve it

Therefore, we switch to manual tuning of the model with some of the print output from the error cells as an assistant and get a better model at last.

Evaluation

Final model: The NeuMF model

```python
def build_NeuMF_model(n_reviewers, n_products, gmf_dim, gmf_regularizer=0, mlp_layers=[32], mlp_regularizer_layers=0, mlp_dropout=0.2):
    # Get the users and items input

    reviewers_input = Input(shape=(1,), dtype='int32', name='reviewers_input')
    products_input = Input(shape=(1,), dtype='int32', name='products_input')

    # Get the embeddings of users and items
    gmf_reviewers_emb = Embedding(input_dim=n_reviewers, output_dim=gmf_dim, embeddings_initializer='uniform',
                            embeddings_regularizer=l2(gmf_regularizer), name='gmf_reviwers_embedding',input_shape=1)
    gmf_products_emb = Embedding(input_dim=n_products, output_dim=gmf_dim, embeddings_initializer='uniform',
                            embeddings_regularizer=l2(gmf_regularizer), name='gmf_products_embedding',input_shape=1)
    mlp_reviewers_emb = Embedding(input_dim=n_reviewers, output_dim=int(mlp_layers[0]/2), embeddings_initializer='uniform',
                            embeddings_regularizer=l2(mlp_regularizer_layers), name='mlp_reviwers_embedding',input_shape=1)
    mlp_products_emb = Embedding(input_dim=n_products, output_dim=int(mlp_layers[0]/2), embeddings_initializer='uniform',
                            embeddings_regularizer=l2(mlp_regularizer_layers), name='mlp_products_embedding',input_shape=1)

    # GMF
    gmf_reviwers_latent = Flatten()(gmf_reviewers_emb(reviewers_input))
    gmf_products_latent = Flatten()(gmf_products_emb(products_input))
    gmf_vector = Multiply()([gmf_reviwers_latent, gmf_products_latent])

    # MLP
    mlp_reviwers_latent = Flatten()(mlp_reviewers_emb(reviewers_input))
    mlp_products_latent = Flatten()(mlp_products_emb(products_input))
    mlp_vector = Concatenate()([mlp_reviwers_latent, mlp_products_latent])
    mlp_vector = Dropout(mlp_dropout)(mlp_vector)
    for index in range(len(mlp_layers)):
        layer = Dense(mlp_layers[index], kernel_regularizer=l2(mlp_regularizer_layers),
                      activation='relu', name="layer%d" %index)
        mlp_vector = layer(mlp_vector)
        # if (index < 2):
            # mlp_vector = BatchNormalization()(mlp_vector)
            # mlp_vector = Dropout(mlp_dropout)(mlp_vector)

    predict_vector = Concatenate()([gmf_vector, mlp_vector])
    prediction = Dense(1, kernel_initializer='lecun_uniform', name="prediction")(predict_vector)

    model = Model(inputs=[reviewers_input, products_input], outputs=prediction)
```

Hyperparameters:

Hyperparameter	values
GMF layer dimension	56
GMF_regularizer, MLP_regularizer	0.001, 0.001
MLP_layer	[32, 16, 8]
MLP_Dropout	0.20
Epoch	3
Batch_size	32

RMSEs:

Train RMSE	Validation RMSE
0.6789221881052284	0.8326316969979083