كليـــــة العلـــــوم
والتقنيات - مراكش
FACULTE DES SCIENCES
ET TECHNIQUES - MARRAKECH

Master IAII
Pr. Sara QASSIMI

# Tutorial: Building a recommendation system using matrix factorization with Keras

### Step 1: Importing Libraries

Let's start by importing the required libraries. Make sure you have the necessary libraries installed. If not, you can install them using **pip**.

```
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
from keras.regularizers import l2
from keras.optimizers import SGD, Adam
```

### Step 2: Loading and Preprocessing Data

Load your dataset into a Pandas DataFrame. We calculate **N** and **M** as the number of users and movies, respectively. Then, shuffle the data and split it into training and testing sets (80% training, 20% testing).

```
# Load the data
df = pd.read_csv('edited_rating.csv')

# Get the number of users and movies
N = df.userId.max() + 1
M = df.movie_idx.max() + 1

# Shuffle the data and split into train and test sets
df = shuffle(df)
cutoff = int(0.8 * len(df))
df_train = df.iloc[:cutoff]
df_test = df.iloc[cutoff:]
```

### Step 3: Preprocess Data

Configure the model parameters such as the latent dimension (**K**), regularization (**reg**), and other hyperparameters. Define the input layers for user (**u**) and movie (**m**) IDs, and create embeddings for users and movies. Calculate user and movie biases and compute the dot product. Compile the model with Mean Squared Error (**MSE**) loss and an **SGD** optimizer.

```
# Set up model parameters
K = 10  # Latent dimensionality
mu = df_train.rating.mean()
epochs = 15
reg = 0.  # Regularization penalty

# Define the Keras model
u = Input(shape=(1,))
m = Input(shape=(1,))
u_embedding = Embedding(N, K, embeddings_regularizer=l2(reg))(u)
m_embedding = Embedding(M, K, embeddings_regularizer=l2(reg))(m)

u_bias = Embedding(N, 1, embeddings_regularizer=l2(reg))(u)
m_bias = Embedding(M, 1, embeddings_regularizer=l2(reg))(m)

x = Dot(axes=2)([u_embedding, m_embedding])
x = Add()([x, u_bias, m_bias])
x = Flatten()(x)

model = Model(inputs=[u, m], outputs=x)

model.compile(
  loss='mse',
  optimizer=SGD(lr=0.08, momentum=0.9),
  metrics=['mse'],
)
```

### Step 4: Model Training

Train the model using the training data (**df_train**) and labels (ratings minus the mean rating). Specify the number of epochs and batch size. Validation data is provided to monitor model performance on the test set during 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 5: Visualizing Results

Visualize the training and validation losses and Mean Squared Error (MSE) to assess the model's performance.

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

# Plot MSE
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()
```

كليـــــة العلـــــوم
والتقنيات - مراكش
FACULTE DES SCIENCES
ET TECHNIQUES - MARRAKECH

Master IAII
Pr. Sara QASSIMI

---

## Step 6: Compare with Different Optimizers

**Assignment: Comparing Optimizers for Movie Recommendation System**

**Due Date**: Before midnight

**Instructions**:

In this assignment, you will explore the impact of different optimizers on the performance of a movie recommendation system built using matrix factorization with Keras. You will be provided with a Python script (similar to the tutorial) that implements the recommendation system using the Stochastic Gradient Descent (SGD) optimizer. Your task is to modify the script to compare this optimizer with two others: Adam and RMSprop.

1. Modify the Script: Open the provided Python script and locate the section labeled "Compare with Different Optimizers." In this section, you will find instructions on how to introduce the Adam and RMSprop optimizers into the model training process. Ensure that you understand the code changes required to implement this comparison.

2. Train the Model: Execute the modified script to train the recommendation system with all three optimizers: SGD, Adam, and RMSprop. Observe and record the training results for each optimizer, including validation loss and Mean Squared Error (MSE).

3. Analysis and Reporting:* Write a brief report (in the body of your email) summarizing your findings. Include insights on how each optimizer performed, whether any optimizer demonstrated faster convergence or better generalization, and any observations you made during the experiment. Feel free to adjust the learning rates if you wish to fine-tune the optimizers.

4. Submission:  Email your report to [sara.qassimi@uca.ac.ma] before midnight on . Ensure the subject of your email is "Optimizer Comparison Assignment - [Your Name]." Late submissions will not be accepted.

This assignment allows you to gain hands-on experience in comparing different optimization algorithms and evaluating their impact on model training.