BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH

----------⁂----------

TS. NGUYỄN THANH HẢI

GIÁO TRÌNH
XỬ LÝ ẢNH

(Ngành Điện - Điện Tử)

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA
THÀNH PHỐ HỒ CHÍ MINH – 2014

Chương 7
TRÍCH ĐẶC TRƯNG VÀ NHẬN DẠNG

Nhận dạng ảnh là khâu quan trọng của hệ thống xử lý ảnh. Nhận dạng là quá trình phân loại các đối tượng được biểu diễn theo một mô hình nào đó. Ảnh thu nhận về thông qua các bước tiền xử lý sẽ được trích đặc trưng. Với những đặc trưng thu được, ta thực hiện huấn luyện và nhận dạng. Chương này trình bày về phương pháp trích đặc trưng cơ bản là PCA. Sau đó, hai mạng nhận dạng được giới thiệu: ANN, SVM.

7.1. PHÂN TÍCH THÀNH PHẦN CHÍNH (PCA)

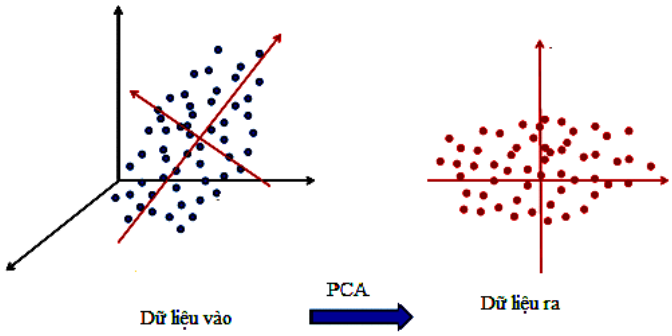
7.1.1. Giới thiệu

Mục đích của phương pháp PCA là loại bỏ đi một số hướng thành phần trong không gian dữ liệu và chỉ giữ lại các thành phần đặc trưng nhất.

PCA giúp làm giảm số chiều của dữ liệu. Hay nói theo cách khác, thay vì sử dụng các trục tọa độ của không gian cũ, PCA xây dựng một không gian mới có số chiều ít hơn, nhưng lại có thể biểu diễn dữ liệu tốt tương đương không gian cũ, nghĩa là đảm bảo độ biến thiên của dữ liệu trên mỗi chiều mới.

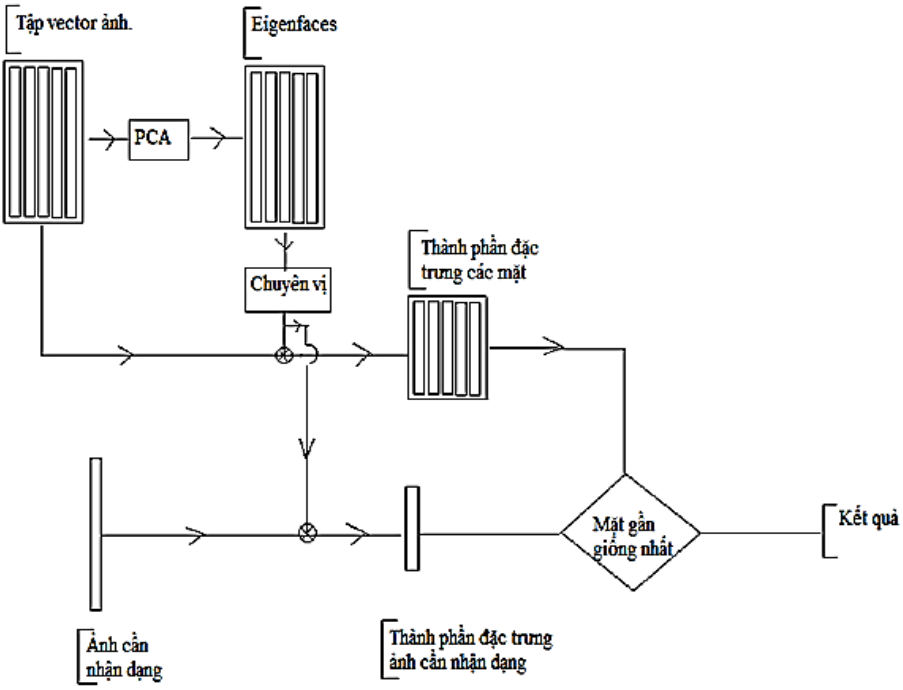
Hình 7.1. *Phép chiếu lên các trục tọa độ khác nhau có thể cho cách nhìn khác nhau về cùng một dữ liệu.*

Hình 7.2 với một tập dữ liệu ban đầu (tập điểm màu xanh) được quan sát trong không gian ba chiều (trục màu đen), ta dễ dàng nhận thấy ba trục này không mô tả được dữ liệu một cách tốt nhất. PCA sẽ tìm một hệ trục tọa độ mới (là trục màu đỏ), với không gian mới, dữ liệu được mô tả rõ ràng hơn (hình bên phải màu đỏ). Rõ ràng hình bên phải chỉ cần hai trục tọa độ nhưng cách thể hiện dữ liệu tốt hơn so với hệ trục ban đầu.



Hình 7.2. *Tìm các trục tọa độ mới sao cho dữ liệu có độ biến thiên cao nhất*

7.1.2. Thuật toán PCA



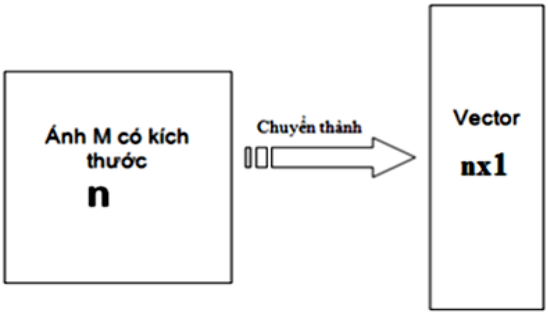
Hình 7.3. *Sơ đồ tiến trình nhận diện dùng PCA cho một khuôn mặt*

7.1.2.1. Chuyển đổi ảnh

Biểu diễn *m* ảnh trong không gian 2D thành 1D. Tạo vectơ có kích thước *n* (số hàng của ảnh × số cột của ảnh) như mô tả sau:

$$\vec{s_i} = [a_1 a_2 .... a_n]^T, \quad i = \overline{1,m} \tag{7.1}$$

với: a_i là giá trị pixel của ảnh, T là chuyển vị của ma trận S_i



Hình 7.4. *Biểu diễn ảnh có kích thước hàng × cột =* ***n*** *thành vectơ* ***n****×1*

Mỗi ảnh là một ma trận cột, ghép *m* ma trận cột ứng với *m* ảnh thành một ma trận có kích thước *n*×*m*.

$$P_{n\times m}=\begin{pmatrix} a_{11}a_{12}a_{13}.....a_{1m} \\ a_{21}a_{22}a_{23}.....a_{2m} \\ a_{31}a_{32}a_{33}.....a_{3m} \\ ... \\ a_{n1}a_{n2}a_{n3}.....a_{nm} \end{pmatrix} \qquad (7.2)$$

Chỉ số đầu là thành phần của vectơ, chỉ số sau là số thứ tự của ảnh.

Ví dụ 7.1: Cho một ảnh 2D gồm có 2 cột và 6 dòng như sau:

$$A=\begin{pmatrix} 118 & 30 \\ 119 & 32 \\ 120 & 33 \\ 121 & 31 \\ 122 & 32 \\ 123 & 34 \end{pmatrix}$$

Biến đổi ảnh A thành vectơ 1D có kích thước 12 phần tử:

$$\vec{s}=(118 \quad 119 \quad 120 \quad 121 \quad 122 \quad 123 \quad 30 \quad 32 \quad 33 \quad 31 \quad 32 \quad 34)^T$$

Như vậy số chiều của ảnh đã được giảm đi, thuận tiện cho việc tính toán nhiều ảnh.

Trong Matlab, ta sử dụng các lệnh sau:

```
>>A=[118 30; 119 32; 120 33; 120 31; 121 32; 122 34]; % Tạo ma trận A
>>S = A(:); % tạo vectơ S từ ma trận A
```

7.1.2.2. Ảnh trung bình

$$\overrightarrow{M} = \frac{1}{m}\sum_{i=0}^{m} x_i \tag{7.3}$$

Hay:

$$\overrightarrow{M} = \frac{1}{m}\begin{pmatrix} a_{11}+a_{12}+a_{13}+.....+a_{1m} \\ a_{21}+a_{22}+a_{23}+.....+a_{2m} \\ a_{31}+a_{32}+a_{33}+.....+a_{3m} \\ ..... \\ a_{n1}+a_{n2}+a_{n3}+.....+a_{nm} \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ .... \\ m_n \end{pmatrix} \tag{7.4}$$

Ví dụ 7.2: Giả sử ta có 4 ảnh mẫu 2D được chuyển đổi thành 4 vectơ 1D như sau:

$$S_1 = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix},\ \overrightarrow{S_2} = \begin{pmatrix} 4 \\ 2 \\ 13 \end{pmatrix},\ \overrightarrow{S_3} = \begin{pmatrix} 7 \\ 8 \\ 1 \end{pmatrix},\ \overrightarrow{S_4} = \begin{pmatrix} 8 \\ 4 \\ 5 \end{pmatrix}$$

Tiến hành ghép 4 vectơ tạo thành ma trận và tính ma trận trung bình:

$$\overrightarrow{M} = \frac{1}{4}\left(\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} + \begin{pmatrix} 4 \\ 2 \\ 13 \end{pmatrix} + \begin{pmatrix} 7 \\ 8 \\ 1 \end{pmatrix} + \begin{pmatrix} 8 \\ 4 \\ 5 \end{pmatrix}\right) = \frac{1}{4}\begin{pmatrix} 20 \\ 16 \\ 20 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 5 \end{pmatrix}$$

Trong Matlab, dùng lệnh `mean(X,dim)` để tính giá trị trung bình của ma trận *X*, với `dim` là chiều lấy trung bình, nếu `dim` bằng 1 lấy trung bình theo cột, nếu `dim` bằng 2 lấy trung bình theo hàng. Không có tham số `dim` thì mặc định `dim` bằng 1.

7.1.2.3. Sai biệt ảnh trung bình

Nhằm mục đích tạo ra sự giãn tương đối giá trị pixel của các ảnh

$$\overrightarrow{n_{1m}} = \begin{pmatrix} a_{11}-m_1 \\ a_{21}-m_2 \\ a_{31}-m_3 \\ ... \\ a_{n1}-m_n \end{pmatrix},\ \overrightarrow{n_{2m}} = \begin{pmatrix} a_{12}-m_1 \\ a_{22}-m_2 \\ a_{32}-m_3 \\ ... \\ a_{n2}-m_n \end{pmatrix},\ .....,\ \overrightarrow{n_{1m}} = \begin{pmatrix} a_{1m}-m_1 \\ a_{2m}-m_2 \\ a_{3m}-m_3 \\ ... \\ a_{nm}-m_n \end{pmatrix} \tag{7.5}$$

Xây dựng ma trận từ các $\overrightarrow{n_{im}}$ vừa tìm được

Đặt $A = \left(\overrightarrow{n_{1m}} \ \overrightarrow{n_{2m}} \ \overrightarrow{n_{3m}} \ ..... \ \overrightarrow{n_{Mm}}\right)$ sẽ được ma trận có kích thước $n \times m$.

Ví dụ 7.3: Từ ví dụ 7.2, ta tính được ma trận A như sau:

$$A = \begin{pmatrix} -4 & -1 & 2 & 3 \\ -2 & -2 & 4 & 0 \\ -4 & 8 & -4 & 0 \end{pmatrix}$$

7.1.2.4.Ma trận hiệp phương sai

Nhằm mục đích thể hiện sự tương quan của từng vectơ đối với các vectơ còn lại trong không gian.

$$\text{cov} = \frac{1}{n-1} AA^T \qquad (7.6)$$

Trị riêng *(eigenvalue* λ_i*),* và vectơ riêng *(eigenvector* x_i*)* của ma trận hiệp phương sai này chính là đặc trưng thành phần thiết yếu của ảnh.

Nhưng thực tế, nếu ảnh có kích thước *200×230,* thì khi đó kích thước của ma trận *cov* là *46000×46000*. Kích thước khá lớn, do đó việc tính trị riêng, vectơ riêng là vấn đề trở ngại khi tính trực tiếp theo cách này. Vì vậy cần phải áp dụng lý thuyết đại số tuyến tính: trị riêng λ , và vectơ riêng có thể tính bằng cách giải quyết trị riêng, và vectơ riêng của ma trận A^TA *(kích thước m×m nhỏ hơn nhiều so với n×n).*

Đặt μ_i và d_i là các trị riêng, và vectơ riêng của ma trận A^TA. Kết quả như sau:

$$A^TAd_i = \mu_i d_i \qquad (7.7)$$

Nhân mỗi vế của (7.6) cho *A* sẽ được:

$$AA^T(Ad_i) = \mu_i(Ad_i) \qquad (7.8)$$

Với $X = Ad_i$

Điều này cho thấy *m* vectơ riêng x_i và, *m* trị riêng λ_i đầu tiên của AA^T tương ứng chính là tích *A* với vectơ riêng d_i của A^TA, và μ_i.

Các vectơ riêng là không gian đặc trưng các tập mẫu trong cơ sở dữ liệu ảnh ban đầu. Các vectơ riêng được sắp xếp theo thứ tự từ cao đến thấp

theo trị riêng tương ứng. Vectơ riêng có trị riêng lớn nhất sẽ mang nhiều đặc trưng thiết yếu nhất của không gian các tập mẫu (tức nó quyết định nhiều nhất sự biến đổi trong ảnh). Ngược lại, vectơ riêng có trị riêng bé nhất sẽ mang thành phần ít đặc trưng nhất trong không gian đặc trưng các tập mẫu. Ở đây cho thấy chỉ với *m* hướng đặc trưng mang trị riêng lớn nhất trong $n \times n$ không gian đặc trưng.

Ví dụ 7.4: Ma trận hợp phương sai được tính từ ma trận *A* trong ví dụ 7.3 theo phương trình (7.6) như sau:

$$\text{cov} = \frac{1}{3}\begin{pmatrix} -4 & -1 & 2 & 3 \\ -2 & -2 & 4 & 0 \\ -4 & 8 & -4 & 0 \end{pmatrix}\begin{pmatrix} -4 & -2 & -4 \\ -1 & -2 & 8 \\ 2 & 4 & -4 \\ 3 & 0 & 0 \end{pmatrix} = \frac{1}{3}\begin{pmatrix} 30 & 18 & 0 \\ 18 & 24 & -24 \\ 0 & -24 & 96 \end{pmatrix} = \begin{pmatrix} 10 & 6 & 0 \\ 6 & 8 & -8 \\ 0 & -8 & 32 \end{pmatrix}$$

Ma trận *cov* có các trị riêng và vectơ riêng như sau:

$$\vec{\lambda} = (1.6057 \quad 13.8430 \quad 34.5513)^T$$

$$\overrightarrow{x_1} = (0.5686 \quad -0.7955 \quad -0.2094)^T$$

$$\overrightarrow{x_2} = (0.8193 \quad 0.5247 \quad 0.2312)^T$$

$$\overrightarrow{x_3} = (-0.0740 \quad -0.3030 \quad 0.9501)^T$$

Trong Matlab, dùng các hàm sau:

`E=eig(X)`: Trả về một vector chứa các giá trị riêng của ma trận vuông X.

`[V, D] = eig(X)`: tạo ra một ma trận đường chéo D của các giá trị riêng và một ma trận V có các cột tương ứng là các vector riêng, do đó: X * V = V * D

7.1.2.5. Phép chiếu

Chiếu lần lượt các ảnh trong cơ sở dữ liệu đến không gian đặc trưng *m* hướng này, để sinh ra các ảnh đặc trưng trong không gian ảnh mới.

$$\Omega_i = [x_1 x_2 .... x_n]^T . \overrightarrow{n_{im}} \quad i = \overline{1, m} \tag{7.9}$$

Với $[X]^T = \left[\overrightarrow{x_1} \quad \overrightarrow{x_2} \quad .... \quad \overrightarrow{x_M}\right]^T$ là ma trận đặc trưng các ảnh mẫu đã rút trích ra được gọi là các eigensignal, $\overrightarrow{n_{im}}$ là vectơ ảnh thứ *i* trừ ảnh trung bình.

7.1.2.6. Nhận dạng

Chuyển đổi ảnh cần nhận dạng thành vectơ 1 chiều:

$$\vec{r} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \dots \\ r_N \end{pmatrix} \tag{7.10}$$

Tính sự sai số của ảnh cần nhận dạng với ảnh trung bình của các ảnh trong cơ sở dữ liệu.

$$\vec{r} = \begin{pmatrix} r_1 - m_1 \\ r_2 - m_2 \\ r_3 - m_3 \\ \dots \\ r_N - m_N \end{pmatrix} \tag{7.11}$$

Chiếu sai số lên không gian đặc trưng của các ảnh mẫu trong cơ sở dữ liệu.

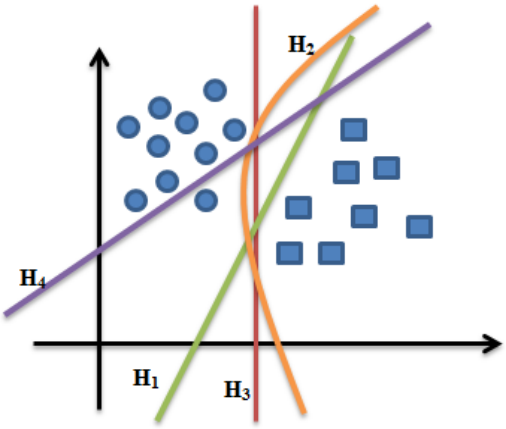
$$\Omega_r = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_M \end{bmatrix}^T . \overrightarrow{r_m} \tag{7.12}$$

Từ đặc trưng đã trích từ phương pháp PCA, các hệ số biểu diễn ảnh có thể dùng để làm ngõ vào của một mạng phục vụ cho việc huấn luyện và phân loại.

7.2. MÁY VECTOR HỖ TRỢ (SVM)

7.2.1. Siêu phẳng - Hyperplane

Siêu phẳng là một yếu tố quan trọng trong thuật toán SVM. Giả sử ta có hai tập dữ liệu với các điểm như trên hình 7.5. Các điểm này thuộc về một trong hai tập. Như vậy vấn đề ở đây là nếu có một điểm dữ liệu mới thì ta sẽ xác định nó thuộc về tập nào.

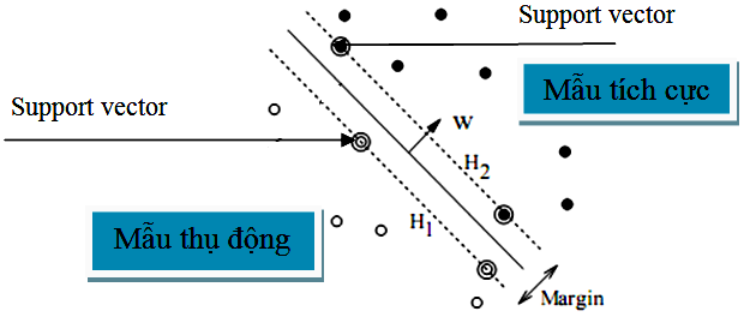


Hình 7.5. *Các đường phân chia hai tập dữ liệu mang đặc trưng khác nhau*

Một cách trực quan trên hình 7.5, ta thấy có thể có ba đường H_1, H_2, H_3 có thể chia hai tập dữ liệu thành hai miền khác nhau. Tuy nhiên, với đường phân chia H_4, đã có một điểm bị phân loại nhầm. Như vậy, việc tính toán các đường để chia các tập dữ liệu là hết sức quan trọng, và các đường này gọi là siêu phẳng. Ta cần tìm các siêu phẳng sao cho nó có độ phân cách lớn nhất hay lề (margin) giữa hai tập dữ liệu.

7.2.2. SVM tuyến tính

SVM tuyến tính thực hiện phân chia hai tập dữ liệu theo siêu phẳng tuyến tính. Giả sử ta có tập huấn luyện $\{\mathbf{x}_i, y_i\}, \quad i = 1,\ldots,l \quad , y_i \in \{-1,1\}, \quad \mathbf{x}_i \in R^d$. Ta muốn phân chia các mẫu tích cực ra khỏi các mẫu thụ động như trong hình 7.6. Các mẫu tích cực nằm trong vùng có $y = 1$ hay vùng H_2, và các mẫu thụ động nằm trong vùng $y = -1$ tương ứng vùng H_1.



Hình 7.6. *Các siêu phẳng H_1 và H_2 phân chia các mẫu tích cực và thụ động thành hai lớp khác nhau*

Các điểm $\mathbf{x}_i$ nằm trên siêu phẳng thỏa mãn $\mathbf{w}.\mathbf{x} + b = 0$. Và $|b|/||\mathbf{w}||$ là khoảng cách từ siêu phẳng đến gốc tọa độ, trong đó $||\mathbf{w}||$ là độ lớn của $\mathbf{w}$.

Đặt d_+ (d_-) là khoảng cách ngắn nhất từ siêu phẳng đến mẫu tích cực (thụ động) gần nhất.

Lề của siêu phẳng là $d_+ + d_-$. Trong trường hợp tuyến tính này, vector hỗ trợ tìm các siêu phẳng phân cách với lề (margin) lớn nhất. Giả sử tất cả dữ liệu huấn luyện thỏa điều kiện:

$$\mathbf{x}_i.w + b \geq +1, \text{với } y_i = +1 \tag{7.13}$$

$$\mathbf{x}_i.w + b \leq -1, \text{với } y_i = -1 \tag{7.14}$$

Kết hợp (7.13) và (7.14)

$$y_i(\mathbf{x}_i.\mathbf{w} + b) - 1 \geq 0 \tag{7.15}$$

Khi dấu bằng trong (7.13) xảy ra, ta có các điểm nằm trên siêu phẳng $H_1 : \mathbf{x}_i.\mathbf{w} + b = 1$ và khoảng cách đến gốc tọa độ là $|1-b| / \| \mathbf{w} \|$. Tương tự, các điểm nằm trên siêu phẳng $H_2 : \mathbf{x}_i.\mathbf{w} + b = -1$ có khoảng cách đến gốc tọa độ là $|-1-b| / \| \mathbf{w} \|$. Vì thế, $d_+ = d_- = 1/\|\mathbf{w}\|$ và độ lớn lề là $2/\|\mathbf{w}\|$. Như vậy, ta có thể tìm cặp siêu phẳng sao cho độ lớn lề là lớn nhất theo phương trình (7.15).

Vấn đề ở đây là tìm **w** và *b*. Công việc tối ưu này được thực hiện bằng cách chuyển (7.15) sang vấn đề tương ứng với tiên đề Lagrange

$$L_p(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \| \mathbf{w} \|^2 - \sum_{i=1}^{l} \alpha_i y_i (\mathbf{x}_i.\mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{7.16}$$

Trong đó, $\alpha_i \geq 0$ là các nhân tử Lagrange.

Lấy đạo hàm L_p theo **w** và *b*, và đặt kết quả bằng 0, ta có

$$\begin{aligned} \frac{\partial L_p(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i = 0 \\ \frac{\partial L_p(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} &= \sum_{i=1}^{l} y_i \alpha_i = 0 \end{aligned} \tag{7.17}$$

Cuối cùng ta được

$$\mathbf{w} = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i \tag{7.18}$$

$$0 = \sum_{i=1}^{l} y_i \alpha_i \tag{7.19}$$

Mỗi mẫu huấn luyện $\mathbf{x}_i$ tương ứng với một hệ số Lagrange α_i. Sau khi huấn luyện các mẫu có $\alpha_i \geq 0$ được gọi là vector hỗ trợ và nằm trên một trong hai siêu phẳng đã nói.

Khi đã có vector hỗ trợ được huấn luyện, ta có thể quyết định một cách đơn giản một mẫu thử **x** thuộc về vùng nào do các siêu phẳng H_1 và H_2 tạo ra bằng cách đặt lớp của **x** là hàm dấu

$$\text{sgn}(\mathbf{w} \cdot \mathbf{x} + b) \tag{7.20}$$

7.2.3. Các điều kiện Karush-Kuhn-Tucker (KKT)

Các điều kiện KKT đóng một vai trò quan trọng trong việc giải quyết tối ưu các vấn đề lập trình phi tuyến trong thực tế. Điều kiện KKT phát biểu như sau:

$$\alpha_i = 0 \Leftrightarrow y_i(\mathbf{x}_i\mathbf{w} + b) \geq 1 \tag{7.21}$$

$$\alpha_i = C \Leftrightarrow y_i(\mathbf{x}_i\mathbf{w} + b) \leq 1 \tag{7.22}$$

$$0 < \alpha_i < C \Leftrightarrow y_i(\mathbf{x}_i\mathbf{w} + b) = 1 \tag{7.23}$$

Việc giải quyết các vấn đề liên quan đến SVM cũng tương tự như việc tìm đáp án cho các điều kiện KKT.

7.2.4. Giải thuật SMO

Để tối ưu thời gian, bộ nhớ trong việc tìm các nhân tử Lagrange, cũng như các trọng số **w** và support vector, có nhiều giải thuật đã được đề ra: Chunking, Osuna, SMO. Giải thuật tối ưu cực tiểu tuần tự - SMO (Sequential Minimal Optimisation) dựa trên ý tưởng của phương pháp phân giải và tối ưu tập cực tiểu chỉ có hai phân tử trong mỗi vòng lặp.

Với điều kiện $\sum_{i=1}^{l} \alpha_i y_i = 0$ thì số lượng nhân tử ít nhất cần phải có là 2, để đảm bảo khi một nhân tử được cập nhật thì nhân tử còn lại cũng phải được điều chỉnh để thỏa điều kiện trên. Như vậy, trong mỗi bước của giải thuật SMO có phần tử α_i và α_j được lựa chọn.

Giả sử ta đã có 2 phần tử được chọn là $\alpha_1$và α_2. Khi thay đổi giá trị này thì giá trị kia phải thay đổi theo để thỏa điều kiện ràng buộc, do đó

$$\alpha_1 y_1 + \alpha_2 y_2 = constant = \alpha_1^{old} y_1 + \alpha_2^{old} y_2 \tag{7.24}$$

Hay nói cách khác, các giá trị mới này nằm trên một đường thẳng trong không gian (α_1,α_2) và $0 \le \alpha_1,\alpha_2 \le C$, trong đó α_1^{old} , α_2^{old} là giá trị trước đó.

Giải thuật sẽ tính giá trị mới α_2^{new}:

$$U \le \alpha_2^{new} \le V \tag{7.25}$$

Trong đó, nếu $y_1 \neq y_2$ thì

$$\begin{aligned} U &= \max(0, \alpha_2^{old} - \alpha_1^{old}) \\ V &= \min(C, C - \alpha_1^{old} + \alpha_2^{old}) \end{aligned} \tag{7.26}$$

Và nếu $y_1 = y_2$ thì

$$\begin{aligned} U &= \max(0, \alpha_1^{old} + \alpha_2^{old} - C) \\ V &= \min(C, \alpha_1^{old} + \alpha_2^{old}) \end{aligned} \tag{7.27}$$

Một cách rõ ràng hơn, đặt E_i là sai biệt giữa hàm ngõ ra và mục tiêu phân loại trên các mẫu huấn luyện $\mathbf{x}_1$ hoặc $\mathbf{x}_2$,

$$E_i = f(\mathbf{x}_i) - y_i = \left(\sum_{j=1}^{l} \alpha_j y_j K(x_j, x_i) + b \right) - y_i, \qquad i = 1,2 \tag{7.28}$$

Và đặt

$$\gamma = K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2) = \| \phi(\mathbf{x}_1) - \phi(\mathbf{x}_2) \|^2 \tag{7.29}$$

Trong đó $K(x_i,x_j)$ là hàm lõi (kernel function).

Tính toán một cách định lượng thì, giá trị mới của α_2 chưa bị giới hạn là

$$\alpha_2^{new,unc} = \alpha_2^{old} + \frac{y_2(E_1 - E_2)}{\gamma} \tag{7.30}$$

Để thỏa điều kiện $U \le \alpha_2^{new} \le V$ thì

$$\alpha_2^{new} = \begin{cases} V, khi & \alpha_2^{new,unc} > V \\ \alpha_2^{new,unc}, khi & U \le \alpha_2^{new,unc} \le V \\ U, khi & \alpha_2^{new,unc} < U \end{cases} \tag{7.31}$$

Và ta tính được giá trị α_1^{new}:

$$\alpha_1^{new} = \alpha_1^{old} + y_1 y_2(\alpha_2^{old} - \alpha_2^{new}) \quad (7.32)$$

Tóm lại, giải thuật SMO thực hiện các bước:

- Bước 1: Tìm một nhân tử Lagrange α_1 không thỏa điều kiện KKT.
- Bước 2: Tìm tiếp nhân tử α_2, tính các giá trị $\alpha_1^{new}, \alpha_2^{new}$.
- Bước 3: Lập lại bước 1, và 2 cho đến khi hội tụ.

Để tăng tốc độ hội tụ, giải thuật SMO sử dụng phỏng đoán heuristic để chọn 2 nhân tử Lagrange trong vấn đề tối ưu. Có 2 phỏng đoán heuristic độc lập nhau trong việc chọn nhân tử thứ nhất và nhân tử thứ hai:

- Phỏng đoán chọn lựa thứ nhất: Đây là vòng lặp ngoài của giải thuật SMO. Vòng lặp ngoài sẽ tìm trên tập huấn luyện các mẫu không thỏa điều kiện KKT. Nếu một mẫu không thỏa điều kiện KKT, nó được dùng cho việc tối ưu. Khi đã tìm xong trên cả tập, vòng lặp ngoài sẽ tìm các mẫu sao cho nhân tử Lagrange của chúng không bằng 0 hoặc không bằng C (các mẫu không biên). Một lần nữa, các mẫu này được kiểm tra và nếu không thỏa điều kiện KKT thì nó được dùng cho việc tối ưu. Vòng lặp ngoài sẽ lặp lại trên các mẫu không biên cho đến khi các điều kiện KKT thỏa sai số ε.
- Phỏng đoán chọn lựa thứ hai: Khi nhân tử Lagrange thứ nhất được chọn, SMO chọn nhân tử thứ 2 sao cho bước nhảy trong phương trình 7.29 là lớn nhất, tức là tối đa $|E_1 - E_2|$. Nếu E_1 dương, SMO chọn một mẫu sao cho E_2 là nhỏ nhất. Nếu E_1 âm, SMO chọn một mẫu với sai số E_2 lớn nhất.

Với việc cập nhật các nhân tử Lagrange, ngưỡng b sẽ được tính lại ở mỗi bước. ngưỡng b_1 được tính:

$$b_1 = E_1 + y_1(\alpha_1^{new} - \alpha_1)K(\mathbf{x}_1, \mathbf{x}_1) + y_1(\alpha_2^{new} - \alpha_2)K(\mathbf{x}_1, \mathbf{x}_2) + b \quad (7.33)$$

Và b_2:

$$b_2 = E_2 + y_1(\alpha_1^{new} - \alpha_1)K(\mathbf{x}_1, \mathbf{x}_2) + y_1(\alpha_2^{new} - \alpha_2)K(\mathbf{x}_2, \mathbf{x}_2) + b \quad (7.34)$$

Khi b_1 và b_2 hợp lệ chúng sẽ bằng nhau.

Vector trọng số $\mathbf{w}$ cũng được cập nhật:

$$\mathbf{w}^{new} = \mathbf{w} + y_1(\alpha_1^{new} - \alpha_1)\mathbf{x}_1 + y_2(\alpha_2^{new} - \alpha_2)\mathbf{x}_2 \quad (7.35)$$

Như vậy, giải thuật SMO được đề ra nhằm góp phần tăng tốc độ hội tụ cũng như thời gian xử lý trên máy tính trong vấn đề huấn luyện trên thuật toán SVM.

Trong MATLAB, công cụ SVM được hỗ trợ thông qua `svmsmoset, svmtrain, svmclassify.`

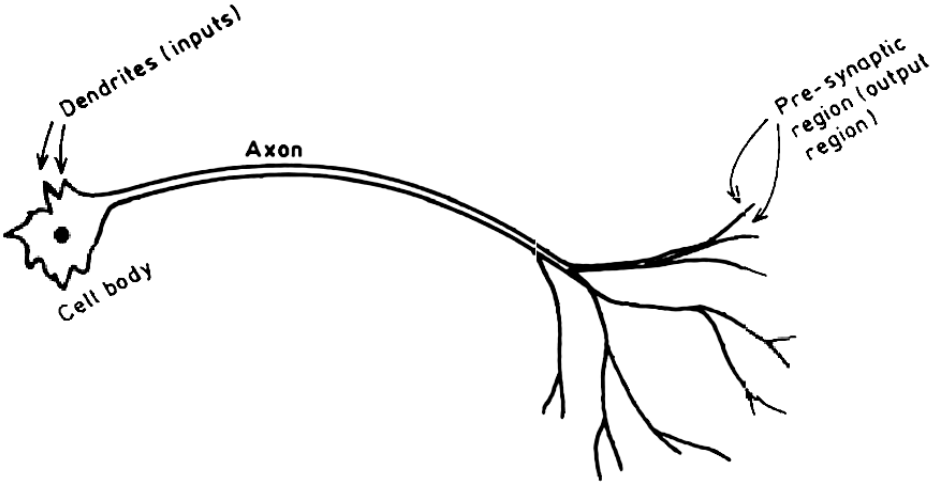
Ví dụ 7.5: Khai báo thông số, huấn luyện và nhận dạng dùng công cụ SVM.

```
SMO_OptsStruct = svmsmoset('TolKKT',0.0005,'MaxIter',10000);
svmStruct = svmtrain(dt,g,'SMO_Opts',SMO_OptsStruct);
classes = svmclassify(svmStruct,dtest);
```

`svmsmoset` cho người dùng khai báo các thông số về điều kiện KKT, số vòng lặp. Trong khi đó, `svmtrain` thực hiện huấn luyện ngõ vào `dt` theo ngõ ra mong muốn `g`. Cuối cùng, với `svmclassify` thực hiện phân loại trên cấu trúc SVM đã khai báo với ngõ vào cần phân loại.

7.3. MẠNG NƠ-RON NHÂN TẠO

Mạng nơ-ron nhân tạo (ANN - Artifical Neural Networks) là một trong những công cụ rất mạnh trong vấn đề phân loại và nhận dạng đối tượng. Dựa trên cấu trúc của nơ-ron sinh học như hình 7.7 trong hệ thống thần kinh người, nhiều dạng mạng nơ-ron nhân tạo đã được xây dựng và phát triển.

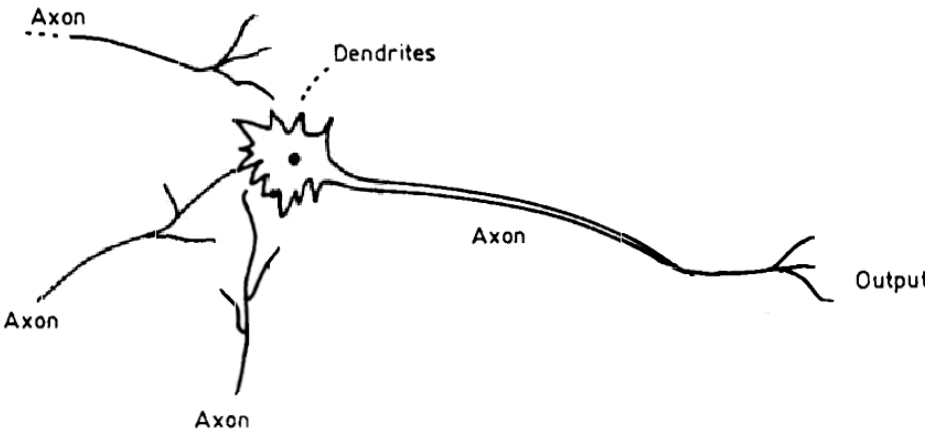


Hình 7.7. *Cấu trúc nơ-ron sinh học: cell body, axon, synaptic*

Mạng ANN đầu tiên được xây dựng đó là Perceptron (Frank Rosenblatt - 1958), tiếp theo đó là Artron, Adaline, Madaline. Bốn dạng cơ bản này, đặc biệt là Perceptron, là cơ sở để phát triển các mạng ANN.

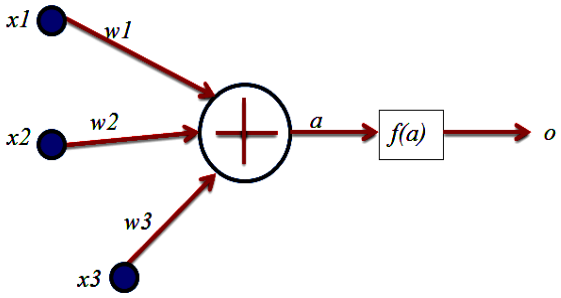
Trong đó, ba loại mạng cơ bản được phát triển tiếp theo là mạng lan truyền ngược (Back-propagation), Hopfield Network, Counter-Propagation Network.

Perceptron có cấu trúc cơ bản của một tế bào thần kinh được trình bày như trong hình 7.8, như ví dụ này thì có ba ngõ vào trọng số được tổng hợp lại và kết nối đến ngõ ra, ngõ ra này lại có thể là ngõ vào của các nơ-ron khác.



Hình 7.8. *Cấu trúc cơ bản của tế bào thần kinh*

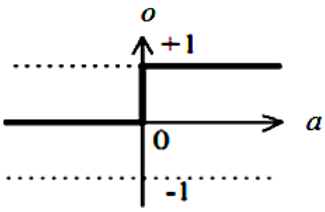
Perceptron này được mô hình hóa theo dạng



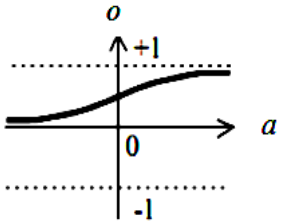
Hình 7.9. *Mô hình perceptron một nơ-ron*

Trong đó *xi*, *i* =1, 2, 3 là các ngõ vào, *wi* là các trọng số với ngõ vào tương ứng, hàm hoạt hóa và ngõ ra $o = f(a)$, trong đó $a = x1.w1 + x2.w2 + x3.w3$.

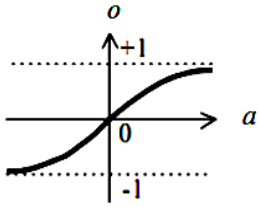
Các hàm hoạt hóa thường gặp được tóm tắt trong hình 7.10, 7.11, 7.12:



Hình 7.10. *Hàm bước*



Hình 7.11. *Hàm sigmoid*



Hình 7.12. *Hàm double sigmoid*

Trong hình 7.10, hàm hoạt hóa là hàm bước

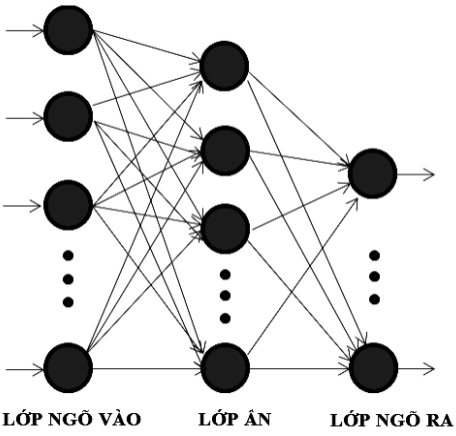
$$o = \begin{cases} 1, & a > 0 \\ 0, & a \le 0 \end{cases} \tag{7.36}$$

Hàm sigmoid và double sigmoid được miêu tả trong hình 7.12 và 7.13 tương ứng với công thức (7.37) và (7.38)

$$o = S(a) = \frac{1}{1+e^{-a}} \tag{7.37}$$

$$o = H(a) = 2S(a) - 1 \tag{7.38}$$

Mạng lan truyền ngược được xây dựng dựa trên cấu trúc Perceptron, trong đó có ít nhất hai lớp, và các hàm hoạt hóa phải là hàm có thể lấy đạo hàm (ví dụ ở đây là hàm sigmoid, double sigmoid). Hình 7.13 là mô hình mạng lan truyền ngược ba lớp (một lớp vào, một lớp ẩn, một lớp ra).



Hình 7.13. *Mạng lan truyền ngược với ba lớp*

Dựa trên giải thuật giảm gradient, mạng lan truyền ngược tìm các trọng số sao cho trung bình bình phương sai số giữa ngõ ra hiện tại và ngõ ra mong muốn đạt kết quả tốt nhất.

$$E = \sum_{p=1}^{P} \left(o_p - d_p\right)^2 \tag{7.39}$$

Trong đó: P là số lượng mẫu, o_p là ngõ ra của mạng, d_p là ngõ ra mong muốn.

Giả sử có I nút ở lớp ngõ vào, J nút ở lớp ẩn, K nút ở lớp ngõ ra. Trọng số $w_{j,i}^{(1,0)}$ là trọng số từ nút thứ i của lớp ngõ vào đến nút thứ j của lớp ngõ ẩn, và $w_{k,j}^{(2,1)}$ là trọng số từ nút thứ j của lớp ẩn đến nút thứ k lớp ngõ ra.

Giải thuật huấn luyện cho mạng ba lớp được tóm tắt trong bảng 7.1 như sau

Bảng 7.1. *Giải thuật cập nhật trọng số cho mạng lan truyền ngược ba lớp*

Chọn ngẫu nhiên giá trị các trọng số

Trong khi điều kiện MSE thỏa mãn hoặc chưa vượt số vòng lặp đặt trước,

Với mỗi ngõ vào x_p, $1 \le p \le P$, (*)

Tính các ngõ vào nút lớp ẩn $net_{p,j}^{(1)}$;

Tính các ngõ ra nút lớp ẩn $x_{p,j}^{(1)}$;

Tính các ngõ vào của nút lớp ngõ ra $net_{p,k}^{(2)}$;

Tính các ngõ ra của mạng $o_{p,k}$;

Điều chỉnh trọng số lớp ngõ ra

$$\Delta w_{k,j}^{(2,1)} = \eta (d_{p,k} - o_{p,k}) S'(net_{p,k}^{(2)}) x_{p,j}^{(1)}$$

Điều chỉnh các trọng số giữa nút ẩn và nút ngõ vào

$$\Delta w_{j,i}^{(1,0)} = \eta \sum_{k=1}^{K} \left((d_{p,k} - o_{p,k}) S'(net_{p,k}^{(2)}) w_{k,j}^{(2,1)} \right) S'(net_{p,j}^{(1)}) x_{p,i}$$

Kết thúc (*)

Kết thúc

Trong đó:$S()$ là hàm hoạt hóa, η là hệ số học hay tốc độ học.

Trong MATLAB, ta có thể sử dụng hàm `newff` để khai báo một mạng truyền thẳng

```
net = newff(P, T, S);
```

trong đó `P`: ma trận các phần tử ngõ vào, `T`: ma trận các phần tử ngõ ra, `S`: kích thước các lớp ẩn.

Tiếp theo, mạng được huấn luyện dùng hàm `train`

```
[neto] = train(net,P,T);
```

Sau khi đã có mạng `neto`, dùng hàm `sim` để phân loại

```
res = sim(neto,dtest);
```

trong đó `dtest` là ngõ vào cần phân loại, `res` là kết quả phân loại.

BÀI TẬP CHƯƠNG 7

7.1. Thu nhận ảnh của 3 người. Thực hiện phân tích thành phần chính PCA để tìm vector đặc trưng.

7.2. Sử dụng mạng nơ-ron 3 lớp (lớp ngõ vào, lớp ẩn, lớp ngõ ra) thực hiện huấn luyện và nhận dạng từ kết quả bài 7.1.

7.3. Trình bày hướng phát triển để thực hiện phân loại SVM trong trường hợp nhiều hơn hai đặc trưng cần phân loại.

TÀI LIỆU THAM KHẢO

1. Nguyễn Quang Hoan, *Xử lý ảnh*, lưu hành nội bộ, Học viện Công nghệ Bưu chính Viễn thông, 2006.
2. Đỗ Năng Tòan, Phạm Việt Bình, *Xử lý ảnh*, Giáo trình môn học, Đại học Thái Nguyên, Khoa Công nghệ thông tin, 2007.
3. Maria Petrou, Panagiota Bosdogianni, *Image Processing: The Fundamentals*, John Wiley & Sons Ltd, 1999.
4. William K. Pratt, *Digital Image Processing*, John Wiley & Sons, Inc., 2001.
5. Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing*, The Third Edition,Prentice Hall, 2008.
6. Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing Using Matlab*, Prentice Hall, 2004.
7. Kayvan N., Robert S., *Biomedical Signal and Image Processing*, Taylor and Francis Group, 2006.

GIÁO TRÌNH
XỬ LÝ ẢNH

TS. NGUYỄN THANH HẢI

NHÀ XUẤT BẢN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
Khu Phố 6, Phường Linh Trung, Quận Thủ Đức, TPHCM
Số 3, Công trường Quốc tế, Quận 3, TP Hồ Chí Minh
ĐT: 38239171 – 38225227 - 38239172
Fax: 38239172 - Email: vnuhp@vnuhcm.edu.vn

*****

PHÒNG PHÁT HÀNH NHÀ XUẤT BẢN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
Số 3 Công trường Quốc tế - Quận 3 – TPHCM
ĐT: 38239170 – 0982920509 – 0913943466
Fax: 38239172 – Website: www.nxbdhqghcm.edu.vn

*****

Chịu trách nhiệm xuất bản:
NGUYỄN HOÀNG DŨNG

Chịu trách nhiệm nội dung:
NGUYỄN HOÀNG DŨNG

Tổ chức bản thảo và chịu trách nhiệm về tác quyền
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM

Biên tập:
PHẠM ANH TÚ

Sửa bản in:
THÙY DƯƠNG

Trình bày bìa
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM

Mã số ISBN: 978-604-73-2582-5

Số lượng 300 cuốn; khổ 16 x 24cm.
Số đăng ký kế hoạch xuất bản: 1007-2014/CXB/05-13/ĐHQGTPHCM.
Quyết định xuất bản số: 107 ngày 28/05/2014 của NXB ĐHQGTPHCM.
In tại Công ty TNHH In và Bao bì Hưng Phú.
Nộp lưu chiểu quý III năm 2014.