Implicit Feedback

```
In [1]:  import numpy as np
         import pandas as pd
```

```
In [2]:  ratings = pd.read_csv("/tf/notebooks/data/data/ratings.csv")
         users = pd.read_csv("/tf/notebooks/data/data/users.csv")
         items = pd.read_csv("/tf/notebooks/data/data/items.csv")
```

```
In [3]:  from recoflow.utils import NegativeSamples
```

```
In [24]: NegativeSamples??
```

```
Signature: NegativeSamples(df, rating_threshold, ratio_neg_per_user=1)
Source:
def NegativeSamples(df, rating_threshold, ratio_neg_per_user=1):
    """ function to sample negative feedback from user-item interaction dataset.

    This negative sampling function will take the user-item interaction data to create
    binarized feedback, i.e., 1 and 0 indicate positive and negative feedback,
    respectively.

    Args:
        df (pandas.DataFrame): input data that contains user-item tuples.
        rating_threshold (int): value below which feedback is set to 0 and above which feedback is set to 1
        ratio_neg_per_user (int): ratio of negative feedback w.r.t to the number of positive feedback for ea
ch user.

    Returns:
        pandas.DataFrame: data with negative feedback
    """

    df.columns = ["USER", "ITEM", "RATING", "unix_timestamp"]

    seed = 42

    df_pos = df.copy()
    df_pos["RATING"] = df_pos["RATING"].apply(lambda x: 1 if x >= rating_threshold else 0)
    df_pos = df_pos[df_pos.RATING>0]


    # Create a dataframe for all user-item pairs
    df_neg = _UserItemCrossJoin(df)

    #remove positive samples from the cross-join dataframe
    df_neg = _FilterBy(df_neg, df_pos, ["USER", "ITEM"])

    #Add a column for rating - setting it to 0
    df_neg["RATING"] = 0

    # Combine positive and negative samples into a single dataframe
    df_all = pd.concat([df_pos, df_neg], ignore_index=True, sort=True)
    df_all = df_all[["USER", "ITEM", "RATING"]]


    # Sample negative feedback from the combined dataframe.
    df_sample = (
        df_all.groupby("USER")
        .apply(
            lambda x: pd.concat(
```

```
                    [
                        x[x["RATING"] == 1],
                        x[x["RATING"] == 0].sample(
                            min(
                                max(
                                    round(len(x[x["RATING"] == 1]) * ratio_neg_per_user), 1
                                ),
                                len(x[x["RATING"] == 0]),
                            ),
                            random_state=seed,
                            replace=False,
                        )
                        if len(x[x["RATING"] == 0] > 0)
                        else pd.DataFrame({}, columns=["USER", "ITEM", "RATING"]),
                    ],
                    ignore_index=True,
                    sort=True,
                )
            )
            .reset_index(drop=True)
            .sort_values("USER")
        )

        df_sample.columns = ["movie_id", "rating", "user_id"]
        return df_sample[["user_id", "movie_id", "rating"]]
File:       /usr/local/lib/python3.6/dist-packages/recoflow/utils.py
Type:       function
```

In [4]: `df_implicit = NegativeSamples(ratings, rating_threshold=3, ratio_neg_per_user=1)`

In [5]: `ratings.shape, df_implicit.shape`

Out[5]: `((100000, 4), (165040, 3))`

In [6]: `df_implicit.head()`

Out[6]:

	user_id	movie_id	rating
0	1	61	1
298	1	866	0
297	1	510	0
296	1	102	0
295	1	886	0

```
In [7]:   df_implicit["unix_timestamp"] = 1
```

```
In [8]:   from recoflow. preprocessing import EncodeUserItem
```

```
In [10]:  # Data encoding

          interaction, n_users, n_items, user_encoder, item_encoder = EncodeUserItem(df_implicit, "user_id", "movie_id",
          "rating", "unix_timestamp")
```

```
          Number of users:  943
          Number of items:  1682
```

```
In [21]:  from recoflow.preprocessing import RandomSplit
```

```
In [22]:  train, test = RandomSplit(interaction, [0.8, 0.2])
```

```
In [11]:  max_rating = interaction.RATING.max()
          min_rating = interaction.RATING.min()
          min_rating, max_rating
```

```
Out[11]:  (0, 1)
```

Implicit Matrix Factorization

```
In [13]:  from keras.models import Model
          from keras.layers import Input, Embedding, Flatten, Dot, Add, Lambda, Activation, Reshape
          from keras.regularizers import l2
          from keras.utils import plot_model
```

```
In [14]:  from recoflow.models import ExplicitMatrixFactorisationBias
```

In [15]: ExplicitMatrixFactorisationBias??

```
Signature:
ExplicitMatrixFactorisationBias(
    n_users,
    n_items,
    n_factors,
    min_rating,
    max_rating,
)
Docstring: <no docstring>
Source:
def ExplicitMatrixFactorisationBias(n_users, n_items, n_factors, min_rating, max_rating):

    # Item Layer
    item_input = Input(shape=[1], name='Item')
    item_embedding = Embedding(n_items, n_factors, embeddings_regularizer=l2(1e-6), name='ItemEmbedding')(item_input)
    item_vec = Flatten(name='FlattenItemE')(item_embedding)

    # Item Bias
    item_bias = Embedding(n_items, 1, embeddings_regularizer=l2(1e-6), name='ItemBias')(item_input)
    item_bias_vec = Flatten(name='FlattenItemBiasE')(item_bias)

    # User Layer
    user_input = Input(shape=[1], name='User')
    user_embedding = Embedding(n_users, n_factors, embeddings_regularizer=l2(1e-6), name='UserEmbedding')(user_input)
    user_vec = Flatten(name='FlattenUserE')(user_embedding)

    # User Bias
    user_bias = Embedding(n_users, 1, embeddings_regularizer=l2(1e-6), name='UserBias')(user_input)
    user_bias_vec = Flatten(name='FlattenUserBiasE')(user_bias)

    # Dot Product of Item and User & then Add Bias
    DotProduct = Dot(axes=1, name='DotProduct')([item_vec, user_vec])
    AddBias = Add(name="AddBias")([DotProduct, item_bias_vec, user_bias_vec])

    # Scaling for each user
    y = Activation('sigmoid')(AddBias)
    rating_output = Lambda(lambda x: x * (max_rating - min_rating) + min_rating)(y)

    # Model Creation
    model = Model([user_input, item_input], rating_output, name="ExplicitMatrixFactorisationBias")

    # Compile Model
    model.compile(loss='mean_squared_error', optimizer=Adam(lr=0.001))

    return model
```

```
File:        /usr/local/lib/python3.6/dist-packages/recoflow/models.py
Type:        function
```

In [16]:
```python
def ImplicitMatrixFactorisationBias(n_users, n_items, n_factors, min_rating, max_rating):

    # Item Layer
    item_input = Input(shape=[1], name='Item')
    item_embedding = Embedding(n_items, n_factors, embeddings_regularizer=l2(1e-6), name='ItemEmbedding')(item_input)
    item_vec = Flatten(name='FlattenItemE')(item_embedding)

    # Item Bias
    item_bias = Embedding(n_items, 1, embeddings_regularizer=l2(1e-6), name='ItemBias')(item_input)
    item_bias_vec = Flatten(name='FlattenItemBiasE')(item_bias)

    # User Layer
    user_input = Input(shape=[1], name='User')
    user_embedding = Embedding(n_users, n_factors, embeddings_regularizer=l2(1e-6), name='UserEmbedding')(user_input)
    user_vec = Flatten(name='FlattenUserE')(user_embedding)

    # User Bias
    user_bias = Embedding(n_users, 1, embeddings_regularizer=l2(1e-6), name='UserBias')(user_input)
    user_bias_vec = Flatten(name='FlattenUserBiasE')(user_bias)

    # Dot Product of Item and User & then Add Bias
    DotProduct = Dot(axes=1, name='DotProduct')([item_vec, user_vec])
    AddBias = Add(name="AddBias")([DotProduct, item_bias_vec, user_bias_vec])

    # Scaling for each user
    y = Activation('sigmoid')(AddBias)
    rating_output = Lambda(lambda x: x * (max_rating - min_rating) + min_rating)(y)

    # Model Creation
    model = Model([user_input, item_input], rating_output, name="ImplicitMatrixFactorisationBias")

    # Compile Model
    model.compile(loss='binary_crossentropy', optimizer="sgd")

    return model
```

In [18]:
```python
n_factors = 40
model = ImplicitMatrixFactorisationBias(n_users, n_items, n_factors, min_rating, max_rating)
```
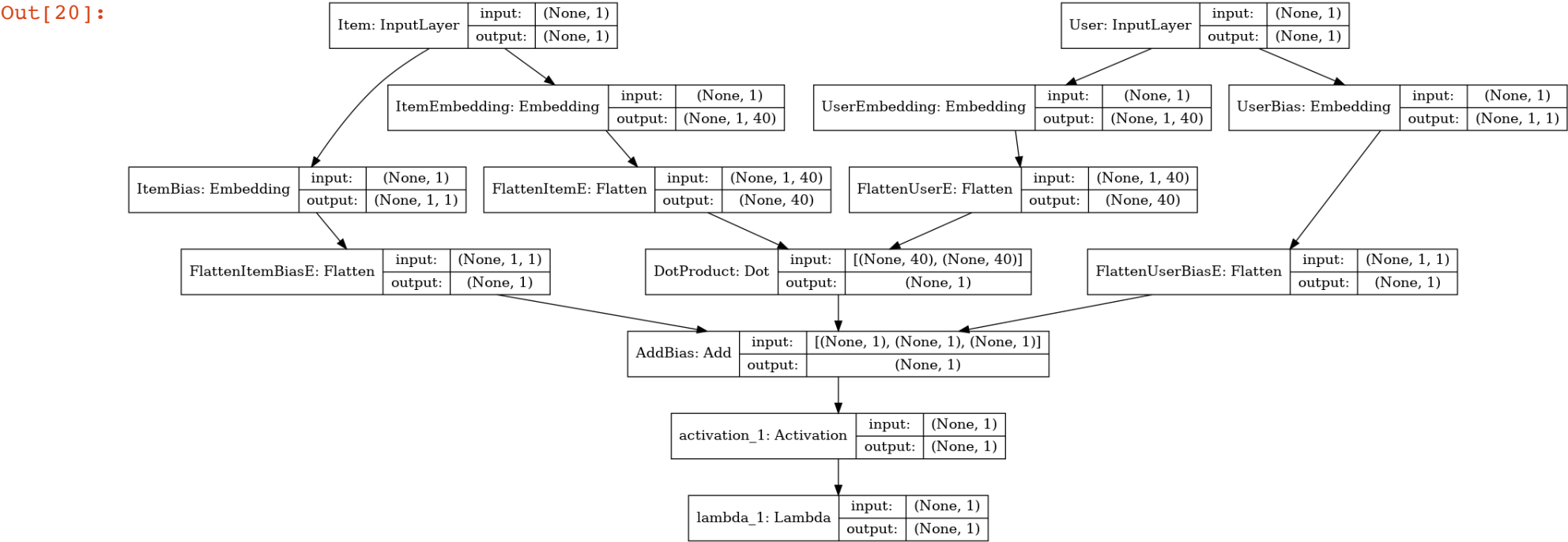
```
In [19]: model.summary()
```

Model: "ImplicitMatrixFactorisationBias"

Layer (type)	Output Shape	Param #	Connected to
Item (InputLayer)	(None, 1)	0	
User (InputLayer)	(None, 1)	0	
ItemEmbedding (Embedding)	(None, 1, 40)	67280	Item[0][0]
UserEmbedding (Embedding)	(None, 1, 40)	37720	User[0][0]
FlattenItemE (Flatten)	(None, 40)	0	ItemEmbedding[0][0]
FlattenUserE (Flatten)	(None, 40)	0	UserEmbedding[0][0]
ItemBias (Embedding)	(None, 1, 1)	1682	Item[0][0]
UserBias (Embedding)	(None, 1, 1)	943	User[0][0]
DotProduct (Dot)	(None, 1)	0	FlattenItemE[0][0] FlattenUserE[0][0]
FlattenItemBiasE (Flatten)	(None, 1)	0	ItemBias[0][0]
FlattenUserBiasE (Flatten)	(None, 1)	0	UserBias[0][0]
AddBias (Add)	(None, 1)	0	DotProduct[0][0] FlattenItemBiasE[0][0] FlattenUserBiasE[0][0]
activation_1 (Activation)	(None, 1)	0	AddBias[0][0]
lambda_1 (Lambda)	(None, 1)	0	activation_1[0][0]

Total params: 107,625
Trainable params: 107,625
Non-trainable params: 0

`plot_model(model, show_layer_names=True, show_shapes=True)`

Item: InputLayer	input:	(None, 1)
	output:	(None, 1)

User: InputLayer	input:	(None, 1)
	output:	(None, 1)

ItemEmbedding: Embedding	input:	(None, 1)
	output:	(None, 1, 40)

UserEmbedding: Embedding	input:	(None, 1)
	output:	(None, 1, 40)

UserBias: Embedding	input:	(None, 1)
	output:	(None, 1, 1)

ItemBias: Embedding	input:	(None, 1)
	output:	(None, 1, 1)

FlattenItemE: Flatten	input:	(None, 1, 40)
	output:	(None, 40)

FlattenUserE: Flatten	input:	(None, 1, 40)
	output:	(None, 40)

FlattenItemBiasE: Flatten	input:	(None, 1, 1)
	output:	(None, 1)

DotProduct: Dot	input:	[(None, 40), (None, 40)]
	output:	(None, 1)

FlattenUserBiasE: Flatten	input:	(None, 1, 1)
	output:	(None, 1)

AddBias: Add	input:	[(None, 1), (None, 1), (None, 1)]
	output:	(None, 1)

activation_1: Activation	input:	(None, 1)
	output:	(None, 1)

lambda_1: Lambda	input:	(None, 1)
	output:	(None, 1)

Train the model

```
In [23]:  %%time
          output = model.fit([train.USER, train.ITEM], train.RATING,
                             batch_size=128, epochs=5, verbose=1,
                             validation_data=([test.USER, test.ITEM], test.RATING))
```

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_grad.py:1250: add_
dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed i
n a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:422: The
name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 132032 samples, validate on 33008 samples
Epoch 1/5
132032/132032 [==============================] - 2s 14us/step - loss: 0.6933 - val_loss: 0.6929
Epoch 2/5
132032/132032 [==============================] - 2s 13us/step - loss: 0.6922 - val_loss: 0.6918
Epoch 3/5
132032/132032 [==============================] - 2s 12us/step - loss: 0.6912 - val_loss: 0.6908
Epoch 4/5
132032/132032 [==============================] - 2s 12us/step - loss: 0.6901 - val_loss: 0.6898
Epoch 5/5
132032/132032 [==============================] - 2s 12us/step - loss: 0.6891 - val_loss: 0.6887
CPU times: user 16.6 s, sys: 1.31 s, total: 17.9 s
Wall time: 8.68 s

Evaluate the Model

```
In [25]:  from recoflow.recommend import GetPredictions
```

```
In [26]:  %%time
          predictions = GetPredictions(model, interaction)
```

CPU times: user 23.8 s, sys: 3.17 s, total: 26.9 s
Wall time: 17.1 s

```
In [27]:  predictions.head()
```

Out[27]:

	USER	ITEM	RATING
0	0	60	0.494863
1	0	865	0.497352
2	0	509	0.502310
3	0	101	0.500138
4	0	885	0.486868

```
In [28]:  from recoflow.metrics import RankingMetrics
```

```
In [29]:  RankingMetrics(test, predictions, k=10)
```

Out[29]:

	k	Precision@k	Recall@k	MAP@k	MRR@k	NDCG@k
0	10	0.3722	0.03	0.0167	0.2472	0.3836

```
In [30]:  from recoflow.datasets import SampleEvaluate
```

```
In [31]:  rating_test, rating_pred = SampleEvaluate()
```

```
In [47]:  rating_test[rating_test.USER == 2]
```

Out[47]:

	USER	ITEM	RATING
3	2	1	5
4	2	4	5
5	2	5	3
6	2	6	3
7	2	7	1

```
In [48]: rating_pred[rating_pred.USER == 2]
```

Out[48]:

	USER	ITEM	RATING
3	2	10	14
4	2	3	13
5	2	11	12
6	2	5	11
7	2	13	10

```
In [36]: from recoflow.metrics import PrecisionK, RecallK
```

```
In [45]: PrecisionK(rating_test, rating_pred, k=3 )
```

Out[45]: 0.222222222222222

```
In [46]: RecallK(rating_test, rating_pred, k=3 )
```

Out[46]: 0.14444444444444446

```
In [ ]:
```