ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN KHOA HỌC MÁY TÍNH

NHẬN DẠNG

BÁO CÁO SEMINAR CUỐI KÌ

FACE RECOGNITION

Giảng viên lý thuyết
TS. Lê Hoàng Thái

Giảng viên hướng dẫn

Nguyễn Ngọc Thảo, Lê Thanh Phong

Sinh viên thực hiện

Nguyễn Thị Thu Hằng, Nguyễn Nam Phong, Đinh Nguyễn Minh Quân

Tháng 6 năm 2021

Mục lục

1 Thông tin chung

- **Tên đề tài:** 2D face recognition
- **Tên môn học:** Nhận dạng
- **Giảng viên lý thuyết:** TS.Lê Hoàng Thái
- **Giảng viên hướng dẫn:** Nguyễn Ngọc Thảo, Lê Thanh Phong
- **Sinh viên thực hiện:**

Họ và tên	Mã số sinh viên	Công việc
Nguyễn Thị Thu Hằng	18120027	Xây dựng demo thực nghiệm, so sánh và đánh giá các phương pháp
Nguyễn Nam Phong	18120506	Tìm hiểu về các phương pháp giảm chiều Kernel PCA, LLE
Đinh Nguyễn Minh Quân	18120506	Tìm hiểu về phương pháp giảm chiều state-of-the-art hiện nay(facenet), phương pháp phân lớp SVM, một số database phổ biến

2 Mở đầu

Face recognition là một quá trình giúp giải quyết bài toán nhận diện (Identify) và xác thực danh tính (Verify) của một người dựa trên ảnh chụp khuôn mặt của họ. Bài toán nhận diện khuôn mặt (face identification) sẽ trả lời cho câu hỏi "người này là ai?". Bài toán này thường được áp dụng trong các hệ thống chấm công, hệ thống giám sát công dân, hệ thống cammera thông minh tại các đô thị. Trong khi đó xác thực khuôn mặt (face verification) sẽ trả lời cho câu hỏi "có phải người này là A không?". Bài toán này thường được dùng trong các hệ thống bảo mật như tính năng mở khóa khuôn mặt trên các mẫu điện thoại thông minh hiện nay.

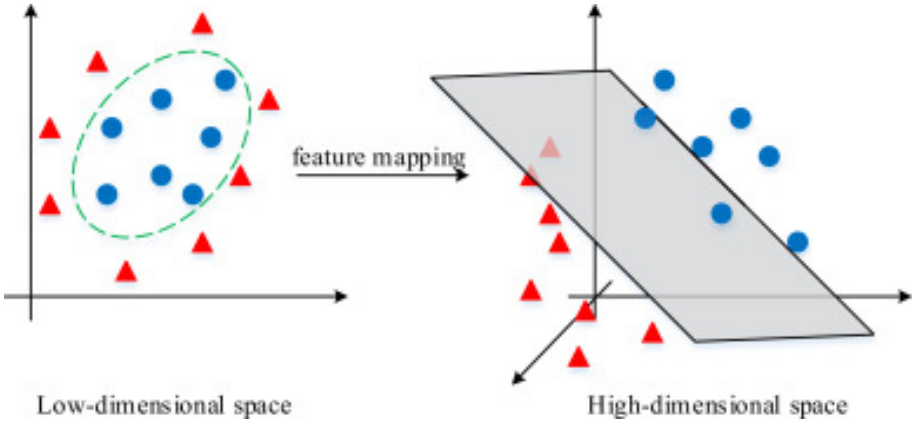
Quá trình face recognition thường bao gồm 4 bước chính là: Face Detection - tách vùng chứa khuôn mặt từ ảnh ban đầu, Preprocessin - tiền xử lý ảnh, Feature Extraction - rút trích đặt trưng và Classification - gán nhãn cho ảnh. Trong quá trình này, việc học các đặt trưng khuôn mặt thường mất khá nhiều thời gian bởi vì dữ liệu được sử dụng trong nhận dạng khuôn mặt thường có số chiều rất lớn. Do đó, để cắt giảm thời gian phân tích mà vẫn không làm ảnh hướng nhiều đến chất lượng của mô hình, ta cần sử dụng các phương pháp giảm chiều dữ liệu nhưng vẫn có thể giữ được phần lớn thông tin để rút trích đặt trưng. Phạm vi bài nghiên cứu của chúng tôi tập trung chủ yếu vào các phương pháp rút trích đặt trưng dựa trên *Chương 3: Face Recognition - Handbook of Biometrics* và nội dung môn học. Cụ thể, ở mục 3 và 4 trong bài báo cáo này chúng tôi trình bày về các phương pháp dùng để rút trích đặt trưng Kernel Principal Component Analysis, Locally Linear Embedding dùng cho dữ liệu có phân bố non-linear. Mục 5 chúng tôi trình bày sơ bộ một phương pháp *state-of-the-art* để rút trích đặt trưng hiện nay là Facenet. Mục 6 chúng tôi giới thiệu về một số database phổ biến về face recognition. Mục 7 chúng tôi trình bày về phương pháp Support Vector Machine dùng để phân lớp và gán nhãn cho ảnh. Mục 8, 9 là tóm tắt về quá trình, các bước xây dựng và kết quả của demo thực nghiệm.

3 Kernel Principal Component Analysis

3.1 Tổng quan về Kernel Principal Component Analysis

Ta đã biết cơ chế để PCA giảm chiều dữ liệu là nó sẽ tìm một hệ trực chuẩn để làm cơ sở mới sao cho thông tin của dữ liệu chủ yếu tập trung ở một vài chiều và một số chiều còn lại chỉ mang một lượng nhỏ thông tin. Cụ thể hơn thì PCA sẽ cố gắn tìm ra các đường hoặc mặt đặc biệt có dạng tuyến tính nào đó và chiếu các điểm dữ liệu lên các đường hoặc mặt tìm được nhằm giảm bớt chiều dữ liệu. Tuy nhiên không phải lúc nào dữ liệu cũng phân bố gần với các đường hoặc mặt đặc biệt có dạng tuyến tính vì vậy việc sử dụng PCA trong trường hợp này là không hiệu quả. Kernel PCA sẽ giải quyết vấn đề này.

Kernel PCA sẽ tiến hành giảm bằng cách tăng chiều. Thoạt đầu nghe có vẻ mâu thuẫn nhưng ta hãy quan sát hình sau đây



Nhìn hình bên trái có các điểm dữ liệu được biểu diễn trong không gian 2 chiều. Với linear PCA, một cách trực quan ta thấy không thể tìm được đường hay mặt nào mà khi ta chiếu các điểm dữ liệu lên đó thì variance vẫn cao. Với phương pháp Kernel PCA, ta giả sử tồn tại mặt tuyến tính nào đó ở không gian cao hơn cho phép ta tiến hành giảm chiều bằng PCA.

Xét vector đặc trưng x $= [x_1, x_2, .., x_D]$ có D chiều và tập dữ liệu có N vector đặc trưng X $= [\mathrm{x}_1^T, x_2^T, .., x_N^T]$.

Ta gọi ϕ là một ánh xạ từ không gian D chiều sang không gian K chiều

$$\phi : \mathbb{R}^D \mapsto \mathbb{R}^K$$

Ma trận Kernel K được định nghĩa như sau:

$$\phi(X)\phi(X)^T = K = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & k(x_1,x_3) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & k(x_2,x_3) & \dots & k(x_2,x_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k(x_d,x_1) & k(x_d,x_2) & k(x_d,x_3) & \dots & k(x_d,x_n) \end{bmatrix}$$

Trong đó $k(x_1, x_2)$ là một hàm kernel và $\phi(x_1)\phi(x_2)^T = k(x_1, x_2)$. Sau đây là một số hàm kernel phổ biến:

- **Tuyến tính:**
$$k(X, Z) = XZ^T \tag{1}$$
- **Đa thức:**
$$k(X, Z) = (XZ^T + r)^d \tag{2}$$
Với d là bậc của đa thức và r là hệ số tự do.
- **Radial Basis Function:**
$$k(X, Z) = e^{-\frac{\|(X-Z)\|_2^2}{2\sigma}} \tag{3}$$
Với σ là tham số đại diện cho "độ dàn trải" của Kernel.

Ví dụ:

Ta có các điểm dữ liệu trong không gian 2 chiều X = $[x_1, x_2]$, Z = $[z_1, z_2]$, ánh xạ $\phi(X) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2]$.

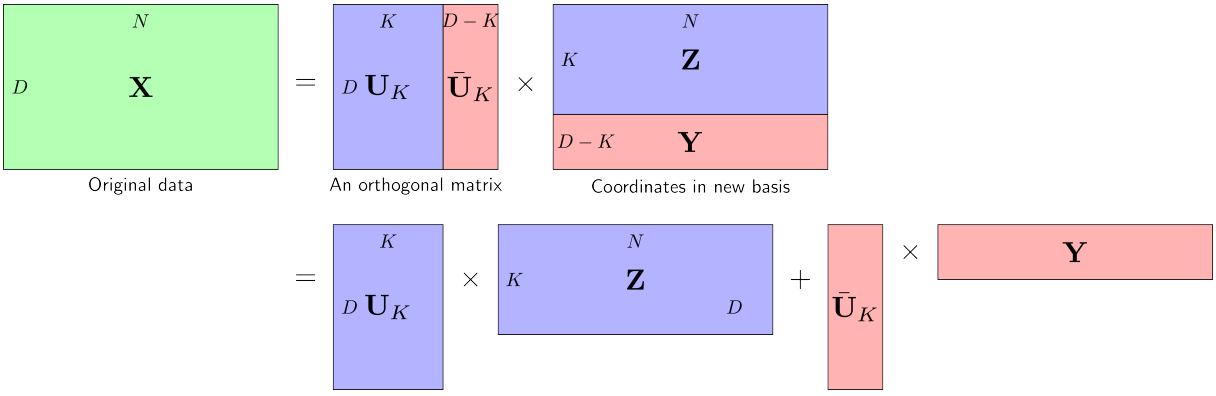
Khi đó $\phi(X)\phi(Z)^T = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2][1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1z_2, z_2^2]^T = 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 = (1 + x_1z_1 + x_2z_2)^2 = (1 + XZ^T)^2 = k(X, Z)$

3.2 Các bước thực hiện Kernel Principal Component Analysis

- **Bước 1: Chọn hàm kernel**
- **Bước 2: Tính ma trận kernel K**

  Gọi X là tập các điểm dữ liệu ban đầu trong không gian D và $\phi(X)$ là tập các điểm dữ liệu trong không gian H. Với phương pháp PCA, đầu tiên ta cần tính ma trận covariance $S = \phi(X)\phi(X)^T$. Tuy nhiên việc ánh xạ tất cả các điểm dữ liệu trong X từ không gian D sang không gian H rồi tính tích vô hướng của các điểm dữ liệu để tìm ma trận S sẽ tốn nhiều chi phí. Sử dụng việc $\phi(x)\phi(x)^T = k(x, x)$ với $k(x, x)$ chính là một hàm Kernel. Ta không cần phải ánh xạ tất cả các điểm dữ liệu sang chiều không gian mới rồi mới có thể tính được ma trận $S = \phi(X)\phi(X)^T$ mà có thể sử dụng ma trận Kernel K được giới thiệu ở phần trên.

- **Bước 3: Centering ma trận Kernel**



Mục đích của PCA là đi tìm ma trận trực giao U sao cho phần lớn thông tin được giữ ở phần U_KZ và phần $\overline{\mathrm{U}}_KY$ sẽ bị lược bỏ. Khi đó ta sẽ xấp xỉ Y bởi một ma trận có toàn bộ các cột là như nhau. Gọi b là một cột của ma trận Y, khi đó ta cần tìm b thỏa mãn

$$b = argmin_b\|(Y - b1^T)\|_2^2 = argmin_b\|(\overline{\mathrm{U}}_K^TX - b1^T)\|_2^2 \tag{4}$$

Giải phương trình đạo hàm theo b của hàm trên bằng 0 ta được:

$$b = \overline{\mathrm{U}}_K^T\overline{\mathrm{x}}$$

Việc tính toán sẽ thuận tiện hơn nhiều nếu vector kỳ vọng $\overline{\mathrm{x}} = 0$. Vậy trước khi tính ma trận covariance chúng ta trừ mỗi vector dữ liệu đi vector kỳ vọng của toàn bộ dữ liệu. Nghĩa là ta ánh xạ điểm dữ liệu X_i từ không gian D gian không gian H sử dụng

$$\overline{\phi}(X_m) = \phi(X_m) - \frac{1}{N}\sum_{m=1}^{N}\phi(X_m)$$

Khi đó hàm Kernel mới sẽ là:

$$\overline{\mathrm{k}}(x_i, x_j) = \overline{\phi}(x_i)\overline{\phi}(x_j)^T = (\phi(x_i) - \frac{1}{N}\sum_{m=1}^{N}\phi(x_m))(\phi(x_j) - \frac{1}{N}\sum_{m=1}^{N}\phi(x_m))^T$$

Hay

$$\bar{\mathrm{k}}(x_i, x_j) = k(x_i, x_j) - \frac{1}{N}\sum_{m=1}^{N} k(x_i, x_m) - \frac{1}{N}\sum_{m=1}^{N} k(x_j, x_m) + \frac{1}{N^2}\sum_{n,m=1}^{N} k(x_n, x_m)$$

Ma trận Kernel K_c mới khi đó là:

$$K_C = K - 1_N K - K 1_N + 1_N K 1_N$$

Trong đó 1_N là ma trận $N \times N$ có mỗi phần tử mang giá trị $\frac{1}{N}$ và K là ma trận Kernel đã tính ở bước 2.

- **Bước 4:**

  Tìm các eigenvector α và eigenvalue λ của ma trận K_c.

$$K\alpha = N\lambda\alpha$$

- **Bước 5:**

  Sắp xếp các eigenvector vừa tìm được theo thứ tự giảm dần của eigenvalue.

- **Bước 6:**

  Chọn ra K eigenvector đầu tiên và chiếu dữ liệu lên chiều không gian mới.

$$y_i = \sum_{i=1}^{D} \alpha_{ij} k(x_i, x), j = 1, 2, .., K$$

3.3 Nhận xét

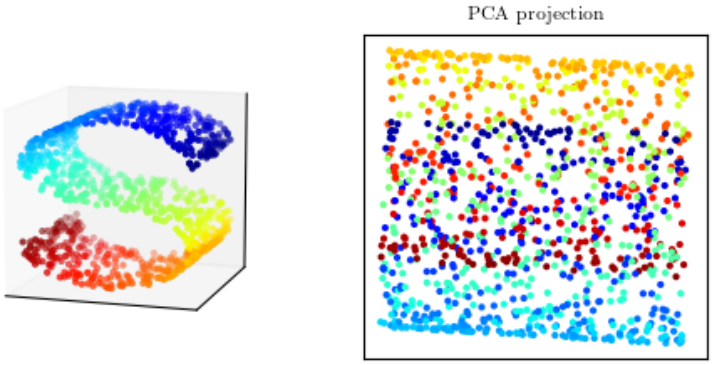
Độ phức tạp của phương pháp Kernel Principal Component Analysis không tăng theo số chiều của feature space do Kernel PCA không cần phải trực tiếp chiếu dữ liệu lên chiều không gian mới rồi tính dot product của các điểm dữ liệu mà dùng hàm kernel để tính toán. Vì vậy Kernel PCA có thể ánh xạ các điểm dữ liệu từ không gian D chiều ban đầu sang không gian H chiều ($D < H$) bất kỳ mà chi phí không tăng đáng kể.

So với các phương pháp optimization như Multi Layer Perceptron hay Deep Learning thì phương pháp Kernel PCA không bị kẹt ở cục bộ địa phương trong quá trình huấn luyện bởi Kernel PCA chỉ cần phải giải quyết vấn đề eigenvalue để tìm ra các eigenvector.

4 Manifold Learning

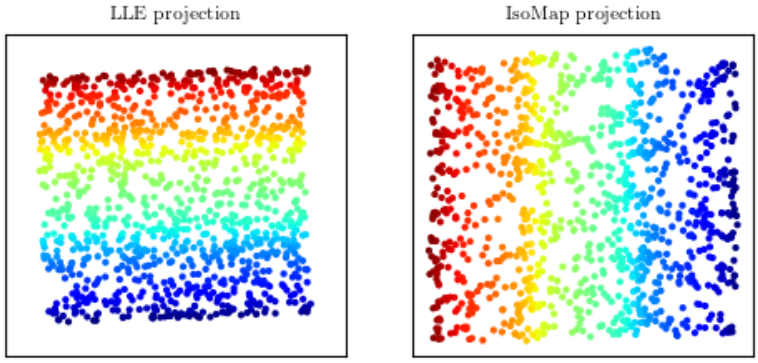
4.1 Tổng quan về Manifold Learning

Có lẽ khi nhắc đến giảm chiều dữ liệu, người ta sẽ nghĩ ngay đến Principal component analysis (PCA), phương pháp giảm chiều dữ liệu khá phổ biến và được sử dụng rộng rãi. Tuy nhiên điểm yếu của phương pháp này chính là nó chỉ có thể áp dụng lên tập dữ liệu có cấu trúc phân bố tuyến tính, không hiệu quả khi tập dữ liệu có cấu trúc không tuyến tính, hay còn gọi là manifold (đa tạp).



Hình 1: Khi áp dụng PCA lên cấu trúc manifold, tập dữ liệu tập tức bị mất cấu trúc và các điểm dữ liệu trộn lẫn vào nhau. Điều này khiến cho mô hình mất đi tính hiệu quả và độ lỗi sẽ cất cao.

Qua nhiều lần phân tích dữ liệu hình ảnh khuôn mặt, người ta thường thấy dữ liệu hình ảnh khuôn mặt với những tư thế và cảm xúc đa dạng thường có cấu trúc manifold. Chính vì vậy, các nhà nghiên cứu đã giới thiệu các phương pháp Manifold Learning, để giữ lại được cấu trúc dữ liệu sao cho các điểm gần nhau vẫn sẽ gần nhau và các điểm xa nhau vẫn sẽ xa nhau sau khi cắt giảm dữ liệu (có thể gọi là bảo toàn cấu trúc cục bộ). Hai phương pháp thường được sử dụng là Isomap và Locally Linear Embedding (LLE). Cụ thể, ta sẽ đi vào phương pháp LLE - một phương pháp sử dụng rộng rãi đối với các tập dữ liệu có cấu trúc manifold.



Hình 2: Khi áp dụng Isomap và LLE, ta thấy được các điểm không còn trộn lẫn vào nhau mà vẫn giữ được cấu trúc xung quanh nó.

4.2 Locally Linear Embedding

Thuật toán Locally Linear Embedding (LLE) lấy ý tưởng dựa trên hình học. Vốn dĩ, ta không biết được hình dáng thực sự của cấu trúc manifold mà tập dữ liệu thể hiện, nhưng điều ta biết được là ở một vùng cục bộ đủ nhỏ của cấu trúc này, các điểm sẽ phân bố tạo thành cấu trúc gần với cấu trúc tuyến tính (tương tự như đường tiếp tuyến với 1 đường cong mà ta đã học ở cấp 2, khoảng xung quanh điểm tiếp xúc càng nhỏ thì có xu hướng trùng với đường tiếp tuyến). Phương pháp LLE dựa vào đặt điểm này để xét vùng cục bộ "tuyến tính" được tạo thành ở mỗi điểm và các điểm láng giềng của nó. Khi đó chỉ việc thực hiện giảm chiều sao cho cấu trúc của vùng cục bộ đó vẫn bảo toàn.

Phương pháp LLE thực hiện theo ba bước:

- **Bước 1:** Với mỗi điểm dữ liệu p chiều, ta tìm k điểm dữ liệu gần nhất với nó (k láng giềng).
- **Bước 2:** Tìm ma trận trọng số W sao cho tổng bình phương lỗi khi tái cấu trúc tuyến tính mỗi x_i sau đạt tối thiểu:

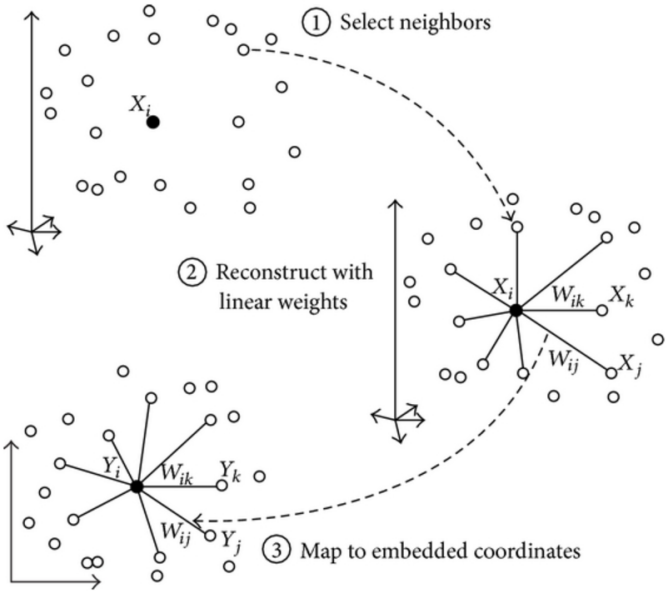
$$\epsilon(W) = \sum_{i=1}^{n} \left\| x_i - \sum_{j=1}^{n} w_{ij} x_j \right\|^2 \tag{5}$$

  Với $w_{ij} = 0$ nếu x_j không nằm trong láng giềng của x_i.

- **Bước 3:** Sử dụng ma trận W đã tìm được, tìm Y sao cho tổng bình phương lỗi của quá trình tái cấu trúc đạt tối thiểu:

$$\Phi(Y) = \sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{ij} y_j \right\|^2 \tag{6}$$

Sau khi hoàn thành 3 bước, ta đã có được bộ dữ liệu Y với số chiều đã giảm với y_i là điểm dữ liệu mới gồm q chiều (q là số chiều ta mong muốn, $q << p$).



Hình 3: Các bước thực hiện của thuật toán LLE.

4.2.1 Tìm k láng giềng của mỗi điểm x_i

Với k là tham số đã biết, ở mỗi điểm x_i, ta sẽ tìm tập k điểm x_j gần nhất với x_i. Để được như vậy, ta sẽ sử dụng thuật toán *k-nearest neighbors* để xác định. Như vậy x_i và k láng giềng của nó sẽ tạo thành 1 vùng cục bộ tuyến tính mà ta cần xét.

4.2.2 Tính ma trận trọng số W

Xét mỗi điểm dữ liệu x_i, ta có k điểm láng giềng x_j cùng trọng số của nó w_{ij} của nó, sẽ tạo thành một vùng cục bộ xấp xỉ tuyến tính. Ta cấu trúc lại x_i như 1 hàm tuyến tính được tạo bởi tổng tuyến tình của các x_j và trọng số của nó:

$$x_i = \sum_j w_{ij} x_j \tag{7}$$

Lúc này, sai số trong tổng bình phương phép xấp xỉ tuyến tính của các x_i càng nhỏ càng tốt:

$$W = \underset{W}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| x_i - \sum_{j=1}^{n} w_{ij} x_j \right\|^2 \tag{8}$$

Tuy nhiên, các w_i không có mối quan hệ gì với nhau, tất cả đều độc lập. Nên chỉ cần ở mỗi x_i ta tính được:

$$w_i = \underset{w_i}{\operatorname{argmin}} \left\| x_i - \sum_j w_{ij} x_j \right\|^2 \tag{9}$$

Để kết quả bất biến với phép tịnh tiến, ta sẽ thêm vào ràng buộc $\sum_j w_{ij} = 1$, thật vậy giả sử ta tịnh tiến dữ liệu 1 khoảng t:

$$(x_i + t) - \sum_j w_{ij}(x_j + t) = x_i - \sum_j w_{ij} x_j + t - t \sum_j w_{ij} = x_i - \sum_j w_{ij} x_j \tag{10}$$

Để tiện với tính toán, ta đặt $z_j = x_j = x_i$, lúc này, công thức trở thành:

$$\epsilon w_i = \left\| \sum_j w_{ij} z_j \right\|^2 = w_i^T z_i z_i^T w_i \tag{11}$$

Đặt $G_i = z_i z_i^T$, ta có $\epsilon_i = w_i^T G_i w_i$ (G_i lúc này là Gram Matrix).

Ngoài ra do có ràng buộc $\sum_j w_{ij} = 1$ (viết lại thành $1^T w_i - 1 = 0$), ta sẽ sử dụng phương pháp nhân tử Lagrange, ta được hàm cần tối tiểu:

$$L(w_i, \lambda) = w_i^T G_i w_i - \lambda(1^T w_i - 1) \tag{12}$$

Để $L(w_i, \lambda)$ đạt cực tiểu, ta lấy đạo hàm theo w_i bằng 0:

$$\frac{\partial L}{\partial w_i} = 2 G_i w_i - \lambda 1 = 0 \tag{13}$$

Khi G khả nghịch ta tìm được kết quả w_i:

$$w_i = \frac{\lambda}{2} G_i^{-1} 1 \tag{14}$$

Lúc này ta có thể dễ dàng tính được w_i bằng cách đặt $\lambda = 1$ rồi giải ra kết quả sau đó chia hệ số sao cho $\sum_j w_{ij} = 1$.

4.2.3 Trường hợp đặc biệt: $k > p$

Khi $k > p$, ta sẽ có vô số cách giải bởi vì số lượng ẩn w_{ij} nhiều hơn số phương trình trong hệ phương trình được tạo từ $x_i = \sum_j w_{ij} x_j$. Để giải quyết trường hợp này, ta sẽ sử dụng một phương pháp gọi là L2 hay Tikhonov regularization. Khi đó hàm cần tối tiểu của ta trở thành:

$$w_i^T G_i w_i + \alpha w_i^T w_i \tag{15}$$

Với α là bậc regularization, thường thuộc khoảng $\left[10^{-3}, 10^{-1}\right]$.

Sử dụng phương pháp Lagrange để thêm ràng buộc:

$$L(w_i, \lambda) = w_i^T G_i w_i + \alpha w_i^T w_i - \lambda(1^T w_i - 1) \tag{16}$$

Lấy đạo hàm bằng không để tìm cực tiểu, ta được:

$$w_i = \frac{\lambda}{2} (G_i + \alpha I)^{-1} 1 \tag{17}$$

Thực hiện với mỗi x_i, ta thu được 1 ma trận trọng số W cần tìm để thực hiện giảm chiều dữ liệu.

4.2.4 Tính tập dữ liệu Y có số chiều giảm dựa vào W đã tìm được

Để tính toán được tập dữ liệu Y với số chiều dữ liệu q mong muốn, ta sẽ tối tiểu hóa hàm $\Phi(Y)$:

$$\Phi(Y) = \sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{ij} y_j \right\|^2 \tag{18}$$

Rút gọn hàm Φ về dạng ma trận, ta được:

$$\Phi(Y) = Y^T (I - W)^T (I - W) Y \tag{19}$$

Đặt $M = (I - W)^T (I - W)$, lúc này M là một ma trận đối xứng $n \times n$

$$\Phi(Y) = Y^T M Y \tag{20}$$

Ta lại nhớ rằng, mô hình của chúng ta bất biến với phép tịnh tiến, chính vì thế, nó sẽ cho ra vô vàn kết quả. Ta cần có một ràng buộc:

$$\frac{1}{n} \sum_{i=1}^{n} y_i = 0 \tag{21}$$

Viêc lấy trung bình bằng 0 ngăn chúng ta có nhiều lời giải khi thực hiện phép tịnh tiến, ngoài ra ta cũng có thêm ràng buộc:

$$\frac{1}{n}Y^TY = I_q \tag{22}$$

Trước hết, ta xét $q = 1$, lúc này $I_q = 1$. Áp dụng phương pháp Lagrange, lúc này hàm mục tiêu của chúng ta sẽ trở thành:

$$L(Y, \mu) = Y^TMY - \mu(\frac{1}{n}Y^TY - 1) \tag{23}$$

Để $L(Y, \mu)$ đạt tối thiểu, ta lấy cực tiểu bằng cách cho đạo hàm bằng 0:

$$\frac{\partial L(Y, \mu)}{\partial Y} = 2MY - \frac{2\mu}{n}Y = 0 \tag{24}$$

$$MY = \frac{\mu}{n}Y \tag{25}$$

Ta thu được một phương trình giống với phương trình của trị riêng và vector riêng. M là một ma trận nxn do đó có thể có n trị riêng tương ứng với n vector riêng. Nhận thấy rằng cặp trị riêng và vector riêng nhỏ nhất lần lượt là 0 và 1:

$$M1 = 0 \tag{26}$$

Vector riêng 1 hoàn toàn là một hằng số, do đó nó không giúp ích được gì cho bài toán. Khi đó ta sẽ tính thêm một vector riêng có trị riêng nhỏ nhì và lấy nó như đáp án bài toán. Lưu ý rằng chúng ta đang xét $q = 1$ thì Y trong phương trình đang là 1 vector cũng là đáp án của bài toán. Với $q > 1$, ta sẽ cần tìm $q + 1$ vector riêng có trị riêng nhỏ nhất (sau đó bỏ đi vector riêng 1 và lấy q vector riêng còn lại làm kết quả). q vector riêng sẽ tạo thành ma trận Y, ta thu về được dữ liệu với số chiều q đã giảm ($q << p$).

4.3 Nhận xét

Manifold Learning là nhóm thuật toán hữu hiệu cho việc giảm chiều dữ liệu trên tập dữ liệu có cấu trúc không tuyến tính. So với PCA, Manifold Learning thích hợp hơn trong các bài toán nhận diện khuôn mặt nơi mà các dữ liệu về khuôn mặt đa dạng về tư thế (nghiêng đầu, ngửa đầu, cúi đầu,...) và cảm xúc (buồn, vui, khóc,..). Locally Linear Embedding mang ý tưởng bảo toàn các vùng "tuyến tính" cục bộ nhờ đó mà các điểm dữ liệu dù được giảm chiều vẫn giữ được mối liên hệ với nhau, không ảnh hưởng xấu đến chất lượng mô hình

5 FaceNet - State of the art method

5.1 Tổng quan về Facenet

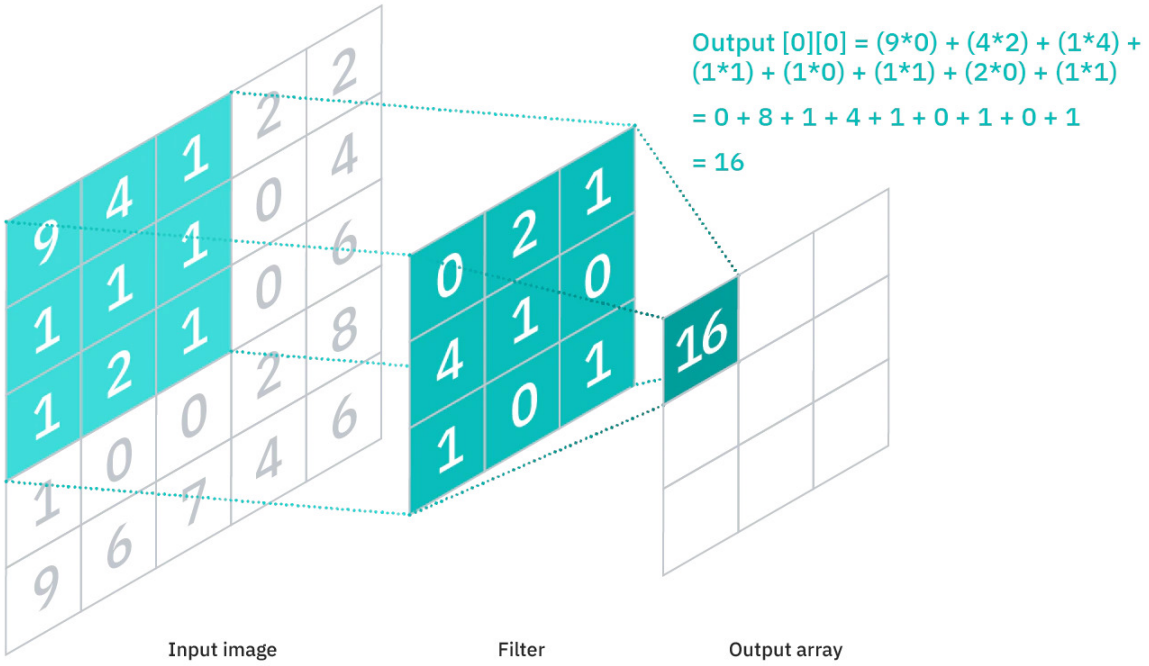
FaceNet, một phương pháp được Google giới thiệu vào năm 2015, phương pháp được xuất phát dựa trên nghiên cứu "FaceNet: Một hệ thống nhúng cho việc nhận dạng và phân cụm khuôn mặt" được đề xuất bởi nhóm tác giả làm việc tại Google đó là Florian Schroff, Dmitry Kalenichenko và James Philbin.

FaceNet dựa trên việc nhúng mỗi ảnh vào không gian Euclide bằng cách sử dụng mạng CNN. Mạng được huấn luyện sao cho khoảng cách bình phương L2 trong không gian nhúng tương ứng với khoảng cách giữa các khuôn mặt: Khuôn mặt của cùng một người có khoảng cách nhỏ và khuôn mặt của những người khác nhau có khoảng cách lớn. Khi đó, mỗi khuôn mặt được đại diện bởi một vector đặc trưng 128 chiều được chuyển đổi thành 128 byte.

5.2 Convolution Neural Network (Mạng Neural Tích Chập)

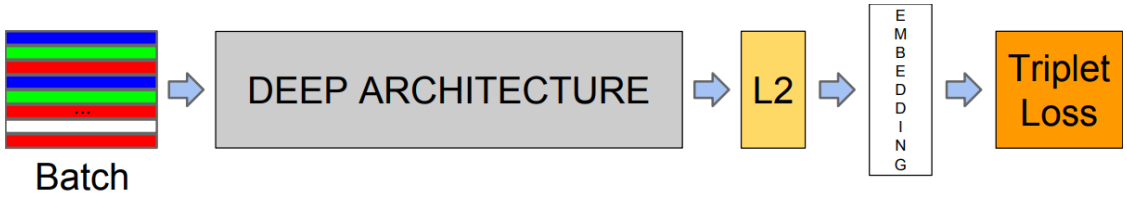
CNN_là một trong những mô hình Deep Learning tiên tiến. CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection).

Là một cửa sổ trượt (Sliding Windows) trên một ma trận như mô tả hình dưới:



Ảnh minh họa cách hoạt động của CNN

5.3 Kiến trúc mạng FaceNet



Ảnh minh họa: Kiến trúc FaceNet

Facenet chính là một dạng siam network có tác dụng biểu diễn các bức ảnh trong một không gian eucledean chiều (thường là 128) sao cho khoảng cách giữa các véc tơ embedding càng nhỏ, mức độ tương đồng giữa chúng càng lớn.

Hầu hết các thuật toán nhận diện khuôn mặt trước facenet đều tìm cách biểu diễn khuôn mặt bằng một véc tơ embedding thông qua một layer bottle neck có tác dụng giảm chiều dữ liệu.

- Tuy nhiên hạn chế của các thuật toán này đó là số lượng chiều embedding tương đối lớn (thường >= 1000) và ảnh hưởng tới tốc độ của thuật toán. Thường chúng ta phải áp dụng thêm thuật toán PCA để giảm chiều dữ liệu để tăng tốc độ tính toán.
- Hàm loss function chỉ đo lường khoảng cách giữa 2 bức ảnh. Như vậy trong một đầu vào huấn luyện chỉ học được một trong hai khả năng là sự giống nhau nếu chúng cùng 1 class hoặc sự khác nhau nếu chúng khác class mà không học được cùng lúc sự giống nhau và khác nhau trên cùng một lượt huấn luyện.

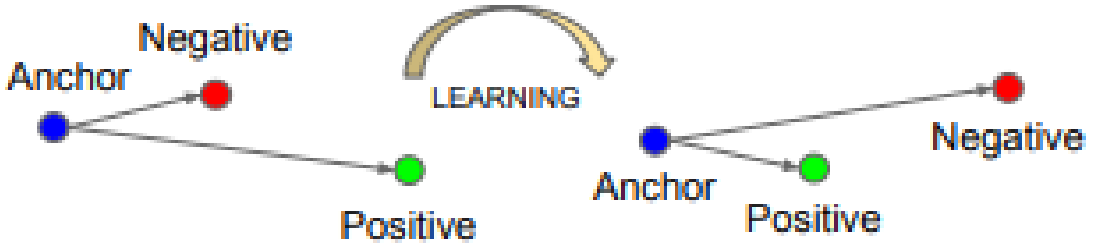
Facenet đã giải quyết cả 2 vấn đề trên bằng các hiệu chỉnh nhỏ nhưng mang lại hiệu quả lớn:

- Base network áp dụng một mạng convolutional neural network và giảm chiều dữ liệu xuống chỉ còn 128 chiều. Do đó quá trình suy diễn và dự báo nhanh hơn và đồng thời độ chính xác vẫn được đảm bảo..
- Sử dụng loss function là hàm triplot loss có khả năng học được đồng thời sự giống nhau giữa 2 bức ảnh cùng nhóm và phân biệt các bức ảnh không cùng nhóm. Do đó hiệu quả hơn rất nhiều so với các phương pháp trước đây.

5.3.1 Triplet Loss

Để áp dụng triple loss, chúng ta cần lấy ra 3 bức ảnh trong đó có một bức ảnh là anchor. Chúng ta sẽ lựa chọn 2 ảnh còn lại sao cho một ảnh là negative (của một người khác với anchor) và một ảnh là positive (cùng một người với anchor). Mục tiêu của hàm loss function là tối thiểu hóa khoảng cách giữa 2 ảnh khi chúng là negative và tối đa hóa khoảng cách khi chúng là positive. Như vậy chúng ta cần lựa chọn các bộ 3 ảnh sao cho:

- Ảnh Anchor và Positive khác nhau nhất: cần lựa chọn để khoảng cách lớn. Điều này cũng tương tự như bạn lựa chọn một ảnh của mình hồi nhỏ so với hiện tại để thuật toán học khó hơn. Nhưng nếu nhận biết được thì nó sẽ thông minh hơn..
- Ảnh Anchor và Negative giống nhau nhất: cần lựa chọn để khoảng cách nhỏ. Điều này tương tự như việc thuật toán phân biệt được ảnh của một người anh em giống bạn với bạn.



Quá trình chọn bộ tham số của Triplet Loss sao cho các ảnh thuộc cùng 1 lớp ở gần nhau và khác lớn ở xa nhau

5.3.2 Triplet Loss tốt nhất

Ý tưởng là chúng ta cần tìm ra bộ ba sao cho là gần đạt được đẳng thức (xảy ra dấu =) nhất. Tức là lớn nhất và nhỏ nhất. Hay nói cách khác với mỗi Anchor.Cần xác định:

- Hard Positive: Bức ảnh Positive có khoảng cách xa nhất với Anchor tương ứng với nghiệm:

$$\arg\max_{\mathrm{P}_i} d(\mathrm{A}_i, P_i) \tag{27}$$

- Negative: Bức ảnh Negative có khoảng cách gần nhất với Anchor tương ứng với nghiệm:

$$\arg\min_{\mathrm{N}_i} d(\mathrm{A}_i, N_i) \tag{28}$$

Với i,j là nhãn của người trong ảnh.

Việc tính toán các trường hợp Hard Positive và Hard Negative có thể được thực hiện offline và lưu vào checkpoint hoặc có thể tính toán online trên mỗi mini-batch.

Chiến lược lựa chọn Triple images sẽ có ảnh hưởng rất lớn tới chất lượng của mô hình Facenet. Nếu lựa chọn Triplet images tốt, Facenet sẽ hội tụ nhanh hơn và đồng thời kết quả dự báo chuẩn xác hơn. Lựa chọn ngẫu nhiên dễ dẫn tới thuật toán không hội tụ.

6 Databases

Có không ít những cơ sở dữ liệu (database) công khai cho mục đích nghiên cứu cộng đồng nhằm phát triển các thuật toán, cung những chuẩn đánh giá kết quả mà thuật toán hay mô hình mang lại. Một vài cơ sở dữ liệu phổ biến trong học nhận dạng khuôn mặt con người như FRGC, FERET, PIE, AR, YALE,...

6.1 Face Recognition Grand Challenge (FRGC)

Face Recognition Grand Challenge (FRGC) được tổ chức bởi NIST với mục tiêu chính là thúc đẩy và nâng cao công nghệ nhận dạng khuôn mặt nhằm phục vụ cho các hệ thống nhận dạng của chính phủ Hoa Kỳ.

Cơ sở dữ liệu FRGC gồm 50000 dữ liệu, là cơ sở dữ liệu tương đối phức tạp với những dữ liệu khác nhau về độ chiếu sáng (do môi trường đa dạng), cảm xúc, và hỗ trợ cả dạng hình ảnh 3D. FRGC chia cơ sở dữ liệu làm 2 phần: huấn luyện và xác thực. Trong đó dữ liệu xác thực gồm 4003 chủ đề với mỗi chủ đề là tập hợp các ảnh về một người: 4 ảnh được kiểm soát, 2 ảnh không được kiểm soát và 1 ảnh 3D. Ảnh được kiểm soát là ảnh được chụp với độ sáng chuẩn từ studio với 2 mức sáng khác nhau, góc thẳng, với 2 biểu cảm cười và bình thường. Ảnh không được kiểm soát là ảnh có độ sáng đa dạng ảnh hưởng bởi môi trường và cũng gồm 2 biểu cảm cười và bình thường.

6.2 Face Recognition Technology (FERET)

Trước FRGC, NIST đã tổ chức chương trình Face Recognition Technology (FERET) sử dụng cơ sở dữ liệu FERET. Mục đích của FERET là để đo lường hiệu suất của các thuật toán của các hệ thống nhận dạng khuôn mặt trong thương mại trên các cơ sở dữ liệu lớn. cơ sở dữ liệu FERET bao gồm 14126 hình ảnh khuôn mặt của 1199 người, được thu thập từ tháng 8/1993 đến tháng 7/1996.

6.3 AR

Cơ sở dữ liệu AR được tạo ra bởi Computer Vision Center (CVC), Trường U.A.B. Nó bao gồm 4000 hình ảnh màu về khuôn mặt của 126 người (70 nam và 56 nữ). Các ảnh được chụp thẳng góc với khuôn mặt. Một điều đặc biệt của AR đó là ngoài đa dạng về biểu cảm khuôn mặt hay độ chiếu sáng, nó còn đa dạng về yếu tố như mắt kính, kiểu tóc, trang điểm,... Điều này khiến nó trở thành cơ sở dữ liệu hiệu quả cho các thuật toán nhận diện khuôn mặt với những yếu tố trên.

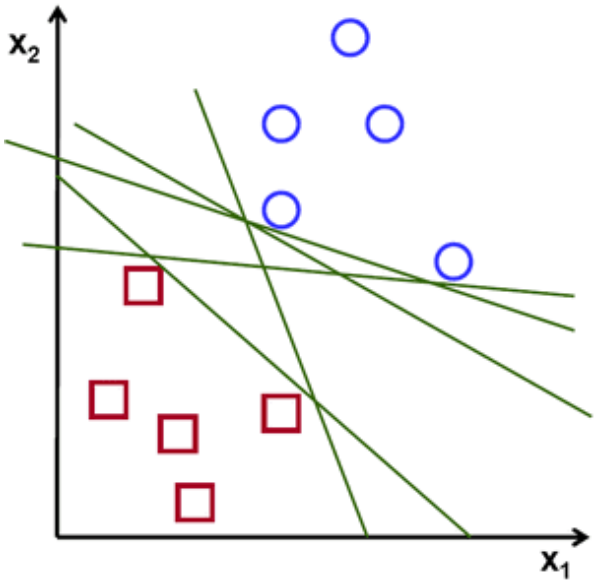
6.4 YALE

YALE là một cơ sở dữ liệu nhỏ gồm 165 hình ảnh xám thuộc 15 chủ đề khác nhau. Mỗi chủ đề về một người với 11 tấm ảnh khuôn mặt khác nhau về biểu hiện: góc sáng giữa, góc sáng trái, góc sáng phải, đeo kính, không đeo kính, vui vẻ, buồn, bình thường, buồn ngủ, ngạc nhiên và nháy mắt. Cơ sở dữ liệu mở rộng của YALE là Yale Face Database B, gồm 5760 hình ảnh của 10 chủ đề với 576 điều kiện khuôn mặt khác nhau (9 tư thế khuôn mặt x 64 điều kiện sáng).

7 Support Vector Machine (SVM)

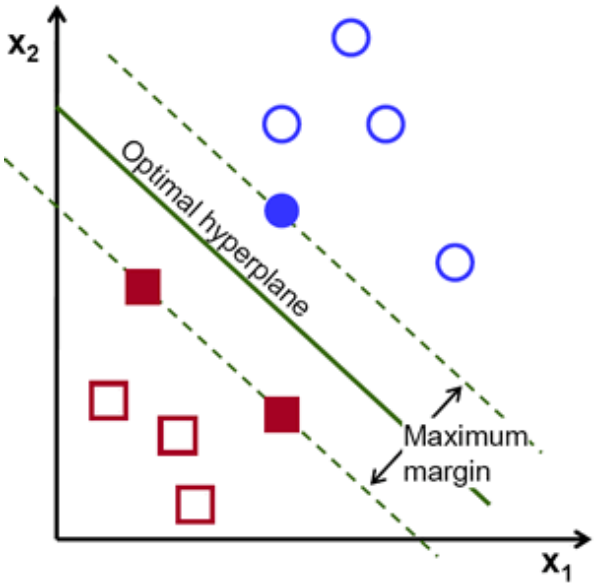
Trong quá trình nhận dạng khuôn mặt, sau khi đã rút trích đặc trưng cũng như giảm chiều, ta tiến hành xây dựng một mô hình phân lớp. Một trong những thuật toán xây dựng mô hình phân lớp phổ biến là Support Vector Machine (SVM).

Một thuật toán cơ bản trong phân lớp đó là thuật toán học Perceptron(PLA), thuật toán này hoạt động trên tập dữ liệu mà 2 lớp cần phân loại có thể được phân chia không gian bởi một siêu phẳng. Tuy nhiên vấn đề xuất hiện ở đây là có thể có vô số siêu phẳng như vậy, do đó chúng ta cần phải tìm được siêu phẳng tối ưu nhất cho mô hình phân lớp.



Hình 4: Ta có thể thấy tồn tại vô số siêu phẳng phân chia giữa các điểm tròn và điểm vuông.

Thuật toán SVM sinh ra để khắc phục vấn đề này. Trong SVM siêu phẳng tối ưu nhất là siêu phẳng sao cho khoảng cách giữa nó và điểm gần nhất (bất kể lớp nào, ta gọi khoảng cách này là margin) là lớn nhất.



Hình 5: Ta có thể thấy được khoảng cách của điểm gần nhất với siêu phẳng là lớn nhất.

Trong không gian 2 chiều, khoảng cách từ 1 điểm (x_0, y_0) đến đường thẳng $w_1x + w_2y + b = 0$ là:

$$\frac{|w_1x_0 + w_2y_0 + b|}{\sqrt{w_1^2 + w_2^2}} \tag{29}$$

Trong không gian 3 chiều, khoảng cách từ 1 điểm (x_0, y_0, z_0) đến mặt phẳng $w_1x+w_2y+w_3z+b = 0$ là:

$$\frac{|w_1x_0 + w_2y_0 + w_3z_0 + b|}{\sqrt{w_1^2 + w_2^2 + w_3^2}} \tag{30}$$

Từ đó ta có công thức tổng quát để tính khoảng cách từ một điểm x_0 có d chiều đến một siêu phẳng là:

$$\frac{|w^Tx_0 + b|}{\|w\|_2} \tag{31}$$

Ở đây nếu ta bỏ trị tuyệt đối thì dấu của giá trị kết quả sẽ cho biết điểm thuộc phía bên nào của siêu thẳng.

Cho tập training set gồm n dữ liệu $(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$ với y là giá trị biểu diễn lớp. Ở đây, ta sẽ cho y là 1 (đối với các điểm thuộc phía dương của siêu phẳng) hoặc -1 (đối với các điểm thuộc phía âm của siêu phẳng) tương ứng với 2 lớp. Lúc này khoảng cách giữa điểm x_i và siêu phẳng sẽ tính bằng:

$$\frac{y_i(w^Tx_i + b)}{\|w\|_2} \tag{32}$$

Bài toán tìm siêu phẳng tối ưu trở thành bài toán tìm (w, b) sao cho margin là lớn nhất:

$$(w, b) = \underset{w,b}{\operatorname{argmax}} \{\min_i \frac{y_i(w^Tx_i + b)}{\|w\|_2}\} \tag{33}$$

Sau khi tìm được bộ (w, b) tối ưu, khi truyền dữ liệu x công thức của siêu phẳng $w^Tx + b$, nếu kết quả cho giá trị âm thì dữ liệu thuộc lớp âm, còn lại thuộc lớp dương. Qua đó, ta chỉ cần xét dấu của kết quả không quan tâm đến giá trị, khi đó:

$$y = sign(w^Tx + b) \tag{34}$$

Lưu ý: Với các bài toán nhiều hơn 2 lớp, ta có thể áp dụng SVM với chiến lược one-vs-one hoặc one-vs-all để tiến hành phân lớp.

8 Xây dựng hệ thống nhận điện đơn giản với pretrain model facenet_keras.h5

8.1 Tổng quan

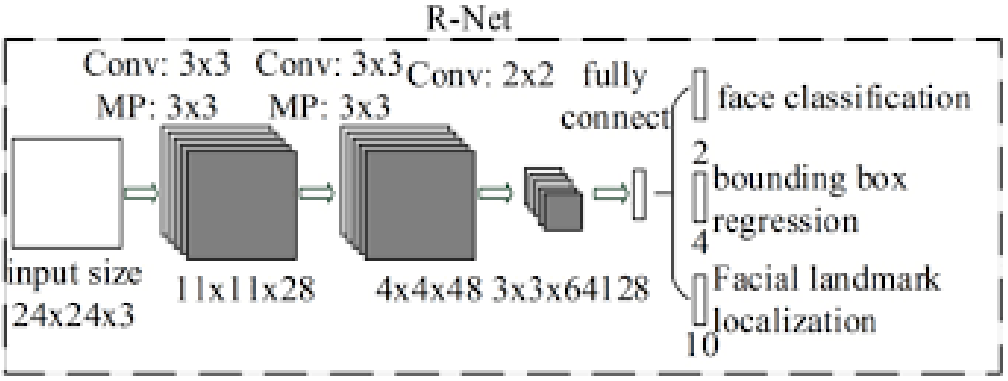
- TÊN: FACE REGCONITON USING FACENET MODEL IN KERAS
- QUY TRÌNH:
  - Bước 1: FACE DETECTION (sử dụng MTCNN)
  - Bước 2: Rút trích đặc trưng với FaceNet
  - Bước 3: Gán nhãn hình ảnh với các phương pháp SVM
  - Bước 4: Tổng kết và phân tích

8.2 Giới thiệu về phương pháp MTCNN trong phát hiện khuôn mặt

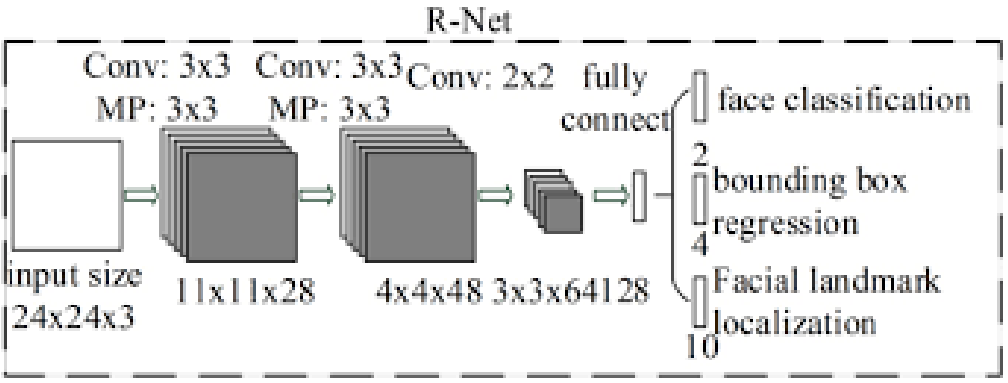
MTCNN là viết tắt của Multi-task Cascaded Convolutional Networks. Nó là bao gồm 3 mạng CNN xếp chồng và đồng thời hoạt động khi detect khuôn mặt. Mỗi mạng có cấu trúc khác nhau và đảm nhiệm vai trò khác nhau trong task. Đầu ra của MTCNN là vị trí khuôn mặt và các điểm trên mặt như: mắt, mũi, miệng...

Qui trình phat hiên khuôn mặt của MTCNN thưc hiên như sau: Thay đổi kích thước hình ảnh ban đầu thành các tỉ lệ khác nhau theo xu hướng giảm dần. Khung quét sẽ quét toàn bộ ảnh và trên nhiều tỉ lệ khác nhau:

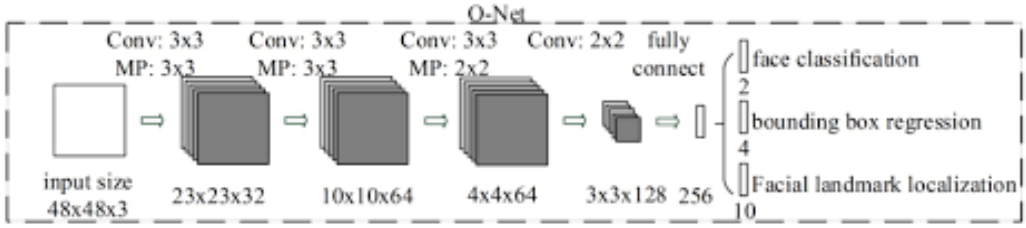
- Tầng 1: Sử dụng mạng CNN, gọi là Mạng đề xuất (P-Net), để thu được các cửa sổ chứa khuôn mặt và các vectơ hồi quy trong các cửa sổ đó. Tiếp theo, các cửa sổ chứa khuôn mặt được hiệu chuẩn dựa trên các vector hồi quy. Cuối cùng, những cửa sổ xếp chồng nhau tại một vùng được hợp nhất thành một cửa sổ (NMS). Sau khi qua tầng 1, ta sẽ thu được các cửa sổ có thể chứa khuôn mặt.



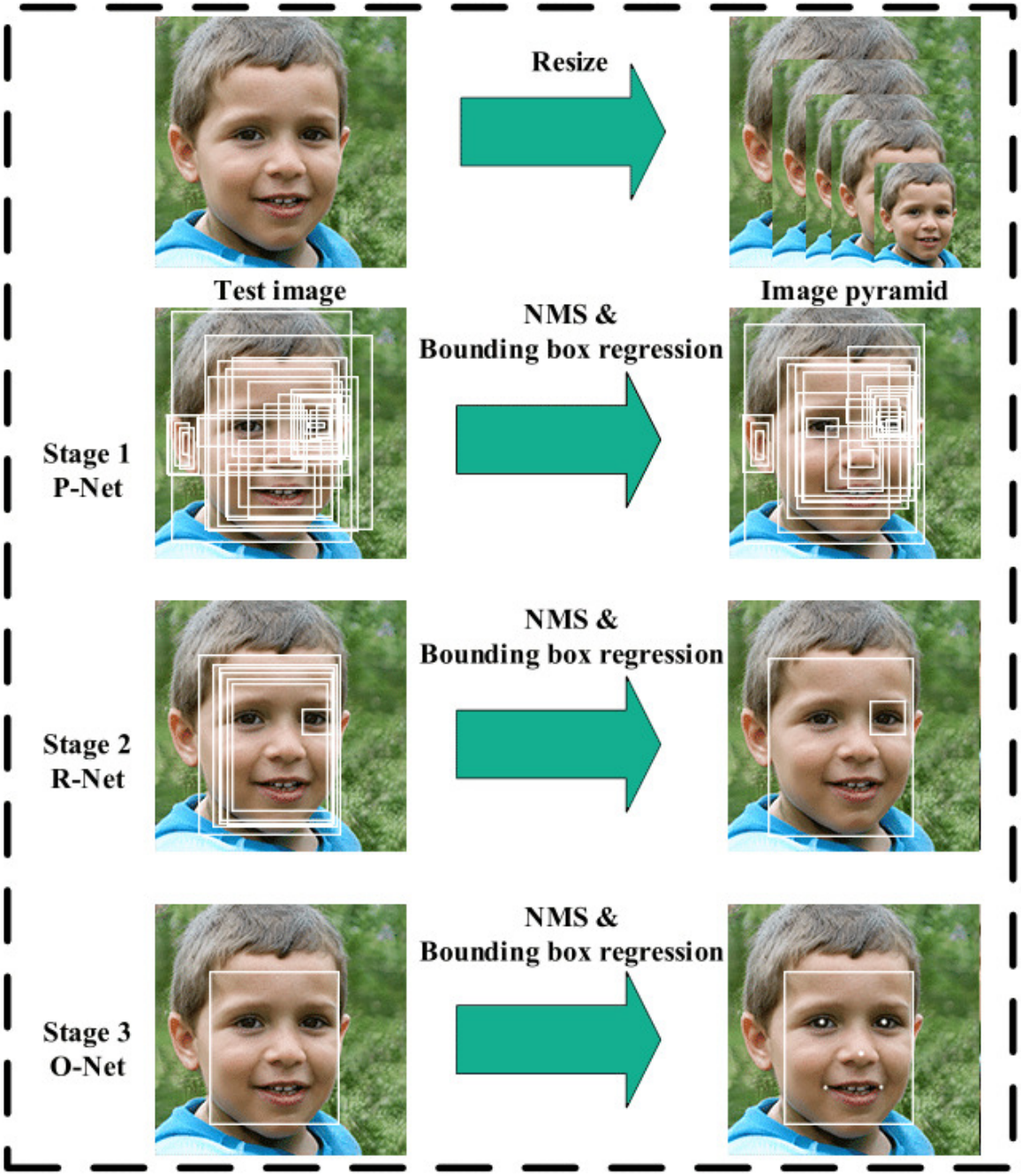
- Tầng 2: Tất cả các cửa sổ chứa khuôn mặt từ tầng 1 sẽ được sàng lọc bằng cách đưa vào một CNN khác phức tạp hơn được gọi là Mạng lọc (R-Net) để tiếp tục loại bỏ một số lượng lớn các cửa sổ không chứa khuôn mặt. Sau đó thực hiện hiệu chuẩn với vector hồi quy và thực hiện hợp nhất các cửa sổ xếp chồng nhau tại một vùng.



- Tầng 3: Tầng này tương tự như tầng 2, nó sử dụng CNN chi tiết nhất được gọi là Mạng đầu ra (O-Net) để lọc kết quả một lần nữa và đánh dấu vị trí năm điểm chính trên khuôn mặt. Những cửa sổ chứa khuôn mặt sau khi đi qua tầng 3 là khuôn mặt được phát hiện.



Quá trình thực tế trên ảnh gốc: Resize => P-Net => R-Net => O-Net



8.3 Giới thiệu pretrain model facenet_keras.h5

Facenet_keras.h5 là một pretrain model được train trên tập dữ liệu MS_Celeb_1M. Với độ lớn của bộ dữ liệu và đặc điểm nét mặt của người châu Âu trong tập dữ liệu được train trước đó. Facenet_keras.h5 sẽ phù hợp với 2 tập dữ liệu YALE và AR Face mà nhóm đã chọn. link download pretrain model: https://drive.google.com/drive/folders/1pwQ3H4aJ8a6yyJHZkTwtjcL4wYWQb7bn

9 Kết quả thực nghiệm và phân tích

Dataset	LBP+SVM	PCA+SVM	LDA+SVM	FaceNet+SVM
YALE	53.333	66.667	100	100
AR face	12.245	26.939	82.857	99.184

Bảng thống kế độ chính xác của Bộ dữ liệu ứng với phương pháp tương ứng

Một số các lý giải về việc sử dụng các kỹ thuật trong kiểm chứng:

1. KFold _ Cross validation là một kỹ thuật lấy mẫu để đánh giá mô hình học máy trong trường hợp dữ liệu không được dồi dào cho lắm.(Ứng với thực tế của demo của nhóm).

2. Đối với các tập dữ liệu nhỏ ta chưa thể đánh giá được độ tốt các các thuật toán. Với tập AR khá lớn các độ chính xác chệnh lệch rõ rệt.

10 Tài liệu tham khảo

1. https://www.researchgate.net/publication/339173834_Face_Recognition_using_FaceNet_Survey_Performance_Test_and_Comparison
2. https://paperswithcode.com/paper/facenet-a-unified-embedding-for-face
3. https://arxiv.org/pdf/1604.02878v1.pdf
4. shorturl.at/aAGIJ
5. https://thetalog.com/machine-learning/locally-linear-embedding/?fbclid=IwAR0GwPVCnvz86kSre3ErUfS
6. https://www.stat.cmu.edu/~cshalizi/350/lectures/14/lecture-14.pdf?fbclid=IwAR2ZHjNaackOHOynz4oKh
7. https://www.cs.mcgill.ca/~dprecup/courses/ML/Lectures/ml-lecture13.pdf?fbclid=IwAR2_7CtwCSnNEG7OspyQAX3Z8GT2UYYhumRLde4xXEcnMVVPrMvYnzW1ocg
8. https://mdav.ece.gatech.edu/ece-6254-spring2017/notes/17-nonlinearDR-2.pdf?fbclid=IwAR1f4Yf0tuZIJo4DihPYdH_UTh_8TNlrhGgWyXMbCcoeF7aFkPFx8Y_k2Ng
9. https://people.eecs.berkeley.edu/~wainwrig/stat241b/scholkopf_kernel.pdf?fbclid=IwAR0xhSCFL1TGxYeK59a4JvXYLeS-MyYFPjtrVIU4Gcj2QGO7pc5j_EyQ3fA
10. https://machinelearningcoban.com/2017/06/15/pca/
11. https://machinelearningcoban.com/2017/06/21/pca2/
12. https://www.sciencedirect.com/science/article/abs/pii/S0950584918302088
13. https://iq.opengenus.org/kernal-principal-component-analysis/
14. http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf?fbclid=IwAR2aOlOv70VK8AJ1OfhXR0wGuwKo7qQsP0g
15. https://nirpyresearch.com/pca-kernel-pca-explained/?fbclid=IwAR0zeb2CkDwPl6sf3ZHjBnUQxqNqDrVXq4x
16. https://machinelearningcoban.com/2017/04/09/smv/?fbclid=IwAR3TfcpKr94UKsTfqOM_4eq0fGSE5MP4cVpclF
17. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithm
18. https://www.nist.gov/programs-projects/face-recognition-grand-challenge-frgc?fbclid=IwAR1WwlnWFm95ntDFj8uTWSBaWtiUf_s2HQgHNvt3jrd0ouyI29BGd20R8Z4
19. https://www.nist.gov/programs-projects/face-recognition-technology-feret?fbclid=IwAR02gd2MHHG1fS3thplmVKAPS_cqVSY-e1n5QpWJcT612WiKWYHmNEd41tg
20. https://www.nist.gov/programs-projects/face-recognition-technology-feret?fbclid=IwAR02gd2MHHG1fS3thplmVKAPS_cqVSY-e1n5QpWJcT612WiKWYHmNEd41tg

21. http://vision.ucsd.edu/~leekc/ExtYaleDatabase/Yale%20Face%20Database.htm?fbclid=IwAR2_7CtwCSnNEG7OspyQAX3Z8GT2UYYhumRLde4xXEcnMVVPrMvYnzW1ocg