Matrix Factorisation

Classical Matrix Factorisation model used for Collaborative Filtering - Popularised by Netflix Competition

In [2]:
```
import sys
sys.path.append("../")
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [4]:
```
%matplotlib inline
```

Step 1: Load & Prepare the Data

Dataset from https://grouplens.org/datasets/movielens/100k/ (https://grouplens.org/datasets/movielens/100k/)

In [5]:
```
df_ratings = pd.read_csv("/tf/notebooks/data/data/ratings.csv")
```

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

Out[6]:

	user_id	movie_id	rating	unix_timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

In [7]:
```
df_ratings.shape
```

Out[7]:
```
(100000, 4)
```

In [8]:
```
#Sparsity
df_ratings.shape[0]/ (df_ratings.user_id.nunique() * df_ratings.movie_id.nunique())
```

Out[8]:
```
0.06304669364224531
```

Data Transformation

- Encoding: Create User and Item Labels (Index) => Label Encoding
- Splitting: How do we split the data in train and test
- Ratings: Explicit or transform them into something else??

In [9]:
```
from reco.preprocess import encode_user_item
```

In [10]:
```
#encode_user_item??
```

In [11]:
```
DATA, user_encoder, item_encoder = encode_user_item(df_ratings, "user_id", "movie_id",
                                                     "rating", "unix_timestamp")
```

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

In [12]:
```
DATA.head()
```

Out[12]:

	user_id	movie_id	RATING	TIMESTAMP	USER	ITEM
0	196	242	3	881250949	195	241
1	186	302	3	891717742	185	301
2	22	377	1	878887116	21	376
3	244	51	2	880606923	243	50
4	166	346	1	886397596	165	345

Data Splitting Strategy

- Random
- Stratified: For each user, split it by train and test
- Chronological: For each user, split it by train and test in chronological order

In [13]:
```
from reco.preprocess import user_split, random_split
```

In [14]:
```
#user_split??
```

In [15]:
```
#train, val, test = user_split(DATA, [0.6, 0.2 ,0.2])
train, test = random_split(DATA, [0.8, 0.2])
```

Step 2: Build Model - Explicit Matrix Factorisation

In [16]:
```
from keras.models import Model
from keras.layers import Input, Embedding, Flatten, Add, Dot, Activation
from keras.regularizers import l2
from keras.utils import plot_model
```

```
Using TensorFlow backend.
```

In [17]:
```
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
    rating = Dot(axes=1, name="DotProduct")([item_vec, user_vec])

    # Model Creation
    model = Model([user_input, item_input], rating)

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

    return model
```

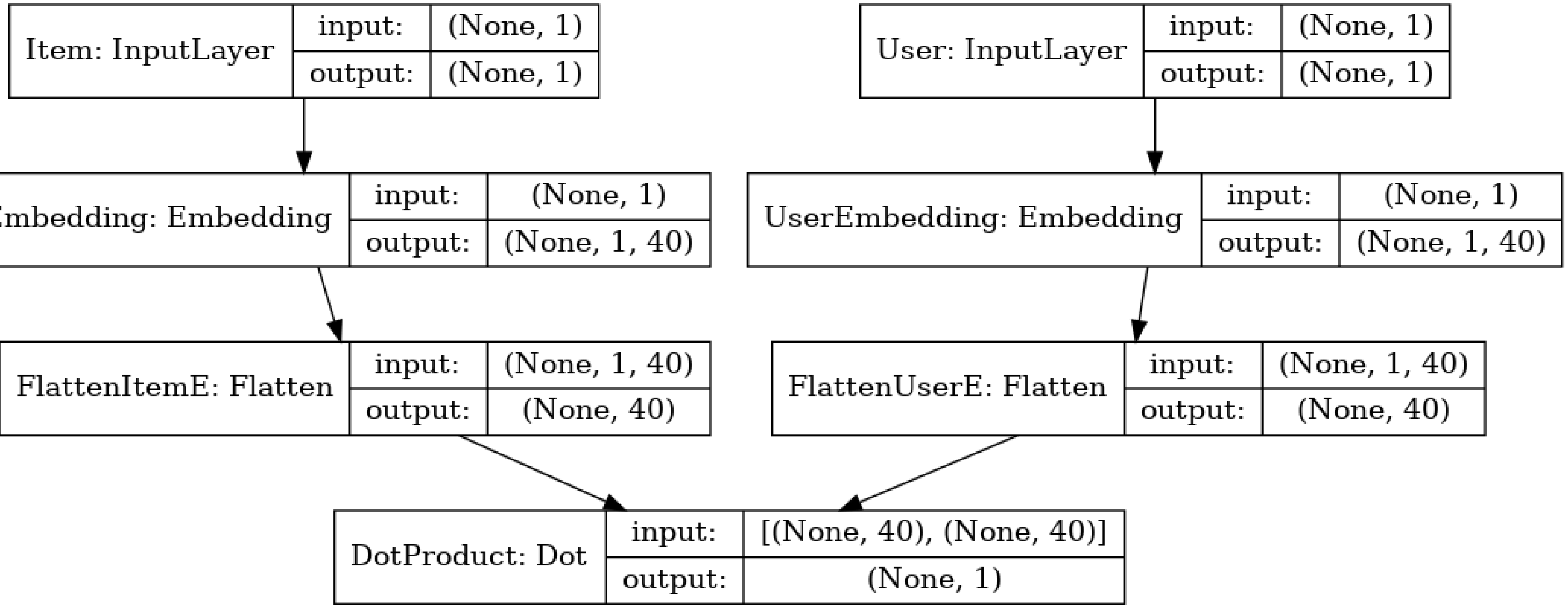
In [18]:
```
n_users = DATA.USER.nunique()
n_items = DATA.ITEM.nunique()
```

In [19]:
```
n_factors = 40
model = ExplicitMF(n_users, n_items, n_factors)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541: The
name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4432: The
name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66: The n
ame tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Op
timizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

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

Out[20]:



In [21]:
```
model.summary()
```

```
Model: "model_1"

________________________________________________________________________________________________
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 [22]:
```
n_users * 40, n_items * 40
```

Out[22]:
```
(37720, 67280)
```

Step: Train the Model

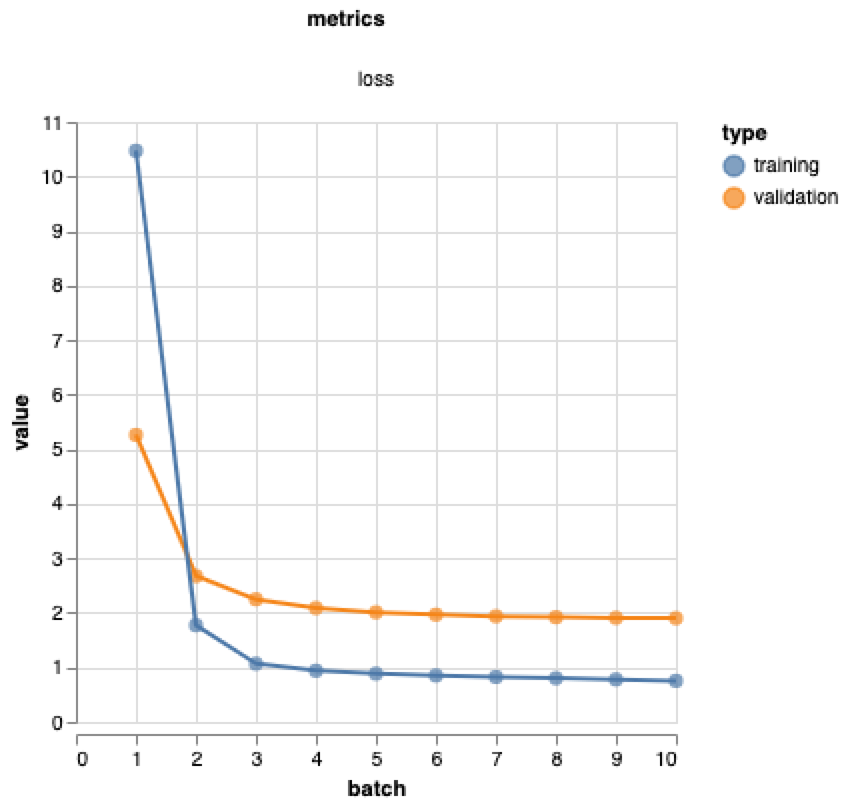
In [23]:
```
%%time
output = model.fit([train.USER, train.ITEM], train.RATING,
                   batch_size=64, epochs=10, verbose=1, validation_split=0.2)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

Train on 64000 samples, validate on 16000 samples
Epoch 1/10
64000/64000 [==============================] - 2s 27us/step - loss: 10.4725 - val_loss: 5.2628
Epoch 2/10
64000/64000 [==============================] - 1s 23us/step - loss: 1.7734 - val_loss: 2.6748
Epoch 3/10
64000/64000 [==============================] - 1s 23us/step - loss: 1.0691 - val_loss: 2.2491
Epoch 4/10
64000/64000 [==============================] - 1s 23us/step - loss: 0.9400 - val_loss: 2.0893
Epoch 5/10
64000/64000 [==============================] - 1s 23us/step - loss: 0.8859 - val_loss: 2.0105
Epoch 6/10
64000/64000 [==============================] - 1s 23us/step - loss: 0.8522 - val_loss: 1.9679
Epoch 7/10
64000/64000 [==============================] - 1s 23us/step - loss: 0.8262 - val_loss: 1.9385
Epoch 8/10
64000/64000 [==============================] - 1s 23us/step - loss: 0.8024 - val_loss: 1.9208
Epoch 9/10
64000/64000 [==============================] - 1s 23us/step - loss: 0.7797 - val_loss: 1.9095
Epoch 10/10
64000/64000 [==============================] - 1s 23us/step - loss: 0.7548 - val_loss: 1.9026
CPU times: user 30 s, sys: 2.2 s, total: 32.2 s
Wall time: 15.1 s
```

In [24]:
```
from reco.vis import metrics
```

In [25]:
```
metrics(output.history)
```

Out[25]:



Getting Simple Recommendation

In [26]:
```
from reco.evaluate import get_embedding
```

In [27]:
```
item_embedding = model.get_layer("ItemEmbedding").get_weights()[0]
```

In [28]:
```
item_embedding.shape
```

Out[28]:
```
(1682, 40)
```

Get Similiar Items

In [29]:
```
from reco.recommend import get_similar, show_similar
```

In [30]:
```python
from sklearn.neighbors import NearestNeighbors
```

In [31]:
```python
def get_similar(embedding, k):
    model_similar_items = NearestNeighbors(n_neighbors=k, algorithm="ball_tree").fit(embedding)
    distances, indices = model_similar_items.kneighbors(embedding)

    return distances, indices
```

In [32]:
```python
item_distance, item_similar_indices = get_similar(item_embedding, 5)
```

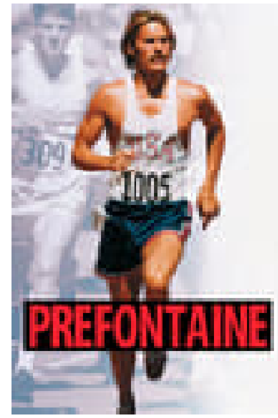
In [33]:
```python
item_similar_indices
```

Out[33]:
```
array([[   0, 1482,  499, 1188, 1472],
       [   1,  722, 1034,  619, 1031],
       [   2, 1237,  722,  591, 1439],
       ...,
       [1679, 1680, 1639, 1543, 1630],
       [1680, 1666, 1679, 1678, 1669],
       [1681, 1626, 1674, 1639, 1603]])
```

In [34]:
```python
import matplotlib.image as mpimage
```

In [35]:
```python
def show_similar(item_index, item_similar_indices, item_encoder):

    s = item_similar_indices[item_index]
    movie_ids = item_encoder.inverse_transform(s)

    images = []
    for movie_id in movie_ids:
        img_path = '/tf/notebooks/data/data/posters/' + str(movie_id) + '.jpg'
        images.append(mpimage.imread(img_path))

    plt.figure(figsize=(20,10))
    columns = 5
    for i, image in enumerate(images):
        plt.subplot(len(images) / columns + 1, columns, i + 1)
        plt.axis('off')
        plt.imshow(image)
```

In [36]:
```
show_similar(0, item_similar_indices, item_encoder)
```



For Non-Negative Matrix Factorisation

Embedding: add Non negative constraints to the embedding layer

Explicit MF with bias -> FastAI Model

- Embedding Dot Product with Bias
- Sigmoid Layer adjustment

In [37]:
```
from keras.layers import Lambda
```

In [38]:
```
max_rating = DATA.RATING.max()
min_rating = DATA.RATING.min()
max_rating, min_rating
```

Out[38]:
```
(5, 1)
```

In [39]:
```
def ExplicitMF_bias (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)

    # 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)

    # 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)

    # Dot Product
    DotProduct = Dot(axes=1, name="DotProduct")([item_vec, user_vec])
    # Add Bias
    AddBias = Add(name="AddBias")([DotProduct, user_bias_vec, item_bias_vec])

    # Scaling trick
    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)

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

    return model
```

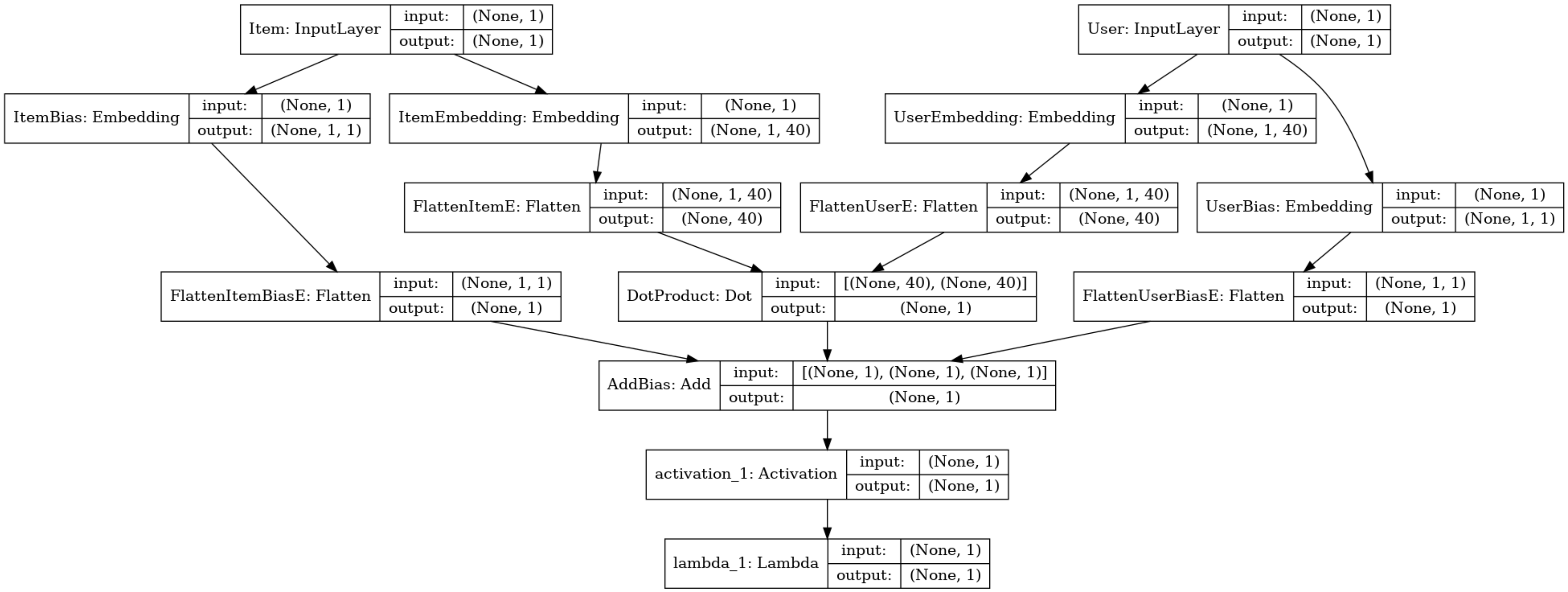
In [40]:
```
n_factors = 40
model_bias = ExplicitMF_bias(n_users, n_items, n_factors)
```

In [41]:
```
model_bias.summary()
```

```
Model: "model_2"
__________________________________________________________________________________________________
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]              
__________________________________________________________________________________________________
UserBias (Embedding)            (None, 1, 1)         943         User[0][0]                       
__________________________________________________________________________________________________
ItemBias (Embedding)            (None, 1, 1)         1682        Item[0][0]                       
__________________________________________________________________________________________________
DotProduct (Dot)                (None, 1)            0           FlattenItemE[0][0]               
                                                                 FlattenUserE[0][0]               
__________________________________________________________________________________________________
FlattenUserBiasE (Flatten)      (None, 1)            0           UserBias[0][0]                   
__________________________________________________________________________________________________
FlattenItemBiasE (Flatten)      (None, 1)            0           ItemBias[0][0]                   
__________________________________________________________________________________________________
AddBias (Add)                   (None, 1)            0           DotProduct[0][0]                 
                                                                 FlattenUserBiasE[0][0]           
                                                                 FlattenItemBiasE[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
__________________________________________________________________________________________________
```

In [42]:
```
plot_model(model_bias, show_layer_names=True, show_shapes=True)
```

Out[42]:

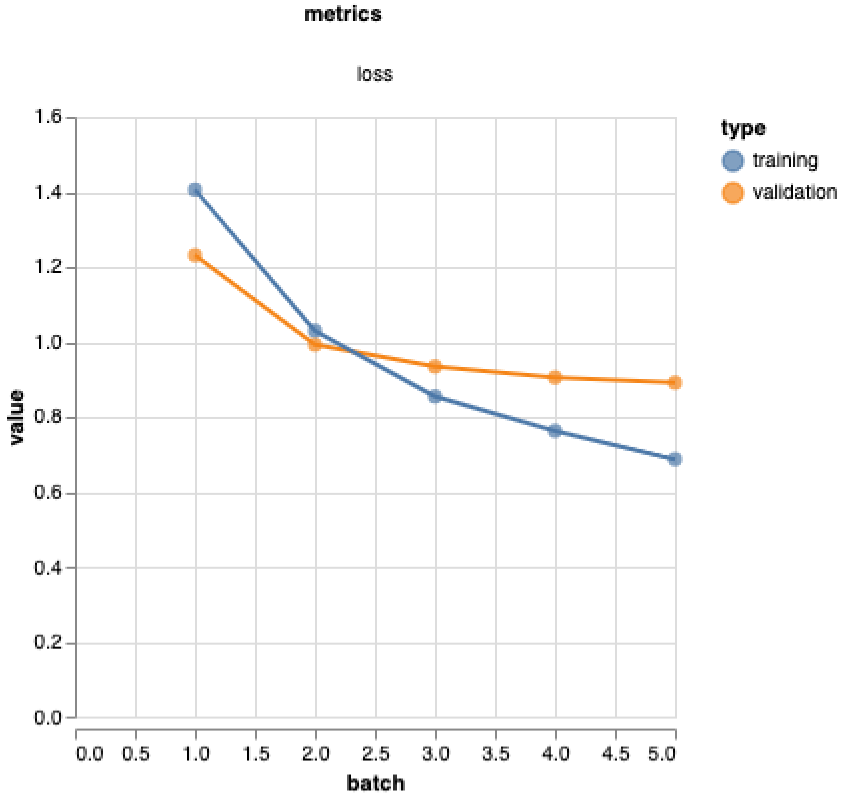


In [43]:
```
%%time
output_bias = model_bias.fit([train.USER, train.ITEM], train.RATING,
                             batch_size=128, epochs=5, verbose=1, validation_split=0.2)
```

```
Train on 64000 samples, validate on 16000 samples
Epoch 1/5
64000/64000 [==============================] - 1s 18us/step - loss: 1.4060 - val_loss: 1.2313
Epoch 2/5
64000/64000 [==============================] - 1s 13us/step - loss: 1.0302 - val_loss: 0.9930
Epoch 3/5
64000/64000 [==============================] - 1s 13us/step - loss: 0.8549 - val_loss: 0.9347
Epoch 4/5
64000/64000 [==============================] - 1s 13us/step - loss: 0.7629 - val_loss: 0.9055
Epoch 5/5
64000/64000 [==============================] - 1s 14us/step - loss: 0.6872 - val_loss: 0.8917
CPU times: user 9.86 s, sys: 742 ms, total: 10.6 s
Wall time: 4.87 s
```

In [44]:
```
metrics(output_bias.history)
```

Out[44]:



Concat Explicit Bias

In [48]:
```
from keras.layers import Concatenate, Dense, Dropout
```

```python
In [52]: def ExplicitMF_bias_concat (n_users, n_items, n_factors):

    # Item Layer
    item_input = Input(shape=[1], name="Item")
    item_embedding = Embedding(n_items, n_factors,
                               embeddings_initializer="he_normal",
                               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),
                               embeddings_initializer="he_normal",
                               name="UserEmbedding")(user_input)
    user_vec = Flatten(name="FlattenUserE")(user_embedding)

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

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

    # Concatenate
    concat = Concatenate(name="Concat")([item_vec, user_vec])
    concatD = Dropout(0.5)(concat)

    # Use Dense
    dense_1 = Dense(32, kernel_initializer="he_normal")(concatD)
    dense_1_drop = Dropout(0.5)(dense_1)
    dense_2 = Dense(1, kernel_initializer="he_normal")(dense_1_drop)

    # Dot Product
    #DotProduct = Dot(axes=1, name="DotProduct")([item_vec, user_vec])
    # Add Bias
    AddBias = Add(name="AddBias")([dense_2, user_bias_vec, item_bias_vec])

    # Scaling trick
```

```
    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)

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

    return model
```

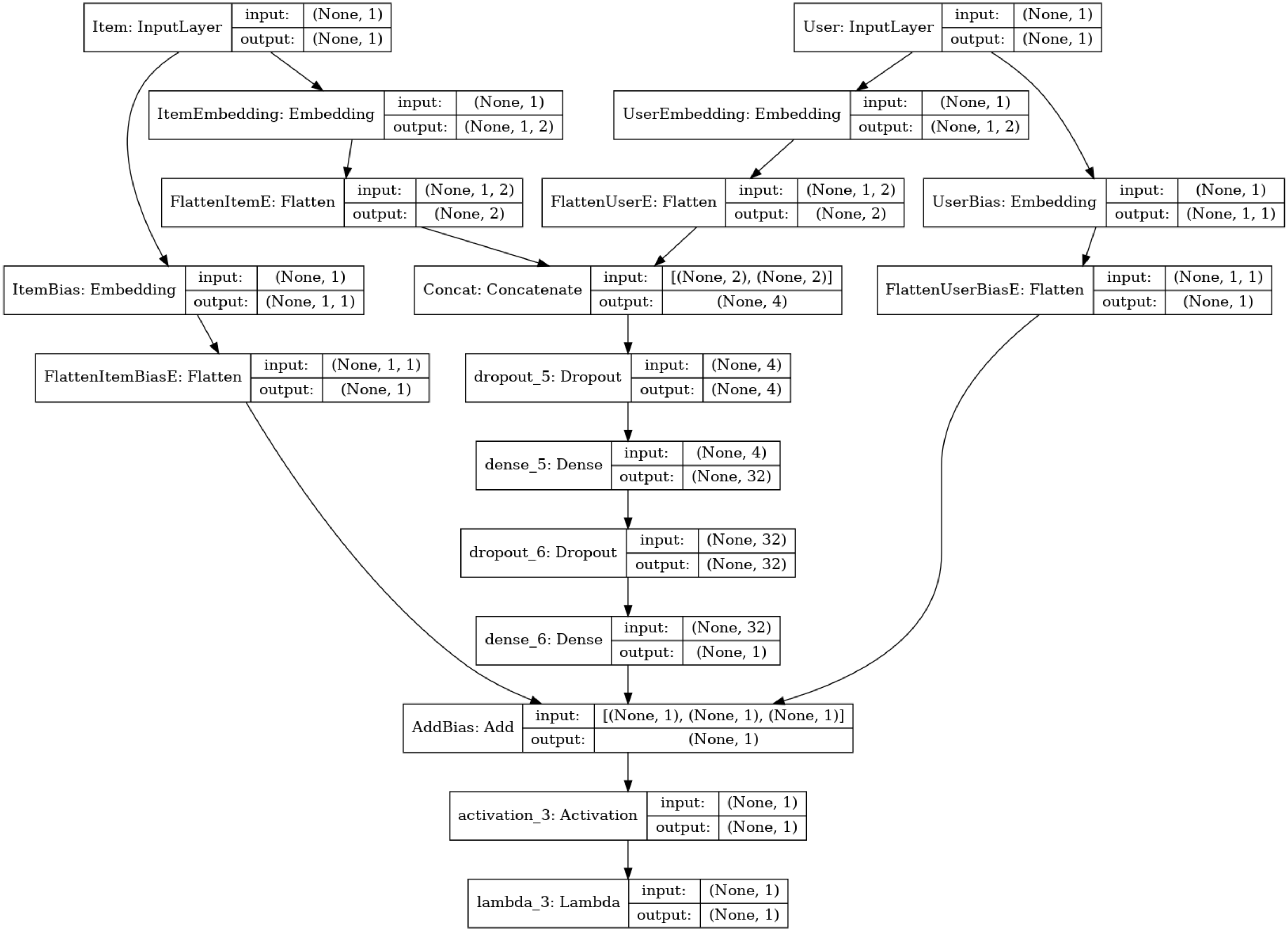
In [64]:
```
n_factors = 2
model_concat = ExplicitMF_bias_concat(n_users, n_items, n_factors)
```

In [65]:
```
model_concat.summary()
```

```
Model: "model_4"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Item (InputLayer)               (None, 1)            0
__________________________________________________________________________________________________
User (InputLayer)               (None, 1)            0
__________________________________________________________________________________________________
ItemEmbedding (Embedding)       (None, 1, 2)         3364        Item[0][0]
__________________________________________________________________________________________________
UserEmbedding (Embedding)       (None, 1, 2)         1886        User[0][0]
__________________________________________________________________________________________________
FlattenItemE (Flatten)          (None, 2)            0           ItemEmbedding[0][0]
__________________________________________________________________________________________________
FlattenUserE (Flatten)          (None, 2)            0           UserEmbedding[0][0]
__________________________________________________________________________________________________
Concat (Concatenate)            (None, 4)            0           FlattenItemE[0][0]
                                                                 FlattenUserE[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 4)            0           Concat[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 32)           160         dropout_5[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 32)           0           dense_5[0][0]
__________________________________________________________________________________________________
UserBias (Embedding)            (None, 1, 1)         943         User[0][0]
__________________________________________________________________________________________________
ItemBias (Embedding)            (None, 1, 1)         1682        Item[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 1)            33          dropout_6[0][0]
__________________________________________________________________________________________________
FlattenUserBiasE (Flatten)      (None, 1)            0           UserBias[0][0]
__________________________________________________________________________________________________
FlattenItemBiasE (Flatten)      (None, 1)            0           ItemBias[0][0]
__________________________________________________________________________________________________
AddBias (Add)                   (None, 1)            0           dense_6[0][0]
                                                                 FlattenUserBiasE[0][0]
                                                                 FlattenItemBiasE[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 1)            0           AddBias[0][0]
__________________________________________________________________________________________________
lambda_3 (Lambda)               (None, 1)            0           activation_3[0][0]
==================================================================================================
Total params: 8,068
Trainable params: 8,068
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [66]:
```
plot_model(model_concat, show_layer_names=True, show_shapes=True)
```

Out[66]:



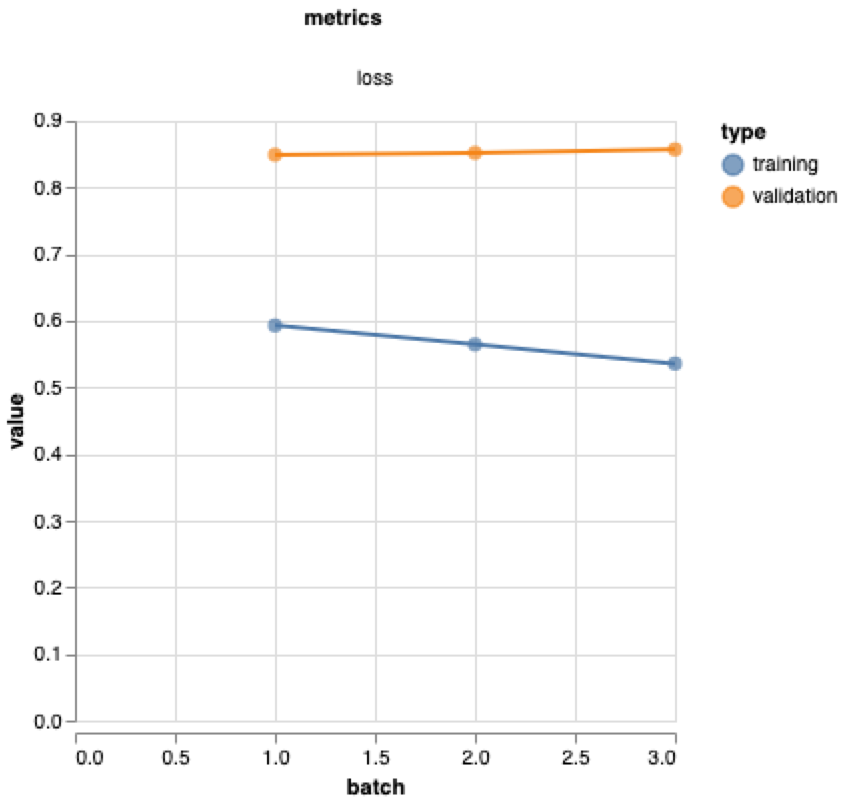
In [67]:
```
trainU, testU = user_split(DATA, [0.8, 0.2])
```

In [68]:
```
%%time
output_concat = model.fit([trainU.USER, trainU.ITEM], trainU.RATING,
                   batch_size=128, verbose=1, epochs=3,
                   validation_data=([testU.USER, testU.ITEM], testU.RATING))
```

```
Train on 80000 samples, validate on 20000 samples
Epoch 1/3
80000/80000 [==============================] - 1s 14us/step - loss: 0.5930 - val_loss: 0.8492
Epoch 2/3
80000/80000 [==============================] - 1s 12us/step - loss: 0.5645 - val_loss: 0.8520
Epoch 3/3
80000/80000 [==============================] - 1s 12us/step - loss: 0.5358 - val_loss: 0.8572
CPU times: user 6.09 s, sys: 434 ms, total: 6.53 s
Wall time: 3.13 s
```

In [69]:
```
metrics(output_concat.history)
```

Out[69]:



In [ ]: