**BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM**

Sinh viên thực hiện: Nhóm 2122CDCL\_04

GV phụ trách: Hồ Thị Hoàng Vy

Đồ án 2  - CHUYÊN ĐỀ CHỌN LỌC TRONG HỆ THỐNG THÔNG TIN

HỌC KỲ II – NĂM HỌC 2021-2022

**môn chuyên đề chọn lọc trong hệ thống thông tin**

**BẢNG THÔNG TIN CHI TIẾT NHÓM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mã nhóm:** | 2122CDCL\_04 | | | |
| **Tên nhóm:** | 2122CDCL\_04 | | | |
| **Số lượng:** | **4** | | | |
| **MSSV** | **Họ tên** | **Email** | **Điện thoại** | **Hình ảnh** |
| 1712486 | Võ Quốc Hưng | 1712486@student.hcmus.edu.vn |  |  |
| 18120418 | Phạm Minh Khoa | 18120418@student.hcmus.edu.vn |  |  |
| 1712850 | Trần Trung | 1712850@student.hcmus.edu.vn |  |  |
| 1712553 | Bùi Tấn Lân | 1712553@student.hcmus.edu.vn |  |  |

**BẢNG PHÂN CÔNG & ĐÁNH GIÁ HOÀN THÀNH CÔNG VIỆC**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bảng phân công & đánh giá hoàn thành công việc** | | | |
| **Công việc thực hiện** | **Người thực hiện** | **Mức độ hoàn thành** | **Đánh giá của nhóm** |
| Tìm phương pháp, nghiên cứu bài làm. Thực hiện đánh giá model. Làm báo cáo. Nộp bài. | Võ Quốc Hưng | 70% | 7/10 |
| Tìm phương pháp, nghiên cứu bài làm. Lên ý tưởng chính cho thuật toán, code thuật toán. | Phạm Minh Khoa | 100% | 10/10 |
| Tìm phương pháp, nghiên cứu bài làm. Thực hiện đánh giá model. | Trần Trung | 60% | 6/10 |
| Tìm phương pháp, nghiên cứu bài làm. Thực hiện lọc dữ liệu, vẽ biểu đồ. | Bùi Tấn Lân | 60% | 6/10 |

Mục Lục

[**A.** **YÊU CẦU CỦA ĐỒ ÁN** 3](#_Toc106314252)

[1. Yêu cầu đồ án 3](#_Toc106314253)

[2. Yêu cầu nộp 3](#_Toc106314254)

[**B.** **Kết quả** 4](#_Toc106314255)

[1. Giải thích dataset, ý nghĩa thuộc tính 4](#_Toc106314256)

[2. Ý tưởng thực hiện thuật toán (mã giả) 8](#_Toc106314257)

[2.1. Giai đoạn offline 10](#_Toc106314258)

[2.2. Giai đoạn online 11](#_Toc106314259)

[3. Kết quả thực hiện 11](#_Toc106314260)

[4. Đánh giá 20](#_Toc106314261)

[4.1. Đánh giá tính chính xác của việc dự đoán rating 20](#_Toc106314262)

[5. Link tham khảo 21](#_Toc106314263)

# **YÊU CẦU CỦA ĐỒ ÁN**

1. Yêu cầu đồ án
2. Dataset: <https://grouplens.org/datasets/movielens/100k/>
3. Hãy cài đặt hệ tư vấn phim theo phương pháp phân rã ma trận - Matrix factorization  
   (model based) sử dụng ngôn ngữ python.

* Mô tả dataset cách chi tiết (word).
* Load data & Data cleaning
* Giai đoạn offline:
  + Xây dựng mô hình: sử dụng SVD++
* Giai đoạn online:
  + Phát sinh các gợi ý
* Đánh giá thuật toán:
  + Lựa chọn độ đo đánh giá độ chính xác của dự đoán

1. Yêu cầu nộp

* doc: báo cáo giải thích về dataset, ý nghĩa thuộc tính, ý tưởng thực hiện thuật  
  toán (mã giả), kết quả thực hiện, đánh giá…
* Video: quay lại quá trình trao đổi và thực hiện của mỗi thành viên có mở camera
* Source code: .pybn và link google colab
* Video run & giải thích lại kết quả thực hiện của nhóm
* Làm bài trên google colab

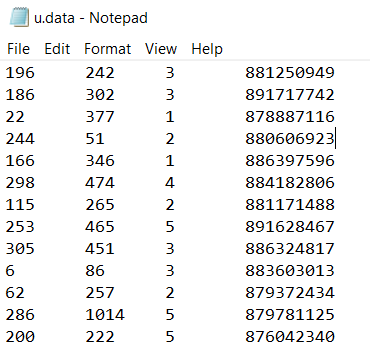
# **Kết quả**

## Giải thích dataset, ý nghĩa thuộc tính

Sử dụng thư viện pandas và file README để tìm hiểu, kiểm tra, lọc data.

Sử dụng hàm read\_csv() để đọc dữ liệu.

* **ml-100k.zip:** chứa tập những file dữ liệu (u.data, u1.base, u1.test…), file chứa script phân data và file thông tin mô tả.
* **u.data:**

****

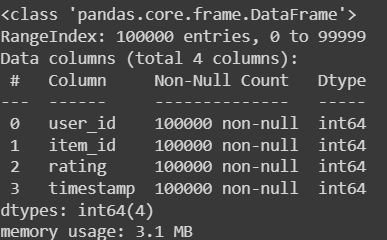
* + Theo mô tả từ file README, file gồm có 100.000 rating (đánh giá từ 1  5) của 943 người dùng về 1682 video. Mỗi user đã đánh giá ít nhất 20 phim. Dữ liệu được sắp xếp ngẫu nhiên và mỗi dòng tuân theo cấu trúc sau: **user id | item id | rating | timestamp**. Các cột cách nhau bởi dấu tab.

***Kiểm tra dữ liệu:***

* + File chỉ chứa dữ liệu, dòng đầu tiên không phải tên của các cột nên sử dụng hàm read\_csv() với tham số names là mảng các tên cột để khỏi nhầm dòng đầu tên là tên các cột. Khi có đối số names, thì tham số chuyển từ mặc định header=0 sang header=non, cho phép lấy mảng names làm tên các cột.
  + Lệnh đọc data từ file u.data trên google drive, sử dụng google colab để thực thi.

pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/moviesLenData/u.data", sep='\t', names=['user\_id','item\_id','rating','timestamp'])

* + Dùng hàm info() để kiểm tra dữ liệu, kết quả:

****

* + - User id, Item id, Rating, timestamp đều là số nguyên.
  + Các **timestamp** là Unix giây kể từ 1/1/1970 UTC, lúc 1/1/1970 là 0 và thời gian sau đó cộng thêm số giây tăng thêm.
  + Kiểm tra giá trị của rating: max=5, min=1

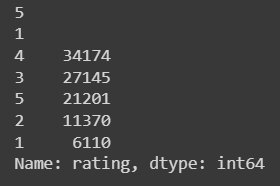
Sử dụng các hàm:

print(df.rating.max())

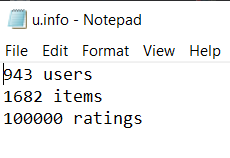
print(df.rating.min())

print(df.rating.value\_counts())

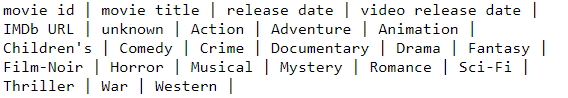
**Kết quả:**

****

* + Tưng tự cho số user và video đều khớp với mô tả trong file README.
  + **Vậy không có dữ liệu sai, null, thiếu sót trong file u.data => không cần lọc dữ liệu.**
* **u.info:** gồm số lượng user, phim, rating trong file **u.data**

****

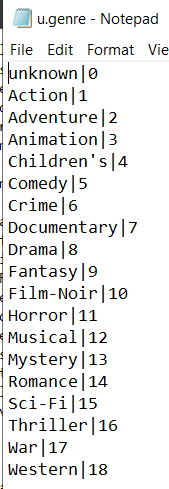
* **u.item:** thông tin về các phim, 19 cột sau cùng là thể loại (từ unknown trở về sau). Giá trị 1 là phim thuộc thể loại, ngược lại, không thuộc là 0. Một phim có thể thuộc nhiều thể loại. movie id dược sử dụng trong file u.data. Cấu trúc như sau:

****

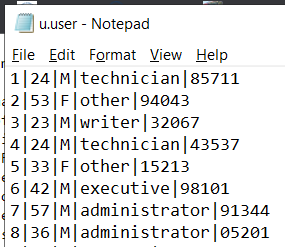
Và dữ liệu được lưu trong file như sau, ngăn cách bằng dấu gạch đứng |, trước cột URL có hai gạch đứng ||:



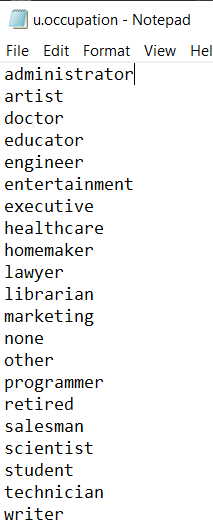
* + Lệnh đọc file u.item: với đối số đầu tiên là đường dẫn tới file u.item, đối số thứ 2 là ký tự ngăn cách (sep), header=None chỉ file dữ liệu không có tên cột cho dữ liệu, tránh việc lấy dòng đầu làm tên cột, names là danh sách tên cột đưa vào, encoding để chỉ loại dữ liệu ký tự bên trong file u.item.
* movieItem = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/moviesLenData/u.item', sep='|',header=None,names=colItem,encoding='latin-1')
* **u.genre:** danh sách các thể loại, gồm có 19 thể loại. Cột 1 là tên, cột 2 là id, 2 cột được ngăn cách bởi dấu gạch đứng |.

****

* **u.user:** Thông tin nhân khẩu về những user, mỗi cột được ngăn cách bởi dấu gạch đứng | và có cấu trúc như sau: user id | age | gender | occupation | zip code. User id được sử dụng trong u.data.

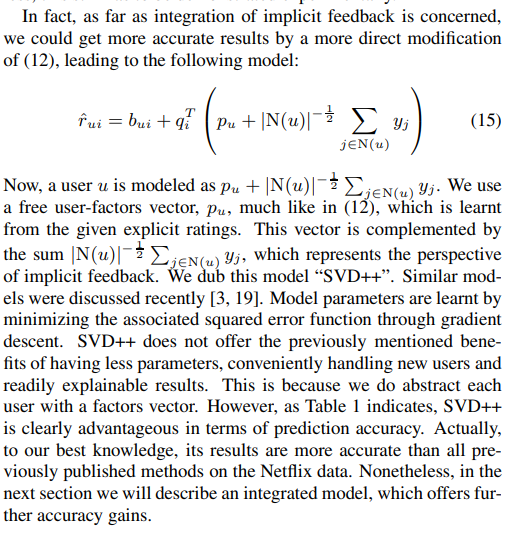
****

* **u.occupation:** danh cách các nghề nghiệp, gồm 21 nghề.

****

* **u1.base  u5.base và u1.test  u5.test:** .base là dữ liệu train, .test là dữ liệu test. Dữ liệu được lấy ra từ u.data chia 80% cho u1.base và 20% cho u1.test. u1  u5 được tạo khi chạy file thực thi mku.sh.
* **ua.base, ua.test, ub.base, ub.test:** .base là tập dữ liệu train, .test là tập dữ liệu test với mỗi user lấy 10 đánh giá của user (tức lấy 10 phim được đánh giá bởi mỗi người dùng). File ua.test bị lỗi mở bằng notepad, lên toàn tiếng trung, nhưng mở bằng visual code thì được. Dữ liệu được lấy từ file u.data khi chạy file mku.sh.
* **allbut.pl:**  script tạo ra các tập training và test trong đó tất cả trừ n rating người dùng nằm trong dữ liệu training.

## Ý tưởng thực hiện thuật toán (mã giả)



*ảnh tử bài báo của Koren*

The other one, denoted by N(u), contains all items for which u provided an implicit preference.

Yj is the (implicit) item factors : numpy array of size (n\_items, n\_factors) tương đương với qi nhưng radom giá trị khác nhau.

qiT =  **; pu** =

bias cua user u

bias cua item i

trung bình rating

**[***m x s***]:** ma trận nhân tố ẩn biểu diễn người dùng (user-factor matrix)

**[***n x s***]:** ma trận nhân tố ẩn biểu diễn sản phẩm (item-factor matrix)

: số factor ẩn

**:**  ma trận factor ẩn

* Model: *u=1…m; i=1…n;*  **[***m x s***]; [***n x s***]; ,**

Model estimation:

Tikhonov regularization: regularization weight

Tối ưu bằng stochastic gradient descent:

(Từ công thức sau chúng ta sẽ tách tường phần để tích ở giai đoàn offline.)

Xét hàm mục tiêu tại 1 điểm dữ liệu:

Lấy đạo hàm theo biến (dưới từ min trong công thức):

### Giai đoạn offline

α (learning rate) có giá trị mặc định là 0.005, λ(regulazation) là 0.02, số factor mặc định là 20.

Step 1: Initialize randomly , **,** , in (0,1).

Step 2: { | }

Step 3:   
Step 4: Repeat until convergence (sẽ cho 1 số mặc định là 20 “epoches”):

Step 5: Randomly shuffle the elements in 𝕐.

Step 6: For each pair in :

Step 7:

Step 8:

Step 9:

Step 10:

Step 11:

Step 12: for each j in all item as u rated:

Step 13: = -

Step 14:

Step 15:

### Giai đoạn online

* Dự đoán film cho user.
* Sử dụng model đã training ở bước offline cho việc dự đoán.
* Mã giả :

bestMovies : chứa danh sách phim và rating dự đoán.

Userid : id của user cần recommend movies

amountMovie : số phim cần recommend cho user.

model : là instance của class SVDpp, đã train dở giai đoạn offline.

predict: là hàm dự đoán rating

* bestMovies = {}
* loop I=1 <= amount of movie in system:
* if user đã rate cho i thì bỏ qua:
* i++
* continue
* bestMovies[i]=model.predict(userid, i)
* i++
* Sort(bestMovies, descrease)
* return bestMovies[0:amountMovie]

## Kết quả thực hiện

# -\*- coding: utf-8 -\*-

"""DA2-SVD++MatrixFactorization.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1hMRn23f1z1AqrupNmNqjxMsgmxeu1GyR

# \*\*Adding library\*\*

"""

#adding library

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

#import random for random between a and b (not include a, b)

import random

"""# \*\*Pull data directly and unzip from grouplens website\*\*"""

DATASET\_LINK='http://files.grouplens.org/datasets/movielens/ml-100k.zip'

!wget -nc http://files.grouplens.org/datasets/movielens/ml-100k.zip

!unzip -n ml-100k.zip

"""# \*\*Describe data, statistics data in general way\*\*

## \*Describe\*

"""

# Viewing u.data file

column\_names1 = ['user id','movie id','rating','timestamp']

df = pd.read\_csv('ml-100k/u.data', sep='\t',header=None,names=column\_names1)

df.head()

#check min, max rating and count how many the distincts value

print("rating:")

print(df.rating.max())

print(df.rating.min())

print(df.rating.value\_counts())

#check how many the distincts value of users id

print("user:")

print(df["user id"].value\_counts())

#check how many the distincts value of video

print("video:")

print(df["movie id"].value\_counts())

df['rating'].describe()

"""## \*Statistics and charts\*"""

plt.hist(df['rating'],log=True);

plt.plot();

plt.show();

"""# \*\*Offline phase\*\*

## \*Build matrix factorization using svd++\*

"""

#@title Default title text

class SVDpp:

# Khởi tạo các biến cho class SVDpp

# matrixDataSet là tập dữ liệu đưa vào (tập train), numFactor là số factor ẩn

# bi là bias của item i, bu là bias của item u

# qi là dictionary chứa những key (id item) và giá trị của key là ma trận (1 cột)

#   numFactor số random trong nữa khoảng [0,1)

# qu là dictionary chứa những key (id user) và giá trị của key là ma trận (1 cột)

#   numFactor số random trong nữa khoảng [0,1)

# yi là dictionary chứa những key (id item) và giá trị của key là ma trận (1 cột)

#   numFactor số 0.1; đây là ma trận những nhân tố ẩn.

# avg là trung bình rating của matrixDataSet đưa vào.

# u\_dict là dictionary chứa các key là user id và value là danh sách các item id

#   mà user đó đã rate.

    def \_\_init\_\_(self, matrixDataSet, numFactor=20):

        self.matrixDataSet = np.array(matrixDataSet)

        self.numFactor = numFactor

        self.bi = {}

        self.bu = {}

        self.qi = {}

        self.pu = {}

        self.avg = np.mean(self.matrixDataSet[:, 2])

        self.yi = {}

        self.u\_dict = {}

        for i in range(self.matrixDataSet.shape[0]):

            user\_id = self.matrixDataSet[i, 0]

            item\_id = self.matrixDataSet[i, 1]

            self.u\_dict.setdefault(user\_id, [])

            self.u\_dict[user\_id].append(item\_id)

            self.bi.setdefault(item\_id, 0)

            self.bu.setdefault(user\_id, 0)

            self.qi.setdefault(item\_id, np.random.random((self.numFactor, 1)))

            self.pu.setdefault(user\_id, np.random.random((self.numFactor, 1)))

            self.yi.setdefault(item\_id, np.zeros((self.numFactor, 1)) + .1)

# Tính căn bậc 2 của Nu và Σyj. It used for predict, train method

# Nu is the amount of item which user rated. Get from u\_dict

# userImplicitFactor is the result of sqrt(Nu) multiply Σyj

    def getNuYj(self, user\_id, item\_id):

            Nu = self.u\_dict[user\_id]

            numItemOfNu = len(Nu)

            sqrt\_Nu = np.sqrt(numItemOfNu)

            y\_u = np.zeros((self.numFactor, 1))

            if numItemOfNu == 0:

                userImplicitFactor = y\_u

            else:

                for idItem in Nu:

                    y\_u += self.yi[idItem]

                userImplicitFactor = y\_u / sqrt\_Nu

            return userImplicitFactor, sqrt\_Nu

# Hàm predict để predict rating

# Phương thức setdefault là dành cho những item, user mới vào hệ thống

#   và thiết lập bi, bu, qi, pu và yi bằng 0. Khởi tạo mảng item rỗng mà user id

#   đã rate (chưa có gì)

# Bởi vì score nằm trong đoạn từ 1 đến 5, khi điểm lớn hơn 5 trả về 5

# hoặc nhỏ hơn 1, 1 sẽ được trả về.

    def predict(self, user\_id, item\_id):

        self.bi.setdefault(item\_id, 0)

        self.bu.setdefault(user\_id, 0)

        self.qi.setdefault(item\_id, np.zeros((self.numFactor, 1)))

        self.pu.setdefault(user\_id, np.zeros((self.numFactor, 1)))

        self.yi.setdefault(item\_id, np.zeros((self.numFactor, 1)))

        self.u\_dict.setdefault(user\_id, [])

        userImplicitFactor, sqrt\_Nu = self.getNuYj(user\_id, item\_id)

        rating = self.avg + self.bi[item\_id] + self.bu[user\_id] + np.sum(

            self.qi[item\_id] \* (self.pu[user\_id] + userImplicitFactor))

        if rating > 5:

            rating = 5

        if rating < 1:

            rating = 1

        return rating

# Train function which is build by the matrix factorization and the svd++

# Lambda thay cho lambda vì bị lỗi do trùng với anonymous function (lambda).

# At the step 2 of docx, we get all pair of user, item, but here, we use number

#   of the rating in dataset source and random to take by np.random.permutation()

#   method which references n number and return an contigent array of the n number from 0 to n-1

# Get all of rating in dataset: self.matrixDataSet.shape[0]

    def train(self, epochs=20, alpha=0.005, Lambda=0.02):

        for epoch in range(epochs):

            print('epoch', epoch + 1, 'is running')

            Yui = np.random.permutation(self.matrixDataSet.shape[0])

            # Stochastic Gradient Descent

            rmse = 0.0

            for i in range(self.matrixDataSet.shape[0]):

                j = Yui[i]

                user\_id = self.matrixDataSet[j, 0]

                item\_id = self.matrixDataSet[j, 1]

                rating = self.matrixDataSet[j, 2]

                predict = self.predict(user\_id, item\_id)

                userImplicitFactor, sqrt\_Nu = self.getNuYj(user\_id, item\_id)

                eui = rating - predict #step 7

                rmse += eui \*\* 2

                tempPu = self.pu[user\_id] - alpha \* (Lambda \* self.pu[user\_id] - eui \* self.qi[item\_id])

                tempQi = self.qi[item\_id] - alpha \* (Lambda \* self.qi[item\_id] - eui \* (self.pu[user\_id] + userImplicitFactor))

                self.bu[user\_id] -= alpha \* (Lambda \* self.bu[user\_id] - eui) #step 10

                self.bi[item\_id] -= alpha \* (Lambda \* self.bi[item\_id] - eui)

                for j in self.u\_dict[user\_id]: #step 14

                    self.yi[j] -= alpha \* (Lambda \* self.yi[j] - eui \* self.qi[j] / sqrt\_Nu)

self.pu[user\_id] = tempPu

               self.qi[item\_id] = tempQi

            print('rmse: ', np.sqrt(rmse / self.matrixDataSet.shape[0]))

    def test(self, test\_data):

        test\_data = np.array(test\_data)

        print('test data size: ', test\_data.shape)

        rmse = 0.0

        for i in range(test\_data.shape[0]):

            user\_id = test\_data[i, 0]

            item\_id = test\_data[i, 1]

            rating = test\_data[i, 2]

            eui = rating - self.predict(user\_id, item\_id)

            rmse += eui \*\* 2

        print('rmse of test data is: ', np.sqrt(rmse / test\_data.shape[0]))

        return np.sqrt(rmse / test\_data.shape[0])

"""## \*Preparing data to train\*

### - Random (or load if right) data to train and test

"""

column\_names1 = ['user id','movie id','rating','timestamp']

datatrain = pd.read\_csv('/content/ml-100k/u1.base', sep='\t',header=None,names=column\_names1)

datatest = pd.read\_csv('/content/ml-100k/u1.test', sep='\t',header=None,names=column\_names1)

"""### - Check, filter data"""

## Lấy dữ liệu và lọc dữ liệu (xóa duplicate, null, NaN)

datatrain.dropna(inplace=True)

datatrain.drop\_duplicates(inplace=True)

datatest.dropna(inplace=True)

datatest.drop\_duplicates(inplace=True)

## Lấy dữ liệu gồm title, id phim từ u.item và lọc

col = 'movie id | movie title | release date | video release date | IMDb URL | unknown | Action '

col+= '| Adventure | Animation | Children | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir '

col+= '| Horror | Musical | Mystery | Romance | Sci-Fi | Thriller | War | Western'

colItem = col.split(' | ')

movieItem = pd.read\_csv('ml-100k/u.item',

                        sep='|',header=None,names=colItem,encoding='latin-1')

movieItem = movieItem[['movie id','movie title']]

movieItem.dropna(inplace=True)

movieItem.drop\_duplicates(inplace=True)

"""## \*Training\*

### 1. Predict before train (testing)

"""

numFactor = 20

beforetrain = SVDpp(datatrain, numFactor)

#196    242 3   881250949

#186    302 3   891717742

#22 377 1   878887116

print (beforetrain.predict(196, 242))

print (beforetrain.predict(186  ,302))

print (beforetrain.predict(22,  377))

"""### 2. To train

epochs=20, numFactor=10,

times~=40'

"""

numFactor = 10

model = SVDpp(datatrain, numFactor)

model.train(epochs=5)

"""### 3. After trained (testing)"""

print (datatest.shape[0])

test\_data = np.array(datatest)

print('test data size', datatest.shape[0])

table={}

table[numFactor] = [model.test(datatest)]

# Create DataFrame

df = pd.DataFrame(table)

# Print the output.

print(df)

# predict rating

#196    242 3

#186    302 3

#22 377 1

print (model.predict(196, 242))

print (model.predict(186    ,302))

print (model.predict(22,    377))

#1  6   5   887431973

#1  10  3   875693118

print("===================")

print (model.predict(1  ,6))

print (model.predict(1, 10))

#247    50  5   893097024

#328    470 4   885046537

print("===================")

print (model.predict(247    ,50))

print (model.predict(328,   470))

"""Cho thấy việc predict ngày càng chính xác hơn nếu chúng ta build với epoch, numfactor cao hơn (không quá cao). Epoch cỡ 100-200, numFactor cỡ 20!?

## \*Train more\*

### \*Train thay đổi epochs, numFactor trên 1 tập dữ liệu train u1.base\*

epochs=50, numFactor=20 => times ~= 1h30

"""

modelFifTwe = SVDpp(datatrain, numFactor=20)

modelFifTwe.train(epochs=50)

# predict rating

#196    242 3

#186    302 3

#22 377 1

print (modelFifTwe.predict(196, 242))

print (modelFifTwe.predict(186  ,302))

print (modelFifTwe.predict(22,  377))

#1  6   5

#1  10  3

print("===================")

print (modelFifTwe.predict(1    ,6))

print (modelFifTwe.predict(1,   10))

#247    50  5

#328    470 4

print("===================")

print (modelFifTwe.predict(247  ,50))

print (modelFifTwe.predict(328, 470))

"""epochs=100 numFactor=20"""

modelHundFif = SVDpp(datatrain, numFactor=20)

modelHundFif.train(epochs=100)

# predict rating

#196    242 3

#186    302 3

#22 377 1

print (modelHundFif.predict(196, 242))

print (modelHundFif.predict(186 ,302))

print (modelHundFif.predict(22, 377))

#1  6   5

#1  10  3

print("===================")

print (modelHundFif.predict(1   ,6))

print (modelHundFif.predict(1,  10))

#247    50  5

#328    470 4

print("===================")

print (modelHundFif.predict(247 ,50))

print (modelHundFif.predict(328,    470))

"""\*Cho thấy build với epochs lớn, và factor lớn thì độ chính xác càng lớn. Nhưng trái lại thời gian build rất lâu; 85 epoch, 20 factor ~= 3h\*

### A testing chains

"""

modelTwTwChains = SVDpp(datatrain, numFactor=20)

modelTwTwChains.train(epochs=10)

modelTwTwChains.train(epochs=20)

modelTwTwChains.train(epochs=20)

# predict rating

#196    242 3

#186    302 3

#22 377 1

print (modelTwTwChains.predict(196, 242))

print (modelTwTwChains.predict(186  ,302))

print (modelTwTwChains.predict(22,  377))

#1  6   5

#1  10  3

print("===================")

print (modelTwTwChains.predict(1    ,6))

print (modelTwTwChains.predict(1,   10))

#247    50  5

#328    470 4

print("===================")

print (modelTwTwChains.predict(247  ,50))

print (modelTwTwChains.predict(328, 470))

"""# Online phase - Recommend films"""

#model = SDVpp(), above

userid=1

amountMovie=3

def checkItemRated(itemid, userid, model):

  listItem = model.u\_dict[userid]

  for item in listItem:

    if item == itemid:

      return True

  return False

def recommendFilm(model, userid, amountMovieSystem=1682, amountMovie=3):

  i=1

  countItemRated=0

  bestMovies = {}

  while i <= amountMovieSystem:

    if checkItemRated(i, userid, model):

      #print("item is rated: ", i)

      i+=1

      countItemRated+=1

      continue

    bestMovies[i]=model.predict(userid, i)

    #print("item is not rated", i, " with predict", bestMovies[i])

    i += 1

  sortedBestMovies = sorted(bestMovies.items(), key=lambda x: x[1], reverse=True)

  print("=======================================")

  print("bias:", model.bu[userid])

  print("number of rated:", countItemRated)

  return sortedBestMovies[0:amountMovie]

print("model", recommendFilm(model, 1))

print("modelFifTwe",recommendFilm(modelFifTwe, 1))

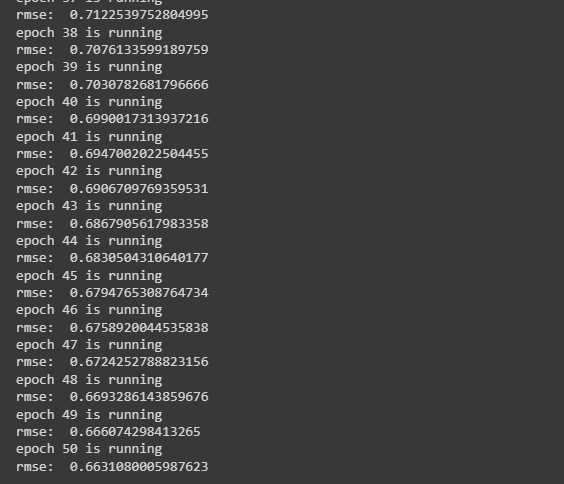
#print("modelTwTwChains",recommendFilm(modelTwTwChains, 1))

print("modelHundFif",recommendFilm(modelHundFif, 1))

## Đánh giá

### Đánh giá tính chính xác của việc dự đoán rating

* Chọn thuật toán RMSE để đánh giá rating dự đoán.
* Từ file u1.base ta radom chọn ngẫu nhiên các dòng trong lúc train dùng SGD.
* Có thể thay đổi epochs, numFactor trên 1 tập dữ liệu train u1.base. Khi số epochs, số Factor càng tăng thì thuật toán có RMSE càng nhỏ. Đổi lại thời gian train sẽ tăng lên rất nhiều.
* Sẽ chèn đánh giá vào các hàm trong model để đưa ra RMSE. Hàm test.
* là giá trị dự đoán.
* là giá trị che (ý là rating trong tập test, giá trị thật)
* là dùng cho những item chưa được user nào rating, từ đó ta dự đoán rating cho nó. Ở đây giả sử là tập test.



Với giá trị epoch càng lớn thì rmse giảm giần thì tỉ lệ dự doán chính xác cao hơn

## Link tham khảo

1. Chuyển dataframe thành item matrix: <https://stackoverflow.com/questions/37576594/rearrange-a-pandas-data-frame-to-create-a-2d-ratings-matrix>
2. Sparse matrix (sparse dataframe) to dense dataframe, or compressed sparse row matrix: <https://discuss.dizzycoding.com/pandas-sparse-dataframe-to-sparse-matrix-without-generating-a-dense-matrix-in-memory/>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html>

1. Tham khảo tham số svd++ trang surprise: <https://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVDpp>
2. Tham khảo code <https://github.com/NicolasHug/Surprise/blob/master/surprise/prediction_algorithms/matrix_factorization.pyx>
3. Tài liệu pdf của Yehuda Koren, nộp chung với đồ án.
4. <https://github.com/NicolasHug/Surprise/blob/master/surprise/prediction_algorithms/matrix_factorization.pyx#:~:text=%22%20processing%20epoch%20%7B%7D%22.format(current_epoch))-,for,-u%2C%20i%2C%20r%20in%20trainset.all_ratings()%3A>
5. <https://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVDpp>
6. <https://github.com/Murtazali05/matrix-factorization/blob/master/svdpp.py>
7. Xem file powerpoint (Presentation.pptx)
8. Link google colab: <https://colab.research.google.com/drive/1hMRn23f1z1AqrupNmNqjxMsgmxeu1GyR?usp=sharing>
9. Link video giải thích bài làm: <https://studenthcmusedu-my.sharepoint.com/personal/1712553_student_hcmus_edu_vn/_layouts/15/onedrive.aspx?id=%2Fpersonal%2F1712553%5Fstudent%5Fhcmus%5Fedu%5Fvn%2FDocuments%2Fvideo1358222819%2Emp4&parent=%2Fpersonal%2F1712553%5Fstudent%5Fhcmus%5Fedu%5Fvn%2FDocuments&ga=1>