**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Logo

Description automatically generated**

**BÁO CÁO LẬP TRÌNH TÍNH TOÁN**

**ĐỀ TÀI:**

**QUÁ TRÌNH TÌNH TOÁN SỬ DỤNG EIGENFACES ĐỂ PHÂN TÍCH TRỰC TIẾP CÁC THÀNH PHẦN PCA**

Sinh viên thực hiện**:**

1. **Dương Văn Chí Bảo Mã SV: 102230334**
2. **Võ Hoàng Thái Bảo Mã SV: 102230335**

Người hướng dẫn**: PGS. TS. Nguyễn Tấn Khôi**

**NHÓM: 7**

*Đà Nẵng, 04/2024*

MỤC LỤC

[MỤC LỤC 1](#_Toc168611615)

[DANH MỤC HÌNH VẼ 2](#_Toc168611616)

[MỞ ĐẦU 3](#_Toc168611617)

[Chương 1. TỔNG QUAN ĐỀ TÀI 6](#_Toc168611618)

[Chương 2. CƠ SỞ LÝ THUYẾT 8](#_Toc168611619)

[2.1 Ý tưởng 8](#_Toc168611620)

[2.2 Cơ sở lý thuyết 9](#_Toc168611621)

[2.2.1 Một số khái niệm toán học 9](#_Toc168611622)

[2.3 Xây dựng chương trình 10](#_Toc168611623)

[2.3.1 Phát biểu biểu bài toán 10](#_Toc168611624)

[2.3.2 Các thuật toán 10](#_Toc168611625)

[Chương 3. CHƯƠNG TRÌNH VÀ KẾT QUẢ THỰC NGHIỆM 17](#_Toc168611626)

[3.1 Giao diện chương trình 17](#_Toc168611627)

[3.2 Kết quả chương trình 18](#_Toc168611628)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 19](#_Toc168611629)

[TÀI LIỆU THAM KHẢO 21](#_Toc168611630)

[PHỤ LỤC 22](#_Toc168611631)

DANH MỤC HÌNH VẼ

Hình 2.1: Minh họa về giảm số chiều 6

Hình 2.2: Minh họa về thuật toán PCA 6

Hình 2.3: Thay đổi hệ tọa độ trong PCA 6

Hình 3.1a: Giao diện chương trình 14

Hình 3.1b: Giao diện chương trình 14

Hình 3.2a: Giao diện kết quả của chương trình 15

Hình 3.2b: Giao diện kết quả của chương trình 15

MỞ ĐẦU

Trước tiên, nhóm chúng tôi xin gửi lời cảm ơn chân thành đến quý thầy cô trong khoa Công nghệ Thông tin, trong đó có **PGS.TS. Nguyễn Tấn Khôi** đã truyền đạt kiến thức và hỗ trợ tận tình trong suốt quá trình học tập và nghiên cứu đề tài này của chúng tôi. Sự hướng dẫn và động viên của các thầy cô đã đóng góp một phần không nhỏ vào sự thành công của đề tài này.

Nhóm chúng tôi cũng xin chân thành cảm ơn đến các bạn sinh viên và bạn bè đã chia sẻ kinh nghiệm, tài liệu và cùng nhau thảo luận, giúp nhóm chúng tôi có thêm nhiều góc nhìn mới và giải quyết những khó khăn gặp phải trong quá trình thực hiện nghiên cứu.

Trong lĩnh vực nhận dạng và phân tích hình ảnh, các kỹ thuật xử lý ảnh tiên tiến đã đóng một vai trò quan trọng trong việc cải thiện độ chính xác và hiệu quả của các hệ thống nhận diện. Một trong những phương pháp nổi bật và được ứng dụng rộng rãi là sử dụng Eigenfaces để phân tích các thành phần chính (Principal Component Analysis - PCA). Eigenfaces, được phát triển dựa trên lý thuyết PCA, cho phép trích xuất các đặc trưng quan trọng từ tập hợp các ảnh, từ đó giúp phân loại và nhận diện đối tượng một cách hiệu quả.

Trong đề tài này, chúng ta sẽ đi sâu vào quá trình tính toán sử dụng Eigenfaces để phân tích trực tiếp các thành phần PCA. Quá trình này không chỉ giúp giảm thiểu dữ liệu cần xử lý mà còn nâng cao khả năng phân biệt và nhận diện các đối tượng khác nhau trong ảnh. Bằng cách chuyển đổi không gian ảnh ban đầu sang không gian Eigenfaces, chúng ta có thể tập trung vào những đặc điểm quan trọng nhất của dữ liệu, loại bỏ nhiễu và tăng cường hiệu suất của các thuật toán nhận dạng.

Mục tiêu của đề tài là cung cấp một cái nhìn tổng quan về lý thuyết và ứng dụng của Eigenfaces trong phân tích PCA, trình bày các bước cụ thể trong quá trình tính toán và minh họa hiệu quả của phương pháp này thông qua các thí nghiệm và ví dụ thực tế. Hy vọng rằng, thông qua nghiên cứu này, chúng ta sẽ hiểu rõ hơn về tầm quan trọng của Eigenfaces trong lĩnh vực nhận dạng ảnh và các ứng dụng tiềm năng của nó trong nhiều lĩnh vực khác nhau.

**1. Mục đích và mục tiêu thực hiện đề tài :**

1.1. Mục đích :

* Tìm hiểu và xây dựng thuật toán phân tích trực tiếp các thành phần PCA (Principal Component Analysis) sử dụng Eigenfaces.
* Đánh giá hiệu suất của thuật toán mới trên các tập dữ liệu hình ảnh khuôn mặt thực tế.
* So sánh hiệu suất của thuật toán mới với các phương pháp phân tích PCA truyền thống.
* Khẳng định tính ứng dụng của thuật toán mới trong lĩnh vực nhận dạng khuôn mặt và xử lý ảnh.

1.2. Mục tiêu:

* Hiểu rõ nguyên lý hoạt động của Eigenfaces và PCA trong công việc phân tích hình ảnh.
* Nghiên cứu các phương pháp phân tích PCA truyền thống và các hạn chế của chúng.
* Phát triển thuật toán mới để phân tích trực tiếp các thành phần PCA sử dụng Eigenfaces.
* Thực hiện mô phỏng và đánh giá hiệu suất của thuật toán mới trên các tập dữ liệu hình ảnh khuôn mặt.
* So sánh hiệu suất của thuật toán mới với các phương pháp phân tích PCA truyền thống.
* Phân tích ưu và nhược điểm của thuật toán mới.
* Đề xuất các ứng dụng tiềm năng của thuật toán mới trong lĩnh vực nhận dạng khuôn mặt và xử lý ảnh.

**2. Đối tượng nghiên cứu:**

* Các nhà nghiên cứu lĩnh vực học máy, xử lý ảnh và nhận dạng khuôn mặt.
* Các nhà phát triển phần mềm và ứng dụng liên quan đến xử lý ảnh và nhận dạng khuôn mặt.
* Sinh viên và học viên quan tâm đến lĩnh vực học máy và xử lý ảnh.

**3. Phương pháp nghiên cứu:**

* Nghiên cứu về cơ sở lý thuyết của phương pháp rút gọn chiều dữ liệu PCA.
* Lựa chọn ngôn ngữ lập trình C++ để thực hiện chương trình
* Thực hiện chương trình trên bộ dữ liệu.
* Đánh giá, kiểm tra kết quả.
* Công bố kết quả thông qua chương trình, bản báo cáo và bài trình chiếu.

**4. Cấu trúc của báo cáo môn học:** Bố cục của báo cáo gồm 3 chương:

Chương 1: Tổng quan đề tài.

Chương 2: Cơ sở lý thuyết.

Chương 3: Chương trình và kết quả thực nghiệm.

# TỔNG QUAN ĐỀ TÀI

#### Giới thiệu chung

Trong thời đại kỹ thuật số hiện nay, công nghệ nhận dạng khuôn mặt đã trở thành một phần không thể thiếu trong nhiều ứng dụng thực tế, từ bảo mật, giám sát đến các tiện ích cá nhân hóa. Một trong những phương pháp hiệu quả và được nghiên cứu rộng rãi trong nhận dạng khuôn mặt là **Eigenfaces**, một ứng dụng của phân tích thành phần chính (Principal Component Analysis - PCA).

#### PCA là gì?

PCA là phương pháp đơn giản nhất dùng để phân tích đa biến dựa trên các vector đặc trưng. Là một thuật toán được sử dụng để tạo ra một ảnh mới từ ảnh ban đầu.

Phương pháp PCA cung cấp cho chúng ta một bức ảnh ít chiều, một cái bóng của vật thể khi quan sát từ chính những đặc trưng cơ bản nhất của vật thể đó.

Mục tiêu của phương pháp PCA là thực hiện giảm số chiều nhưng vẫn đảm bảo tối đa sự phân tán dữ liệu.

Bản chất của PCA là tìm ra một không gian mới theo hướng biến thiên mạnh nhất của một tập hợp các vector trong không gian cho trước.

#### Eigenfaces là gì?

Eigenfaces là một phương pháp dựa trên PCA để nhận diện và phân tích khuôn mặt. PCA là một kỹ thuật thống kê sử dụng để giảm thiểu chiều dữ liệu, tập trung vào việc tìm ra các hướng quan trọng nhất (các thành phần chính) trong không gian dữ liệu. Trong ngữ cảnh của nhận diện khuôn mặt, PCA giúp chuyển đổi không gian ảnh ban đầu thành một không gian có số chiều thấp hơn nhưng vẫn giữ được những thông tin quan trọng nhất.

Eigenfaces được tạo ra bằng cách tính toán các trị riêng và vector riêng từ ma trận hiệp phương sai của tập dữ liệu ảnh khuôn mặt. Các vector riêng này tạo thành một cơ sở mới, trong đó mỗi ảnh có thể được biểu diễn như một tổ hợp tuyến tính của các Eigenfaces. Điều này cho phép việc nhận diện và phân loại khuôn mặt trở nên hiệu quả hơn do chỉ cần so sánh trong không gian có chiều thấp hơn.

#### Quá trình tính toán

Thu thập và tiền xử lý dữ liệu: Tập hợp các ảnh khuôn mặt được thu thập và tiền xử lý để chuẩn hóa về kích thước và điều kiện ánh sáng.

Tính toán ma trận hiệp phương sai: Từ tập dữ liệu đã chuẩn hóa, ma trận hiệp phương sai được tính toán để xác định mối quan hệ giữa các điểm ảnh.

Tính toán các trị riêng và vector riêng: Các trị riêng và vector riêng của ma trận hiệp phương sai được tính toán. Các vector riêng này chính là các Eigenfaces.

Chiếu ảnh vào không gian Eigenfaces: Các ảnh khuôn mặt ban đầu chiếu vào không gian Eigenfaces để chuyển đổi chúng thành các vector có số chiều thấp hơn.

Nhận diện và phân loại: Dựa trên các vector trong không gian Eigenfaces, các thuật toán nhận diện và phân loại khuôn mặt được áp dụng để so sánh và nhận diện đối tượng.

#### Ứng dụng

**Giảm thiểu dữ liệu**: Kích thước dữ liệu giảm đáng kể mà vẫn giữ được thông tin quan trọng.

**Tăng hiệu quả xử lý**: Các thuật toán nhận diện hoạt động nhanh hơn và hiệu quả hơn trong không gian có số chiều thấp.

**Cải thiện độ chính xác**: Giảm thiểu nhiễu và các yếu tố không liên quan giúp cải thiện độ chính xác của hệ thống nhận diện.

# CƠ SỞ LÝ THUYẾT

## Ý tưởng

Mục tiêu của phương pháp PCA là “giảm số chiều” của 1 tập vector sao cho vẫn đảm bảo được “tối đa thông tin quan trọng nhất” phương pháp PCA sẽ giữ lại K thuộc tính “mới” từ M các thuộc tính ban đầu (K<M).

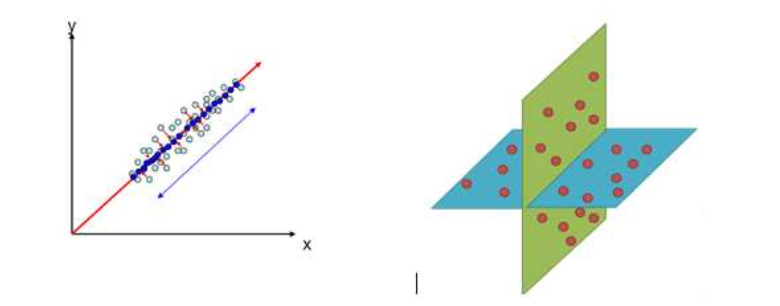
Vector y chỉ còn K chiều

Vector x ban đầu có N chiều

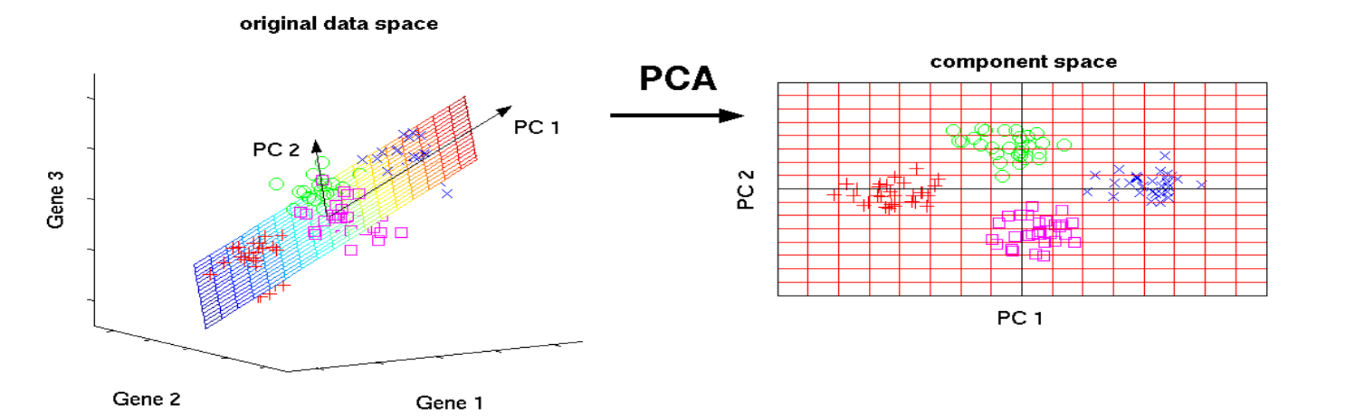




Hình 2.1: Minh họa về giảm số chiều



Hình 2.2: Minh họa về thuật toán PCA



Hình 2.3: Thay đổi hệ tọa độ trong PCA

## Cơ sở lý thuyết

### Một số khái niệm toán học

**Độ lệch chuẩn:**

Trong đó : n là số lượng phần tử trong tập hợp.

là ký hiệu trung bình của mẫu.

**Phương sai**: là bình phương độ lệch chuẩn

**Vector khuôn mặt trung bình:**

**Ma trận phương sai (Covariance) C:**

## Xây dựng chương trình

### Phát biểu biểu bài toán

Input: Một tập hợp dữ liệu hình ảnh dưới dạng một ma trận kích thước N×M bao gồm các vector dạng cột thể hiện hình ảnh, trong đó N là số chiều của vector và M là số lượng hình ảnh được nhập vào.

Output: Tập hợp dữ liệu hình ảnh dưới dạng một ma trận kích thước K×M bao gồm các vector dạng cột thể hiện hình ảnh, trong đó K là số chiều mới của vector và M là số lượng hình ảnh đã được nhập vào trước đó và hình ảnh tạo ra được từ ma trận kích thước K×M ở trên

### Các thuật toán

***Bước 1: Nhập dữ liệu***

* Ví dụ: Ta có 4 ảnh trong tập huấn luyện với kích thước mỗi ảnh là (9 x 1)

T1 = T2 = T3 = T4 =

***Bước 2: Tính vector trung bình:***

* Thuật toán

+ Lặp i từ 0 tới N – 1

Khởi tạo biến sum = 0

Lặp j từ 0 tới M - 1

Sum += dataMatrix[i, j]

avgVector[i, 0] = sum / M;

+ Trả về ma trận avgVector.

* Ví dụ:

= = = 171.5

= = = 176.5

….

Cuối cùng, ta sẽ thu được có kích thước (9 x 1) như sau:

=

* Mã nguồn minh họa:

MatrixXd averageVector(int N, int M, MatrixXd dataMatrix) {

MatrixXd avgVector(N, 1);

foru(i, 0, N - 1){

double sum = 0;

foru(j, 0, M - 1){

sum += dataMatrix(i, j);

}

avgVector(i, 0) = sum / M;

}

return avgVector;

}

***Bước 3: Trừ vector khuôn mặt trung bình:***

* Thuật toán:

+ Khởi tạo mảng N×M

+ Lặp i từ 0 tới N - 1

Lặp j từ 0 tới M -1

diffMatrix(i, j) = dataMatrix(i, j) - avgVector(i, 0);

+ Trả về ma trận N×M

* Ví dụ minh họa:

= T1 - = - =

= T2 - = - =

= T3 - = - =

= T4 - = - =

* Mã nguồn minh họa:

MatrixXd differenceMatrix(int N, int M, MatrixXd dataMatrix, MatrixXd avgVector) {

MatrixXd diffMatrix(N, M);

foru(i, 0, N - 1){

foru(j, 0, M - 1){

diffMatrix(i, j) = dataMatrix(i, j) - avgVector(i, 0);

}

}

return diffMatrix;

}

***Bước 4: Tính ma trận hiệp phương sai:***

* Thuật toán:

Nhân ma trận chuyển vị của các vector đã trừ với chính nó AT×A. Trong đó, A =

* Ví dụ minh họa:

A = AT×A = × =

* Mã nguồn minh họa:

tranMatrix = transposeMatrix(diffMatrix);

matrixProduct = multipleMatrix(tranMatrix, diffMatrix);

Trong đó:

***// Tích hai ma trận***

MatrixXd multipleMatrix(MatrixXd matrix1, MatrixXd matrix2) {

MatrixXd matrixProduct(matrix1.rows(), matrix2.cols());

foru(i, 0, matrix1.rows() - 1) {

foru(j, 0, matrix2.cols() - 1) {

double sum = 0;

foru(k, 0, matrix1.cols() - 1) {

sum += matrix1(i, k) \* matrix2(k, j);

}

matrixProduct(i, j) = sum;

}

}

return matrixProduct;

}

***//Ma trận chuyển vị***

MatrixXd transposeMatrix(MatrixXd matrix) {

MatrixXd transposedMatrix(matrix.cols(), matrix.rows());

foru(i, 0, matrix.rows() - 1) {

foru(j, 0, matrix.cols() - 1) {

transposedMatrix(j, i) = matrix(i, j);

}

}

return transposedMatrix;

}

***Bước 5: Tìm trị riêng và các vector riêng sử dụng thuật toán Jacobi:***

* Thuật toán:

Nhận ma trận vuông A, ma trận đầu ra cho giá trị riêng (eigenValues) và vectơ riêng (eigenVectors), số lượng lặp tối đa tùy chọn (maxIter), và độ dung sai tùy chọn (tolerance) làm đầu vào.

Khởi tạo eigenVectors thành ma trận đơn vị (MatrixXd::Identity(n, n)) để biểu thị cơ sở ban đầu.

Lặp lại đến maxIter lần hoặc cho đến khi hội tụ (dựa trên tolerance):

Tìm phần tử ngoài đường chéo lớn nhất A(p, q) trong A.

Kiểm tra xem phần tử ngoài đường chéo lớn nhất có nhỏ hơn độ dung sai (maxOffDiag < tolerance) hay không. Nếu có, đạt được hội tụ và vòng lặp kết thúc.

Tính toán góc xoay theta bằng hàm arctangent (std::atan2).

Tính toán cosin (c) và sin (s) của góc xoay.

Tạo ma trận xoay R dùng c và s để loại bỏ phần tử ngoài đường chéo A(p, q).

Cập nhật A bằng cách áp dụng phép xoay từ cả hai phía (R.transpose()\*A\* R).

Cập nhật eigenVectors bằng cách nhân nó với ma trận xoay (eigenVectors\*R).

Sau vòng lặp, trích xuất giá trị riêng từ đường chéo của ma trận A cuối cùng và lưu trữ chúng trong eigenValues.

* Ví dụ:

A =

Từ ma trận này, ta dễ dàng tìm ra được phần tử có trị tuyệt đối lớn nhất và nằm ngoài đường chéo đó là ở hàng 2, cột 3 => maxOffDiag = 50914, p = 2, q = 3

Xét thấy maxOffDiag < tolerance => Tiếp tục chương trình

Tiếp tục tính toán, ta được:

theta = 0.5 x atan2(2 x A[p, q], A[p, p] – A[q, q])

= 0.5 x atan2(2 x 50914, 82867 – 70771)

= 0.5 x atan2(101828, 12096)

= 0.5148

c = cos(theta)

= 0.8704

s = sin(theta)

= 0.4924

Ma trận R có giá trị như sau

R =

Cập nhật lại ma trận A

A = RT×A×R = ××

=

eigenVector = eigenVector x R = x =

Lặp lại cho tới khi các số ngoài đường chéo của mảng A gần tới 0 hoặc đã đạt tới giới hạn số lần lặp. Khi đó, ta thu được các phần tử trên đường chéo là các value, còn eigenVector là bảng các vector tương ứng với các trị riêng tìm được

Ma trận A thu được:

A =

eigenVector =

* Mã nguồn minh họa:

void jacobiEigen(MatrixXd A, MatrixXd &eigenValues, MatrixXd &eigenVectors, int maxIter = 100, double tolerance = 1e-8) {

int n = A.rows();

eigenVectors = MatrixXd::Identity(n, n

int iter = 0;

while (iter < maxIter) {

double maxOffDiag = 0.0;

int p, q;

for (int i = 0; i < n - 1; ++i) {

for (int j = i + 1; j < n; ++j) {

if (std::abs(A(i, j)) > maxOffDiag) {

maxOffDiag = std::abs(A(i, j));

p = i;

q = j;

}

}

}

if (maxOffDiag < tolerance) break;

double theta = 0.5 \* std::atan2(2 \* A(p, q), A(p, p) - A(q, q));

double c = std::cos(theta);

double s = std::sin(theta);

MatrixXd R = MatrixXd::Identity(n, n);

R(p, p) = c;

R(q, q) = c;

R(p, q) = -s;

R(q, p) = s;

A = R.transpose() \* A \* R;

eigenVectors = eigenVectors \* R;

iter++;

}

eigenValues.resize(n, 1);

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

eigenValues(i, 0) = A(i, i);

}

}

***Bước 6: Lọc ra các trị riêng và vector riêng phù hợp:***

* Thuật toán:

Sắp xếp theo thứ tự dãy giảm dần các eigenvalues tìm được.

Theo dõi sự biến thiên của dãy trên, khi không còn biến thiên (hoặc xấp xỉ bằng không) thì lúc đó ta đã chọn đủ K.

* Mã nguồn minh họa:

void filterEigen(MatrixXd& eigenValues, MatrixXd& eigenVectors) {

for (int i = 0; i < eigenValues.rows() - 1; ++i) {

for (int j = i + 1; j < eigenValues.rows(); ++j) {

if (eigenValues(i, 0) < eigenValues(j, 0)) {

double tempValue = eigenValues(i, 0);

eigenValues(i, 0) = eigenValues(j, 0);

eigenValues(j, 0) = tempValue;

MatrixXd tempVector = eigenVectors.col(i);

eigenVectors.col(i) = eigenVectors.col(j);

eigenVectors.col(j) = tempVector;

}

}

}

int index = 0;

MatrixXd tempEigenValues(eigenValues.rows(), 1);

MatrixXd tempEigenVectors(eigenVectors.rows(), eigenVectors.cols());

for (int i = 0; i < eigenValues.rows() - 1; ++i) {

if (fabs(eigenValues(i, 0) - 0.0) > ESP && fabs(eigenValues(i, 0) - eigenValues(i + 1, 0)) > ESP) {

tempEigenValues(index, 0) = eigenValues(i, 0);

tempEigenVectors.col(index) = eigenVectors.col(i);

index++;

}

}

eigenValues.resize(index, 1);

eigenVectors.resize(eigenVectors.rows(), index);

eigenValues = tempEigenValues.block(0, 0, index, 1);

eigenVectors = tempEigenVectors.block(0, 0, eigenVectors.rows(), index);

}

***Bước 7: Nhân ma trận matrixProduct với các vector riêng vừa tìm được:***

* Mã nguồn minh họa:

MatrixXd ansVector = multipleMatrix(diffMatrix, eigenVectors);

***Bước 8: Chuẩn hóa các vector kết quả vừa tìm được ở bước 6:***

* Thuật toán:
* Mã nguồn minh họa:

MatrixXd normalizeVector(MatrixXd ansVector) {

vector <double> normVector;

foru(i, 0, ansVector.cols() - 1) {

double norm = 0;

foru(j, 0, ansVector.rows() - 1) {

norm += ansVector(j, i) \* ansVector(j, i);

}

normVector.push\_back(sqrt(norm));

}

foru(i, 0, ansVector.cols() - 1) {

foru(j, 0, ansVector.rows() - 1) {

ansVector(j, i) /= normVector[i];

}

}

return ansVector;

}

***Bước 9: Xuất kết quả***

* Mã nguồn minh họa xuất ra ma trận K×M bằng ngôn ngữ C++:

printOutput(N, M, dataMatrix, avgVector, diffMatrix, matrixProduct, eigenValues, eigenVectors, ansVector);

* Mã nguồn minh họa tạo ra hình ảnh từ ma trận K×M ở trên bằng ngôn ngữ Python:

import numpy as np

from PIL import Image

from sklearn.decomposition import PCA

data = np.loadtxt('data.out')

pca = PCA(n\_components=2)

transformed\_data = pca.fit\_transform(data)

transformed\_data -= np.min(transformed\_data, axis=0)

transformed\_data /= np.max(transformed\_data, axis=0)

image\_data = np.zeros((100, 100, 3), dtype=np.uint8)

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

    x, y = transformed\_data[i] \* 99

    x, y = int(x), int(y)

    image\_data[y, x] = [255, 255, 255]

image = Image.fromarray(image\_data)

image.show()

image.save('generated\_image.png')

# CHƯƠNG TRÌNH VÀ KẾT QUẢ THỰC NGHIỆM

## Giao diện chương trình

A screenshot of a computer

Description automatically generated

Hình 3.1a: Giao diện chương trình

A screenshot of a computer

Description automatically generated

Hình 3.1b: Giao diện chương trình

## Kết quả chương trình

A screenshot of a computer

Description automatically generated

Hình 3.2a: Giao diện kết quả chương trình

A screenshot of a computer

Description automatically generated

Hình 3.2b: Giao diện kết quả chương trình

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

#### Kết luận

Eigenfaces là một kỹ thuật hiệu quả để giảm chiều dữ liệu và trích xuất các đặc trưng quan trọng từ ảnh khuôn mặt. Nó có nhiều ứng dụng trong xử lý ảnh và nhận dạng khuôn mặt.

Eigenfaces là một kỹ thuật mạnh mẽ với nhiều tiềm năng phát triển trong tương lai. Các nghiên cứu và phát triển tiếp theo sẽ tập trung vào việc cải thiện hiệu suất, tính chính xác, mở rộng ứng dụng và kết hợp với các phương pháp học máy khác để mang lại hiệu quả tối ưu trong các lĩnh vực xử lý ảnh, nhận dạng khuôn mặt và trí tuệ nhân tạo.

#### Hướng phát triển

##### Nâng cao hiệu suất:

Tích hợp các kỹ thuật học sâu: Kết hợp Eigenfaces với các mạng nơ-ron tích chập (CNN) để tăng cường khả năng nhận diện khuôn mặt. Việc sử dụng các mô hình học sâu có thể cải thiện đáng kể độ chính xác và khả năng khái quát hóa của hệ thống.

Tối ưu hóa thuật toán PCA: Áp dụng các phương pháp PCA nâng cao như Kernel PCA hoặc Sparse PCA để xử lý các tập dữ liệu phức tạp hơn và tăng cường khả năng phân biệt.

##### Mở rộng ứng dụng:

Ứng dụng trong giám sát an ninh: Phát triển các hệ thống nhận diện khuôn mặt dựa trên Eigenfaces để sử dụng trong các hệ thống giám sát và an ninh, giúp nhận diện nhanh chóng và chính xác các đối tượng.

Cá nhân hóa dịch vụ người dùng: Sử dụng công nghệ này trong các hệ thống cá nhân hóa dịch vụ, chẳng hạn như trong thương mại điện tử, dịch vụ y tế và giáo dục, nơi mà việc nhận diện và phân loại khuôn mặt có thể mang lại trải nghiệm người dùng tốt hơn.

##### Xử lý dữ liệu lớn

Phân tán và điện toán đám mây: Tích hợp công nghệ Eigenfaces vào các nền tảng điện toán đám mây và xử lý phân tán để xử lý các tập dữ liệu lớn và phức tạp, đồng thời đảm bảo khả năng mở rộng và tính linh hoạt của hệ thống.

Tối ưu hóa thời gian thực: Phát triển các thuật toán và hệ thống có khả năng xử lý và nhận diện khuôn mặt trong thời gian thực, đáp ứng yêu cầu của các ứng dụng như nhận diện trong video trực tiếp.

##### Nghiên cứu các biến thể của Eigenfaces:

Localized Eigenfaces: Phân chia ảnh thành các vùng nhỏ và tính toán Eigenface cho từng vùng để phân tích các đặc trưng chi tiết hơn.

Discriminant Eigenfaces: Tìm kiếm Eigenface tối ưu hóa khả năng phân biệt giữa các lớp trong tập dữ liệu.

Tensor Eigenfaces: Mở rộng Eigenfaces sang dữ liệu đa chiều, ví dụ như video có xuất hiện khuôn mặt.

##### Phát triển các giao diện và công cụ hỗ trợ

Xây dựng giao diện người dùng: Phát triển các giao diện người dùng thân thiện để dễ dàng tích hợp và sử dụng công nghệ Eigenfaces trong các ứng dụng thực tế.

Công cụ phân tích và đánh giá: Tạo ra các công cụ hỗ trợ việc phân tích và đánh giá hiệu suất của hệ thống, giúp người dùng có thể dễ dàng theo dõi và cải thiện chất lượng nhận diện.

TÀI LIỆU THAM KHẢO

Tài liệu lý thuyết:

<https://www.academia.edu/34931265/Eigenfaces_-_LE_HOANG_THANH_1_> [https://www.turing.com/kb/guide-to-principal-component-analysis](https://www.turing.com/kb/guide-to-principal-component-analysis%20) [https://www.gatevidyalay.com/tag/principal-component-analysis-numerical-example/](https://www.gatevidyalay.com/tag/principal-component-analysis-numerical-example/%20)

<https://machinelearningcoban.com/2017/06/15/pca/>

Code tham khảo:

<https://github.com/toosyou/PCA/tree/master>

# PHỤ LỤC

Link Github:

<https://github.com/CulesBao/PBL_DUT>

Chương trình đầy đủ bằng ngôn ngữ C++:

#include <bits/stdc++.h>

#include "./eigen-3.4.0/Eigen/Dense"

#include <windows.h>

using namespace Eigen;

using namespace std;

#define ESP 1e-3

#define endl '\n'

#define ll long long

#define fori(x, v) for (auto &x : v)

#define foru(i, a, b) for (int i = a; i <= b; i++)

#define ford(i, a, b) for (int i = a; i >= b; i--)

bool isPrintToConsole = true;

ifstream input;

ofstream output;

//Doc du lieu

void readData(int &N, int &M, MatrixXd &dataMatrix, int choose) {

if (choose == 1) {

input >> N >> M;

dataMatrix.resize(N, M);

foru(i, 0, N - 1) {

foru(j, 0, M - 1) input >> dataMatrix(i, j);

}

}

else{

cin >> N >> M;

dataMatrix.resize(N, M);

foru(i, 0, N - 1) {

foru(j, 0, M - 1) cin >> dataMatrix(i, j);

}

}

}

void welcome() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* DO AN LAP TRINH TINH TOAN 1 \*" << endl;

cout << "\* PHAN TICH TRUC TIEP CAC THANH PHAN PCA \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

cout << "Thong tin ve chuong trinh: " << endl;

cout << "- Giang vien huong dan: PGS.TS Nguyen Tan Khoi" << endl;

cout << "- Sinh vien thuc hien: Duong Van Chi Bao va Vo Hoang Thai Bao" << endl;

cout << "- Lop sinh hoat: 23T-DT4" << endl << endl;

}

void inputSeclection(int &N, int &M, MatrixXd &dataMatrix) {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* CHON DU LIEU TUY CHINH \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

cout << "Da co san 3 bo du lieu: " << endl;

cout << "1. FaceDataset.inp (Bo du lieu khuon mat tham khao tu Le Hoang Thanh)" << endl;

cout << "2. IrisDataset.inp (Bo du lieu phan tich cac loai hoa, day la bo du lieu huyen thoai trong phan tich PCA)" << endl;

cout << "2. GithubDataset.inp (du lieu tham khao tu Github cua toosyou)" << endl;

cout << "3. WikiDataset.inp (du lieu tham khao tu Wikipedia)" << endl;

cout << "4. SydneyDataset.inp (du lieu tham khao tu truong dai hoc Sydney thong qua Google Dataset Search Engine)" << endl << endl;

cout << "--------------------------------------------------------------------------------" << endl;

cout << "| QUY DINH CACH NHAP DU LIEU |" << endl;

cout << "| Dong dau tien: 2 so nguyen N va M. |" << endl;

cout << "| N dong tiep theo: Moi dong gom M , cac so cach nhau boi mot khoang trang |" << endl;

cout << "--------------------------------------------------------------------------------" << endl << endl;

cout << "Nhan 1 de chon viec nhap tu file" << endl;

cout << "Nhan 2 de chon viec nhap tu ban phim" << endl;

int c;

do{

c = 0;

cout << endl << "Moi chon: ";

cout << flush;

cin >> c;

if (c < 1 || c > 2) {

cout << "So vua nhap khong phu hop. Vui long chon lai!" << endl;

}

}

while (c < 1 || c > 2);

string nameFile;

switch (c) {

case 1:

cout << "Ban da chon: Du lieu se duoc nhap tu file!" << endl;

do{

cout << "Moi nhap ten file: ";

cout << flush;

cin >> nameFile;

input.open(nameFile);

if (input .fail()) {

cout << "Khong tim thay file. Vui long nhap lai!" << endl;

}

}

while (input .fail());

break;

case 2:

cout << "Ban da chon: Du lieu se duoc nhap tu ban phim!" << endl;

cout << "Moi nhap du lieu o ben duoi" << endl;

break;

}

cout << flush;

readData(N, M, dataMatrix, c);

if (c == 1){

input.close();

}

cout << "Du lieu da duoc nhap xong!" << endl << endl;

}

void outputSeclection() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* LUA CHON DINH DANG XUAT \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

cout << "An 1: Xuat du lieu ra tep data.out" << endl;

cout << "An 2: Xuat du lieu ra Terminal" << endl;

int choose;

do{

choose = 0;

cout << endl << "Moi chon kieu du lieu: ";

cout << flush;

cin >> choose;

if (choose < 1 || choose > 2) {

cout << "So vua nhap khong phu hop. Vui long chon lai!" << endl;

}

}

while (choose < 1 || choose > 2);

switch (choose) {

case 1:

cout << "Ban da chon: Du lieu se duoc xuat ra o tep data.out" << endl;

output.open("data.out");

isPrintToConsole = false;

break;

case 2:

cout << "Ban da chon: Du lieu se duoc xuat ra o Terminal" << endl << endl;

break;

}

}

void instruction() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* KET QUA TINH TOAN \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

cout << "Viec tinh toan da hoan tat! Sau day la cac buoc ma chuong trinh da thuc hien:" << endl << endl;

cout << "Buoc 1: Doc du lieu" << endl;

cout << "Buoc 2: Tinh vector trung binh" << endl;

cout << "Buoc 3: Tinh su chenh lech giua dataMatrix va avgVector" << endl;

cout << "Buoc 4: Tinh tich 2 ma tran" << endl;

cout << "Buoc 5: Tim tri rieng cua ma tran" << endl;

cout << "Buoc 6: Tinh cac vector tuong ung voi cac tri rieng" << endl;

cout << "Buoc 7: Tinh tich 2 ma tran ansVector = matrixProduct \* eigenVectors" << endl;

cout << "Buoc 8: Rut gon ansVector" << endl;

cout << "Buoc 9: Xuat ket qua tuy theo lua chon" << endl << endl;

}

//Ham tinh tich co huong 2 ma tran

MatrixXd multipleMatrix(MatrixXd matrix1, MatrixXd matrix2) {

MatrixXd matrixProduct(matrix1.rows(), matrix2.cols());

foru(i, 0, matrix1.rows() - 1) {

foru(j, 0, matrix2.cols() - 1) {

double sum = 0;

foru(k, 0, matrix1.cols() - 1) {

sum += matrix1(i, k) \* matrix2(k, j);

}

matrixProduct(i, j) = sum;

}

}

return matrixProduct;

}

//Ma tran chuyen vi

MatrixXd transposeMatrix(MatrixXd matrix) {

MatrixXd transposedMatrix(matrix.cols(), matrix.rows());

foru(i, 0, matrix.rows() - 1) {

foru(j, 0, matrix.cols() - 1) {

transposedMatrix(j, i) = matrix(i, j);

}

}

return transposedMatrix;

}

//Buoc 0: Lua chon dau vao

void inputoutputSetup(int &N, int &M, MatrixXd &dataMatrix) {

welcome();

Sleep(1000);

inputSeclection(N, M, dataMatrix);

Sleep(500);

outputSeclection();

Sleep(500);

instruction();

Sleep(500);

}

//Buoc 2: Tinh vector trung binh

MatrixXd averageVector(int N, int M, MatrixXd dataMatrix) {

MatrixXd avgVector(N, 1);

foru(i, 0, N - 1){

double sum = 0;

foru(j, 0, M - 1){

sum += dataMatrix(i, j);

}

avgVector(i, 0) = sum / M;

}

return avgVector;

}

//Buoc 3: Tinh su chenh lech giua dataMatrix va avgVector

MatrixXd differenceMatrix(int N, int M, MatrixXd dataMatrix, MatrixXd avgVector) {

MatrixXd diffMatrix(N, M);

foru(i, 0, N - 1){

foru(j, 0, M - 1){

diffMatrix(i, j) = dataMatrix(i, j) - avgVector(i, 0);

}

}

return diffMatrix;

}

// Bước 5 và 6: Tìm eigenValues và eigenVectors bằng phương pháp jac

void jacobiRotate(MatrixXd &A, MatrixXd &V, int p, int q) {

double c, s;

if (A(p, q) != 0.0) {

double d = (A(q, q) - A(p, p)) / (2.0 \* A(p, q));

double t = (d >= 0) ? 1.0 / (std::fabs(d) + std::sqrt(1.0 + d\*d)) : -1.0 / (std::fabs(d) + std::sqrt(1.0 + d\*d));

double t2 = t / std::sqrt(1.0 + t\*t);

double t1 = t \* t2;

c = 1.0 / std::sqrt(1.0 + t\*t);

s = c \* t;

} else {

c = 1.0;

s = 0.0;

}

// Thực hiện ma trận xoay Jacobi

double a\_pp = A(p, p);

double a\_qq = A(q, q);

double a\_pq = A(p, q);

A(p, p) = c\*c\*a\_pp - 2.0\*c\*s\*a\_pq + s\*s\*a\_qq;

A(q, q) = s\*s\*a\_pp + 2.0\*c\*s\*a\_pq + c\*c\*a\_qq;

A(p, q) = 0.0;

A(q, p) = 0.0;

for (int k = 0; k < A.rows(); ++k) {

if (k != p && k != q) {

double a\_pk = A(p, k);

double a\_qk = A(q, k);

A(p, k) = c\*a\_pk - s\*a\_qk;

A(k, p) = A(p, k);

A(q, k) = s\*a\_pk + c\*a\_qk;

A(k, q) = A(q, k);

}

double v\_pk = V(k, p);

double v\_qk = V(k, q);

V(k, p) = c\*v\_pk - s\*v\_qk;

V(k, q) = s\*v\_pk + c\*v\_qk;

}

}

// Sửa lại hàm Jacobi để trả về eigenvalues và eigenvectors

void jacobiEigen(MatrixXd A, MatrixXd &eigenValues, MatrixXd &eigenVectors, int maxIter = 100, double tolerance = 1e-8) {

int n = A.rows();

eigenVectors = MatrixXd::Identity(n, n); // Khởi tạo ma trận eigenVectors là ma trận đơn vị

int iter = 0;

while (iter < maxIter) {

double maxOffDiag = 0.0;

int p, q; // Vị trí của phần tử ngoại đường chéo lớn nhất

// Tìm phần tử ngoại đường chéo lớn nhất

for (int i = 0; i < n - 1; ++i) {

for (int j = i + 1; j < n; ++j) {

if (std::abs(A(i, j)) > maxOffDiag) {

maxOffDiag = std::abs(A(i, j));

p = i;

q = j;

}

}

}

// Kiểm tra điều kiện dừng

if (maxOffDiag < tolerance) break;

// Tính góc xoay

double theta = 0.5 \* std::atan2(2 \* A(p, q), A(p, p) - A(q, q));

double c = std::cos(theta);

double s = std::sin(theta);

// Tạo ma trận xoay

MatrixXd R = MatrixXd::Identity(n, n);

R(p, p) = c;

R(q, q) = c;

R(p, q) = -s;

R(q, p) = s;

// Cập nhật ma trận A và ma trận eigenVectors

A = R.transpose() \* A \* R;

eigenVectors = eigenVectors \* R;

iter++;

}

// Lấy eigenvalues từ ma trận A (ma trận đường chéo chứa eigenvalues)

eigenValues.resize(n, 1);

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

eigenValues(i, 0) = A(i, i);

}

}

//Buoc 7: Loc ra cac eigenValues va eigenVectors phu hop

void filterEigen(MatrixXd& eigenValues, MatrixXd& eigenVectors) {

for (int i = 0; i < eigenValues.rows() - 1; ++i) {

for (int j = i + 1; j < eigenValues.rows(); ++j) {

if (eigenValues(i, 0) < eigenValues(j, 0)) {

double tempValue = eigenValues(i, 0);

eigenValues(i, 0) = eigenValues(j, 0);

eigenValues(j, 0) = tempValue;

MatrixXd tempVector = eigenVectors.col(i);

eigenVectors.col(i) = eigenVectors.col(j);

eigenVectors.col(j) = tempVector;

}

}

}

// Khai báo các ma trận phụ

int index = 0;

MatrixXd tempEigenValues(eigenValues.rows(), 1);

MatrixXd tempEigenVectors(eigenVectors.rows(), eigenVectors.cols());

for (int i = 0; i < eigenValues.rows() - 1; ++i) {

if (fabs(eigenValues(i, 0) - 0.0) > ESP && fabs(eigenValues(i, 0) - eigenValues(i + 1, 0)) > ESP) {

tempEigenValues(index, 0) = eigenValues(i, 0);

tempEigenVectors.col(index) = eigenVectors.col(i);

index++;

}

}

// Sao chép kết quả vào ma trận gốc

eigenValues.resize(index, 1);

eigenVectors.resize(eigenVectors.rows(), index);

eigenValues = tempEigenValues.block(0, 0, index, 1);

eigenVectors = tempEigenVectors.block(0, 0, eigenVectors.rows(), index);

}

//Buoc 8: Rut gon ansVector

MatrixXd normalizeVector(MatrixXd ansVector) {

vector <double> normVector;

foru(i, 0, ansVector.cols() - 1) {

double norm = 0;

foru(j, 0, ansVector.rows() - 1) {

norm += ansVector(j, i) \* ansVector(j, i);

}

normVector.push\_back(sqrt(norm));

}

foru(i, 0, ansVector.cols() - 1) {

foru(j, 0, ansVector.rows() - 1) {

ansVector(j, i) /= normVector[i];

}

}

return ansVector;

}

//Buoc 9: Xuat ket qua

void printOutput(int N, int M, MatrixXd dataMatrix, MatrixXd avgVector, MatrixXd diffMatrix, MatrixXd matrixProduct, MatrixXd eigenValues, MatrixXd eigenVectors, MatrixXd ansVector) {

ofstream output;

output.open("data.out");

cout << "Nhan 1 de in ra toan bo qua trinh" << endl;

cout << "Nhan 8 de in ra ket qua cuoi cung" << endl;

cout << "Nhan cac phim tu 2 toi 7 de in ra ket qua o cac buoc tuonng ung" << endl;

cout << "Nhan 0 de thoat chuong trinh" << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

int choose;

do{

cout << "Vui long chon: ";

cout << flush;

cin >> choose;

if (choose < 0 || choose > 8) {

cout << "Vui long chon lai!" << endl;

} else{

if (isPrintToConsole) {

switch (choose) {

case 0:

cout << "Ket thuc chuong trinh, xin cam on!" << endl;

break;

case 1:

cout << "Buoc 2: Vector trung binh: " << endl << avgVector << endl;

cout << "Buoc 3: Su chenh lech giua dataMatrix va avgVector: " << endl << diffMatrix << endl;

cout << "Buoc 4: Tich 2 ma tran: " << endl << matrixProduct << endl;

cout << "Buoc 5: Tri rieng cua ma tran: " << endl << eigenValues << endl;

cout << "Buoc 6: Cac vector tuong ung voi cac tri rieng: " << endl << eigenVectors << endl;

cout << "Buoc 7: Tich 2 ma tran ansVector = matrixProduct \* eigenVectors: " << endl << matrixProduct \* eigenVectors << endl;

cout << "Buoc 8: Rut gon ansVector, ket qua cuoi cung la: " << endl << normalizeVector(ansVector) << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 2:

cout << "Buoc 2: Vector trung binh: " << endl << avgVector << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 3:

cout << "Buoc 3: Su chenh lech giua dataMatrix va avgVector: " << endl << diffMatrix << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 4:

cout << "Buoc 4: Tich 2 ma tran: " << endl << matrixProduct << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 5:

cout << "Buoc 5: Tri rieng cua ma tran: " << endl << eigenValues << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 6:

cout << "Buoc 6: Cac vector tuong ung voi cac tri rieng: " << endl << eigenVectors << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 7:

cout << "Buoc 7: Tich 2 ma tran ansVector = matrixProduct \* eigenVectors: " << endl << matrixProduct \* eigenVectors << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 8:

cout << "Ket qua cuoi cung la: " << endl << normalizeVector(ansVector) << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

}

}

else {

switch (choose) {

case 0:

cout << "Ket thuc chuong trinh, xin vui long mo file data.out de xem ket qua" << endl;

break;

case 1:

cout << "Ban da chon: In ra toan bo qua trinh" << endl;

output << "Buoc 2: Vector trung binh: " << endl << avgVector << endl;

output << "Buoc 3: Su chenh lech giua dataMatrix va avgVector: " << endl << diffMatrix << endl;

output << "Buoc 4: Tich 2 ma tran: " << endl << matrixProduct << endl;

output << "Buoc 5: Tri rieng cua ma tran: " << endl << eigenValues << endl;

output << "Buoc 6: Cac vector tuong ung voi cac tri rieng: " << endl << eigenVectors << endl;

output << "Buoc 7: Tich 2 ma tran ansVector = matrixProduct \* eigenVectors: " << endl << matrixProduct \* eigenVectors << endl;

output << "Ket qua cuoi cung la: " << endl << normalizeVector(ansVector) << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 2:

cout << "Ban da chon: In ra buoc 2" << endl;

output << "Buoc 2: Vector trung binh: " << endl << avgVector << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 3:

cout << "Ban da chon: In ra buoc 3" << endl;

output << "Buoc 3: Su chenh lech giua dataMatrix va avgVector: " << endl << diffMatrix << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 4:

cout << "Ban da chon: In ra buoc 4" << endl;

output << "Buoc 4: Tich 2 ma tran: " << endl << matrixProduct << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 5:

cout << "Ban da chon: In ra buoc 5" << endl;

output << "Buoc 5: Tri rieng cua ma tran: " << endl << eigenValues << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 6:

cout << "Ban da chon: In ra buoc 6" << endl;

output << "Buoc 6: Cac vector tuong ung voi cac tri rieng: " << endl << eigenVectors << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 7:

cout << "Ban da chon: In ra buoc 7" << endl;

output << "Buoc 7: Tich 2 ma tran ansVector = matrixProduct \* eigenVectors: " << endl << matrixProduct \* eigenVectors << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

case 8:

cout << "Ban da chon: In ra buoc 8" << endl;

output << "Buoc 8: Rut gon ansVector, ket qua cuoi cung la: " << endl << normalizeVector(ansVector) << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

break;

}

}

}

Sleep(500);

}

while (choose != 0);

output.close();

}

int main() {

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

int N, M;

MatrixXd dataMatrix;

//Buoc 0: Cai dat thong tin

inputoutputSetup(N, M, dataMatrix);

// Buoc 1: Doc du lieu

MatrixXd avgVector(N, 1), diffMatrix(N, M), matrixProduct(N, M), tranMatrix(N, M), eigenValues(N, 1), eigenVectors(N, M);

//Buoc 2: Tinh vector trung binh

avgVector = averageVector(N, M, dataMatrix);

//Buoc 3: Tinh su chenh lech giua dataMatrix va avgVector

diffMatrix = differenceMatrix(N, M, dataMatrix, avgVector);

//Buoc 4: Tinh tich 2 ma tran

tranMatrix = transposeMatrix(diffMatrix);

matrixProduct = multipleMatrix(tranMatrix, diffMatrix);

//Buoc 5: Tim eigenvalues va eigenvectors

eigenValues.resize(N, 1);

eigenVectors.resize(N, N);

jacobiEigen(matrixProduct, eigenValues, eigenVectors);

//Buoc 6: Loc ra cac eigenValues va eigenVectors phu hop

filterEigen(eigenValues, eigenVectors);

//Buoc 7: Tinh tich 2 ma tran ansVector = matrixProduct \* eigenVectors

MatrixXd ansVector = multipleMatrix(diffMatrix, eigenVectors);

//Buoc 8: Rut gon ansVector

//Buoc 9: Xuat ket qua

printOutput(N, M, dataMatrix, avgVector, diffMatrix, matrixProduct, eigenValues, eigenVectors, ansVector);

//Buoc 10: An phim bat ki de thoat chuong trinh

cout << flush << endl;

system("pause");

return 0;

}

Chương trình đầy đủ bằng ngôn ngữ Python:

import numpy as np

from PIL import Image

from sklearn.decomposition import PCA

data = np.loadtxt('data.out')

pca = PCA(n\_components=2)

transformed\_data = pca.fit\_transform(data)

transformed\_data -= np.min(transformed\_data, axis=0)

transformed\_data /= np.max(transformed\_data, axis=0)

image\_data = np.zeros((100, 100, 3), dtype=np.uint8)

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

    x, y = transformed\_data[i] \* 99

    x, y = int(x), int(y)

    image\_data[y, x] = [255, 255, 255]

image = Image.fromarray(image\_data)

image.show()

image.save('generated\_image.png')

HẾT