Master IAII
Pr. Sara QASSIMI
08 Decembre 2023

Tutorial: Autoencoder for Collaborative Filtering

Objective:

This tutorial provides a comprehensive walkthrough of building a Residual Network for recommendation with the MovieLens 20M dataset. Understand the code structure, model construction, training, and result visualization to apply similar techniques to other recommendation problems.

Step 1: Import Libraries

```
from __future__ import print_function, division
from builtins import range, input

import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.utils import shuffle

from keras.models import Model
from keras.layers import Input, Embedding, Dot, Add, Flatten, Dense , Concatenate
from keras.layers import Dropout, BatchNormalization, Activation
from keras.regularizers import l2
from keras.optimizers import SGD, Adam
```

Step 2: Load Data

```
df = pd.read_csv('../../movielens-20m-dataset/edited_rating.csv')
N = df.userId.max() + 1  # number of users
M = df.movie_idx.max() + 1  # number of movies
df = shuffle(df)
cutoff = int(0.8 * len(df))
df_train = df.iloc[:cutoff]
df_test = df.iloc[cutoff:]
```

Step 3: Model Configuration

```
K = 10  # latent dimensionality
mu = df_train.rating.mean()
epochs = 15
reg = 0.  # regularization penalty
```

Step 4: Building the Residual Network Model

```
# Keras Model Construction
u = Input(shape=(1,))
m = Input(shape=(1,))
u_embedding = Embedding(N, K)(u)
m_embedding = Embedding(M, K)(m)

# Main Branch
u_bias = Embedding(N, 1)(u)
m_bias = Embedding(M, 1)(m)
x = Dot(axes=2)([u_embedding, m_embedding])
x = Add()([x, u_bias, m_bias])
x = Flatten()(x)

# Side Branch
u_embedding = Flatten()(u_embedding)
m_embedding = Flatten()(m_embedding)
y = Concatenate()([u_embedding, m_embedding])
y = Dense(400)(y)
y = Activation('elu')(y)
y = Dense(1)(y)

# Merge Main and Side Branches
x = Add()([x, y])
```

```
# Compile the Model
model = Model(inputs=[u, m], outputs=x)
model.compile(
  loss='mse',
  optimizer=SGD(lr=0.08, momentum=0.9),
  metrics=['mse'],
)
```

Step 5: Model Training

```
r = model.fit(
  x=[df_train.userId.values, df_train.movie_idx.values],
  y=df_train.rating.values - mu,
  epochs=epochs,
  batch_size=128,
  validation_data=(
    [df_test.userId.values, df_test.movie_idx.values],
    df_test.rating.values - mu
  )
)
```

Step 6: Visualize Results

```
# Plot Losses
plt.plot(r.history['loss'], label="train loss")
plt.plot(r.history['val_loss'], label="test loss")
plt.legend()
plt.show()

# Plot Mean Squared Error
plt.plot(r.history['mean_squared_error'], label="train mse")
plt.plot(r.history['val_mean_squared_error'], label="test mse")
plt.legend()
plt.show()
```