Phân Tích Thành Phần Chính và Ứng Dụng Trong Nhận Diện Khuôn Mặt

Ngày 8 tháng 7 năm 2024

Thành viên nhóm

Lê Quốc Đạt – 21110259

Đào Huy Hoàng – 21110297

Đặng Thị Kim Anh – 21280084

Lê Hồ Hoàng Anh – 21280085

Nguyễn Phúc Loan – 21280098

Mục lục

Chương 1

Giới Thiệu

Trong thế giới ngày nay, sự phát triển nhanh chóng của kỹ thuật số và mạng Internet đã mang lại nhiều lợi ích, nhưng cùng với đó cũng là sự gia tăng đáng kể của các mối đe dọa về an toàn thông tin và vật chất. Những vụ đánh cắp thông tin thẻ tín dụng, tài khoản ngân hàng, và các cuộc tấn công vào hệ thống máy tính của nhà nước và chính phủ không còn xa lạ. Năm 1998, hơn 100 triệu đô la đã bị thất thoát ở Mỹ do các vụ xâm nhập phạm pháp này. Các tội phạm thường lợi dụng những lỗ hổng cơ bản trong quá trình truy cập vào các hệ thống thông tin và kiểm soát để thực hiện hành vi xâm nhập.

Để đối phó với những mối đe dọa này, công nghệ hiện đại đã phát triển nhiều phương pháp xác thực dựa trên những đặc trưng riêng biệt và độc đáo của mỗi cá nhân. Một trong những phương pháp hiệu quả trong lĩnh vực này là nhận dạng mặt người. Nhận dạng mặt người không chỉ được sử dụng trong an ninh mà còn trong nhiều ứng dụng khác như mở khóa thiết bị, chấm công nhân viên, và theo dõi hành vi khách hàng.

Trong chủ đề này, chúng ta sẽ đi sâu vào các phương pháp đã được sử dụng trong lĩnh vực nhận dạng mặt người, với trọng tâm là phương pháp phân tích thành phần chính (Principal Components Analysis - PCA). Phương pháp PCA không chỉ là một công cụ mạnh mẽ trong việc giảm thiểu dữ liệu mà còn giúp trích xuất các đặc trưng quan trọng từ hình ảnh mặt người. Qua đó, chúng ta có thể xây dựng các hệ thống nhận dạng mặt người với độ chính xác cao và hiệu quả.

Chương 2

Thuật Toán PCA

2.1 Giới thiệu về PCA

PCA là thuật toán tìm một không gian mới (với số chiều nhỏ hơn không gian cũ) với các trục tọa độ trong không gian mới được xây dựng sao cho trên mỗi trục, độ biến thiên của dữ liệu là lớn nhất có thể.

Hình 2.1: Ví dụ minh họa PCA

Cùng là 1 chú lạc đà, tuy nhiên với các góc nhìn khác nhau (trục thông tin), chúng ta có những cách thu nhận thông tin khác nhau và cho ta những kết luận khác nhau

2.2 Khái niệm

Phân tích thành phần chính là một phương pháp được sử dụng thường xuyên khi các nhà phân tích thống kê phải đối mặt với những bộ số liệu với số chiều lớn (big data) để giảm thiểu chiều dữ liệu mà vẫn không mất đi thông tin và giữ lại được những thông tin cần thiết cho việc xây dựng các mô hình bằng một thuật toán thống kê sử dụng phép biến đổi trực

giao để biến đổi một tập hợp dữ liệu từ một không gian nhiều chiều sang một không gian mới ít chiều hơn (2 hoặc 3 chiều) nhằm tối ưu hóa việc thể hiện sự biến thiên của dữ liệu.

2.3 Đặc tính

- **Xây dựng không gian mới:** Thay vì sử dụng các trục tọa độ của không gian dữ liệu ban đầu, PCA tạo ra một không gian mới ít chiều hơn nhưng vẫn có khả năng biểu diễn dữ liệu gần như tương đương. Các thành phần chính trong không gian mới là các vectơ riêng của ma trận hiệp phương sai, đại diện cho các hướng của biến thiên lớn nhất trong dữ liệu.
- **Tính tuyến tính của các thành phần chính:** Các trục tọa độ trong không gian mới là tổ hợp tuyến tính của các biến gốc trong không gian cũ. Điều này có nghĩa là PCA xây dựng các thành phần mới dựa trên sự kết hợp tuyến tính của các biến gốc và vẫn giữ được tính chất biểu diễn của dữ liệu ban đầu.
- **Khám phá các liên kết tiềm ẩn:** Trong không gian mới, PCA có thể giúp phát hiện các mối quan hệ phức tạp, liên kết tiềm ẩn giữa các biến mà trong không gian ban đầu có thể không rõ ràng hoặc khó phát hiện được. Điều này có thể cải thiện khả năng diễn giải và hiểu được cấu trúc dữ liệu.
- **Ứng dụng trong khai phá dữ liệu và máy học:** PCA là công cụ quan trọng để giảm số chiều dữ liệu, trực quan hóa dữ liệu, phát hiện biến động quan trọng và tối ưu hóa các mô hình học máy.

2.4 Cơ sở toán học

2.4.1 Kỳ vọng (mean)

Kỳ vọng của một biến ngẫu nhiên X, thường được ký hiệu là $\mathbb{E}[X]$, là giá trị mong đợi của biến đó. Đối với một tập hợp N giá trị $x_1, x_2, \ldots, x_N$, kỳ vọng được tính bằng:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Kỳ vọng (expectation) trong lý thuyết xác suất và thống kê là một cách mô tả giá trị trung bình của một tập hợp các giá trị ngẫu nhiên. Khi áp dụng vào dữ liệu mẫu (như dữ liệu bạn đã đề cập), kỳ vọng chính là giá trị trung bình (mean) của các giá trị trong tập dữ liệu đó. Để làm rõ hơn, hãy xem xét hai trường hợp: kỳ vọng của biến ngẫu nhiên và giá trị trung bình của dữ liệu mẫu.

2.4.2 Phương sai (variance)

Phương sai là một chỉ số đo lường mức độ phân tán của một tập dữ liệu xung quanh giá trị trung bình của nó. Phương sai càng nhỏ thì các điểm dữ liệu càng gần với kỳ vọng, tức các điểm dữ liệu càng giống nhau. Phương sai càng lớn thì ta nói dữ liệu càng có tính phân tán. Công thức tính phương sai của một tập dữ liệu $x_1, x_2, \ldots, x_N$ là:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2$$

Phương sai (variance) cũng được dùng để tính toán độ lệch chuẩn (standard deviation), là căn bậc hai của phương sai.

2.4.3 Hiệp phương sai (covariance)

Hiệp phương sai là độ đo sự biến thiên cùng nhau của hai biến ngẫu nhiên. Nếu 2 biến có xu hướng thay đổi cùng nhau (nghĩa là, khi một biến có

giá trị cao hơn giá trị kỳ vọng thì biến kia có xu hướng cũng cao hơn giá trị kỳ vọng), thì hiệp phương sai giữa hai biến này có giá trị dương. Mặt khác, nếu một biến nằm trên giá trị kỳ vọng còn biến kia có xu hướng nằm dưới giá trị kỳ vọng, thì hiệp phương sai của hai biến này có giá trị âm. Nếu hai biến này độc lập với nhau thì giá trị bằng 0. Công thức tính hiệp phương sai giữa hai biến X và Y là:

$$\text{cov}(X, Y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})$$

2.4.4 Ma trận hiệp phương sai

Cho N điểm dữ liệu được biểu diễn bởi các vector cột $x_1, x_2, \ldots, x_N$, vector kỳ vọng và ma trận hiệp phương sai của toàn bộ dữ liệu được định nghĩa là:

$$C = \frac{1}{N-1}(X - \bar{X})^\top (X - \bar{X})$$

trong đó X là ma trận dữ liệu với mỗi hàng là một quan sát, $\bar{X}$ là vector trung bình của các cột của X, và $X^\top$ là ma trận chuyển vị của X.

2.4.5 Trị riêng và vector riêng của ma trận hiệp phương sai

Cho một ma trận vuông A, nếu số vô hướng λ và vector $v \neq 0$ thỏa mãn phương trình:

$$Av = \lambda v$$

thì λ được gọi là trị riêng của ma trận A, và v được gọi là vector riêng ứng với trị riêng λ.

Trị riêng λ và vector riêng v là các giá trị và vectơ đặc biệt của ma trận A. Phương trình $Av = \lambda v$ có nghĩa là việc nhân ma trận A với vector v tạo ra một vector mới chỉ có thay đổi về tỉ lệ và hướng của v. Trong ngữ cảnh của PCA, ma trận hiệp phương sai C được sử dụng để tìm trị riêng

và vector riêng, giúp chúng ta hiểu rõ hơn về hướng và mức độ biến thiên của dữ liệu trong không gian mới sau khi thực hiện PCA.

2.5 Các bước phân tích thành phần chính

1. Chuẩn bị dữ liệu:
   - Chuẩn bị ma trận dữ liệu X, trong đó mỗi hàng biểu diễn một quan sát, mỗi cột biểu diễn một biến.
2. Chuẩn hóa dữ liệu (nếu cần):
   - Nếu các biến trong X có phạm vi hoặc đơn vị đo khác nhau, thực hiện chuẩn hóa để các biến có cùng phạm vi hoặc đơn vị đo.
3. Tính giá trị trung bình của từng biến:
   - Tính vector giá trị trung bình $\bar{x}$ của các biến.
4. Tính ma trận hiệp phương sai:
   - Tính ma trận hiệp phương sai S bằng cách sử dụng công thức:

$$S = \frac{1}{n-1}(X - \bar{X})^\top (X - \bar{X})$$

     Trong đó n là số lượng quan sát.
5. Tính các trị riêng và vectơ riêng của ma trận hiệp phương sai:
   - Giải bài toán trị riêng-vectơ riêng của ma trận S. Trị riêng λ_i và vectơ riêng tương ứng v_i phải được sắp xếp theo thứ tự giảm dần của giá trị riêng.
6. Chọn số chiều của không gian mới:
   - Chọn số lượng thành phần chính (số lượng trị riêng và vectơ riêng) mà bạn muốn giữ lại để giảm chiều dữ liệu. Đây thường là một lựa chọn dựa trên tỉ lệ phần trăm phương sai giải thích hoặc các tiêu chí khác.

7. Tạo ma trận chiếu (ma trận vectơ riêng):

   - Xây dựng ma trận chiếu V_k với các cột là k vectơ riêng (đơn vị) ứng với k trị riêng lớn nhất.

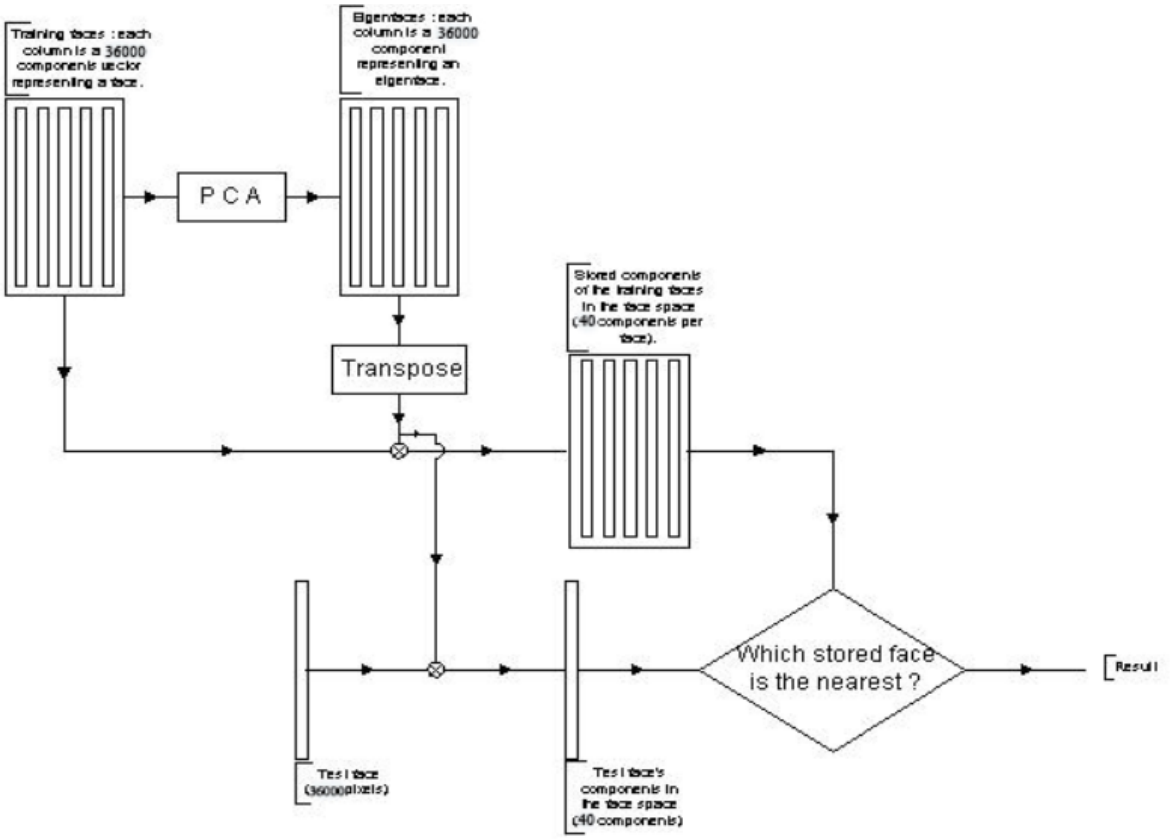
8. Biến đổi dữ liệu gốc sang không gian mới:

   - Biến đổi dữ liệu ban đầu X thành dữ liệu mới Y trong không gian được xác định bởi các thành phần chính bằng công thức:

$$Y = (X - \bar{X})V_k$$

   Trong đó V_k là ma trận chứa các vectơ riêng ứng với k trị riêng lớn nhất.

9. Sơ đồ:



Hình 3. Sơ đồ thuật toán nhận dạng mặt người dùng PCA

2.5.1 Các bước chi tiết hơn trong PCA

Để hiểu rõ hơn các bước cụ thể trong PCA, chúng ta có thể mô tả chi tiết từng bước như sau:

1. Chuẩn bị dữ liệu : - Dữ liệu ban đầu X có kích thước $n \times p$, trong đó n là số quan sát và p là số biến.

2. Chuẩn hóa dữ liệu : - Chuẩn hóa dữ liệu nếu cần, để mỗi biến có giá trị trung bình bằng 0 và phương sai bằng 1. Điều này có thể thực hiện bằng cách trừ đi giá trị trung bình của mỗi biến và chia cho độ lệch chuẩn của biến đó.

3. Tính giá trị trung bình của từng biến : - Tính vector giá trị trung bình $\bar{x}$ của các biến:

$$\bar{x}_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$$

trong đó x_{ij} là giá trị của biến j tại quan sát i.

4. Tính ma trận hiệp phương sai : - Tính ma trận hiệp phương sai S:

$$S = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^\top$$

5. Tính các trị riêng và vectơ riêng của ma trận hiệp phương sai : - Giải các trị riêng λ_i và vectơ riêng v_i của ma trận S bằng cách giải phương trình đặc trưng:

$$Sv_i = \lambda_i v_i$$

Các trị riêng λ_i sẽ được sắp xếp theo thứ tự giảm dần, và các vectơ riêng v_i tương ứng sẽ được sắp xếp theo các trị riêng đó.

6. Chọn số chiều của không gian mới : - Xác định số lượng thành phần chính cần giữ lại bằng cách kiểm tra tổng phần trăm phương sai được giải thích bởi các thành phần chính đó. Ví dụ, nếu mục tiêu là giữ lại 95

7. Tạo ma trận chiếu (ma trận vectơ riêng) : - Tạo ma trận chiếu V_k, trong đó các cột là k vectơ riêng tương ứng với k trị riêng lớn nhất. Ma trận này sẽ được sử dụng để chiếu dữ liệu ban đầu vào không gian mới.

8. Biến đổi dữ liệu gốc sang không gian mới : - Biến đổi dữ liệu ban đầu X thành dữ liệu mới Y trong không gian thành phần chính bằng cách nhân ma trận dữ liệu đã chuẩn hóa với ma trận chiếu V_k:

$$Y = (X - \bar{X})V_k$$

Dữ liệu Y là dữ liệu mới trong không gian thành phần chính, với số chiều đã được giảm từ p xuống k.

2.6 Ứng dụng PCA trong nhận dạng khuôn mặt

Trong lĩnh vực nhận dạng khuôn mặt, PCA được sử dụng để giảm chiều dữ liệu của các hình ảnh khuôn mặt, giúp tăng hiệu quả và độ chính xác của các thuật toán nhận dạng. Các bước thực hiện PCA trong nhận dạng khuôn mặt bao gồm:

1. Chuẩn bị dữ liệu hình ảnh : - Tập hợp dữ liệu hình ảnh khuôn mặt, mỗi hình ảnh được biểu diễn dưới dạng một vector hàng. Ví dụ, nếu mỗi hình ảnh có kích thước $m \times n$ pixel, thì mỗi hình ảnh sẽ được biến đổi thành một vector có $m \times n$ phần tử.

2. Chuẩn hóa dữ liệu hình ảnh : - Chuẩn hóa dữ liệu hình ảnh để mỗi vector có giá trị trung bình bằng 0. Điều này giúp đảm bảo rằng các thành phần chính phản ánh sự biến đổi trong dữ liệu, thay vì bị ảnh hưởng bởi giá trị trung bình.

3. Tính ma trận hiệp phương sai của dữ liệu hình ảnh : - Tính ma trận hiệp phương sai của các vector hình ảnh chuẩn hóa để tìm ra các hướng chính của sự biến đổi trong dữ liệu hình ảnh.

4. Tính các trị riêng và vectơ riêng của ma trận hiệp phương sai : - Giải các trị riêng và vectơ riêng của ma trận hiệp phương sai để xác định các hướng chính (thành phần chính) của sự biến đổi trong dữ liệu hình ảnh.

5. Chọn số lượng thành phần chính cần giữ lại : - Xác định số lượng thành phần chính cần giữ lại để đảm bảo rằng phần lớn phương sai của dữ liệu được giữ lại trong không gian mới. Trong thực tế, số lượng thành

phần chính thường được chọn dựa trên tỉ lệ phần trăm phương sai giải thích hoặc qua phân tích thực nghiệm.

6. Biến đổi dữ liệu hình ảnh sang không gian mới : - Sử dụng các vectơ riêng tương ứng với các thành phần chính để chiếu dữ liệu hình ảnh ban đầu vào không gian mới, giảm số chiều của dữ liệu từ $m \times n$ xuống số lượng thành phần chính được chọn.

7. Sử dụng dữ liệu trong không gian thành phần chính cho nhận dạng khuôn mặt : - Dữ liệu trong không gian thành phần chính có thể được sử dụng làm đầu vào cho các thuật toán nhận dạng khuôn mặt, chẳng hạn như các thuật toán học máy hoặc các phương pháp đối sánh mẫu.

Sử dụng PCA để giảm chiều dữ liệu hình ảnh khuôn mặt giúp tăng hiệu quả tính toán và độ chính xác của các thuật toán nhận dạng, đồng thời giảm thiểu nhiễu và sự dư thừa trong dữ liệu.

2.7 Ứng dụng của phân tích thành phần chính trong lĩnh vực chuyên môn

- **Kinh tế:** PCA được sử dụng rộng rãi trong dự đoán thị trường chứng khoán và cổ phiếu. Bằng cách giảm số chiều của dữ liệu từ các chỉ số kinh tế và thông tin tài chính phức tạp, PCA giúp nhận diện các yếu tố chính ảnh hưởng đến biến động của thị trường. Điều này có thể cải thiện khả năng dự báo và quản lý rủi ro trong đầu tư và giao dịch.

- **Sinh học:** Trong ngành sinh học, PCA được dùng để phân tích dữ liệu phân tử và di truyền. Ví dụ, trong phân loại các mã gen của tế bào, PCA có thể giúp xác định những đặc điểm quan trọng và nhận biết các nhóm gen có tính chất tương đồng nhau. Điều này quan trọng trong nghiên cứu về di truyền học và phát triển thuốc.

- **Y học và Y tế:** PCA được sử dụng để phân tích dữ liệu y tế, từ việc phân tích hình ảnh chẩn đoán trong điều trị hình ảnh đến phân tích dữ liệu lâm sàng và dữ liệu y sinh học để tìm ra các yếu tố quan trọng và mối quan hệ giữa các biến.

- **Khoa học xã hội:** PCA được áp dụng để phân tích dữ liệu khảo sát và nghiên cứu hành vi. Bằng cách giảm số chiều của dữ liệu từ các câu hỏi khảo sát phức tạp, PCA giúp nhận diện các yếu tố chính ảnh hưởng đến hành vi và quan điểm của con người. Điều này hỗ trợ các nhà nghiên cứu trong việc hiểu rõ hơn về các mô hình hành vi xã hội và phát triển các chính sách công hiệu quả.

- **Khoa học môi trường:** PCA được sử dụng để phân tích dữ liệu môi trường như chất lượng không khí, nước và đất. Bằng cách giảm số chiều của dữ liệu từ các chỉ số môi trường, PCA giúp xác định các nguồn gây ô nhiễm chính và đánh giá tác động của chúng lên môi trường. Điều này quan trọng trong việc phát triển các chiến lược quản lý và bảo vệ môi trường.

- **Kỹ thuật:** Trong lĩnh vực kỹ thuật, PCA có thể được áp dụng vào nhận diện hình ảnh và xử lý tín hiệu. Ví dụ, trong nhận diện khuôn mặt, PCA giúp giảm chiều dữ liệu hình ảnh và trích xuất các đặc trưng quan trọng nhất của khuôn mặt để phân loại và nhận dạng. Điều này cũng áp dụng trong việc nén hình ảnh và xử lý tín hiệu số.

Chương 3

Ứng Dụng Của PCA Trong Nhận Diện Khuôn Mặt

3.1 Ý tưởng chính

Mục tiêu của phương pháp PCA (Phân tích thành phần chính) là giảm số chiều của một tập hợp các vector dữ liệu sao cho vẫn giữ được những thông tin quan trọng nhất. PCA thực hiện điều này bằng cách chuyển đổi không gian dữ liệu ban đầu sang một không gian mới, gọi là không gian con của không gian ban đầu, sao cho phương sai của dữ liệu trên các thành phần chính (các trục chính) giảm dần theo thứ tự.

Trong bối cảnh nhận diện khuôn mặt, PCA được áp dụng để biểu diễn và nhận dạng các đặc trưng chính của khuôn mặt. Dữ liệu hình ảnh khuôn mặt thường có số lượng pixel lớn, và việc xử lý trực tiếp trên dữ liệu này có thể gặp khó khăn về mặt tính toán và hiệu quả. Thay vào đó, PCA giúp giảm chiều dữ liệu mà vẫn giữ được những đặc trưng quan trọng của khuôn mặt, như hình dáng chung, các đường nét, và tỷ lệ các phần của khuôn mặt.

Đặc biệt, PCA giúp tạo ra các biểu diễn gọn gàng của khuôn mặt bằng cách chuyển đổi các pixel hình ảnh thành các thành phần chính, mỗi thành phần biểu thị một dạng biến đổi hoặc cấu trúc chính của khuôn mặt. Các thành phần này có thể được sử dụng để so sánh, phân loại và nhận diện khuôn mặt trong các hệ thống nhận diện.

3.2 Ứng dụng của PCA trong nhận diện khuôn mặt

Phương pháp phân tích thành phần chính (PCA) có nhiều ứng dụng quan trọng trong việc nhận diện khuôn mặt, bao gồm:

- **Tiền xử lý:** PCA thường được sử dụng để chuẩn hóa kích cỡ của các ảnh trong cơ sở dữ liệu nhằm đảm bảo sự thống nhất và tăng tính đồng nhất của dữ liệu đầu vào.
- **Tách khuôn mặt:** PCA có thể được áp dụng để tách phần mặt từ các ảnh chụp ban đầu, giúp phân biệt và tách riêng khuôn mặt khỏi nền ảnh.
- **Trích chọn đặc trưng:** PCA giúp trích xuất các đặc trưng chính của khuôn mặt từ dữ liệu hình ảnh, như hình dáng chung, các nét mặt quan trọng như mắt, mũi, miệng, và tỷ lệ các phần của khuôn mặt.
- **Đối sánh:** Sau khi trích chọn các vector đặc trưng của khuôn mặt bằng PCA, chúng có thể được sử dụng để đối sánh và nhận diện khuôn mặt giữa các ảnh khác nhau. PCA giúp giảm chiều dữ liệu mà vẫn giữ được những đặc điểm quan trọng để dễ dàng so sánh và xử lý.

PCA không chỉ giúp cải thiện độ chính xác trong việc nhận diện khuôn mặt mà còn giảm thiểu chi phí tính toán và tăng tốc quá trình xử lý dữ liệu hình ảnh.

3.3 Ưu điểm của PCA

- **Giảm chiều dữ liệu:** PCA giúp giảm số chiều của dữ liệu mà vẫn giữ được những đặc trưng quan trọng, làm giảm độ phức tạp tính toán và giúp cải thiện hiệu suất.
- **Loại bỏ đặc trưng tương quan:** PCA có thể giúp loại bỏ hoặc giảm sự tương quan giữa các đặc trưng, từ đó giảm thiểu vấn đề đa cộng tuyến trong dữ liệu.

- **Hiệu quả tính toán:** PCA có thể giảm thiểu thời gian tính toán và lưu trữ so với việc xử lý trực tiếp trên dữ liệu gốc.
- **Trực quan hóa dữ liệu:** Sau khi áp dụng PCA, dữ liệu có thể được trực quan hóa dễ dàng hơn do đã giảm chiều.

3.4 Nhược điểm của PCA

- **Tuyến tính hóa:** PCA giả định rằng mối quan hệ giữa các biến là tuyến tính, điều này có thể không phù hợp với dữ liệu phi tuyến tính.
- **Mất thông tin quan trọng:** PCA có thể làm mất đi một số thông tin quan trọng khi giảm chiều dữ liệu.
- **Độ nhạy với tỷ lệ dữ liệu:** PCA nhạy cảm với tỷ lệ và phương sai của dữ liệu, đôi khi dẫn đến kết quả không mong muốn nếu dữ liệu không được chuẩn hóa đúng cách.
- **Không hiệu quả với dữ liệu phi tuyến tính:** PCA không hiệu quả khi áp dụng vào dữ liệu có các mối quan hệ phi tuyến tính phức tạp.

3.5 Các thử thách mà nhận diện khuôn mặt gặp phải

- **Điều kiện ánh sáng:** Biến đổi ánh sáng có thể làm ảnh hưởng đáng kể đến chất lượng và độ chính xác của nhận diện khuôn mặt.
- **Sự thay đổi hướng khuôn mặt:** Khuôn mặt có thể xuất hiện ở nhiều góc độ và hướng khác nhau, điều này đặt ra thách thức trong việc xác định và nhận dạng chính xác.
- **Nhận diện khuôn mặt dựa trên video:** Dữ liệu video đòi hỏi hệ thống phải xử lý nhanh và chính xác để nhận diện và theo dõi các khuôn mặt trong thời gian thực.

- **Điều kiện lão hóa:** Sự thay đổi của ngoại hình và cấu trúc khuôn mặt do quá trình lão hóa là một thử thách khó khăn cho các hệ thống nhận diện khuôn mặt.
- **Độ phân giải thấp:** Ảnh khuôn mặt có độ phân giải thấp có thể gây khó khăn trong việc nhận dạng và trích xuất các đặc trưng quan trọng của khuôn mặt.
- **Quản lý dữ liệu lớn:** Xử lý và quản lý các cơ sở dữ liệu lớn chứa thông tin khuôn mặt đòi hỏi hệ thống phải có khả năng xử lý một lượng dữ liệu lớn một cách hiệu quả và an toàn.

3.6 Ứng dụng nhận diện khuôn mặt trong thực tế

- **Nhận dạng tội phạm:** Các hệ thống nhận diện khuôn mặt được sử dụng để nhận dạng tội phạm đã bị truy nã hoặc người bị tạm giữ.
- **Thẻ căn cước chứng minh thư nhân dân:** Sử dụng để xác minh danh tính của cá nhân trong các thủ tục hành chính, chứng minh thư nhân dân.
- **Hệ thống theo dõi quan sát và bảo vệ:** Giúp giám sát và bảo vệ an ninh trong các khu vực công cộng và nhà máy.
- **Kiểm soát truy cập vào các hệ thống máy tính:** Dùng để xác thực người dùng và giới hạn quyền truy cập vào các hệ thống máy tính, bảo vệ thông tin quan trọng.
- **Giải pháp bảo mật cho các giao dịch rút tiền tự động:** Xác thực danh tính của người dùng để đảm bảo an toàn và ngăn chặn gian lận.
- **Hệ thống tương tác giữa người và máy:** Sử dụng nhận diện khuôn mặt để cải thiện trải nghiệm người dùng và tương tác tự động với máy móc.

- **Kiểm tra trạng thái người lái xe:** Các hệ thống giám sát lái xe dựa trên nhận diện khuôn mặt để đánh giá trạng thái như mệt mỏi, tập trung hoặc phát hiện ngủ gật.
- **Tìm kiếm và tổ chức dữ liệu liên quan đến con người:** Hỗ trợ trong việc quản lý và truy xuất thông tin cá nhân trong các cơ sở dữ liệu lớn.
- **Ứng dụng trong video phone:** Tự động nhận diện và theo dõi khuôn mặt trong cuộc gọi video, cải thiện trải nghiệm người dùng.
- **Phân loại trong lưu trữ hình ảnh trong điện thoại di động:** Sử dụng để nhận diện và phân loại các mặt của người thân trong hình ảnh cá nhân.

Chương 4

Ví dụ code Python

4.1

```
# -*- coding: utf-8 -*-
"""PPS.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1
      FnSDEXBHO3zAFNpLnW4kliIA-YbN66fI
"""

# Clone the repository and mount Google Drive
!git clone https://github.com/alaminanik/PCA-Based-
  Face-Recognition.git
from google.colab import drive
drive.mount('/content/drive')

# Import necessary libraries
import os
from matplotlib import pyplot as plt
from matplotlib.image import imread
```

```
import numpy as np

# Path to the images folder in your Google Drive
images_folder = '/content/PCA-Based-Face-Recognition/
    images1'
TRAIN_IMG_FOLDER = '/content/PCA-Based-Face-
    Recognition/images1/Training_images/'
TEST_IMG_FOLDER = '/content/PCA-Based-Face-
    Recognition/images1/Test/'

# Load training and testing image files
train_set_files = os.listdir(TRAIN_IMG_FOLDER)
test_set_files = os.listdir(TEST_IMG_FOLDER)

# Parameters for image dimensions
width = 128
height = 128

# Ensure the train and test sets have overlapping
    identities
train_id_file = set([f.split('_')[0] for f in
    train_set_files])
test_id_file = set([f.split('_')[0] for f in
    train_set_files])
print(train_id_file <= test_id_file)

# Preprocess training images
print('Train_Images:')
train_image_names = os.listdir(TRAIN_IMG_FOLDER)
training_tensor = np.ndarray(shape=(len(
    train_image_names), height*width), dtype=np.float64
```

```
    )

for i in range(len(train_image_names)):
    img = plt.imread(TRAIN_IMG_FOLDER +
        train_image_names[i])
    training_tensor[i, :] = np.array(img, dtype='
        float64').flatten()
    plt.subplot(5, 5, 1 + i)
    plt.imshow(img, cmap='gray')
    plt.tick_params(labelleft=False, labelbottom=
        False, bottom=False, top=False, right=False,
        left=False, which='both')
plt.show()

# Preprocess test images
print('Test Images:')
test_image_names = os.listdir(TEST_IMG_FOLDER)
testing_tensor = np.ndarray(shape=(len(
    test_image_names), height*width), dtype=np.float64)

for i in range(len(test_image_names)):
    img = imread(TEST_IMG_FOLDER + test_image_names[i
        ])
    testing_tensor[i, :] = np.array(img, dtype='
        float64').flatten()
    plt.subplot(8, 5, 1 + i)
    plt.imshow(img, cmap='gray')
    plt.subplots_adjust(right=1.2, top=1.2)
    plt.tick_params(labelleft=False, labelbottom=
        False, bottom=False, top=False, right=False,
        left=False, which='both')
```

```
plt.show()

# Calculate the mean face
mean_face = np.zeros((1, height*width))

for i in training_tensor:
    mean_face = np.add(mean_face, i)

mean_face = np.divide(mean_face, float(len(
  train_image_names))).flatten()

plt.imshow(mean_face.reshape(height, width), cmap='
  gray')
plt.tick_params(labelleft=False, labelbottom=False,
  bottom=False, top=False, right=False, left=False,
  which='both')
plt.show()

# Normalize the training tensor
normalised_training_tensor = np.ndarray(shape=(len(
  train_image_names), height*width))

for i in range(len(train_image_names)):
    normalised_training_tensor[i] = np.subtract(
      training_tensor[i], mean_face)

for i in range(len(train_image_names)):
    img = normalised_training_tensor[i].reshape(
      height, width)
    plt.subplot(5, 5, 1 + i)
    plt.imshow(img, cmap='gray')
```

```
        plt.tick_params(labelleft=False, labelbottom=
            False, bottom=False, top=False, right=False,
            left=False, which='both')
    plt.show()

# Compute the covariance matrix
cov_matrix = np.cov(normalised_training_tensor)
cov_matrix = np.divide(cov_matrix, 25.0)
print('Covariance Matrix Shape:', cov_matrix.shape)

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
print('eigenvalues.shape: {} eigenvectors.shape: {}'.
    format(eigenvalues.shape, eigenvectors.shape))

# Sort the eigenvalues and eigenvectors
eig_pairs = [(eigenvalues[index], eigenvectors[:,
    index]) for index in range(len(eigenvalues))]
eig_pairs.sort(reverse=True)
eigvalues_sort = [eig_pairs[index][0] for index in
    range(len(eigenvalues))]
eigvectors_sort = [eig_pairs[index][1] for index in
    range(len(eigenvalues))]

sorted_ind = sorted(range(eigenvalues.shape[0]), key=
    lambda k: eigenvalues[k], reverse=True)
eigvalues_sort = eigenvalues[sorted_ind]
eigvectors_sort = eigenvectors[sorted_ind]
train_set_files_sort = np.array(train_set_files)[
    sorted_ind]
```

```
# Cumulative variance explained
var_comp_sum = np.cumsum(eigvalues_sort) / sum(
    eigvalues_sort)
print("Cumulative proportion of variance explained
    vector: \n%s" % var_comp_sum)

# Plot cumulative variance explained
num_comp = range(1, len(eigvalues_sort) + 1)
plt.title('Cum. Prop. Variance Explain and Components
    Kept')
plt.xlabel('Principal Components')
plt.ylabel('Cum. Prop. Variance Explained')
plt.scatter(num_comp, var_comp_sum)
plt.show()

# Project training images onto the principal
    components
reduced_data = np.array(eigvectors_sort[:25]).
    transpose()
reduced_data.shape
print(training_tensor.transpose().shape, reduced_data
    .shape)

proj_data = np.dot(training_tensor.transpose(),
    reduced_data)
proj_data = proj_data.transpose()
proj_data.shape

# Visualize projected data
for i in range(proj_data.shape[0]):
    img = proj_data[i].reshape(height, width)
```

```
        plt.subplot(5, 5, 1 + i)
        plt.imshow(img, cmap='gray')
        plt.tick_params(labelleft=False, labelbottom=
            False, bottom=False, top=False, right=False,
            left=False, which='both')
    plt.show()

    # Compute weights
    w = np.array([np.dot(proj_data, i) for i in
        normalised_training_tensor])
    print(w.shape)

    # Face recognizer function
    def recogniser(test_image_names, train_image_names,
        proj_data, w, t0=2e8, prn=False):
        count = 0
        num_images = 0
        correct_pred = 0
        result = []
        wts = []
        FMR_count = 0
        FNMR_count = 0
        test_image_names2 = sorted(test_image_names)

        for img in test_image_names2:
            unknown_face = plt.imread(TEST_IMG_FOLDER +
                img)
            num_images += 1
            unknown_face_vector = np.array(unknown_face,
                dtype='float64').flatten()
            normalised_uface_vector = np.subtract(
```

```
    unknown_face_vector, mean_face)
w_unknown = np.dot(proj_data,
    normalised_uface_vector)
diff = w - w_unknown
norms = np.linalg.norm(diff, axis=1)
index = np.argmin(norms)
wts.append([count, norms[index]])

if prn: print('Input:' + '.'.join(img.split('
    .')[:2]), end='\t')
count += 1

match = img.split('_')[0] ==
    train_image_names[index].split('_')[0]
if norms[index] < t0:  # It's a face
    if match:
        if prn: print('Matched:' +
            train_image_names[index], end='\t')
        correct_pred += 1
        result.append(1)
    else:
        if prn: print('F/Matched:' +
            train_image_names[index], end='\t')
        result.append(0)
        FMR_count += 1
else:
    if match:
        if prn: print('Unknown face!' +
            train_image_names[index], end='\t')
        FNMR_count += 1
    else:
```

```
                pass
                correct_pred += 1

            if prn: print(norms[index], end=' ')
            if prn: print()

        FMR = FMR_count / num_images
        FNMR = FNMR_count / num_images

        print('Correct predictions: {}/{} = {} \t\t'.
            format(correct_pred, num_images, correct_pred /
             num_images), end=' ')
        print('FMR: {} \t'.format(FMR), end=' ')
        print('FNMR: {} \t'.format(FNMR))

        return wts, result, correct_pred, num_images, FMR
            , FNMR

wts, result, correct_pred, num_images, FMR, FNMR =
    recogniser(test_image_names, train_image_names,
    proj_data, w, t0=2e8, prn=True)

# Visualization function
count = 0
num_images = 0
correct_pred = 0

def Visualization(img, train_image_names, proj_data,
    w, t0):
        global count, num_images, correct_pred
        unknown_face = plt.imread(TEST_IMG_FOLDER + img)
```

```
num_images += 1
unknown_face_vector = np.array(unknown_face,
    dtype='float64').flatten()
normalised_uface_vector = np.subtract(
    unknown_face_vector, mean_face)

plt.subplot(40, 2, 1 + count)
plt.imshow(unknown_face, cmap='gray')
plt.title('Input:' + '.'.join(img.split('.')[:2])
    )
plt.tick_params(labelleft=False, labelbottom=
    False, bottom=False, top=False, right=False,
    left=False, which='both')
count += 1

w_unknown = np.dot(proj_data,
    normalised_uface_vector)
diff = w - w_unknown
norms = np.linalg.norm(diff, axis=1)
index = np.argmin(norms)

plt.subplot(40, 2, 1 + count)
if norms[index] < t0:  # It's a face
    match = img.split('_')[0] ==
        train_image_names[index].split('_')[0]
    if match:
        plt.title('Matched:', color='g')
        plt.imshow(imread(TRAIN_IMG_FOLDER +
            train_image_names[index]), cmap='gray')
        correct_pred += 1
    else:
```

```
                plt.title('False matched:', color='r')
                plt.imshow(imread(TRAIN_IMG_FOLDER +
                    train_image_names[index]), cmap='gray')
        else:
            if img.split('_')[0] not in [i.split('_')[0]
                for i in train_image_names]:
                plt.title('Unknown face', color='g')
                correct_pred += 1
            else:
                plt.title('Unknown face', color='r')

        plt.tick_params(labelleft=False, labelbottom=
            False, bottom=False, top=False, right=False,
            left=False, which='both')
        plt.subplots_adjust(right=1.2, top=2.5)
        count += 1

fig = plt.figure(figsize=(5, 30))

test_image_names2 = sorted(test_image_names)
for i in range(len(test_image_names2)):
        Visualization(test_image_names2[i],
            train_image_names, proj_data, w, t0=2.7e7)

plt.show()
```

Listing 4.1: Example Python Code

Chương 5

Kết Luận

Nhận diện khuôn mặt đã trở thành một lĩnh vực nghiên cứu quan trọng và có ứng dụng rộng rãi trong nhiều ngành công nghiệp hiện đại, từ an ninh và giám sát đến tương tác người-máy và các dịch vụ cá nhân hóa. Trong bối cảnh này, phương pháp Phân Tích Thành Phần Chính (PCA) nổi lên như một công cụ hiệu quả và hữu ích để giảm chiều dữ liệu và trích xuất các đặc trưng quan trọng từ hình ảnh khuôn mặt.

Trong quá trình nghiên cứu, chúng ta đã tìm hiểu về cơ sở toán học của PCA, bao gồm các khái niệm về kỳ vọng, phương sai, hiệp phương sai, và ma trận hiệp phương sai. Việc áp dụng các công cụ toán học này giúp chuyển đổi dữ liệu từ không gian ban đầu sang một không gian mới ít chiều hơn nhưng vẫn giữ được phần lớn thông tin quan trọng. Chúng ta cũng đã thấy cách PCA giúp trích xuất các thành phần chính từ dữ liệu khuôn mặt, cho phép xây dựng các hệ thống nhận diện khuôn mặt với độ chính xác cao và hiệu quả tính toán.

PCA không chỉ giúp giảm bớt độ phức tạp của dữ liệu mà còn tối ưu hóa quá trình nhận diện khuôn mặt bằng cách loại bỏ những đặc trưng ít quan trọng, giữ lại những đặc trưng quan trọng nhất. Điều này làm giảm thời gian xử lý, tiết kiệm tài nguyên và cải thiện hiệu suất của hệ thống. Các ứng dụng thực tế của PCA trong nhận diện khuôn mặt rất đa dạng, từ an ninh, giám sát đến các dịch vụ tiện ích hàng ngày như mở khóa thiết bị và chấm công.

Tuy nhiên, PCA cũng có những hạn chế như giả định tính tuyến tính

của các mối quan hệ giữa các biến và khả năng mất thông tin quan trọng trong quá trình giảm chiều dữ liệu. Để khắc phục những hạn chế này, các phương pháp nâng cao như Kernel PCA đã được phát triển để xử lý dữ liệu phi tuyến tính và cải thiện độ chính xác của việc trích xuất đặc trưng.

Trong tương lai, sự phát triển của công nghệ và các phương pháp học máy mới sẽ tiếp tục cải tiến và hoàn thiện các hệ thống nhận diện khuôn mặt. PCA và các phương pháp liên quan sẽ vẫn giữ vai trò quan trọng trong việc xử lý và phân tích dữ liệu lớn, đóng góp vào sự phát triển của các ứng dụng nhận diện khuôn mặt ngày càng thông minh và hiệu quả hơn.

Tóm lại, PCA là một phương pháp mạnh mẽ và hiệu quả trong việc xử lý và phân tích dữ liệu khuôn mặt. Việc hiểu và áp dụng đúng PCA không chỉ giúp chúng ta xây dựng các hệ thống nhận diện khuôn mặt tốt hơn mà còn mở ra nhiều cơ hội ứng dụng khác trong các lĩnh vực nghiên cứu và công nghiệp.