Matrix Factorization - Full Movielens data (100k)

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
import numpy as np
import pandas as pd
```

Prepare data

In [6]:
```
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 [7]:
```
from recoflow.preprocessing import EncodeUserItem, StratifiedSplit
```

In [10]:
```
# Encoding the data
interaction, n_users, n_items, user_encoder, item_encoder = EncodeUserItem(ratings,
                                                                           "user_id",
                                                                           "movie_id",
                                                                           "rating",
                                                                           "unix_timestamp")
```

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

In [11]:
```
train, test = StratifiedSplit(interaction, [0.8, 0.2])
```

In [12]:
```
train.shape, test.shape
```

Out[12]:
```
((80000, 7), (20000, 7))
```

Build the Model

In [14]:

```
from keras.models import Model
from keras.layers import Embedding, Dot, Input, Flatten
from keras.regularizers import l2
```

In [15]:

```
def ExplicitMF(n_users, n_items, n_factors):

    # 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)

    # 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)

    # Dot Product of Item and User
    rating = Dot(axes=1, name="DotProduct")([item_vec, user_vec])

    # Create the Model
    model = Model([user_input, item_input], rating, name="ExplicitMF")

    # Compile the Model
    model.compile(loss="mean_squared_error", optimizer="adam")

    return model
```

In [17]:

```
n_factors = 40
model = ExplicitMF(n_users, n_items, n_factors)
```

In [18]:

```
model.summary()
```

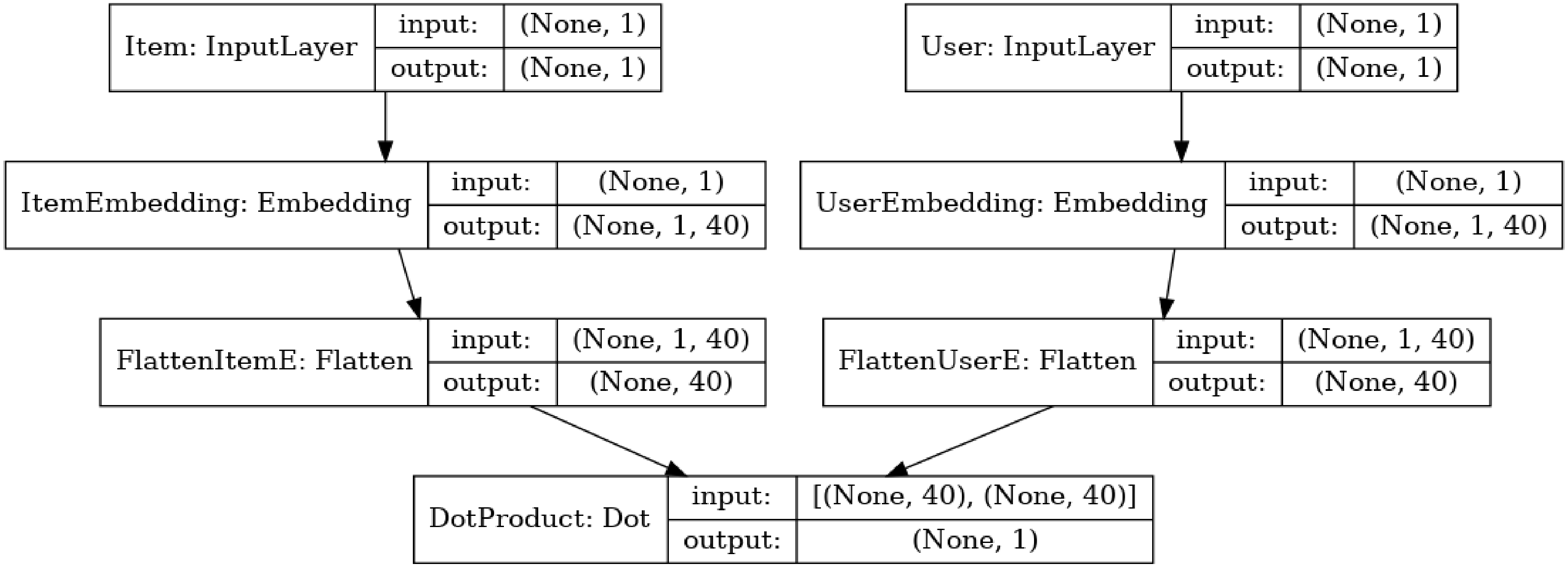
```
Model: "ExplicitMF"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Item (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]
__________________________________________________________________________________________________
DotProduct (Dot)                (None, 1)            0           FlattenItemE[0][0]
                                                                 FlattenUserE[0][0]
==================================================================================================
Total params: 105,000
Trainable params: 105,000
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [19]:

```
from keras.utils import plot_model
```

```
In [20]: plot_model(model, show_layer_names=True, show_shapes=True)
```

Out[20]:



```
In [22]: %%time
         output = model.fit([train.USER, train.ITEM], train.RATING, shuffle=True,
                            batch_size=32, epochs=5, verbose=1,
                           validation_data=([test.USER, test.ITEM], test.RATING))
```

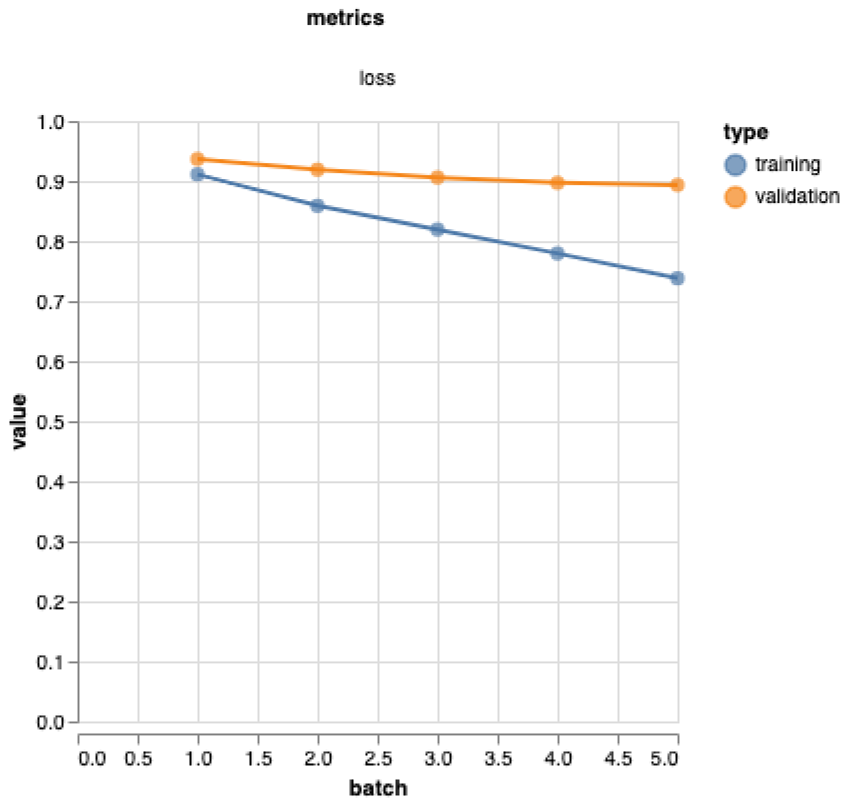
```
Train on 80000 samples, validate on 20000 samples
Epoch 1/5
80000/80000 [==============================] - 4s 47us/step - loss: 0.9112 - val_loss: 0.9367
Epoch 2/5
80000/80000 [==============================] - 4s 46us/step - loss: 0.8589 - val_loss: 0.9190
Epoch 3/5
80000/80000 [==============================] - 4s 45us/step - loss: 0.8194 - val_loss: 0.9061
Epoch 4/5
80000/80000 [==============================] - 4s 45us/step - loss: 0.7795 - val_loss: 0.8971
Epoch 5/5
80000/80000 [==============================] - 4s 45us/step - loss: 0.7381 - val_loss: 0.8937
CPU times: user 36.3 s, sys: 3.11 s, total: 39.4 s
Wall time: 18.3 s
```

```
In [23]: from recoflow.vis import MetricsVis
```

In [24]:

```
MetricsVis(output.history)
```

Out[24]:



In [26]:

```
score = model.evaluate([test.USER, test.ITEM], test.RATING, verbose=1)
score
```

```
20000/20000 [==============================] - 0s 12us/step
```

Out[26]:

```
0.8936678772449493
```

Get Predictions

In [28]:

```
from recoflow.recommend import GetPredictions
```

In [30]:

```
%time
predictions = GetPredictions(model, interaction)
```

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.96 µs
```

Explicit Ratings with Bias

In [31]:

```
from recoflow.models import ExplicitMatrixFactorisationBias
```

In [40]:

```
max_rating = interaction.RATING.max()
min_rating = interaction.RATING.min()
max_rating, min_rating
```

Out[40]:

```
(5, 1)
```

In [52]:

```
from keras.layers import Add, Activation, Lambda
from keras.optimizers import Adam
```

In [53]:

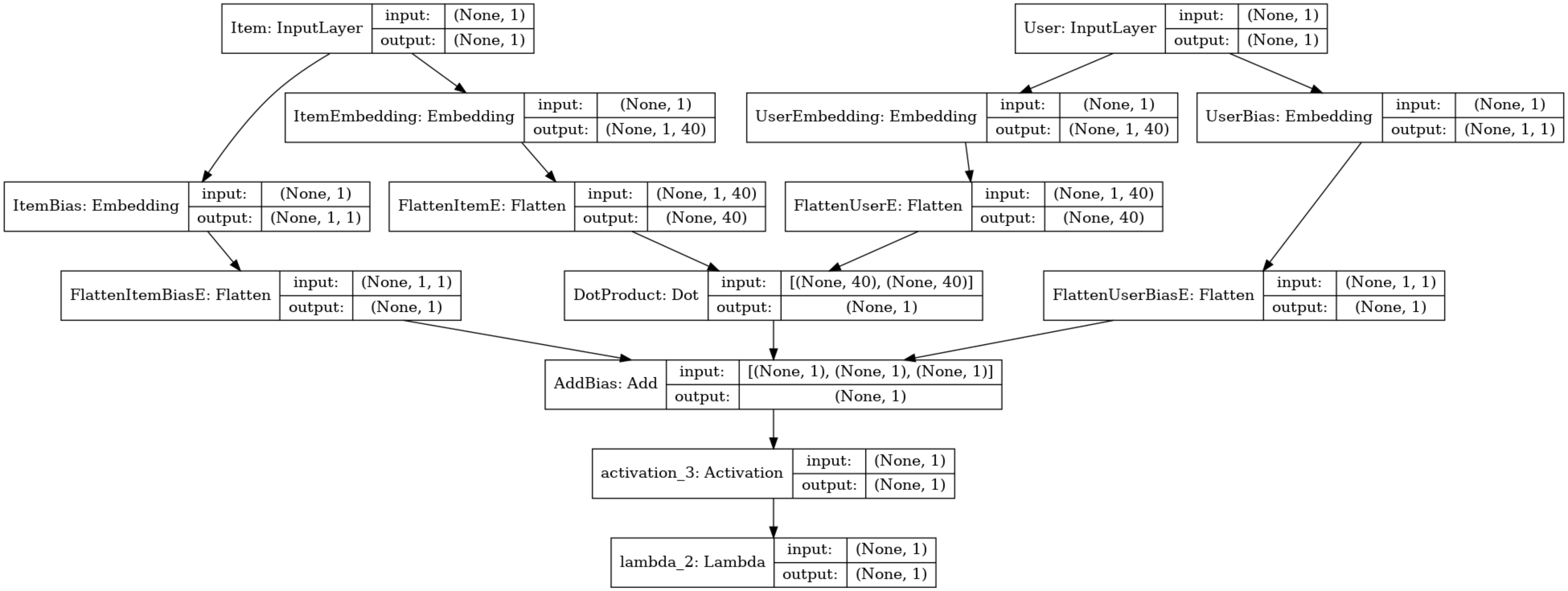
```
def ExplicitMatrixFactorisationBias(n_users, n_items, n_factors, max_rating, min_rating):

    # 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
```

In [54]:

```
n_factors = 40
model = ExplicitMatrixFactorisationBias(n_users, n_items, n_factors, max_rating, min_rating)
```

In [55]:
```
plot_model(model, show_layer_names=True, show_shapes=True)
```

Out[55]:


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_3: Activation | input: (None, 1) | output: (None, 1)
lambda_2: Lambda | input: (None, 1) | output: (None, 1)

In [56]:

```python
model.summary()
```

```
Model: "ExplicitMatrixFactorisationBias"
__________________________________________________________________________________________________
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_3 (Activation)       (None, 1)            0           AddBias[0][0]
__________________________________________________________________________________________________
lambda_2 (Lambda)               (None, 1)            0           activation_3[0][0]
==================================================================================================
Total params: 107,625
Trainable params: 107,625
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [57]:

```
%%time
output = model.fit([train.USER, train.ITEM], train.RATING, shuffle=True,
                   batch_size=32, epochs=5, verbose=1,
                  validation_data=([test.USER, test.ITEM], test.RATING))
```

```
Train on 80000 samples, validate on 20000 samples
Epoch 1/5
80000/80000 [==============================] - 4s 53us/step - loss: 1.1839 - val_loss: 0.9472
Epoch 2/5
80000/80000 [==============================] - 4s 51us/step - loss: 0.8356 - val_loss: 0.8547
Epoch 3/5
80000/80000 [==============================] - 4s 51us/step - loss: 0.7134 - val_loss: 0.8305
Epoch 4/5
80000/80000 [==============================] - 4s 51us/step - loss: 0.6086 - val_loss: 0.8318
Epoch 5/5
80000/80000 [==============================] - 4s 52us/step - loss: 0.5123 - val_loss: 0.8487
CPU times: user 44.9 s, sys: 3.53 s, total: 48.4 s
Wall time: 21.2 s
```

In [58]:

```
score = model.evaluate([test.USER, test.ITEM], test.RATING, verbose=1)
score
```

```
20000/20000 [==============================] - 0s 14us/step
```

Out[58]:

```
0.8486886561393738
```

In [ ]:

```
```