|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  **BÁO CÁO BÀI TẬP LỚN**  **KỸ THUẬT LẬP TRÌNH**  **Người hướng dẫn: Nguyễn Đức Minh**  **Sinh viên thực hiện: Ngô Xuân Lộc - 20192979**  Hà Nội, 10-2021 |

**MỤC LỤC**

[CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI 2](#_Toc97136054)

[1.1 Tổng quan về Recommendation System 2](#_Toc97136055)

[1.2 Neighborhood-Based Collaborative Filtering: 2](#_Toc97136056)

[1.3 Những biến thể của Neighborhood-Based Collaborative Filtering 3](#_Toc97136057)

[CHƯƠNG 2. User – User Collaborative Filtering 3](#_Toc97136058)

[2.1 Similarity Funcitons 4](#_Toc97136059)

[2.1.1 Chuẩn hóa dữ liệu: 4](#_Toc97136060)

[2.1.2 Cosine Similarity 5](#_Toc97136061)

[2.2 Rating Prediction: 5](#_Toc97136062)

[CHƯƠNG 3. Code C ++ và QT Creator 7](#_Toc97136063)

[3.1 Dữ liệu được sử dụng 7](#_Toc97136064)

[3.2 Tạo class 8](#_Toc97136065)

[3.3 Normalize data 10](#_Toc97136066)

[3.4 Similarity data 11](#_Toc97136067)

[3.5 Rating predict 12](#_Toc97136068)

[3.6 Recommend 14](#_Toc97136069)

[3.7 Evaluation chất lượng mô hình trên tập test set 15](#_Toc97136070)

[CHƯƠNG 4. NHỮNG KỸ THUẬT DÙNG TRONG OOP C++ 15](#_Toc97136071)

# GIỚI THIỆU ĐỀ TÀI

## Tổng quan về Recommendation System

Ảnh có chứa văn bản

Mô tả được tạo tự động

Figure 1: Introduction

Với các hệ thống Client – User hiện nay như Youtube, Facebook, Shoppe, … Mọi hệ thống đều luôn muốn gợi ý, đem lại những thông tin, sản phẩm mà User có thể quan tâm nhất. Từ đó có thể tăng trải nghiệm của người dùng, cũng như cải thiện doanh thu cho nhà cung cấp. Vậy nên Recommendation System được ra đời.

Thuật toán được dùng trong Recommendation System có nhiều loại, nhưng mục tiêu của nó thì chỉ có một, đó là: dựa vào những thông tin đã có của User, có thể recommend những sản phẩm, thông tin khác mà User đó mong muốn.

Ví dụ, trong 1 User là 1 người hay đọc sách về khoa học, vậy từ đó hệ thống có thể đưa ra dự đoán và recommend những sản phẩm liên quan đến khoa học cho người đó.

## Neighborhood-Based Collaborative Filtering:

Hiện nay có rất nhiều thuật toán được áp dụng vào các hệ thống Recommendation System. Có thể kể đến như thuật toán Content-based Recommendation Systems – hệ thống này dựa trên data có sẵn của 1 User và từ đó đưa ra dự đoán cho người đó. Ví dụ người đó thích truyện tranh hoạt hình, từ đó hệ thống sẽ gợi ý cho User đó những loại truyện tranh hoạt hình có khả năng người đó sẽ thích.

Có 1 thuật toán khá hay và nó được áp dụng tối ưu hàm lồi đó là Matrix Factorization Collaborative Filtering. Thuật toán này có kết quả khá tốt vì nó sẽ tối ưu thuật toán thông qua loss function.

Ở đây chúng ta sẽ đề cập đến thuật toán Neighborhood-Based Collaborative Filtering. Đây là 1 thuật toán có sự cải tiến của Content-based Recommend Systems và tốc độ nhanh hơn Matrix Factorization. Vốn dĩ nó có sự cải tiến hơn Content-based Recommenda Systems vì Content-based chỉ dựa trên thông tin có sẵn của 1 user duy nhất và đưa ra dự đoán cho User đó. Còn Neighborhood-Based sẽ tận dụng được thông tin của các User có những thông tin liên quan.

Ví dụ, User 1 thích xem phim liên quan đến hình sự và người đó đã xem những phim như là Người phán sử, Hồ sơ cá sấu. User 2 cũng thích xem phim hình sự và người đó đã xem phim Người phán sử, Hồ sơ cá sấu, Chạy án, … Vậy khả năng User 1 cũng thích phim chạy án, từ đó sẽ recommend phim chạy án cho User 1.

## Những biến thể của Neighborhood-Based Collaborative Filtering

Hiện nay có 2 biến thể của thuật toán này đó là User – User Collaborative Filtering. Đó là recommend cho user dựa trên những user có thông tin tương tự.

Biến thể thứ 2 đó là Item – Item Collaborative Filtering. Thuật toán là sự đảo ngược của User – User Collaborative Filtering. Recommend cho item dựa trên những item có thông tin tương tự.

# User – User Collaborative Filtering

Thuật toán này được xây dựng dựa trên Similarity Functions:

Ảnh có chứa bàn

Mô tả được tạo tự động

Figure : Similarity Functions

Ta giả sự độ thích thú và quan tâm của Users đó là các sao đánh giá (có giá trị từ 1 đến 5). Figure 1 cho ta thấy. có 7 users từ 0 đến 6. Và có 5 items từ 0 đến 4. Các items đã được các users đánh giá sao. Dấu hỏi chấm thể hiện việc user đó chưa biết đến item đó hoặc chưa đánh giá sao.

Công việc trong thuật toán này đó là tìm các dấu hỏi chấm đó. Để giải quyết vấn đề đó thì ta dùng similarity functions.

## Similarity Funcitons

### Chuẩn hóa dữ liệu:

Ảnh có chứa bàn

Mô tả được tạo tự động

Figure : Dữ liệu ban đầu

Để phục vụ sự tối ưu của thuật toán ta cần chuẩn hóa dữ liệu và đánh giá trị cho các ô có dấu hỏi chấm.

Ảnh có chứa bàn

Mô tả được tạo tự động

Figure : Dữ liệu đã được chuẩn hóa

Để chuẩn hóa dữ liệu ta sẽ trừ đi cho mỗi cột (mỗi user) đi giá trị trung bình của mỗi cột. Sau đó ta sẽ có được giá trị dương thể hiện cho user đó thích sản phẩm, và giá trị âm thể hiện user đó không thích sản phẩm. Ta có thể đánh giá trị 0 cho những ô có giá trị là hỏi chấm. Vì giá trị 0 thể hiện sự trung lập của sản phẩm đó đối với user đó.

### Cosine Similarity

Cosine là hàm được sử dụng hiều nhất để tạo ra Similarity Matrix giữa các user

Ảnh có chứa bàn

Mô tả được tạo tự động

Figure : Similarity Matrix

Ta có công thức cosine\_similarity là (giả sử xem độ giống nhau giữa u0 và u1):

Ta thấy u0 và u1 có mức độ giống nhau cao, u2 giống u4, u5, u6.

## Rating Prediction:

Từ Similarity Matrix ta cần tìm ra các giá trị để điền vào các giá trị 0 ở ma trận đã được chuẩn hóa.

Ta có các bước để predict rating như sau: giả sử ta tìm predict rating của user1 cho item1

* Bước 1: chọn k user có nearest neighborhood gần nhất với u1. giả sử chọn k = 2. Ta có những người ra rated i1 là u0, u3, u5 tương ứng với độ giống nhau là 0.83, -0.40, -0.23. Từ ta thấy k user gần với u1 nhất là u0 u5 ứng với 0.83, -0.23
* Bước 2: Ta có u0 và u5 đã rated cho i1 là 0.75 và 0.5.
* Bước 3: Đưa ra dự đoán:

* Bước 4: làm tương tự với các user và item khác ta sẽ điền được hết giá trị còn thiếu trong matrix

Ảnh có chứa bàn

Mô tả được tạo tự động

Figure : Matrix chuẩn hóa đã được predict

* Bước 5: Sau đó ta cộng các giá trị từng cột với giá trị trung bình đã tính được từ trước. Từ đó ta sẽ được Matrix đã được predict chưa chuẩn hóa.

Ảnh có chứa bàn

Mô tả được tạo tự động

Figure : Matrix đã được predict chưa chuẩn hóa

# Code C ++ và QT Creator

## Dữ liệu được sử dụng

Dữ liệu được lấy từ MovieLens dataset của 56 users và 1089 items. Training set gồm 5000 mẫu được đọc từ file some\_handle\_train\_data. Test set gồm 5000 mẫu được đọc từ file some\_handle\_test\_data.

Mỗi mẫu có 4 giá trị là id user, id item, rating, timestamp.

// Read data from file url and return data

MatrixXf getData(string fileName, int heightSize, int widthSize) {

MatrixXf data(heightSize, widthSize);

string myText;

ifstream MyReadFile(fileName);

int heightCount = 0;

while (getline(MyReadFile, myText)) {

string s = myText;

string delim = " ";

int widthCount = 0;

auto start = 0U;

auto end = s.find(delim);

while (end != string::npos)

{

int value = stoi(s.substr(start, end - start));

//cout << value << endl;

data(heightCount, widthCount) = value;

//data[heightCount][widthCount] = value;

widthCount++;

start = end + delim.length();

end = s.find(delim, start);

}

int value = stoi(s.substr(start, end));

data(heightCount, widthCount) = value;

//data[heightCount][widthCount] = value;

widthCount++;

heightCount++;

}

MyReadFile.close();

return data;

}

// Initial Train, Test set

MatrixXf Y\_train\_data(N, M);

MatrixXf Y\_test\_data(H, M);

Y\_train\_data = getData("some\_handled\_train\_data.txt", N, M);

Y\_test\_data = getData("some\_handled\_test\_data.txt", H, M);

## Tạo class

class RecommendSystem {

protected:

float Y\_data[N][M]; // original data .shape NxM

float Y\_data\_T[M][N]; // original data transpose .shape MxN

float Ybar\_data[N][M]; // normalize data. shape NxM

float ratingThreshold; // rating threshold to recommend

int n\_users; // number of users

int n\_items; // number of items

int K; // k neighborhood

int uuFC; // type model

vector<float> mu; // average of user columns

MatrixXd Ybar; // normalize data. shape n\_items x n\_users

MatrixXd S; // similarity data

class UserUserRecommendSystem :public RecommendSystem {

protected:

public:

UserUserRecommendSystem(MatrixXf y\_data, int k, float rating = 0) :RecommendSystem(y\_data, k) {

// Y\_data

for (int i = 0; i < N; i++) {

for (int j = 0; j < M; j++) {

Y\_data[i][j] = y\_data(i, j);

}

}

// Y\_data\_T

int i, j;

for (i = 0; i < M; i++)

for (j = 0; j < N; j++)

Y\_data\_T[i][j] = Y\_data[j][i];

// n\_users, n\_items

n\_users = largest(Y\_data\_T[0], N) + 1;

n\_items = largest(Y\_data\_T[1], N) + 1;

cout << n\_users << endl;

cout << n\_items << endl;

uuFC = 1;

ratingThreshold = rating;

}

void displaySimilarityMatrix();

void displayNormalizeMatrix();

void displayInfo();

};

class ItemItemRecommendSystem :public RecommendSystem {

public:

ItemItemRecommendSystem(MatrixXf y\_data, int k, float rating = 0) :RecommendSystem(y\_data, k) {

// Y\_data

for (int i = 0; i < N; i++) {

for (int j = 0; j < M; j++) {

if (j == 0) Y\_data[i][j] = y\_data(i, 1);

else if (j == 1) Y\_data[i][j] = y\_data(i, 0);

else Y\_data[i][j] = y\_data(i, j);

}

}

// Y\_data\_T

int i, j;

for (i = 0; i < M; i++)

for (j = 0; j < N; j++)

Y\_data\_T[i][j] = Y\_data[j][i];

// n\_users, n\_items

n\_users = largest(Y\_data\_T[0], N) + 1;

n\_items = largest(Y\_data\_T[1], N) + 1;

uuFC = 0;

ratingThreshold = rating;

}

void displaySimilarityMatrix();

void displayNormalizeMatrix();

void displayInfo();

};

## Normalize data

void RecommendSystem::normalize\_Y() {

float users[N];

assignArray(users, Y\_data\_T[0], N);

assignMatrix(Ybar\_data, Y\_data);

mu = createZerosVector(mu, n\_users);

for (int n = 0; n < n\_users; n++) {

// Ids

vector< int > ids;

for (int i = 0; i < N; i++)

if (users[i] == n) ids.push\_back(i);

// item\_ids

vector< int > item\_ids;

for (int i = 0; i < ids.size(); i++) {

item\_ids.push\_back(Y\_data\_T[1][ids[i]]);

}

// ratings

vector< float > ratings;

for (int i = 0; i < ids.size(); i++) {

ratings.push\_back(Y\_data\_T[2][ids[i]]);

}

// take mean ratings

float m = average(ratings);

mu[n] = m;

// Ybar\_data

for (int i = 0; i < ids.size(); i++) {

Ybar\_data[ids[i]][2] = ratings[i] - mu[n];

}

}

displayMatrix(Ybar\_data);

MatrixXd tmp = MatrixXd::Zero(n\_items, n\_users);

for (int i = 0; i < N; i++) {

tmp(int(Y\_data[i][1]), int(Y\_data[i][0])) = Ybar\_data[i][2];

}

Ybar = tmp;

cout << Ybar << endl;

}

Sau khi normalize data ta được Ybar có số chiều là n\_items x n\_users

## Similarity data

void RecommendSystem::similarity() {

MatrixXd S\_Ref(n\_users, n\_users);

MatrixXd Ybar\_T = Ybar.transpose();

for (int i = 0; i < n\_users; i++) {

int tmp = i \* 100 / n\_users;

show\_percent(tmp);

cout << string(width + 6, '\b');

for (int j = 0; j < n\_users; j++) {

vector<float> vec1;

vector<float> vec2;

for (int k = 0; k < n\_items; k++) {

vec1.push\_back(Ybar\_T(i, k));

vec2.push\_back(Ybar\_T(j, k));

}

S\_Ref(i, j) = cosine\_similarity(vec1, vec2);

}

}

show\_percent(100);

cout << string(width + 6, '\b');

S = S\_Ref;

}

Sau khi similarity ta sẽ được matrix S có số chiều là n\_users x n\_users

## Rating predict

float RecommendSystem::pred(int u, int i, int normalized) {

if (uuFC) return \_\_pred(u, i, normalized);

return \_\_pred(i, u, normalized);

}

float RecommendSystem::\_\_pred(int u, int i, int normalized) {

// Ids

vector<float> ids;

for (int k = 0; k < N; k++)

if (int(Y\_data[k][1]) == int(i)) ids.push\_back(k);

//users\_rated\_i

vector<float > users\_rated\_i;

for (int k = 0; k < ids.size(); k++) {

users\_rated\_i.push\_back(Y\_data[int(ids[k])][0]);

}

// sim

vector<float> sim;

for (int k = 0; k < users\_rated\_i.size(); k++) {

sim.push\_back(S(u, int(users\_rated\_i[k])));

}

// a

vector<float> a = slicing(argsort(sim), K);

// nearest\_s

vector<float> nearest\_s;

for (int k = 0; k < a.size(); k++) {

nearest\_s.push\_back(sim[int(a[k])]);

}

//////////////

vector<float> r;

for (int k = 0; k < a.size(); k++) {

int index = users\_rated\_i[int(a[k])];

r.push\_back(Ybar(i, index));

}

// Return result

float result;

if (normalized == 1) {

result = multiplyVectors(r, nearest\_s) / (addVector(nearest\_s) + 1e-8);

}

else {

result = multiplyVectors(r, nearest\_s) / (addVector(nearest\_s) + 1e-8) + mu[u];

}

return result;

}

Hàm pred có 1 tham số là normalized. Normalized bằng 1 tương ứng với việc dự đoán và trả lại giá trị đã được normalize. Nếu Normalized bằng 0 tương ứng với việc dự đoán trả lại giá trị đưa normalize.

## Recommend

vector<float> RecommendSystem::recommend(int u) {

vector<float> recommended\_items;

// Ids

vector< float > ids;

for (int i = 0; i < N; i++)

if (int(Y\_data[i][0]) == int(u)) ids.push\_back(i);

//Items\_rated\_by\_u

vector< float > items\_rated\_by\_u;

for (int i = 0; i < ids.size(); i++) {

items\_rated\_by\_u.push\_back(Y\_data[int(ids[i])][1]);

}

//find predict rating and recommended

for (int i = 0; i < n\_items; i++) {

if (!isInVector(items\_rated\_by\_u, i)) {

float rating = \_\_pred(u, i);

if (rating > ratingThreshold) recommended\_items.push\_back(i);

}

}

return recommended\_items;

}

void RecommendSystem::print\_recommendation() {

cout << "Recommendation: " << endl;

for (int u = 0; u < n\_users; u++) {

vector<float> recommended\_items = recommend(u);

if (recommended\_items.size() > 0) {

if (uuFC == 1) {

cout << " \t\tRecommend item(s):";

for (int i = 0; i < recommended\_items.size(); i++) cout << recommended\_items[i] << " ";

cout << "for user " << u << endl;

}

if (uuFC == 0) {

cout << " \t\tRecommend item " << u << " ";

cout << "to user(s) : ";

for (int i = 0; i < recommended\_items.size(); i++) cout << recommended\_items[i] << " ";

cout << endl;

}

}

}

}

Sau khi đã có hàm pred ta sẽ recommend những cho user những sản phẩm tiềm năng.

## Evaluation chất lượng mô hình trên tập test set

// Predict on testset

float SE = 0;

for (int n = 0; n < H; n++) {

float pred = uuRS.pred(Y\_test\_data(n, 0), Y\_test\_data(n, 1), 0);

SE += pow(pred - Y\_test\_data(n, 2), 2);

}

float RMSE = sqrt(SE / H);

cout << "User-user RecommendSystem, RMSE = " << RMSE;

# NHỮNG KỸ THUẬT DÙNG TRONG OOP C++

Những kỹ thuật trong OOP C++ đó là

* Tính trừu tượng
* Tính đóng gói
* Tính thừa kế
* Hàm chồng

Ngoài ra về GUI, sử dụng QT creator:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Figure 8: Qt Creator