TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG


# BÀI TẬP LỚN

## Big Data

### Đề tài

## Amazon Product Recommendation System

**GVHD: Thầy Trần Vĩnh Đức**

**Sinh viên thực hiện: Trần Văn Thành - MSSV 20133561**

**Phan Xuân Đức - MSSV 20146212**

*Hà Nội, tháng 1 năm 2018*

# Mục Lục



## Hình Ảnh

# 1. Hệ gợi ý (Recommendation System hay Recommender System)

Recommendation System (RS) là một mảng khá rộng của Machine Learning, có tuổi đời ít hơn so với classification hay regression vì internet chỉ thực sự bùng nộ khoảng 15-20 năm trở lại đây. Có 2 thực thể chính trong một hệ Recommendation system là user và item. User là người dùng, item là sản phẩm, ví dụ như các bộ phim, bài hát, cuốn sách (amazon product RS), videos clip (youtube RS) hoặc cũng có thể là chính các user khác trong bài toán gợi ý kết bạn của Facebook. Mục đích chính của RS chính là dự đoán chính xác mức độ quan tâm của một người dùng tới một sản phẩm nào đó, qua đó có chiến lược recommendation phù hợp, cho mục đích thương mại.

## 1.1 Các phương pháp học máy

- **Content-based System (CBS)**

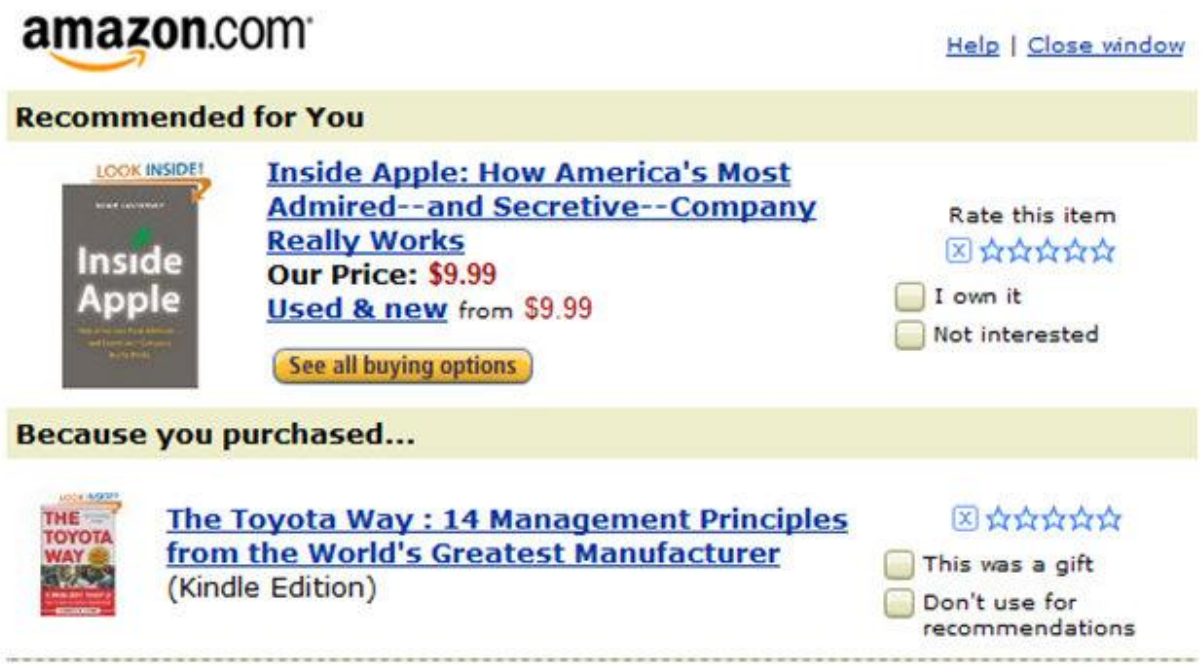
Khuyến nghị dựa trên đặc tính của sản phẩm. Ví dụ một người dùng  xem nhiều các bộ phim về cảnh sát hình sự, vậy thì gợi ý một  bộ phim trong CSDL có chung đặc tính "hình sự" cho người dùng này. Ở phương pháp này yêu cầu việc sắp xếp các sản phẩm vào từng nhóm hoặc đi tìm các đặc trưng của từng sản phẩm. Tuy nhiên sẽ có những sản phẩm không có nhóm cụ thể sẽ dẫn đến việc xác định nhóm hoặc đựac trưng của từng sản phẩm là rất khó khăn.

- **Collaborative Filtering (CF)**

Gọi là lọc cộng tác, phương pháp này khuyến nghị các sản phẩm dựa trên sự tương quan giữa người dùng và/hoặc sản phẩm (*user-user*, *user-item*, *item-item*). Một sản phẩm được giới thiệu tới người dùng dựa trên những người dùng khác có hành vi tương tự.

## 1.2 Bài Toán

Hệ gợi ý sản phẩm của Amazon - ***Amazon Product Recommendation System***

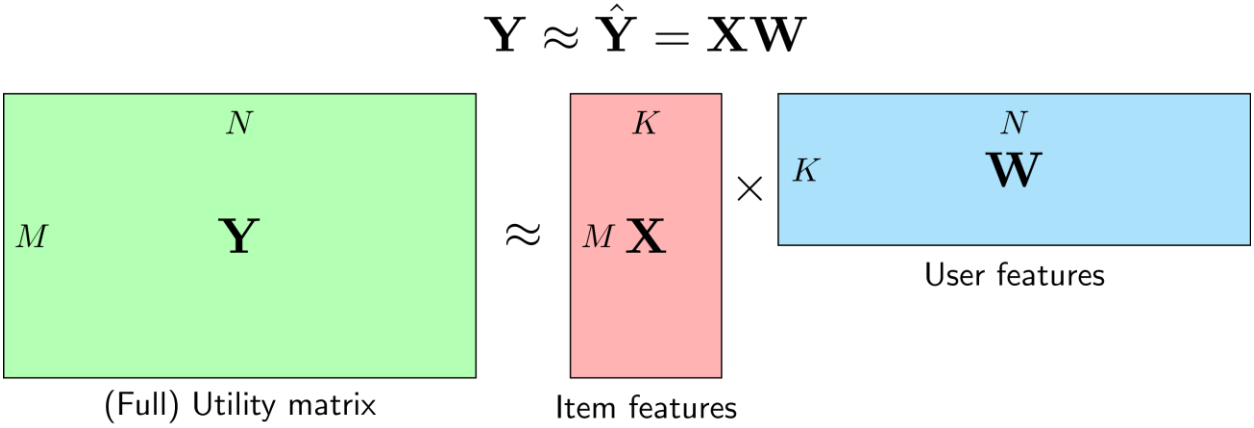



**Hình 1: Recommended for you.**

**DataSets**: bộ dữ liệu bán hàng của Amazon bao gồm hoạt động bán hàng và xếp hạng người dùng cho mỗi sản phẩm.

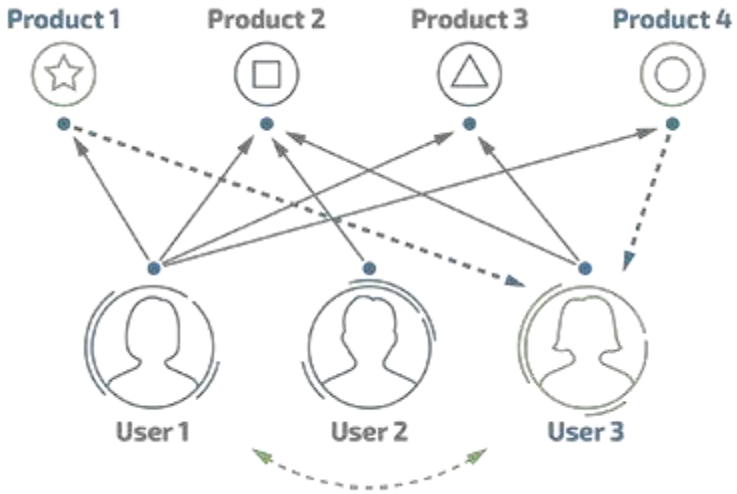
**Ý tưởng** là tạo ra một hệ thống gợi ý / giới thiệu sản phẩm cho mỗi người dùng dựa trên những lần mua hàng trước đây của anh ấy và xếp loại của anh ấy cho mỗi sản phẩm. Mô hình lọc cộng tác - *Collaborative Filtering* (CF) dựa trên bài toán phân tích ma trận thành nhân tử (*Matrix Factorization*) được xây dựng để dự đoán rate ảo cho sản phẩm mà người dùng không mua. Phương pháp này được gọi là *Matrix Factorization Collaborative Filtering* (MFCF). Hệ thống dự đoán xếp hạng của người dùng cho tất cả các mặt hàng và hiển thị các sản phẩm mà người dùng có thể mua và có đánh giá cao hơn. Đề xuất được tạo tự động cho những người dùng hiện có lịch sử và xếp hạng mua hàng. Đối với người dùng mới, mặc định các sản phẩm được xếp hạng cao nhất trong danh mục này được lên kế hoạch để được đề xuất.

Sử dụng Thuật toán "Xoay vòng Ít nhất" - *Alternating Least Squares* (ALS). Logic thuật toán ALS này được cài đặt để thực hiện bằng tay thay vì sử dụng thư viện ***spark.mllib*** có sẵn. Đề tài này được phát triển trên môi trường phân tán sử dụng Apache Spark và môi trường Python v2.7 sử dụng Hadoop v2.2.4.2

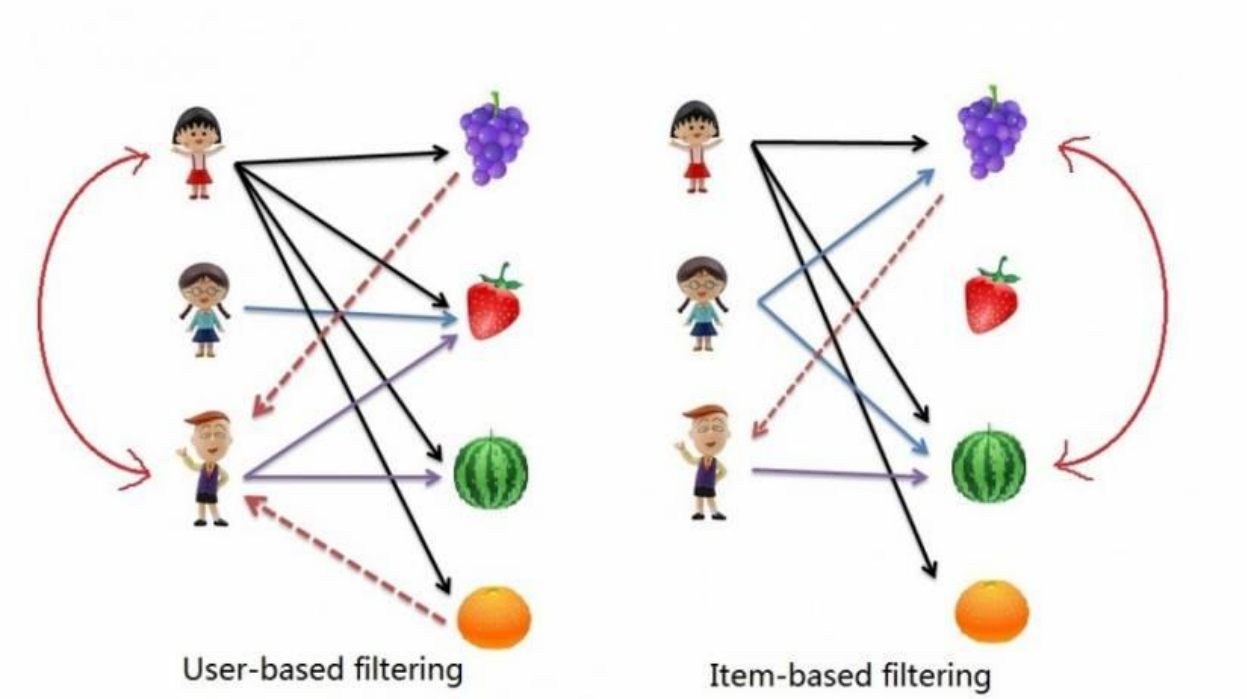



**Hình 2: Matrix Factorization. Utility matrix Y được phân tích thành tích của hai ma trận low-rank X và \W. link ảnh**

## Collaborative Filtering

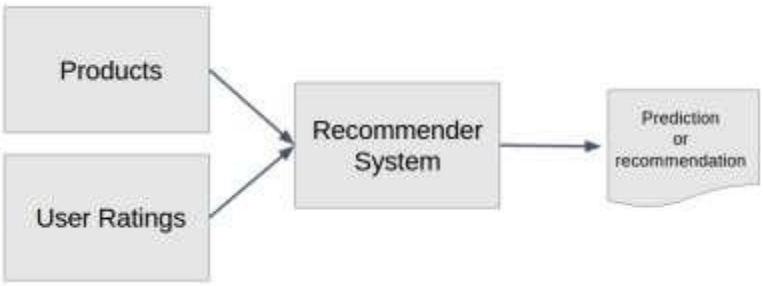



**Hình 3: User-Item.**




## 2. Cài đặt

Project bao gồm ba tệp đầu vào là ***ratings.dat***, ***users.dat*** và ***products.dat***. Các tệp đầu vào được định dạng ***.json*** lồng nhau được chuyển đổi thành tệp ***.dat*** bằng cách sử dụng kịch bản python. Sau đó các tập tin được chuyển tới hệ thống lưu trữ file Hadoop trong một thư mục đầu vào. Các tệp đầu vào sau đó được đọc vào một *array/collection/dictionary* và được phân phối qua các nhóm Hadoop để tạo ra một Bộ dữ liệu phân tán - Distributed Dataset có khả năng hồi phục. Sử dụng kỹ thuật lọc cộng tác - Collaborative Filtering để tính toán sự tương đồng của người dùng giữa các mục và thu hút sự quan tâm của người dùng có hành vi tương tự. Lọc cộng tác là một trong những trường hợp của phương pháp Neighborhood. Ở đây sử dụng kỹ thuật phân tích ma trận sử dụng phương pháp thay thế ít nhất để thực hiện lọc cộng tác. Ma trận dự đoán được tạo ra sau *n* số lần lặp lại và sau đó chúng tôi tương quan người dùng với các sản phẩm không được xếp hạng sớm hơn và có tỷ lệ dự đoán cao hơn giá trị, như xếp hạng 3 hoặc 4. Mỗi khuyến nghị sản phẩm cụ thể của người dùng cụ thể được ghi vào một tệp ***out.dat***, thường được lưu trữ và sử dụng trong môi trường thời gian thực để đưa ra các đề xuất.




**Hình 4: Cấu trúc Project.**

### 2.1 DataSets

Data Source and Metadata Description: http://jmcauley.ucsd.edu/data/amazon/links.html

Bộ dữ liệu được lấy từ dữ liệu sản phẩm của Amazon trong 18 năm qua của các loại sản phẩm khác nhau. Trong tập dữ liệu, chỉ chọn dữ liệu **rating** (500,176)**, products, user** của sản phẩm Nhạc cụ (Musical Instruments) được bán trên Amazon.

http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/ratings_Musical_Instruments.csv

### 2.2 Bóc tách json

- Ratings metadata : Các trường được đánh dấu đỏ được rút ra thành tệp *ratings.dat*


```
{
"reviewerID": "A2SUAM1J3GNN3B",
"asin": "0000013714",
"reviewerName": "J. McDonald",
"helpful": [2, 3],
"reviewText": "I am having a wonderful time playing these old hymns. The music is at times hard to read
because we think the book was published for singing from more than playing from. Great purchase
```

though!",

"overall": 5.0,

"summary": "Heavenly Highway Hymns",

"unixReviewTime": 1252800000,

"reviewTime": "09 13, 2009"

}

- Product metadata : Trường đánh dấu đỏ được rút ra thành một tệp tin *products.dat*

{

"asin": "0000031852",

"title": "Girls Ballet Tutu Zebra Hot Pink",

"price": 3.17,

"imUrl": "http://ecx.images-amazon.com/images/I/51fAmVkTbyL._SY300_.jpg",

"salesRank": {"Toys & Games": 211836},

"brand": "Coxlures",

"categories": [["Sports & Outdoors", "Other Sports", "Dance"]]

}

- Trích xuất dữ liệu

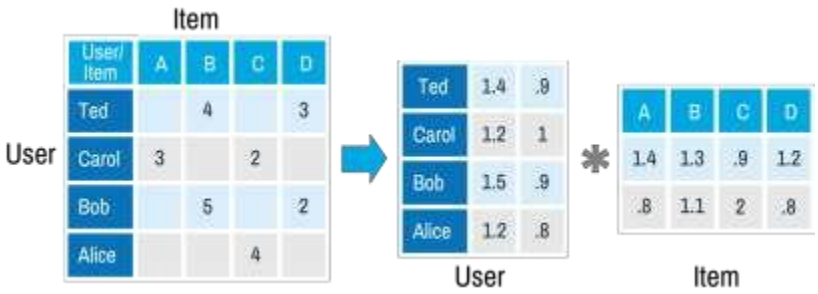
Nguồn dữ liệu có sẵn được nén và định dạng *.json* lồng nhau. Dữ liệu này được trích xuất thành tệp *.csv* sử dụng thư viện json, pandas trong Python. Do hạn chế khả năng xử lý bằng cách sử dụng máy ảo Hadoop có 2 clusters, vì vậy sử dụng tập dữ liệu này phù hợp.

- Matrix Factorization - Alternative Least Square

Chọn ngẫu nhiên giá trị U, V và điều chỉnh  các giá trị của U, V để cho RMSE nhỏ hơn.

- ✓ Step1: Sửa V và Cập nhật tất cả các mục của U.
- ✓ Step2: Sửa U và Cập nhật tất cả các mục của V.

✓ Step3: Lặp lại bước đầu tiên và bước thứ hai cho đến khi đạt đến độ bão hòa ít nhất của giá trị RMSE.




**Hình 5: Matrix Factorization.**

## 2.3 ALS - Alternating Least Squares

```
var U = sc.broadcast(User Matrix)

var V = sc.broadcast(Item Matrix)

var R = sc.broadcast(Ratings Matrix)

     for ( i in range(Iterations)):

          us = sc.parallelize(range(U), partitions)

               .map(lambda x: updateUV(x, vsb.value,
          Rb.value))

               .collect()

          usb = sc.broadcast(us)

          vs = sc.parallelize(range(V), partitions)

               .map(lambda x: updateUV(x, usb.value,
          Rb.value.T))

               .collect()

          vsb = sc.broadcast(vs)
```

### 2.4 Mã Giả

```
# import libraries
    numpy, pyspark
# check for sufficient arguments
    if (number of arguments passed !=5):
        sys.exit(1)
# declare functions for loading, parsing input files
    def loadRating(ratingsFile), def parseRating(line), def
parseProduct(line), def parseUser(line)
# declare function for compute RMSE
   def computeRMSE(R, us, vs):
    diff = R - us * vs.T
    rmse_val = np.sqrt(np.sum(np.power(diff, 2))/(U * V))
    return rmse_val
# declare function to calculate update U,V matrix after
each iteration
   def updateUV(i, uv, r):
    uu = uv.shape[0]
    ff = uv.shape[1]
    xtx = uv.T * uv
    xty = uv.T * r[i, :].T
    for j in range(ff):
        xtx[j, j] += lambdas * uu
    updated_val = np.linalg.solve(xtx, xty)
    return updated_val
```

```
# setup Environment

   conf                                                    =
SparkConf().setAppName("AmazonALS").set("spark.executor.mem
ory", "2g")

   sc = SparkContext(conf=conf)

# read ratings, products, users files into respective RDD's

    ratings        =        sc.textFile(join(hdfs_src_dir,
"ratings.dat")).map(parseRating)

    products       =        sc.textFile(join(hdfs_src_dir,
"products.dat")).map(parseProduct

    users          =        sc.textFile(join(hdfs_src_dir,
"users.dat")).map(parseUser)

# declare number of partitions, iterations, parameters

    Partitions_num  =  4,  Iterations_num  =  20,  seed,
regularization parameters

# Generate Random Matrix U,V

    U = matrix(rand(u, n_factors))

    V = matrix(rand(v, n_factors))

# convert ratings data into a Users vs Products and Values
matrix R

    Z = np.zeros((U,V))

    R = np.matrix(Z)

    for i in range(numRatings):

        r_local = r_array[i]

        R[(r_local[0]-1),(r_local[1]-1)] = r_local[2]

# broadcast the values to all the processors

    Rb = sc.broadcast(R)
```

```
    ub = sc.broadcast(U)
    vb = sc.broadcast(V)
# run for 20 iterations while adjusting and obtain optimum RMSE value
    for i in range(n_iterations):
        U = parallelize(Fix V and solve for U value)
        V = parallelize(Fix U and solve for V value)
        RMSE_current = computeRMSE(R, us, vs)
# Print the Iteration and RMSE
    print("Iteration Number”)
    print("RMSE value”)
# prediction Matrix
    RP = np.dot(pb.ub);
# write the products recommended to each user into a file
    output = open(outputfile,'w')
    for i in range(U):
        for j in range(V):
        pRating = recoB.value[i,j]
            if((Rb.value[i,j]==0 and pRating>3)):
                output.write(str(i)+","+str(j)+","+str(pR
            ating)+"\n")
    output.close()
# stop spark context
# reading the data
    def parse(path):
```

```
g = gzip.open(path, 'r')
for l in g:
    yield eval(l)
```

### 2.5 Hướng dẫn chạy chương trình

- **Step1**: Mở Hadoop on Virtual box, đã cài sẵn môi trường spark 1.2.1 và python 2.7
- **Step2**: Cài đặt thư viện numpy, scipy
- **Step3**: Chạy lệnh sau "export SPARK_HOME=/usr/hdp/2.2.4.2-2/spark" trên terminal
- **Step4**: Chuyển 4 file AmazonALS.py, ratings.dat, products.dat, users.dat sang thư mục: /workspace/AmazonRecommendation của máy ảo hadoop
- **Step5**: Tạo thư mục trong HDFS có dạng /user/AZ_P/input, và chuyển 3 file .dat vào thư mục input vừa tạo bằng các lệnh:
  $ hdfs dfs -mkdir /user/AZ_P/input
  $ hdfs dfs –put <file_name> <path>
- **Step6**: Vào thư mục Demo chứa file AmazonALS.py, mở terminal ở đây và chạy lệnh:
  $ su root
  $ spark-submit AmazonALS.py <InputDirectory> <OutputFileName> <Iterations> <Partitions>
  $ spark-submit AmazonALS.py /user/AZ_P/input outputAmazonReco.dat 10 4
- **Step7**: Một file outputAmazonReco.dat được tạo ra trong thư mục hiện tại với các cột:

  userID, recommendedProduct, predictedRating

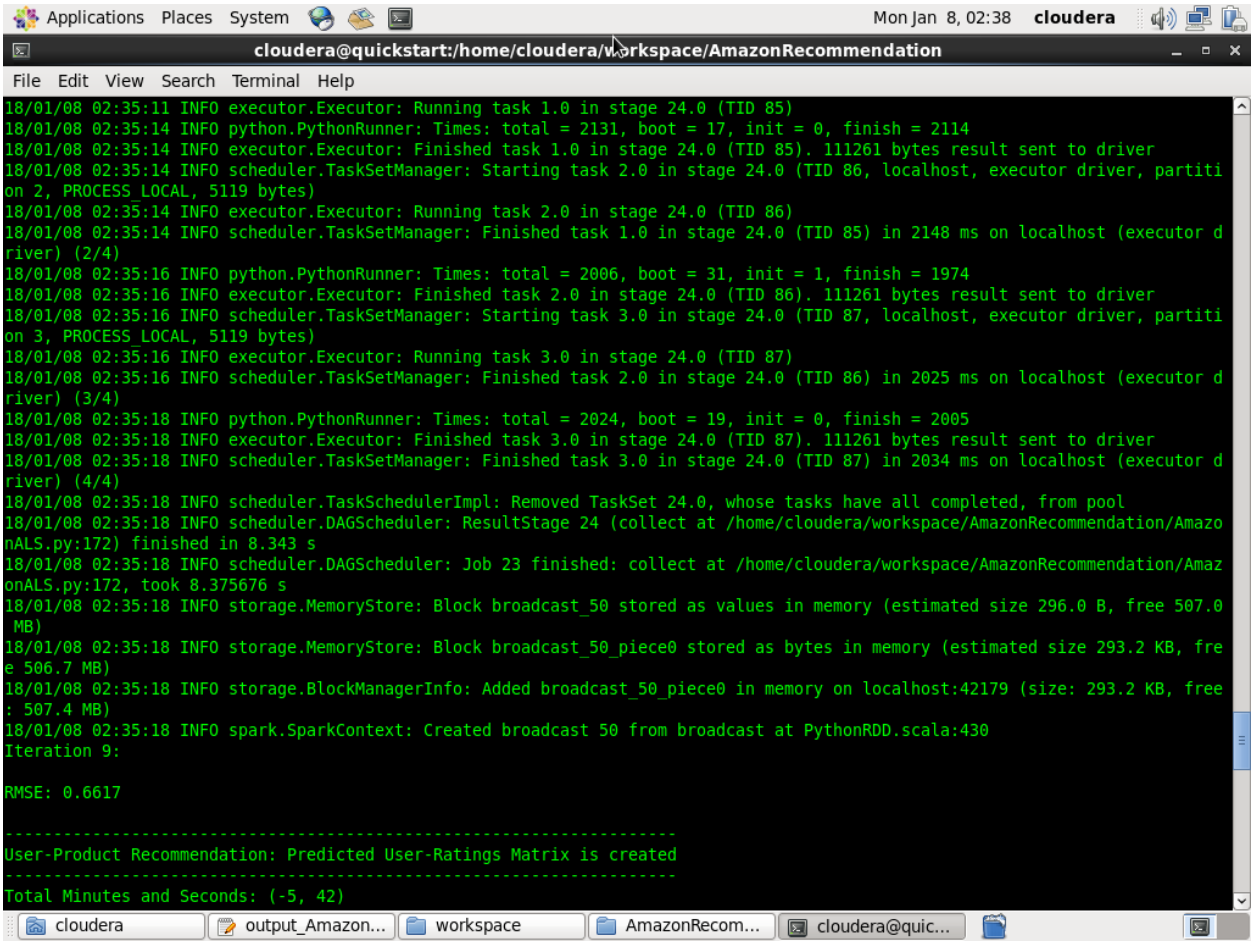
# 3. Kết quả

Trong thời gian chạy chương trình, giá trị RMSE cho mỗi lần lặp được hiển thị và dữ liệu gợi ý được ghi vào một tệp với 3 cột: userID, recommendedProduct, predictedRating.

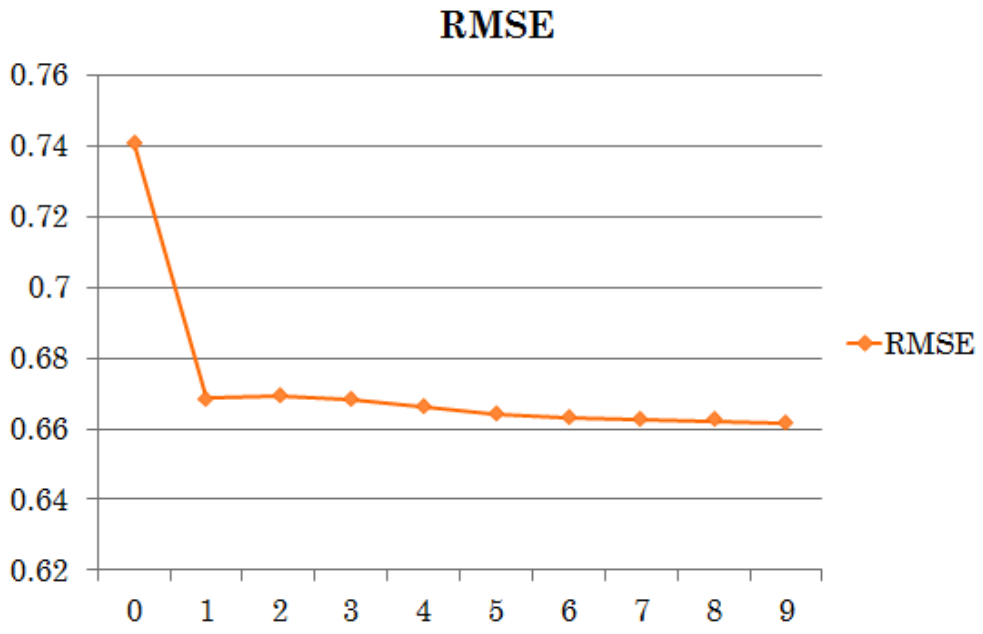
Kết quả khi run :

$ spark-submit AmazonALS.py /user/AZ_P/input outputAmazonReco.dat 10 4

Giá trị RMSE giảm dần từ lần lặp lại thứ nhất đến lần lặp lại 20. Chỉ thấy một phút giảm trong giá trị từ lần lặp lại thứ 15 và độ bão hòa của người quan sát ở mức 20.



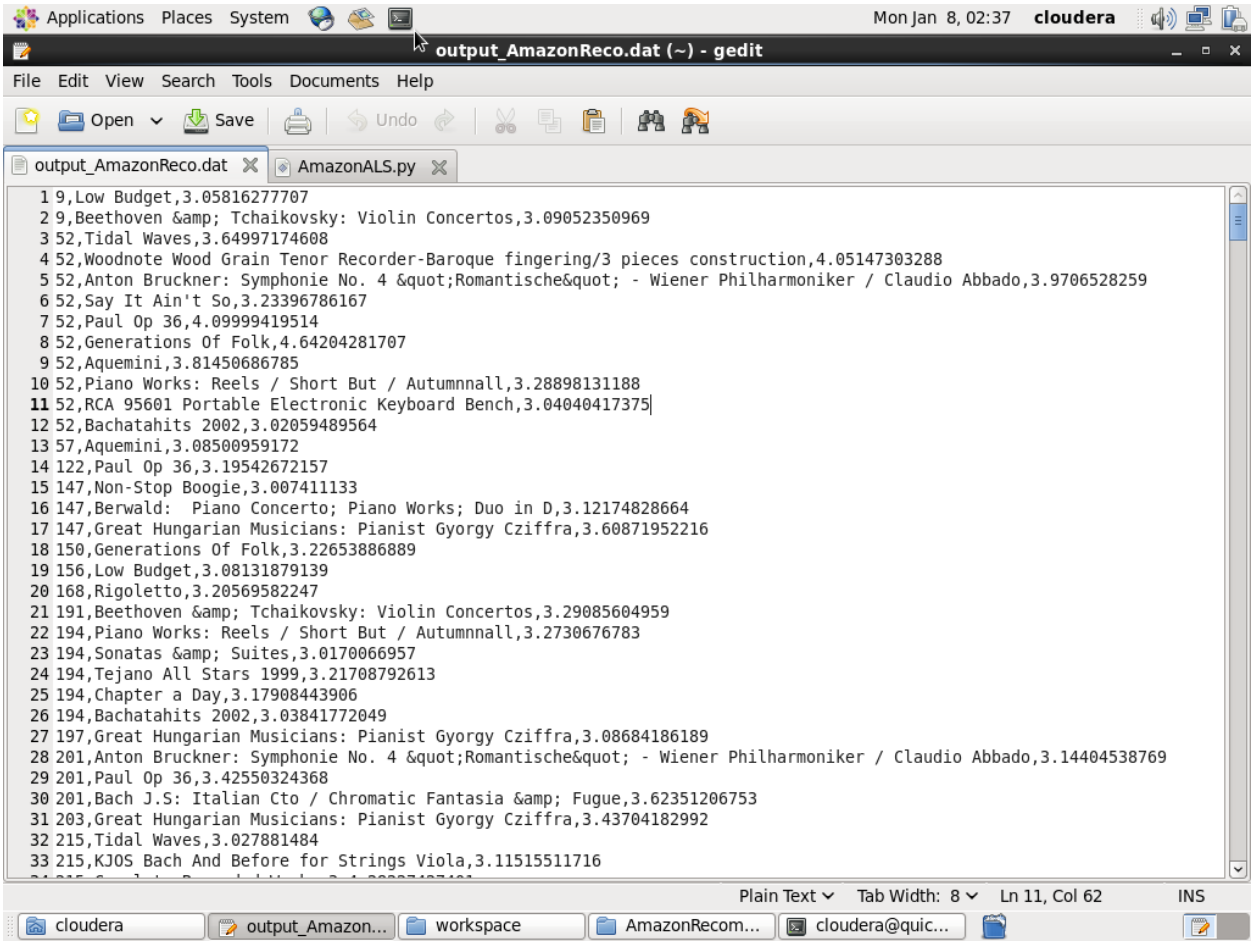

Một giá trị RMSE được cải thiện được quan sát với thông số định chuẩn 0,01; 0,001



| Loop | RMSE |
|---|---|
| 0 | 0.7410 |
| 1 | 0.6686 |
| 2 | 0.6693 |
| 3 | 0.6684 |
| 4 | 0.6663 |
| 5 | 0.6643 |
| 6 | 0.6633 |
| 7 | 0.6627 |
| 8 | 0.6622 |
| 9 | 0.6617 |

Thời gian chạy: 5 phút 42 giây.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$




**Hình 6: output_AmazonReco.dat < userID>< recommendedProduct>< predictedRating>**

## 4. Tài liệu tham khảo


1) https://machinelearningcoban.com/

2) https://math.stackexchange.com/questions/1072451/analytic-solution-for-matrix-factorizationusing-alternating-least-squares

3) http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-andimplementation-in-python/

4) https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-basedrecommender-systems/

5) https://datascience.stackexchange.com/questions/2598/item-based-and-user-basedrecommendation-difference-in-mahout

6) http://jmcauley.ucsd.edu/data/amazon/links.html

7) https://github.com/Mendeley/mrec/blob/master/mrec/mf/wrmf.py