Neural Collaborative Filtering: Small Dataset Results

Import Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import keras
import keras.layers
import tensorflow as tf
from keras import regularizers
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

train_data = pd.read_csv("cleandata/train_small.csv")
test_data = pd.read_csv("cleandata/test_small2.csv")
train_data = train_data.rename(columns={"userId": "user_id", "movieId": "business_id"})
test_data = test_data.rename(columns={"userId": "user_id", "movieId": "business_id"})


train_data['is_train'] = True
test_data['is_train'] = False
all_data = pd.concat([train_data,test_data])
all_data.user_id = all_data.user_id.astype('category').cat.codes.values
all_data.business_id = all_data.business_id.astype('category').cat.codes.values
is_train = all_data['is_train'] == True
train_data = all_data[is_train]
test_data = all_data[~is_train]


all_data
```

	user_id	business_id	date_review	rating_review	is_train
0	16727	73770	2012-09-14 08:38:47	5.0	True
1	3511	73770	2017-08-11 18:52:05	5.0	True
2	7049	73770	2012-02-04 21:13:37	2.0	True
3	19208	73770	2011-01-22 01:58:35	1.0	True
4	3801	73770	2015-05-08 20:55:25	2.0	True
...	...	...	...	...	...
59995	1903	37747	2016-07-05 21:54:34	4.0	False
59996	4129	12394	2016-12-11 01:29:08	4.0	False
59997	16956	35860	2012-08-07 15:35:03	4.0	False
59998	14735	13544	2014-04-11 22:46:39	5.0	False
59999	1057	39160	2018-10-05 23:43:16	1.0	False

318881 rows × 5 columns

Model

```
n_latent_factors_user = 8
n_latent_factors_business = 10
n_latent_factors_mf = 3
n_users, n_business = len(all_data.user_id.unique()), len(all_data.business_id.unique())


business_input = keras.layers.Input(shape=[1],name='Item')
business_embedding_mlp = keras.layers.Embedding(n_business + 1, n_latent_factors_business, na
business_vec_mlp = keras.layers.Flatten(name='FlattenBusiness-MLP')(business_embedding_mlp)
business_vec_mlp = keras.layers.Dropout(0.2)(business_vec_mlp)

business_embedding_mf = keras.layers.Embedding(n_business + 1, n_latent_factors_mf, name='bus
business_vec_mf = keras.layers.Flatten(name='Flattenbusiness-MF')(business_embedding_mf)
business_vec_mf = keras.layers.Dropout(0.2)(business_vec_mf)

user_input = keras.layers.Input(shape=[1],name='User')
user_vec_mlp = keras.layers.Flatten(name='FlattenUsers-MLP')(keras.layers.Embedding(n_users +
user_vec_mlp = keras.layers.Dropout(0.2)(user_vec_mlp)

user_vec_mf = keras.layers.Flatten(name='FlattenUsers-MF')(keras.layers.Embedding(n_users + 1
user_vec_mf = keras.layers.Dropout(0.2)(user_vec_mf)


concat = keras.layers.concatenate([business_vec_mlp, user_vec_mlp], name='Concat')

concat_dropout = keras.layers.Dropout(0.2)(concat)
```

```
dense = keras.layers.Dense(128, name='FullyConnected')(concat_dropout)
dense_act = keras.layers.advanced_activations.LeakyReLU(alpha=0.3)(dense)
dense_batch = keras.layers.BatchNormalization(name='Batch')(dense_act)
dropout_1 = keras.layers.Dropout(0.2,name='Dropout-1')(dense_batch)
dense_2 = keras.layers.Dense(64,name='FullyConnected-1')(dropout_1)
dense_batch_2 = keras.layers.BatchNormalization(name='Batch-2')(dense_2)

dropout_2 = keras.layers.Dropout(0.2,name='Dropout-2')(dense_batch_2)
dense_3 = keras.layers.Dense(50,name='FullyConnected-2')(dropout_2)
dense_4 = keras.layers.Dense(20,name='FullyConnected-3')(dense_3)
dense_4_act = keras.layers.advanced_activations.LeakyReLU(alpha=0.3)(dense_4)

pred_mf = keras.layers.concatenate([business_vec_mf, user_vec_mf], name='Dot')
pred_mlp = keras.layers.Dense(1, activation='relu',name='Activation')(dense_4_act)

combine_mlp_mf = keras.layers.concatenate([pred_mf, pred_mlp],name='Concat-MF-MLP')
result_combine = keras.layers.Dense(100, name='Combine-MF-MLP')(combine_mlp_mf)
deep_combine = keras.layers.Dense(16, kernel_regularizer=regularizers.l2(0.001), name='FullyC
dropout_3 = keras.layers.Dropout(0.3,name='Dropout-3')(deep_combine)

result = keras.layers.Dense(1, name='Prediction')(deep_combine)


model = keras.Model([user_input, business_input], result)
model.compile(optimizer='adam',loss= 'mse', metrics =["accuracy", "mse"])


history = model.fit([train_data.user_id.values, train_data.business_id.values], train_data.ra
```

```
A:\Anaconda\lib\site-packages\tensorflow_core\python\framework\indexed_slices.py:424: Us
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
Train on 207104 samples, validate on 51777 samples
Epoch 1/20
207104/207104 [==============================] - 192s 928us/step - loss: 1.5995 - accura
Epoch 2/20
207104/207104 [==============================] - 188s 908us/step - loss: 1.2680 - accura
Epoch 3/20
207104/207104 [==============================] - 187s 903us/step - loss: 1.1750 - accura
Epoch 4/20
207104/207104 [==============================] - 172s 830us/step - loss: 1.1279 - accura
Epoch 5/20
207104/207104 [==============================] - 169s 814us/step - loss: 1.0985 - accura
Epoch 6/20
207104/207104 [==============================] - 128s 619us/step - loss: 1.0781 - accura
Epoch 7/20
207104/207104 [==============================] - 94s 452us/step - loss: 1.0594 - accurac
Epoch 8/20
207104/207104 [==============================] - 88s 427us/step - loss: 1.0483 - accurac
Epoch 9/20
207104/207104 [==============================] - 88s 425us/step - loss: 1.0379 - accurac
Epoch 10/20
207104/207104 [==============================] - 88s 425us/step - loss: 1.0286 - accurac
Epoch 11/20
207104/207104 [==============================] - 101s 487us/step - loss: 1.0217 - accura
Epoch 12/20
207104/207104 [==============================] - 103s 498us/step - loss: 1.0176 - accura
Epoch 13/20
207104/207104 [==============================] - 103s 498us/step - loss: 1.0092 - accura
Epoch 14/20
207104/207104 [==============================] - 105s 509us/step - loss: 1.0041 - accura
Epoch 15/20
207104/207104 [==============================] - 106s 512us/step - loss: 1.0003 - accura
Epoch 16/20
207104/207104 [==============================] - 102s 493us/step - loss: 0.9971 - accura
Epoch 17/20
207104/207104 [==============================] - 106s 514us/step - loss: 0.9947 - accura
Epoch 18/20
207104/207104 [==============================] - 113s 548us/step - loss: 0.9908 - accura
Epoch 19/20
207104/207104 [==============================] - 103s 498us/step - loss: 0.9870 - accura
Epoch 20/20
207104/207104 [==============================] - 104s 504us/step - loss: 0.9870 - accura
```

Evaluating Model

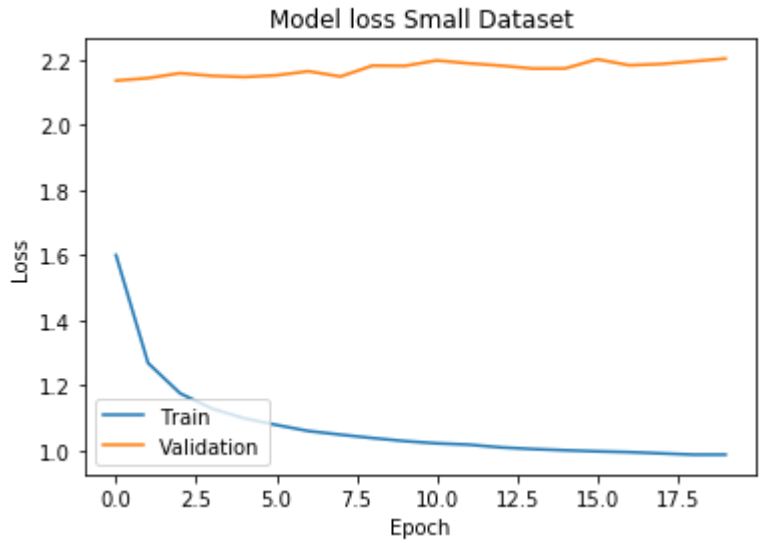
All Review Results

```
from sklearn.metrics import mean_squared_error
prediction = model.predict([test_data.user_id.values, test_data.business_id.values])
```

```
y_hat_rounded = np.round(prediction, 0)
print(np.sqrt(mean_squared_error(test_data.rating_review, y_hat_rounded)))
print(np.sqrt(mean_squared_error(test_data.rating_review, prediction)))
```

```
1.4701813947038418
1.4386594227617993
```

```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss Small Dataset')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='lower left')
plt.show()
```




```
prediction
```

```
array([[3.7071753],
       [3.9028566],
       [4.347692 ],
       ...,
       [3.4718204],
       [3.3162842],
       [3.4221017]], dtype=float32)
```

```
# pd.DataFrame(prediction).to_csv('gs://zw2624-bucket/output/leaky_relu_prediction.csv')
pd.DataFrame(prediction).to_csv('small_dataset_leaky_relu_prediction_copy.csv')
```

```
test_data['predicted'] = pd.DataFrame(prediction)
```

```
A:\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_g
  """Entry point for launching an IPython kernel.
```

User Coverage

```
test_data['predicted'] = pd.DataFrame(prediction)
correct = 0
for user in test_data.user_id:
    real = np.argsort(test_data.loc[test_data['user_id'] == user]['rating_review'])
    pred = np.argsort(test_data.loc[test_data['user_id'] == user]['predicted'])
    correct += (list(real) == list(pred))
```

```
A:\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_g
  """Entry point for launching an IPython kernel.
```

```
correct/3 / len(test_data.user_id.unique())
```

```
0.2037
```

Last Review Results

```
idx = test_data.groupby(['user_id'])['date_review'].transform(max) == test_data['date_review'
test_data_latest = test_data[idx]
print(np.sqrt(mean_squared_error(test_data_latest.rating_review, test_data_latest.predicted))
```

```
1.4710930978825
```