TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Nhóm sinh viên thực hiện:
VŨ ĐĂNG QUÂN MSSV:20173314
LƯU TUẤN LINH MSSV:20173234
DƯƠNG PHÚC THẮNG MSSV:20173368
NGUYỄN TIẾN ANH MSSV: 20172941
Giáo viên hướng dẫn: Nguyễn Linh Giang

Đề Tài: NHẬN DẠNG VÂN TAY

Chuyên ngành : Kỹ thuật máy tính

Bộ môn: nhập môn An Toàn Thông Tin

Hà Nội, tháng 6 năm 2020

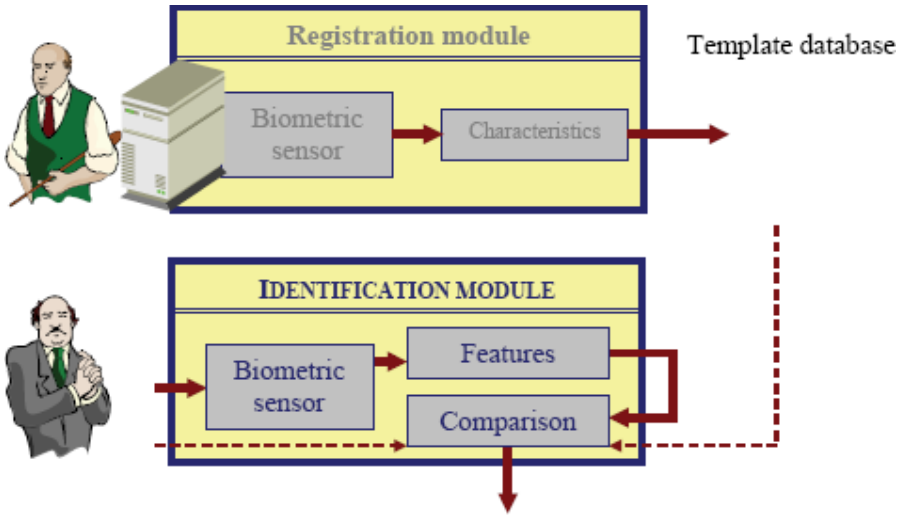
GIỚI THIỆU

1. Giới thiệu
2. Tổng quan nghiên cứu tình hình
3. Ý nghĩa đề tài

CHƯƠNG 1: GIỚI THIỆU

1.1 GIỚI THIỆU

Ngày nay, các kỹ thuật sinh trắc học ngày càng được ứng dụng rộng rãi. Trong đó, nhận dạng vân tay được xem là một trong những kỹ thuật hoàn thiện và đáng tin cậy nhất để xác nhận một người. Gần đây, kỹ thuật này được chú ý nhiều và người ta thấy rằng nó thích hợp với những ứng dụng có cơ sở dữ liệu nhỏ, nhưng không thuận tiện cho những ứng dụng có phạm vi lớn. Đa số các hệ thống bảo mật hiện nay được bảo vệ bằng password và PIN (Personal Identification Number), nhưng các phương pháp này đã được chứng minh là không hiệu quả. Bởi vì, password là những con số khó nhớ, dễ quên và dễ bị đánh cắp. Bằng cách sử dụng vân tay và mật mã, việc xác nhận một người có thể được thực hiện bằng một hệ thống nhận dạng vân tay an toàn và thuận tiên. Hình 1.1 là cấu trúc cơ bản của hệ thống nhận dạng dấu vân tay. Đầu tiên, dấu vân tay của một người cần được lấy mẫu (bằng một thiết bị có thể chụp được vân tay – Biometric sensor) và lưu vào cơ sở dữ liệu (Registration module). Sau đó, khi cần xác nhận người đó cung cấp lại một dấu vân tay khác, dấu vân tay này sẽ được so sánh với dấu vân tay trong cơ sở dữ liệu để quyết định chấp nhận hay từ chối dựa trên một giá trị ngưỡng đối sánh.



Hình 1.1: Cấu trúc cơ bản cả hệ thống nhận dạng vn tay

Hiện nay, trên thị trường thế giới đã có bán nhiều loại thiết bị chụp vân tay(fingerprint reader, fingerprint scanner) với các chất lượng khác nhau. Bảng 1.1 giới thiệu một số loại thiết bị chụp vân tay và các thông số kỹ thuật của chúng. Hình 1.2 là ảnh vân tay được chụp từ các thiết bị này.

	Technology	Company	Model	Dpi	Area (h×w)	Pixels
Optical	FTIR	Biometrika www.biometrika.it/eng/	FX2000	569	0.98"×0.52"	560×296 (165,760)
	FTIR	Digital Persona www.digitalpersona.com	UareU2000	440	0.67"×0.47"	316×228 (72,048)
	FTIR (sweep)	Kinetic Sciences www.kinetic.bc.ca	K-1000	up to 1000	0.002"×0.6"	2×900 (H×900)
	FTIR	Secugen www.secugen.com	Hamster	500	0.64"×0.54"	320×268 (85,760)
	Sheet prism	Identix www.identix.com	DFR 200	380	0.67"×0.67"	256×256 (65,535)
	Fiber optic	Delsy www.delsy.com	CMOS module	508	0.71"×0.47"	360×240 (86,400)
	Electro-optical	Ethentica www.ethentica.com	TactilSense T-FPM	403	0.76"×0.56"	306×226 (69,156)
Solid-state	Capacitive (sweep)	Fujitsu www.fme.fujitsu.com	MBF300	500	0.06"×0.51"	32×256 (H×256)
	Capacitive	Infineon www.infineon.com	FingerTip	513	0.56"×0.44"	288×224 (64,512)
	Capacitive	ST-Microelectronics us.st.com	TouchChip TCS1AD	508	0.71"×0.50"	360×256 (92,160)
	Capacitive	Veridicom www.veridicom.com	FPS110	500	0.60"×0.60"	300×300 (90,000)
	Thermal (sweep)	Atmel www.atmel.com	FingerChip AT77C101B	500	0.02"×0.55"	8×280 (H×280)
	Electric field	Authentec www.authentec.com	AES4000	250	0.38"×0.38"	96×96 (9,216)
	Piezoelectric	BMF www.bm-f.com	BLP-100	406	0.92"×"0.63	384×256 (98,304)

Bảng 1.1: Một số loại thiết bị chụp vân tay và các thông số kỹ thuật của chung

Hình 1.2: Ảnh vân tay được chụp từ các thiết bị trên: a) Biometrika FX2000,
b) Digital Persona UareU2000, c) Identix DFR200,
d) Ethentica TactilSense T-FPM,
e) STMicroelectronics TouchChip TCS1AD, f) Veridicom FPS110,
g) Atmel FingerChip AT77C101B, h) Authentec AES4000.

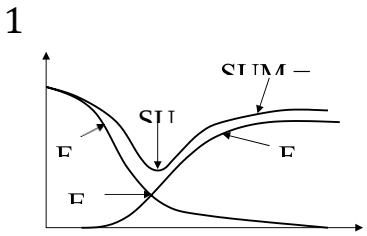
Để đánh giá một hệ thống nhận dạng vân tay ta cần phân tích hai loại lỗi đó là: lỗi từ chối nhầm (False Reject Rate: FRR) và lỗi chấp nhận nhầm (False Accept Rate: FAR)

$$FAR = \frac{\text{Số lỗi chấp nhận nhầm của các vân tay khác nhau}}{\text{Tổng số lần đối sánh của các vân tay khác nhau}}$$

$$FRR = \frac{\text{Số lỗi từ chối nhầm của các vân tay khác nhau}}{\text{Tổng số lần đối sánh của các vân tay khác nhau}}$$

Giá trị của hai loại lỗi này có mối quan hệ với nhau thông qua giá trị ngưỡng đối sánh T (threshold) là sai lệch cho phép giữa mẫu cần đối sánh với mẫu được lưu trong cơ sở dữ liệu. Khi chọn giá trị ngưỡng thấp thì lỗi từ chối nhầm sẽ tăng, lỗi chấp nhận nhầm sẽ giảm và ngược lại.
Hệ thống thường được đánh giá theo hai cách:

1. Tỷ lệ lỗi cực tiểu $SUM_{min} = (FAR + FRR)_{min}$: theo quan điểm dù là loại lỗi gì thì cũng là lỗi, do đó tỷ lệ lỗi cực tiểu SUM_{min} là hệ số lỗi nhỏ nhất màhệ thống có thể đạt được.
2. Mức độ lỗi cân bằng (Equal Error Rate: EER): đó là điểm mà FAR và FRR bằng nhau.



Hình 1.3: Mối quan hệ giữa FAR, FRR, SUM và EER theo ngưỡng T

1.2 TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU

Các phương pháp nhận dạng vân tay kinh điển đều dựa vào việc đối sánh (matching) các điểm đặc trưng (feature) trên vân tay. Có nhiều phương pháp đối sánh khác nhau. Trong bài này, chúng tôi nghiên cứu phương pháp đối sánh bằng mạng neural nhân tạo (Artificial Neural Network).

1.3 Ý NGHĨA ĐỀ TÀI

Đề tài giới thiệu một hướng nghiên cứu và ứng dụng lĩnh vực nhận dạng vân tay vào thực tiễn. Một lĩnh vực đã khá phổ biến trên thế giới nhưng còn hạn chế ở Việt Nam.

PHƯƠNG PHÁP NHẬN DẠNG VÂNTAY

1. *Các điểm đặc trưng trên ảnh vân tay*
2. *Trích các điểm đặc trưng*
3. *Làm nổi ảnh vân tay*
4. *Đối sánh (matching)*

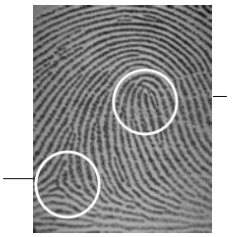
CHƯƠNG 2:

PHƯƠNG PHÁP NHẬN DẠNG VÂN TAY

2.1 CÁC ĐIỂM ĐẶC TRƯNG TRÊN ẢNH VÂN TAY

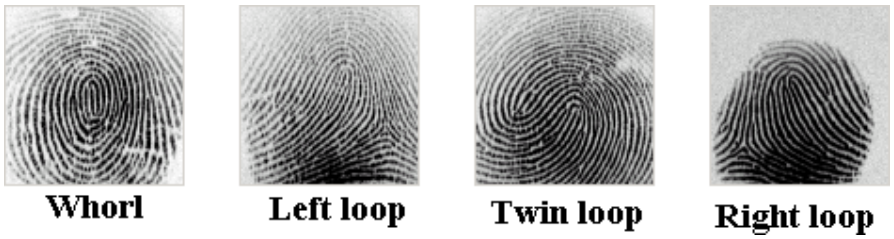
Trên các ảnh vân tay có các điểm đặc trưng (là những điểm đặc biệt mà vị trí của nó không trùng lặp trên các vân tay khác nhau) được phân thành hai loại: singularity và minutiae

- **Singularity:** Trên vân tay có những vùng có cấu trúc khác thường so với những vùng bình thường khác (thường có cấu trúc song song), những vùng như vậy goi là singularity. Có hai loại singularity là core và delta.



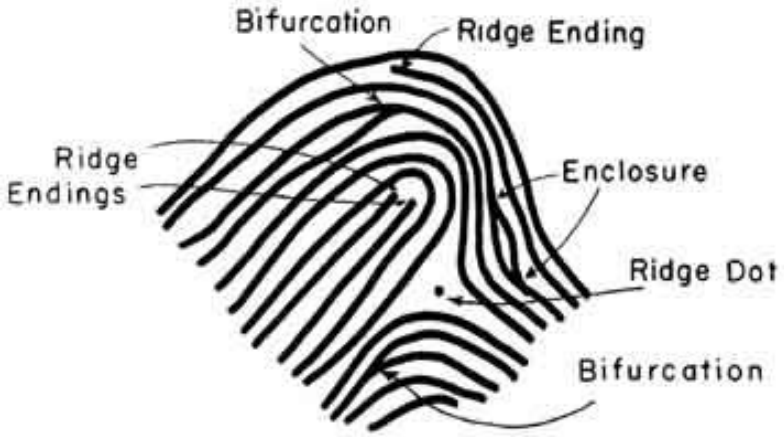
Hình 2.1: Các điểm singularity core và

delta Core thường có một số dạng như sau:



Hình 2.2: Một số loại core thường gặp.

- **Minutiae:** Khi dò theo từng đường vân ta sẽ thấy có những điểm đường vân kết thúc (Ridge Ending) hoặc rẽ nhánh (Bifurcation), những điểm này được gọi chung là minutiaae.



Hình 2.3: Các điểm minutiae Ridge Ending (điểm kết thúc) và Bifurcation (điểm rẽ nhánh)

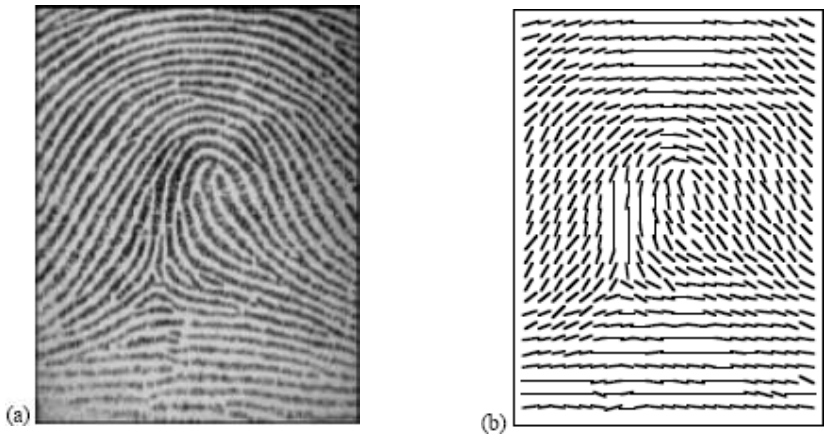
2.2 TRÍCH CÁC ĐIỂM ĐẶC TRƯNG

Bằng các phương pháp xử lý ảnh ta có thể tìm được vị trí các điểm đặc trưng trên các ảnh vân tay.

2.2.1 Trích các điểm singularity

a. Trường định hướng (orientation field)

Ảnh vân tay là ảnh định hướng, các đường vân là các đường cong theo các hướng xác định. Góc hợp bởi phương của một điểm trên đường vân với phương ngang được gọi là hướng của điểm đó. Tập hợp các hướng của các điểm trên ảnh vân tay gọi là trường định hướng của ảnh vân tay đó.



Hình 2.4: ảnh vân tay (a) và trường định hướng của nó (b)

Phương pháp xác định trường định hướng như sau [5], [14]:

- Chia ảnh vân tay thành các khối nhỏ hơn kích thước WxW
- Tính gradient theo hai hướng x, y là Gx, Gy tại mỗi điểm (pixel) trong khối
- Khi đó hướng của điểm chính giữa của khối được xác định theo công thức:

$$\varphi = \frac{1}{2}\tan^{-1}\left(\frac{\sum_{i=1}^{W}\sum_{j=1}^{W}2G_x(i,j)G_y(i,j)}{\sum_{i=1}^{W}\sum_{j=1}^{W}\left(G_x^2(i,j)-G_y^2(i,j)\right)}\right)$$

Hàm *orientation.m* thực hiện tính trường định hướng được giới thiệu trong phần phụ lục.

b. *Xác định các điểm singularity bằng chỉ số Poincare (Poincare index)* [3]
Giả sử (i,j) là một điểm bất kỳ trên ảnh vân tay, C là một đường cong khép kính xung quanh (i,j) thì chỉ số Poincare tại (i,j) là tổng đại số các độ sai lệch hướng của các điểm liền kề nhau trên đường cong C.

$$Poincare(i,j) = \sum_{k=0}^{Np-1} \Delta(k)$$

$$\Delta(k) = \begin{cases} d(k) & |d(k)| < \pi/2 \\ d(k)+\pi & d(k) \le -\pi/2 \\ d(k)-\pi & \end{cases}$$

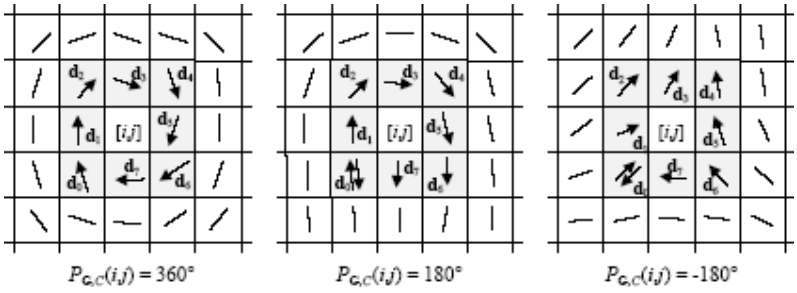
$$d(k) = \varphi(x_{k+1}, y_{k+1}) - \varphi(x_k, y_k)$$

Trong đó: N_p là tổng số điểm trên đường cong "số" C
j(x,y) là hướng tại điểm (x,y)

Dựa vào chỉ số Poincare ta có thể xác định các điểm singularity như sau:

$$Poincare(i,j) = \begin{cases} 0^0 & \text{(i,j) không phải là điểm} \\ 360^0 & \text{singularity (i,j) là điểm} \\ & \text{whorl} \\ 180^0 & \text{(i,j) là điểm} \\ & \text{loop (i,j) là} \\ -180^0 & \text{điểm delta} \end{cases}$$

Hình 2.5 minh họa cách tính chỉ số poincare tại điểm (i,j) với số điểm trên đường cong "số" $N_p = 8$



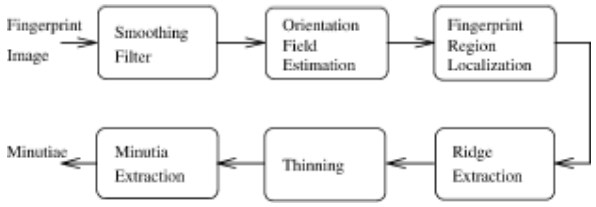
Hình 2.5: Cách tính chỉ số poincare tại điểm (i,j) với $N_p = 8$

Hàm *poincare.m* thực hiên việc tính chỉ số Poincare theo thuật toán trên và hàm *singularity.m* xác định các điểm singularity dựa vào chỉ số Poincare (phụ lục).

2.2.2. Trích các điểm minutiae

Có hai phương pháp chính để tìm các điểm minutiae: trích các điểm minutiae từ ảnh binary và trích các điểm minutiae trực tiếp từ ảnh xám.

a. Trích các điểm minutiae từ ảnh binary [5]

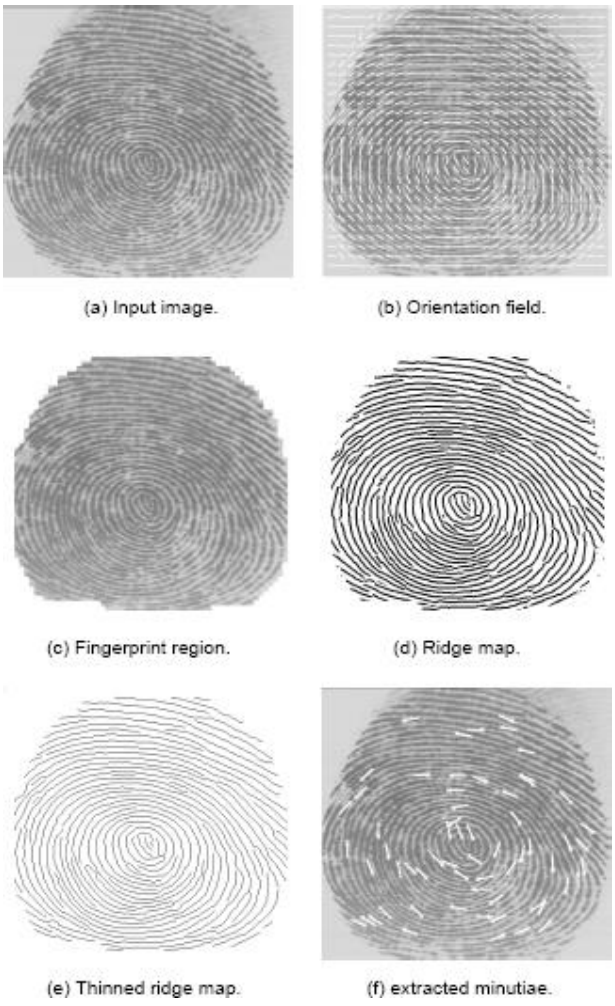


Hình 2.6: Sơ đồ mô tả thuật toán trích các điểm minutiae từ ảnh binary

Ý tưởng chính của phương pháp này là từ ảnh xám ban đầu ta sử dụng các bộ lọc thích hợp để phát hiện và làm mảnh đường vân dưới dạng một pixel (ridge detection), biến đổi ảnh xám ban đầu thành ảnh binary (có giá trị là 0 hoặc 1) tương ứng.

Sau đó, các điểm minutiae sẽ được trích như sau: giả sử (x,y) là một điểm trên đường vân đã được làm mãnh và N_0, N_1, ..., N_7 là 8 điểm xung quanh nó thì

- (x,y) là một điểm kết thúc nếu $\sum_{i=0} N_i = 1$
- (x,y) là một điểm rẽ nhánh nếu $\sum_{i=0} N_i > 2$

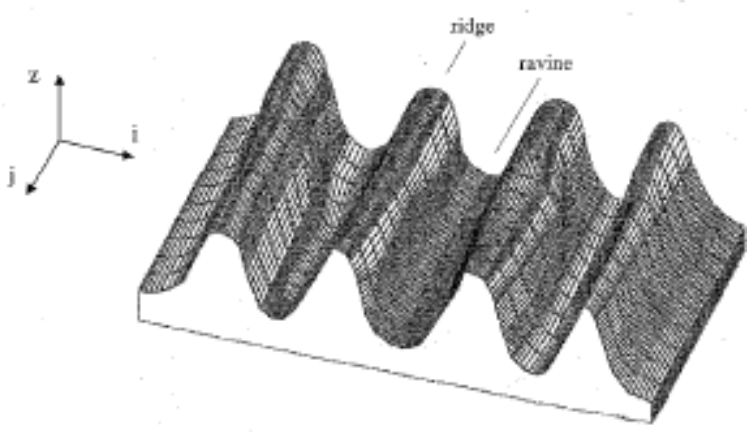


Hình 2.7: Các kết quả của thuật toán

b. Trích các điểm minutiae trực tiếp từ ảnh xám [1]

- Dò theo đường vân (Ridge line following)

Giả sử I là một ảnh xám có kích thước là $m_x n$ và nếu coi chiều thứ ba z là mức xám tại điểm (i,j) thì bề mặt của ảnh vân tay I có dạng như sau:



Hình 2.8: Bề mặt của ảnh vân tay với các đường vân (ridge) và các rãnh (ravine)

Theo quan điểm toán học thì đường vân là tập hợp các điểm cực đại dọc theo một hướng xác định. Việc xác định các điểm minutiae trực tiếp từ ảnh xám dựa vào thuật toán dò theo đường vân. Thuật toán này dựa vào việc xác định các điểm cực đại dọc theo hướng của đường vân.

- Xác định điểm cực đại

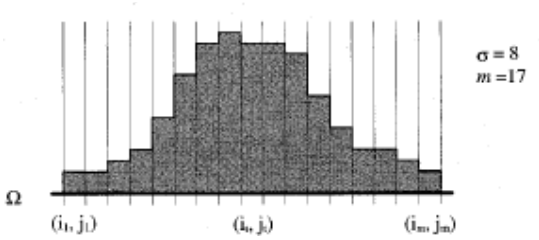
Giả sử $((i_t, j_t), f, s)$ là thiết diện của đường vân có điểm chính giữa là (i_t, j_t), hướng của thiết diện $f = j_t + p/2$ (j_t là hướng của đường vân tại (i_t, j_t)) và bề rộng của thiết diện m = 2s+1 pixel (hình 2.9). Khi đó, được xác định như sau:

$$\Omega = \{ (i, j) \mid (i, j) \in \mathbf{I}, (i, j) \in segment((i_{start}, j_{start}), (i_{end}, j_{end})) \}$$

$$(i_{start}, j_{start}) = (round (i_t - \sigma \cdot cos\ \phi), round (j_t - \sigma \cdot sin\ \phi))$$
$$(i_{end}, j_{end}) = (round (i_t + \sigma \cdot cos\ \phi), round (j_t + \sigma \cdot sin\ \phi))$$

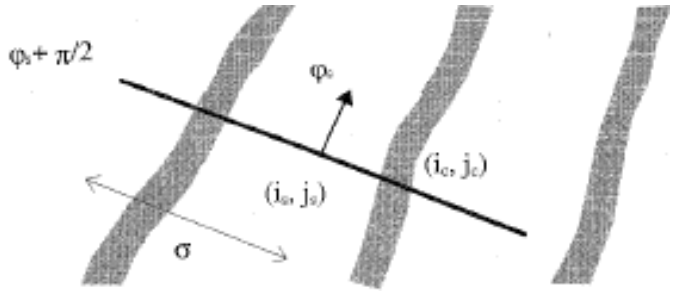
$$round(x) = \begin{cases} \lfloor x + 0.5 \rfloor & \text{if } x \geq 0 \\ \lfloor x + 0.5 \rfloor & \text{otherwise} \end{cases}$$

và điểm cực đại có thể được xác định bằng cách so sánh mức xám giữa các điểm trong



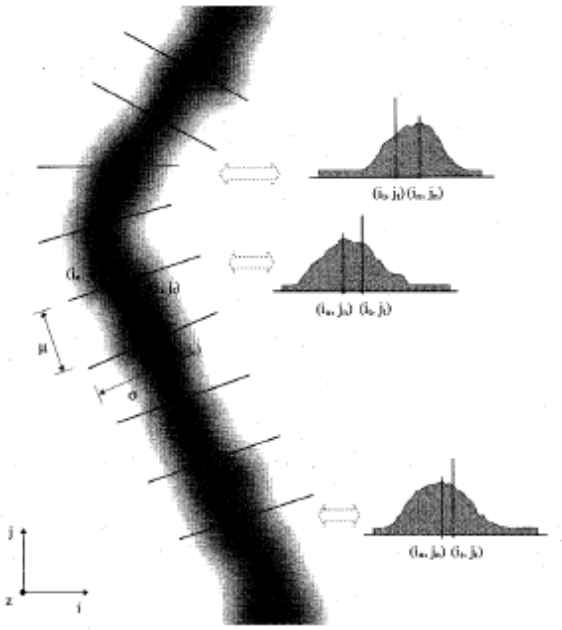
Hình 2.9: Thiết diện của đường vân tại

- Tóm lại việc tìm các điểm minutiae bằng thuật toán dò theo đường vân được thực hiện như sau (chi tiết xem ở tài liệu tham khảo[1]):
  - Lấy một điểm bất kì (is,js) trên ảnh I
  - Tìm hướng j_s tại điểm (is,js)
  - Tìm điểm cực đại (ic,jc) gần (is,js) nhất



Hình 2.10: Điểm cực đại (ic,jc) tương ứng với (is,js)

  - Tìm hướng j_c tại điểm (ic,jc)
  - Dịch chuyển theo hướng j_c một đoạn m
  - Tinh chỉnh lại điểm cực đại (ic,jc) và hướng j_c
  - Tiếp tục quá trình này để dò theo đường vân (ridge following) cho đến khi không phát hiện được điểm cực đại (ic,jc) thì đó là điểm Ridge Ending hoặc chạm vào một đường vân khác thì đó là điểm Bifurcation (mỗi đường vân sau khi được dò sẽ được gán nhãn)
  - Tiếp theo chọn một điểm (is,js) khác và thực hiện lại quá trình trên cho đến khi dò hết tất cả các đường vân.



Hình 2.11: Dịch chuyển theo đường vân từng đoạn m

Tất cả các thuật toán trên được thực hiện bằng hàm *minutiae.m* (phụ lục)

2.3 LÀM NỔI ẢNH VÂN TAY

Các ảnh vân tay thường được lấy bằng hai phương pháp: từ mực hoặc từ các sensor. Các ảnh vân tay được lấy từ mực thường có chất lượng thấp và không đồng đều. Phần này sẽ giới thiệu phương pháp dùng bộ lọc Gabor để cải thiện chất lượng của ảnh vân tay [8], [13], [14].

Hàm Gabor là một công cụ hữu dụng cho việc xử lý ảnh. Nó có đặc tính chọn lọc trong miền không gian lẫn tần số. Hàm Gabor 2_D thực có dạng như sau:

$$g(x,y;T,\phi) = \exp\left(-\frac{1}{2}\left[\frac{x_\phi^2}{\sigma_x^2}+\frac{y_\phi^2}{\sigma_y^2}\right]\right)\cos\left(\frac{2\pi x_\phi}{T}\right)$$

$$x_\phi = x\cos\phi + y\sin\phi$$

$$y_\phi = -x\sin\phi + y\cos\phi$$

Trong đó:

f là hướng của bộ lọc
T là chu kỳ của hàm cos (thường được chọn từ thực nghiệm có giá trị [0,1])
s_x, s_y là các độ lệch chuẩn (thường được chọn từ thực nghiệm có giá trị [0,4])

Các bước thực hiện:

1. Chuẩn hóa mức xám: nếu I(x,y) là mức xám tại điểm (x,y) của ảnh I thì mức xám chuẩn hóa $N_i(x,y)$ được xác định rheo công thức sau:

$$N_i(x,y)=\begin{cases} M_0+\sqrt{\dfrac{V_0\times(I(x,y)-M_i^2)}{V_i}}, if\ I(x,y)>M_i \\ M_0-\sqrt{\dfrac{V_0\times(I(x,y)-M_i^2)}{V_i}}, otherwise \end{cases}$$

trong đó:

M_0, V_0 là mean và variance mong muốn (thường được chọn là 100) M_i, V_i là mean và variance của ảnh I

Chú ý: nếu mức xám của các vùng khác nhau trên ảnh I không đồng đều thì có thể chia I thành các khối nhỏ và chuẩn hoá theo từng khối.

Hình 2.12: ảnh I và ảnh chuẩn hóa của nó

(Hàm *normalize.m* thực hiện chuẩn hóa mức xám được giới thiệu ở phụ lục)

2. Xác định trường định hướng theo phương pháp đã giới thiệu ở trên
3. Sử dụng hàm lọc Gabor cho ảnh đã chuẩn hóa trong miền tần số
   - Chia ảnh cần lọc thành từng khối nhỏ kích thước WxW
   - Xác định hướng của khối (dựa vào trường định hướng)
   - Hướng f của bộ lọc là hướng của khối
   - Sử dụng phép biến đổi FFT và phép biến đổi IFFT cho từng khối ảnh và hàm Gabor

Hình 2.13: Kết quả lọc bằng hàm *gabor_filter.m* (phụ lục) với T = 0.6, $s_x = 1$, $s_y = 2$

2.4 ĐỐI SÁNH (MATCHING)

Hầu hết các phương pháp nhận dạng vân tay đều dựa vào việc đối sánh vị trí các điểm đặc trưng. Gần đây, một số tác giả đã kết hợp thêm một số đặc tính khác của ảnh vân tay để nâng cao hiệu quả đối sánh như: Orientation field [9] hoặc Density map [10]. Chi tiết xem ở tài liệu tham khảo, ở đây tôi xin giới thiệu phương pháp đối sánh vị trí các điểm đặc trưng mà tôi đã sử dụng, phương pháp này gần giống với các phương pháp được nêu ở [4] và [11]. Hàm *matching.m* (phụ lục) thực hiện đối sánh hai ảnh vân tay theo phương pháp này.

Giả sử I và I' lần lượt là các ảnh vân tay mẫu và ảnh vân tay cần đối sánh, $m = \{x, y, q\}$ là các điểm đặc trưng được xác định bởi tọa độ (x,y) và hướng q.

$$I = \{m_1, m_2, ..., m_m\}, \quad m_i = \{x_i, y_i, \theta_i\}, \quad i = 1...m$$
$$I' = \{m'_1, m'_2, ..., m'_n\}, \quad m'_j = \{x'_j, y'_j, \theta'_j\}, \quad j = 1...n$$

trong đó: m, n lần lượt là số điểm đặc trưng của I và I'.

Khi đó, điểm $m' \in I'$ được coi là "giống" với điểm $m \in I$ nếu độ sai lệch về không gian và độ sai lệch về hướng nhỏ hơn các giá trị ngưỡng r_0 và θ_0 :

$$sd(m'_j, m_i) = \sqrt{(x'_j - x_i)^2 + (y'_j - y_i)^2} \le r_0$$
$$dd(m'_j, m_i) = |\theta'_j - \theta_i| \le \theta_0$$

Nếu

$$\frac{\text{Tổng số điểm của I – số điểm giống nhau}}{\text{Tổng số điểm của I}} < \text{ngưỡng T}$$

thì I' được coi là giống I. Trong đó T là phần trăm số điểm sai lệch cho phép.

MẠNG NEURAL NHÂN TẠO

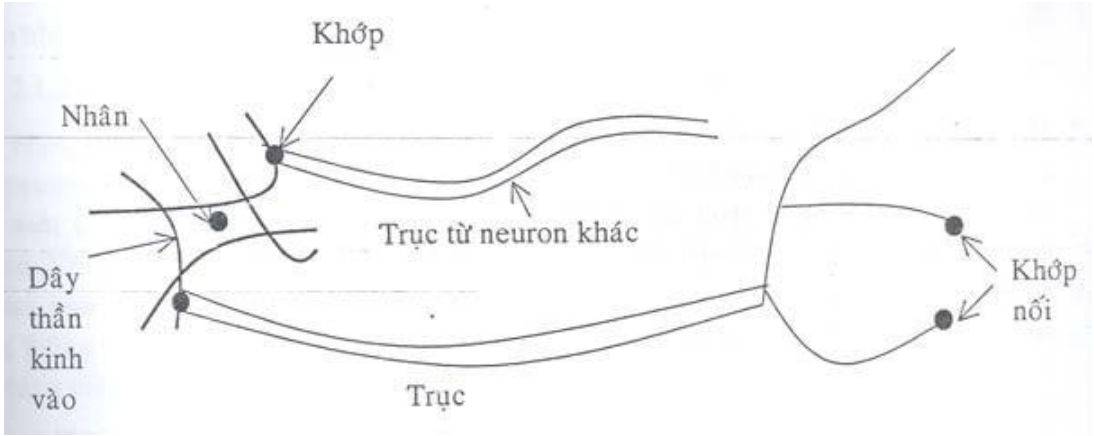
1. *Tổng quan về neural – mạng neural*
2. *Một số mô hình mạng neural*

CHƯƠNG 3: MẠNG NEURAL NHÂN TẠO

3.1 TỔNG QUAN VỀ NEURAL – MẠNG NEURAL

1. Bộ não và neuron sinh học

Tế bào thần kinh còn gọi là "neuron". Nghiên cứu sinh học về bộ não con người cho thấy rằng các neuron là đơn vị cơ sở đảm nhiệm những chức năngn xử lý nhất định trong hệ thần kinh, bao gồm: não, tủy sống và các dây thần kinh . Mỗi neuron có phần thân và nhân bên trong (gọi là soma), một đầu thần kinh ra (gọi là dendrite). Các dây thần kinhvào tạo thành một lưới dày đặc xung quanh thân tế bào, chiếm diện tích khoảng 0,25mm^2, còn dây thần kinh tạo rathành trục dài có thể từ 1 cm đến hàng mét. Đường kính nhân tế bào thường chỉ là 10^{-4} m. trục dây thần kinh ra cũng có thể phân nhánh theo dạng cây để nối với dây thần kinh vào hoặc trực tiếp với nhân tế bào các neuron khác thông qua các khớp nối (gọi là synapse). Thông thường, mỗi neuron có thể gồm vài chục cho tới hàng trăm khớp nối để nối với các neuron khác. Người ta ước lượng rằng lưới các dây thần kinh ra cùng với các khớp nối bao phủ diện tích khoảng 90% bề mặt neuron.



Hình 3.1 Cấu tạo mạng neural sinh học

Các tín hiệu truyền trong dây thần kinh vào và dây thần kinh ra của các neural là tính hiệu điện, được thực hiện thông qua các quá trình phản ứng và giải phóng các chất hữu cơ. Các chất này được phát ra từ các khớp nối

dẫn tới các

dây thần kinh vào sẽ làm tăng hay giảm điện thế của nhân tế bào. Khi điện thế này đạt đến một mức ngưỡng nào đó sẽ tạo ra một xung điện dẫn tới trục dây thần kinh ra. Xung này được truyền theo trục, tới các nhánh rẽ khi chạm tới các khớp nối với các neuron khác và sẽ giải phóng các chất truyền điện. Thường chia khớp nối thành 2 loại: khớp nối kích thích (excitatory) và khớp nối ức chế (inhibitory).

Phát hiện quan trọng nhất về bộ não sinh học là các liên kết khớp thần kinh khá mềm dẻo, có thể biến động và sửa đổi theo thời gian tùy thuộc vào các dạng kích thích. Hơn nữa, các neuron có thể sản sinh các liên kết mới với các neuron khác; đôi khi, lưới các neuron có thể di trú từ vùng này sang vùng khác trong bộ não. Đây là cơ sở quan trọng để giải thích cho cơ chế học của bộ não con người.

Các chức năng cơ bản của bộ não bao gồm:

- Bộ nhớ được tổ chức theo các bó thông tin và truy cập theo nội dung.
- Bộ não có thể tổng quát hóa, có thể truy xuất các tri thức hay các mối liên kết chung của các đối tượng tương ứng với một khái niệm chung nào đó.
- Bộ não có khả năng điều chỉnh hoặc tiếp tục thực hiện ngay khi có những sai do thông tin bị thiếu hay thiếu chính xác. Ngoài ra, bộ não còn có thể phát hiện và phục hồi các thông tin bị mất dựa trên sự tương tự giữa các đối tượng.
- Bộ não có khả năng xuống cấp và thay thế dần. Khi có những trục trặc tại các vùng não (do chấn thương) hoặc bắt gặp những thông tin hoàn toàn mới lạ, bộ não vẫn có thể được tiếp tục làm việc.
- Bộ não có khả năng học.

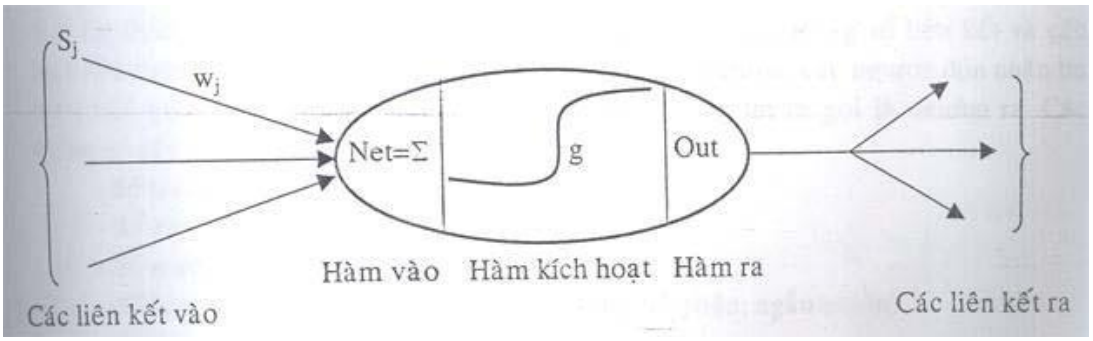
Nhìn chung, tính toán sơ bộ cho thấy rằng dù bộ vi xử lý máy tính điện tử có thể tính toán nhanh hơn hàng triệu lần so với neuron của bộ não, nhưng xét tổng thể thì bộ não lại tính toán nhanh hơn hàng tỷ lần. Ngoài ra, cũng dễ thấy rằng bộ não con người có thể lưu trữ nhiều thông tin hơn các máy tính hiện đại, dù rằng điều này không phải đúng mãi mãi bởi lẽ bộ não tiến hóa chậm còn bộnhớ máy tính thì được nâng cấp rất nhanh nhờ những tiến bộ của khoa học kỹ thuật.

2. Mô hình neuron nhân tạo và mạng neuron nhân tạo

mạng neuron nhân tạo (Artificial neural network – ANN) là mạng bao gồm các nút (neuron, đơn vị xử lý) được nối với nhau bởi các liên kết neuron. Mỗi liên kết kèm theo một trọng số nào đó, đặc trưng cho đặc tính kích hoạt hoặc ức chế giữa các neuron. Có thể xem các trọng số là phương tiện để lưu thông tin dài hạn trong mạng neuron, còn nhiệm vụ của quá trình huấn luyện (học) là cập nhật các trọng số khi có thêm thông tin về các mẫu học, hay nói cách khác, cá

trọng số được điều chỉnh sao cho dáng điệu vào ra của nó mô phỏng hoàn toàn phù hợp môi trường đang xem xét.

a. Mô hình nhân tạo



Hình 3.2 Mô hình neural nhân tạo

Mỗi nueron được nối với các neuron khác và nhận được các tín hiệu từ chúng với các trọng số wj.

- Tổng thông tin vào có trọng số là:
- Net = $w_j s_j$, đây là thành phần tuyến tính của neuron
- Hàm kích hoạt g đóng vai trò biến đổi từ Nét sang tín hiệu đầu ra out
- Out = g(Net), đây là thành phần phi tuyến của mạng neuron
- Một số dạng hàm kích hoạt thường dùng trong thực tế:

+ Hàm bước: $\text{step}(x)=\begin{cases}1, x \geq 0\\ 0, x < 0\end{cases}$ (3.1)

+ Hàm dấu : $\text{sign}(x)=\begin{cases}1, x \geq 0\\ -1, x < 0\end{cases}$ (3.2)

+ Hàm sigmoid 1: $\text{sigmoid}(x)=\frac{1}{1+e^{-\alpha(x+\theta)}}$ (3.3)

+ Hàm sigmoid 2: $\text{sigmoid}(x)=a \cdot \frac{e^{-\alpha(x+\theta)}-1}{e^{-\alpha(x+\theta)}+1}$ (3.4)

Trong đó :

$S=(s_1, s_2, ..., s)$: vector tín hiệu vào

$W=(w_1, w_2, ...w_n)$: vector trọng số

θ: là ngưỡng đóng vai trò làm tăng tính thích nghi và khả năng tính toán của mạng neuron

b. Mạng neuron nhân tạo

Mạng neuron nhâan tạo, sau đây gọi tắt l mạng neuron,được xây dựng trên cơ sở mạng neuron sinh học, l hệ thống bao gồm nhiều phần tử xử lý đơn giản (neuron), hoạt động song song. Tính năng của hệ thống này tùy thuộc vào cấu trúc của hệ, các trọng số liên kết và cấu trúc của chúng cho phép hợp với mẫu học. trong mạng neuron, các neuron đón nhận tín hiệu vo gọi là neuron vào, còn các neuron đưa thông tin ra gọi là nueron ra. Các thông số cấu trúc mạng neuron bao gồm:

- Số tín hiệu vào, số tín hiệu ra.
- Số lớp neuron
- Số neuron trên mỗi lớp ẩn
- Số lượng liên kết của mỗi neuron (đầy đủ, bộ phận, ngẫu nhiên)
- Các trọng số liên kết

b.1 Phân loại mạng neuron

- Theo kiểu liên kết neuron, ta có mạng neuron truyền thẳng (feed-forward neural network) và mạng neuron hồi qui (recurrent neural network). Trong mạng neuron truyền thẳng, các liên kết neuron đi theo một hướng nhất định , không có chu trình. Ngược lại, mạng neuron hồi qui cho phép các liên kết neuron tạo thành chu trình. Vì các thông tin ra của các neuron được truyền lại cho chính các neuron nên đã góp phần kích hoạt cho chung và tạo ra khả năng lưu giữ trạng thi trong của nó dưới dạng các ngưỡng kích hoạt ngồi các trọng số liên kết neuron.
- Theo số lớp, ta có mạng neuron một lớp (single-layer) và mạng neuron đa lớp (multi-layer). Trong đó, thông thường lớp neuron vo chỉ chịu trách nhiệm truyền đưa tín hiệu vào, không thực hiện một tính toán nào, nên khi tính số lớp của mạng ta không tính lớp này vào.

b.2 Cách nhìn về mạng neuron

+ Có thể xem mạng neuron như một công cụ tốn học, một bảng tra. Giả sử mạng neuron NN có m neuron vào và n neuron ra, khi đó với mỗi vector tín hiệu

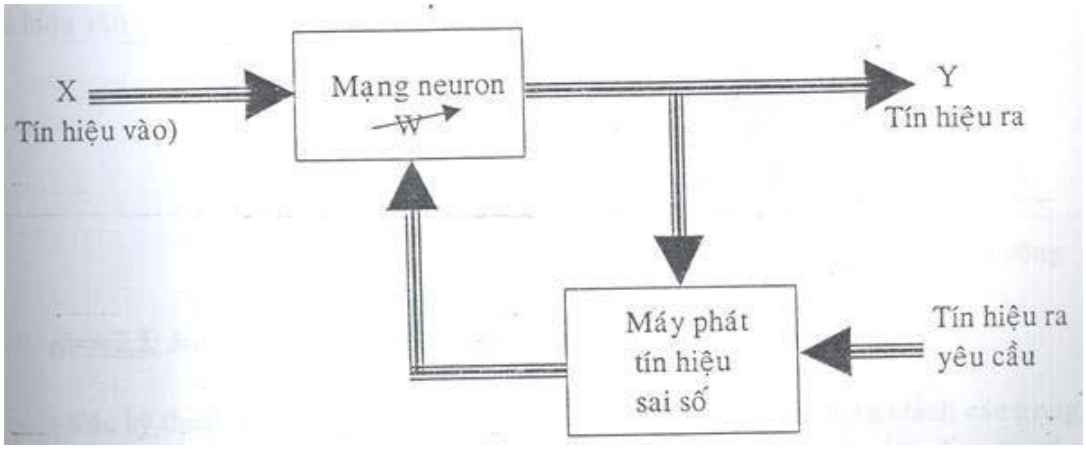
vào X=(x_1 , x_2 ,..., x_m) sau quá trình tính tốn tại các neuron ẩn sẽ nhận được kết quả

ra Y= (y_1 , y_2 ,..., y_n) và ta qui ước viết Y = out (X,NN).

+ Mạng neuron như một hệ thống thích nghi, có khả năng học

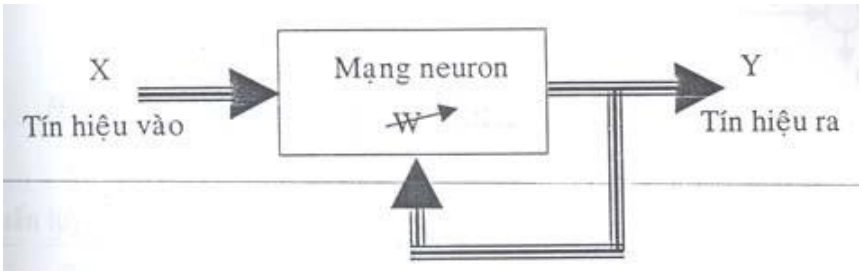
(huấn luyện) để tinh chỉnh các trọng số liên kết cũng như cấu trúc của chung sao cho phù hợp với các mẫu học (samples). Thường phân biệt 3 kỹ thuật học:

+ Học có giám sát (supervised learning), còn gọi là học có thầy: Mạng được cung cấp một tập mẫu học {(x,d)} theo nghĩa x là các tín hiệu vào thì kết quả đúng của hệ phải là d. Ở mỗi lần học, vector tín hiệu vào x được đưa vào mạng, sau đó so sánh sự sai khc giữa cc kết quả ra đúng d với kết quả tính tốn Y. Sai số này được dùng để hiệu chỉnh lại các trọng số liên kết trong mạng. quá trình cứ tiếp tục cho đến khi thỏa mãn một tiêu chuẩn nào đó. Có 2 cách sử dụng tập mẫu học: hoặc dùng các mẫu lần lượt – hết mẫu này đến mẫu khác, hoặc sử dụng đồng thời tất cả các mẫu cùng một lúc.



Hình 3.3 Mô hình học có giám sát

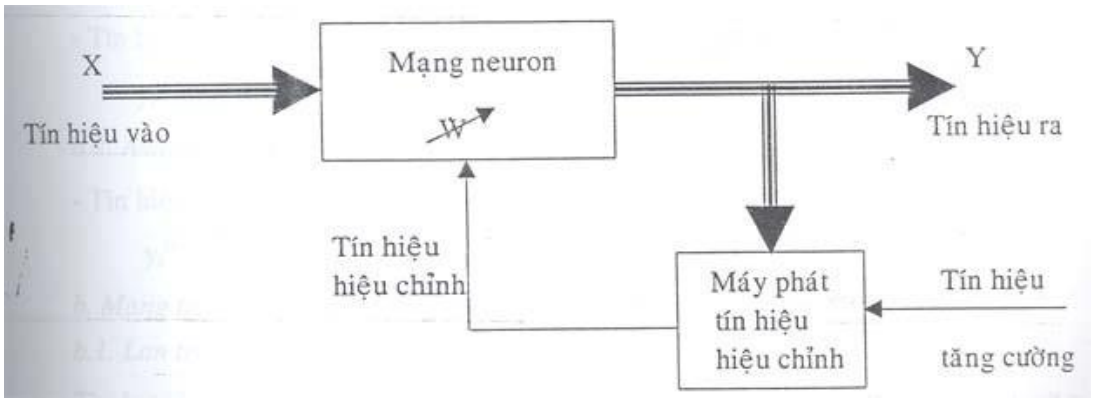
+ Học không có giám sát (unsupervised learning), còn gọi là học khôang thầy: Trong kỹ thuật học này, sẽ không có sự hồi tiếp từ môi trường để cho biết tín hiệu ra yêu cầu của mạng nêan như thế nào, hoặc chúng có đúng chưa – giống như học có giám sát, mà nói chung mạng neuron phải tự nó phát hiện ra bất cứ mối liên hệ có liên quan có thể tồn tại trong giữ liệu vào (chẳng hạn như: các mẫu, các đặc trưng, các quy tắc, sự tương quan) và chuyển mối liên hệ để phát hiện này sang đầu ra. Mạng học với cơ chế này gọi là mạng tự tổ chức. Thí dụ, mạng neuron học không thầy có thể cho chung ta biết một mẫu đầu vào mới đồng dạng như thế nào với mẫu đặc trưng để thấy trong quá khứ (sự đồng dạng); hoặc một dạng neuron khác có thể xây dựng một tập những cái rìu trên cơ sở sự tương tự của những thí dụ trước đó (phân tích thành phần chủ yếu)v.v…



Hình 3.4 Mô hình học không có giám sát

+ Học tăng cường (Reinforced learning): Như đã giới thiệu ở

trên, kỹ thuật học có giám sát là hiệu chỉnh dần giá trị tín hiệu đầu ra tương ứng với từng cặp mẫu tín hiệu vào-ra. Tuy nhiên thực tế nhiều khi không thể có được các thông tin chi tiết này. Do đó thường phải sử dụng thuật tốn "học tăng cường". trong học tăng cường
, dữ liệu huấn luyện rất thô và chúng chỉ " ước lượng" để so sánh với "sự truyền kiến thức" hồi tiếp trong học có giám sát.



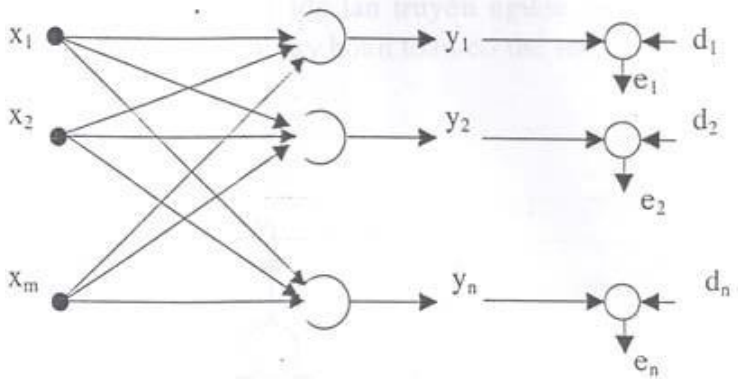
Hình 3.5 Mô hình huấn luyện tăng cường

+ Các kỹ thuật học trong mạng neuron có thể nhằm vào việc hiệu chỉnh các trọng số liên kết – gọi là học tham số; hoặc nhằm vào việc điều chỉnh, sửa đổi cấu trúc của mạng, bao gồm số lớp, số neuron, kiểu và trọng số các liên kết – gọi là học cấu trúc.

3.2 MỘT SỐ MƠ HÌNH MẠNG NEURAL

1. Mạng truyền thẳng (Feedforward neural Networks)

a. Mạng Perceptron đơn lớp



Hình 3.6 Mạng perceptron đơn lớp

- Tập mẫu tín hiệu vào: $x^{(k)}=[x_1^{(k)},x_2^{(k)},\ldots,x_m^{(k)}]^T$;k=1..p, với p là số tập mẫu huấn luyện; m là số tín hiệu vào trong một tập mẫu
- Tập mẫu tín hiệu ra tương ứng với tập mẫu tín hiệu vào: $d^{(k)}=[d_1^{(k)},d_2^{(k)},\ldots,d_n^{(k)}]^T$
- Tập tín hiệu ra thực tế ứng với tín hiệu mẫu $x^{(k)}$: $y^{(k)}=[y_1^{(k)},y_2^{(k)},\ldots,y_n^{(k)}]^T$
- Vector trọng số: $w_i=[w_{i1},w_{i2},\ldots,w_{im}]$. i =1..n với n là số tín hiệu ra.

- Hàm kích hoạt neuron: a(.)
- Mục tiêu của quá trình huấn luyện mạng là:

$$Y_i^{(k)}=a(w_i^T x^{(k)}) = a\left[\sum_{j=1}^{m} w_{ij} x_j^{(k)}\right] = d_i^{(k)}$$

a.1 Qui luật học Perceptron

Tín hiệu ra$_{mẫu}$ có dạng tuyến tính ngưỡng, chỉ nhận gi trị ± 1.

từ (2.4) ta có: $Y^{(k)} = sgn(w^T x^{(k)}) = d^{(k)}$ (3.5)

a.2 Adaline (Adaptive linear Element)

Tính hiệu ra$_{mẫu}$ cĩ dạng tuyến tính dốc, từ (3.4) ta có:

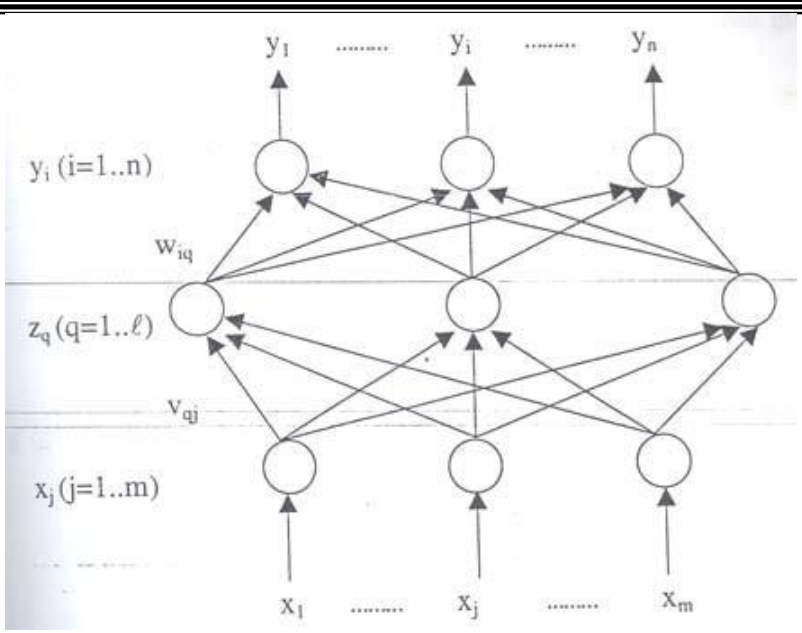
$$Y_i^{(k)} = (w_i^T x^{(k)}) = d_i^{(k)} \quad (3.6)$$

b. Mạng truyền thẳng đa lớp (Multilayer feedforward networks)

b.1 Lan truuyền ngược (Back propagation)

Thuật tốn huấn luyện lan truyền ngược có một ý nghĩa quan trọng trong lịch sử phát triển mạng neuron. Các mạng neuron được huấn luyện được huấn luyện bằng thuật tốn này gọi là mạng lan truyền ngược. với các tập học {($x^{(k)}$, $d^{(k)}$}, k=1..p, thuật tốn lan truyền ngược đưa ra các thủ tục để thay đổi trọng số trong mạng lan truyền ngược. Cơ sở của việc cập nhật các trọng số này là cơ chế suy giảm gradient.

Với một cặp tính hiệu mẫu vào-ra ($x^{(k)}$, $d^{(k)}$, thuật tốn lan truyền ngược thực hiện 2 giai đoạn. Đầu tiên, mẫu $x^{(k)}$ được lan truyền từ lớp vào đến lớp ra và cho ra tín hiệu thật ở lớp ra là $y^{(k)}$. Sai số giữa tín hiệu thật ở lớp ra $y^{(k)}$ và tín hiệu ra mẫu $d^{(k)}$ được lan truyền ngược trở lại từ lớp ra đến những lớp trước đó để cập nhật lại trọng số cho chung. Xét cụ thể một mạng neuron 3 lớp lan truyền ngược (hình 3.7) để minh họa thuật tốn lan truyền ngược và kết quả này hoàn toàn có thể mở rộng cho những mạng có số lớp nhiều hơn.



Hình 3.7 Mạng neural 3 lớp lan truyền ngược

Mạng neuron 3 lớp lan truyền ngược hình 3.7 có m neuron ở lớp vào, l
neuron ở lớp ẩn vàe n neuron ở lớp ra.

Đầu tiên, xét cặp mẫu huấn luyện (x,d). với tín hiệu mẫu đầu vào x ở lớp vào, neuron q ở lớp ẩn sẽ nhận được tín hiệu:

$$\text{Net}_q = \sum_{j=1}^{m} v_{qj} x_j \qquad (3.7)$$

Tín hiệu của neuron q ở lớp ẩn là:

$$Z_q = a(\text{net}_q) = a\left[\sum_{j=1}^{m} v_{qj} x_j\right] \qquad (3.8)$$

Tín hiệu vào của neuron I ở lớp ra:

$$\text{Net}_i = \sum_{q=1}^{\ell} w_{iq} z_q = \sum v_{iq} .a\left[\sum_{j=1}^{m} v_{qj} x_j\right] \qquad (3.9)$$

Tín hiệu ra của neuron I ở lớp ra:

$$Y_i = a(\text{net}_i) = a\left[\sum_{q=1}^{\ell} w_{iq} z_q\right] = a\left[\sum_{q=1}^{\ell} w_{iq} \cdot \left[a\sum_{j=1}^{m} v_{qj} z_j\right]\right] \qquad (3.10)$$

Định nghĩa hàm sai số:

$$E(w) = \tfrac{1}{2}\sum_{i=1}^{n}(d_i - y_i)^2 = \tfrac{1}{2}\sum_{i=1}^{n}\left[d_i - a\left[\sum_{q=1}^{\ell} w_{iq} z_q\right]\right]^2 \qquad (3.11)$$

Theo phương pháp suy giảm gradient, trọng số các neuron giữa lớp ẩn ra được cập nhật như sau:

$$\begin{aligned}\Delta w_{iq} &= -\eta\left[\frac{\partial E}{\partial w_{iq}}\right] \\ &= -\eta\left[\frac{\partial E}{\partial y_i}\right]\left[\frac{\partial y_i}{\partial net_i}\right]\left[\frac{\partial net_i}{\partial wiq}\right] \\ &= \eta\left[d_i - y_i\right]\left[a'(net_i)\right]\left[z_q\right] = \eta\delta_{oi} z_q\end{aligned} \qquad (3.12)$$

trong đó δ_{oi} là sai số tín hiệu ở lớp ra, được định nghĩa như sau:

$$\begin{aligned}\delta_{oi} &= -\frac{\partial E}{\partial net_i} = -\left[\frac{\delta E_i}{\delta y_i}\right]\left[\frac{\partial net_i}{\partial net_i}\right] \\ &= \left[d_i - y_i\right]\left[a'(net_i)\right]\end{aligned} \qquad (3.13)$$

Để cập nhật trọng số lin kết giữa lớp vo với lớp ẩn ta cũng dùng phương pháp suy giảm gradient, cụ thể giữa neuron thứ j ở lớp vo với neuron q ở lớp ẩn như sau:

$$\Delta v_{qj} = -\eta\left[\frac{\partial E}{\partial v_{qj}}\right] = \eta\left[\frac{\partial E}{\partial net_q}\right]\left[\frac{\partial net_q}{\partial v_{qj}}\right]$$

$$= -\eta\left[\frac{\partial E}{\partial z_q}\right]\left[\frac{\partial z_q}{\partial net_q}\right]\left[\frac{\partial net_q}{\partial v_{qj}}\right]$$

$$= \eta\sum_{i=1}^{n}[(d_i - y_i).a'(net_i)w_{iq}].a'(net_q).x_j \quad (3.14)$$

$$\Delta v_{qj} = \eta\sum_{i=1}^{n}[d_{oi}w_{iq}].a'(net_q)x_j = \eta\delta_{hq}x_j \quad (3.15)$$

trong đó δ_{hq} là sai số tín hiệu ở lớp ẩn, được định nghĩa như sau:

$$\delta_{hq} = -\frac{\partial E}{\partial net_q} = -\left[\frac{\partial E}{\partial z_q}\right]\left[\frac{\partial z_q}{\partial net_q}\right] = a'(net_q)\sum_{i=1}^{n}\delta_{oi}w_{iq} \quad (3.16)$$

Nhận thấy rằng sai số tín hiệu lớp ẩn khác với sai số tín hiệu lớp ra. Và chính sự khác nhau này nên thủ tục để cập nhật các trọng số liên kết được gọi là "qui luật học tổng quát hóa delta".

Tổng quát, với mạng neuron có số lớp tùy ý, quy luật cập nhật lan truyền ngược sẽ có dạng:

$$\Delta w_{ij} = \eta \delta_i x_j = \eta \delta_{output-i} \cdot x_{input-j} \qquad (3.17)$$

trong đó , "output-i" và "input-j" là 2 điểm kết thc nối từ neuron j đến neuron i;x_j l tín hiệu vo; d_i l sai số tín hiệu, được xác định theo (2.13) cho lớp ra hoặc (2.16) cho lớp vào.

b.2 Các hệ số học trong thuật tốn lan truyền ngược

- Khởi tạo giá trị trọng số liên kết neuron (Intial weights): Trong mạng truyền thẳng, giá trị trọng số được khởi tạo ban đầu có ảnh hưởng quan trọng đến kết quả cuối cùng. Chúng được gán ngẫu nhiên với giá trị tương đối nhỏ, vì nếu quá lớn thì hàm sigmoid sẽ dễ bão hòa ngay lúc bắt đầu, dẫn đếan hệ thống sẽ bị "nghẽn" tại giá trị địa phương nhỏ nhất hoặc rất ổn định ở lân cận điểm bắt đầu. Do đó,tầm hợp lý của gi trị trọng số khởi tạo ban đầu thường nằm trong khoảng [$-3/k_i$, $3/k_i$], trong đó k_i l số đầu vào nối vào neuron I [Wessels and Barnard, 1992]
- Hằng số học (learning constant): Một hệ số quan trọng khác cũng làm ảnh hưởng đến hiệu quả và sự hội tụ của thuật tốn lan truyền ngược, đó là hằng số học . Sẽ không có một giá trị hằng số học cố định cho các trường hợp huấn luyện khác nhau, mà thường chúng được chọn thử nghiệm cho từng bi tốn cụ thể. Một hằng số học có giá trị lớn có thể làm gia tăng tốc độ hội tụ, nhưng kết quả cũng có thể sẽ bị cường điệu; Trong khi một hằng số học có giá trị nhỏ hơn thì có tác dụng ngược lại. tầm giá trị hằng số học thường dao động trong khoảng từ 10^{-3} đến 10.

Một vấn đề khác cũng được đặt ra là hằng số hằng sẽ được tốt nhất ở lúc bắt đầu huấn luyện, tuy nhiên sẽ không còn tốt nữa sau vài lần huấn luyện. Do đó, tốt nhất là dùng hằng số học thích nghi. Phương pháp trực giác để xác định hằng số học này là kiểm sốt riêng lẻ quá trình cập nhật trọng số để làm giảm hàm sai số; nếu không thì giảm

dần chung nếu kết quả cường điệu; hoặc trong trường hợp khi nhiều bước lặp đều có sự suy giảm hàm sai số dẫn đến bi tốn quá hội tụ, thì nên tăng dần hằng số học lên. Cụ thể nhất, hằng số học nên được cập nhật theo các quy luật sau:

$$\Delta\eta = \begin{cases} +a, \Delta E < 0 \\ -b\eta, \Delta E > 0 \\ 0, \end{cases} \qquad (3.18)$$

trong đó:

E là độ lệch hm sai số; a,b l cc hằng số dương

Hoặc trong trường hợp dựa trên các bước huấn luyện trước thì:

$$\begin{cases} +a, & \tilde{\lambda}(t=1)\lambda(t) > 0 \\ -b\eta(t), & \tilde{\lambda}(t-1)\lambda(t) < 0 \\ 0, & \end{cases} \qquad (3.19)$$

trong đó:

$$\lambda(t) = \frac{\partial E}{\partial w_{ij}} \qquad (3.20)$$

$$\tilde{\lambda}(t) = (1-c)\lambda(t) + c\tilde{\lambda}(t-1), c \in [0,1] \text{ là hằng số} \qquad (3.21)$$

- Hàm sai số (Cost functions): Trong công thức tính hàm sai số (3.11), thnh phần sai số bình phương $(d_i-y_i)^2$có thể thay thế bởi bất kỳ một hàm $F(d_i,y_i)$ nào khác sao cho hàm này có thể đạt cực tiểu khi argument của d_i v y_i bằng nhau. Và tương ứng với hàm sai số mới này.

- Động lượng (Momentum): Sự suy giảm gradient có thể sẽ rất chậm nếu hằng số học quá lớn. Do đó , phương pháp làm chênh lệch đi khả năng dao động biên độ nhưng vẫn cho php sử dụng hằng số học lớn là đưa thêm vào trọng số thành phần "động lượng" theo công thức sau:

$$w(t) = -h\text{Đ}E(t) + aw(t-1) \qquad (3.22)$$

trong đó: a Ỵ[0,1] l tham số động lượng và thường chọn gi trị l 0,9.

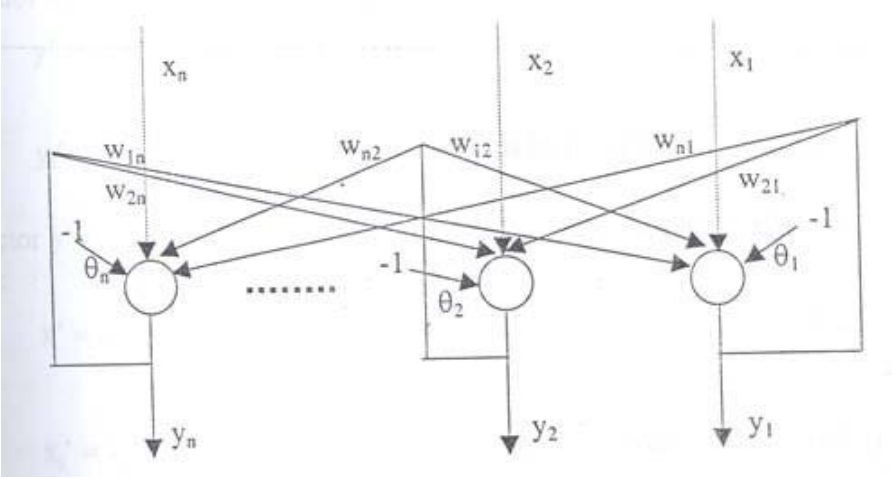
2. Mạng hồi qui (Recurrent/feedback neural networks)

a. Mạng hồi tiếp đơn lớp (Single-layer feedback networks)

a.1 Mạng Hopfield rời rạc

Mỗi một nút có một tín hiệu vào x_j v một ngưỡng q_j , trong đó j=1..n. ng ra của nút thứ j sẽ được nối tới ngõ vào của các nút khác với chính nó. Các liên kết này mang trọng số w_{ij}, với $i \neq j$ (nghĩa l w_{ii} = zero). Hơn nữa, trọng số liên i=1..n v

kết mạng là đối xứng, tức l $w_{ij} = w_{ji}$.



Hình 3.8: Cấu trc của mạng Hopfield

Qui luật cập nhật tại mỗi cực trong mạng Hopfield rời rạc (l mạng Hopfield hoạt động với thời gian rời rạc) như sau:

$$y_i^{(k+1)} = \mathrm{sgn}\left[\sum_{j=1, j\neq i}^{n} w_{ij} y_j^{(k)} + x_i - \theta_i\right], \quad i = 1,2,\ldots, n \qquad (3.23)$$

Mạng Hopfield rời rạc cĩ thể tổng qut hĩa thnh mơ hình lin tục khi thời gian được cho l biến lin tục.

a.2 Mạng Hopfield với bộ nhớ tự kết hợp (Autoassociative memory – AM): Thuật toán lưu trữ để xác định ma trận trọng số:

$$W_{ij} = \sum_{k=1}^{p} x_i^k x_j^k \qquad (3.24)$$

Trong đó:

$$\begin{cases} i \neq j \\ w_{ii} = 0 \\ x^k = (x_1^k, x_2^k, \ldots x_n^k)^T \\ p \end{cases}$$

Trường hợp x_i l vector nhị phân đơn cực, thì x^k Ỵ {0,1}v quy luật lưu trữ như sau:

$$w_{ij} = \sum_{k=1}^{p}(2x_i^k - 1)(x_j^k - 1) \qquad (3.25)$$

với:

$$\begin{cases} i \neq j \\ w_{ii} = 0 \end{cases}$$

với:

a.3 Mạng Hopfield với bộ nhớ 2 chiều kết hợp (Bidirectional associative memory)

$$y' = a(w_x) \qquad (3.26)$$

hoặc:

$$y_i' = a\left[\sum_{j=1}^{m} w_{ij}x_j\right] \text{ với I = 1..n; a là hàm kích hoạt} \qquad (3.27)$$

Đưa vector y' truyền ngược trở lại neuron lớp X, khi đó tín hiệu đầu ra neuron lớp X như sau:

$$x' = a(W^T y') \qquad (3.28)$$

hoặc:

$$x_j' = a\left[\sum w_{ji}y_j'\right] \text{ ,với j = 1..m; a là hàm kích hoạt} \qquad (3.29)$$

Tổng quát, ở bước truyền tới và truyền ngược thứ k/2, tín hiệu đầu ra neuron lớp Y và lớp X lần lượt là:

$$y^{(k-1)} = a(W x^{(k-2)}) \qquad (3.30)$$

$$x^{(k)} = a(W^T y^{(k-1)}) \qquad (3.31)$$

xét p cặp vector kết hợp đã được lưu trữ trong BAM:

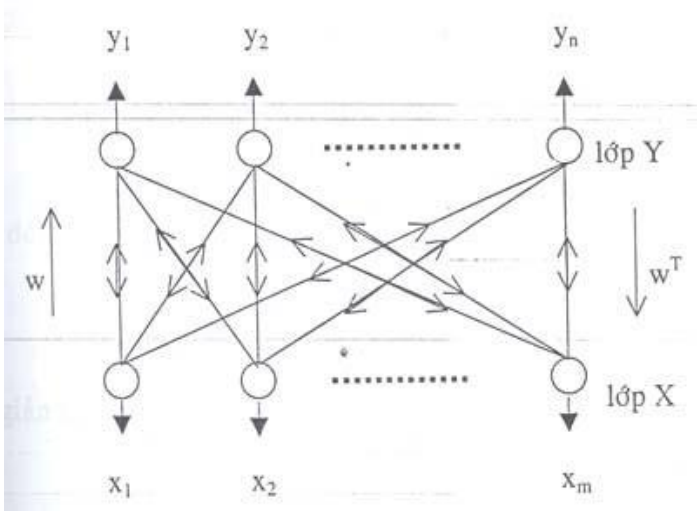
$$\{(x^1,y^1), (x^2,y^2),\ldots, (x^p,y^p)\} \qquad (3.32)$$

trong đó $x^k=(x_1^k,x_2^k,\ldots, x_m^k)^T$ và $y^k=(y_1^k,y_2^k,\ldots, y_n^k)^T$ là các vector đơn cực và lưỡng cực. Qui luật học trọng số cho BAM như sau:

$$W = \begin{cases} \sum_{k=1}^{p} y^k (x^k)^T \\ \sum_{k=1}^{p} (2y^k - 1)(2x^k - 1)^T \end{cases} \qquad (3.33)$$

hoặc:

$$w_{ij} = \begin{cases} \sum_{k=1}^{p} y_{pi}^k x_j^k \\ \sum_{k=1} (2y_i^k - 1)(2x_j^k - 1) \end{cases} \qquad (3.34)$$



Hình 3.9 Cấu trc của mạng Hopfield với bộ nhớ hai chiều kết hợp (BAM)

b. Mạng hồi qui lan truyền ngược (Recurrent back-propagation networks- RBP)

Tổng quát, xét mạng n nút với hàm kích hoạt a(.) v trọng số liên kết w_{ij} nối từ nút i đến nút j. trong số các nút này có nút đầu vào nhận tín hiệu vào x_i, định nghĩa x_i=o cho các nút không phải đầu vào. Cũng vậy, trong số các nút của mạng ta có nút đầu ra với gi trị yêu cầu (mẫu) d_i.

Quá trình kích hoạt tại mỗi nút được mô tả bởi phương trình vi phân:

$$W = \begin{cases} \sum_{k=1}^{p} y^k (x^k)^T \\ \sum_{k=1}^{p} (2y^k - 1)(2x^k - 1)^T \end{cases}$$

hoặc:

$$w_{ij} = \begin{cases} \sum_{k=1}^{p} y_{pi}{}^k x_j{}^k \\ \sum_{k=1}^{p} (2y_i{}^k - 1)(2x_j{}^k - 1) \end{cases}$$

cũng như các mạng lan truyền ngược khác, hàm sai số trong mạng

hồi qui lan truyền ngược cũng được tối thiểu hóa. Ta có:

$$E_{\cdot} = 1/_{k=1} 2E^2 k$$

trong đó:

$$E_k = \begin{cases} d_k - y_k, k \\ o, \end{cases} \tag{3.37}$$

dùng phương pháp suy giảm gradient, trọng số được cập nhật như sau:

$$\Delta w_{pq} = -\eta \frac{\partial E}{\partial w_{pq}} = \eta \sum_k E_k \frac{\partial y_k}{\partial w_{pq}} \tag{3.38}$$

để ước lượng $\partial y_k / \partial w_{pq}$, ta vi phân (3.36):

$$\frac{\partial y_i}{\partial w_{pq}} = a'(h_i)\left[\delta_{ip} y_q + \sum_j w_{ij} \frac{\partial y_j}{\partial w_{pq}}\right] \tag{3.39}$$

và được viết lại:

$$\frac{\partial y_i}{\partial w_{pq}} - \sum_j a'(h_i) w_{ij} \frac{\partial y_i}{\partial w_{pq}} = \sum_j \left[\delta_{ij} - a'(h_i) w_{ij}\right] \frac{\partial y_i}{\partial w_{pq}}$$

$$= \delta_{ip} a'(h_i) y_q \tag{3.40}$$

trong đó:

$$\delta_{ij} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \tag{3.41}$$

Đơn giản ký hiệu, ta có;

$$\sum_j L_{ij} \frac{\partial y_j}{\partial w_{pq}} = \delta_{ip} a'(h_i) y_q \qquad (3.42)$$

trong đó:

$$L_{ij} = \delta_{ij} - a'(h_i) w_{ij} \qquad (3.43)$$

Như vậy, ta có:

$$\frac{\partial y_k}{\partial w_{pq}} = [L^{-1}]_{kp} a'(h_p) y_q = \eta \delta_p y_q \qquad (3.44)$$

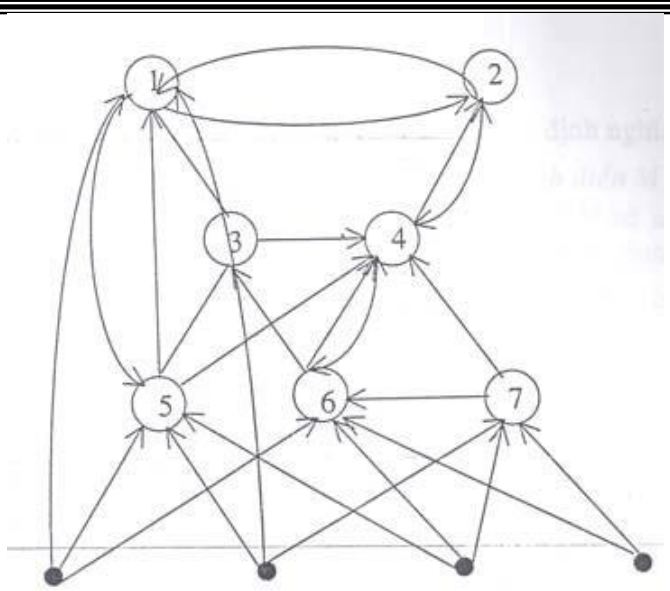
và qui luật học cho mạng RBP trở thành:

$$\Delta w_{pq} = \eta \sum_k \mathrm{E}_k [L^{-1}]_{kp} a'(h_p) y_q = \eta \delta_p y_q \qquad (3.45)$$

trong đó:

$$\delta_p = a'(h_p) \sum_k \mathrm{E}_k [L^{-1}]_{kp} \qquad (3.46)$$

Cùng họ với mạng hồi qui lan truyền ngược vừa nêu, còn có nhiều kiểu mạng hồi qui/hồi tiếp khác như: Mạng hồi qui một phần (Partially Recurrent networks), mạng hồi qui hồn tồn (Fully Recurrent Networks); hoặc mạng với thuật tốn học tăng cường (Reinforced learning).



Hình 3.10 Mạng hồi qui lan truyền ngược

Chương 4

NHẬN DẠNG VÂNTAY BẰNGMẠNGNEURAL

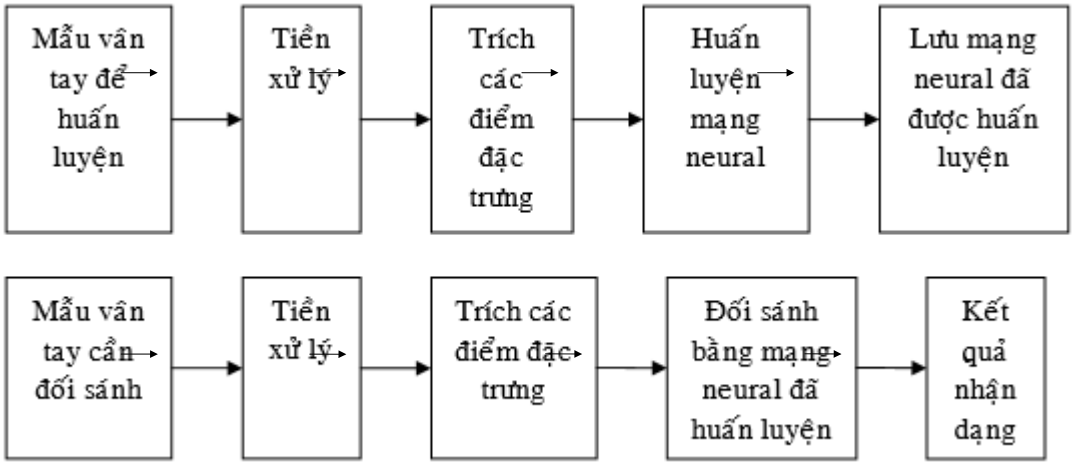
1. *Giới thiệu*
2. *Phương pháp đề nghị*
3. *Thuật toán huấn luyện mạng neural*

CHƯƠNG 4:

NHẬN DẠNG VÂN TAY BẰNG MẠNG NEURAL

4.1 GIỚI THIỆU:

Phương pháp nhận dạng vân tay bằng mạng neural nhân tạo (Artificial Neural Network) sẽ huấn luyện mạng neural dựa vào các mẫu dữ liệu vào là vị trí của các điểm đặc trưng của ảnh vân tay. Mạng neural sau khi được huấn luyện sẽ được dùng để đối sánh các mẫu vân tay cần nhận dạng. Lưu đồ thực hiện như sau:



PHƯƠNG PHÁP ĐỀ NGHỊ

1. *Lựa chọn mạng sử dụng: một ngõ ra hay nhiều ngõ ra?*

Nếu chọn mạng nhiều ngõ ra, mỗi ngõ ra tương ứng với một mẫu thì có sự bất cập:

- Bao nhiêu ngõ ra thì đủ?
- Mỗi lần cập nhật thêm một mẫu mới thì phải huấn luyện lại toàn mạng. Vì vậy, ở đây tôi chọn mạng truyền thẳng Perceptron một ngõ ra, mỗi mạng tương ứng với một mẫu. Như vậy, khi cần đối sánh một mẫu ta phải so sánh mẫu đó qua tất cả các mạng trong cơ sở dữ liệu. Bởi vì, việc so một mẫu qua các mạng đơn giản và nhanh hơn thời gian huấn luyện một mạng lớn nên phương pháp này khả thi hơn.

Trên cơ sở lựa chọn mạng như vậy tôi chọn hàm kích hoạt lớp ra là hàm tuyến tính và được huấn luyện về 0 đối với từng mẫu.

2. Xây dựng tập mẫu ngõ vào.

Ngõ vào của mạng là vị trí của các điểm đặc trưng. Để xác định vị trí của một điểm ta phải có một điểm gốc "tương đối" cố định. Ở đây, tôi chọn điểm core làm gốc tọa độ, bởi vì điểm core luôn tồn tại và tương đối cố định trong ảnh vân tay.

Việc đối sánh bằng mạng neural có một nhược điểm đó là thứ tự các điểm đặc trưng khi đưa vào mạng phải chính xác, chỉ cần sai lệch một vị trí sẽ làm sai toàn bộ mạng. Nhưng sai lệch là không thể tránh khỏi trong quá trình xác định các điểm đặc trưng đối với các ảnh có chất lượng không đảm bảo.

Để khắc phục nhược điểm này, tôi đề nghị một phương pháp đó là: không đưa trực tiếp vị trí của các điểm minutiae vào mạng (ngoại trừ điểm delta) mà sử dụng vị trí trung bình cộng của các điểm minutiae. Cụ thể như sau:

- Chọn điểm core làm gốc tọa độ, khi đó điểm core sẽ chia mặt phẳng ảnh thành bốn phần.
- Trong mỗi phần tư của mặt phẳng ảnh ta tìm vị trí trung bình của các điểm minutiae trong phần tư đó. Bốn vị trí trung bình của các điểm minutiae ở bốn phần tư của mặt phẳng ảnh sẽ được đưa vào tám ngõ vào của mạng (sử dụng tọa độ decac).
- Để gia tăng độ phân biệt ta có thể đưa thêm số điểm minutiae trong mỗi phần tư của mặt phẳng ảnh vào bốn ngõ vào khác của mạng.

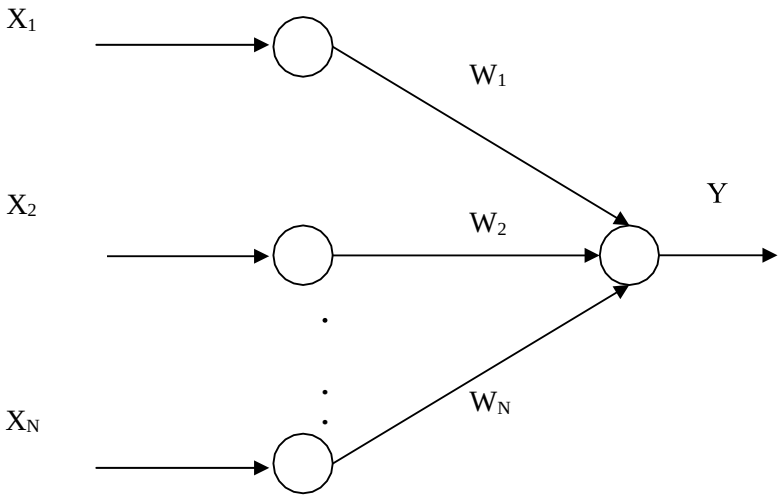
3. Số lớp sử dụng

Từ kinh ngiệm và thực nghiệm sử dụng mạng neural người ta nhận thấy là việc sử dụng mạng Perceptron nhiều hơn hai lớp là không cần thiết. Vì vậy, ở đây tôi sẽ thử nghiệm các kết quả đối sánh trên các mạng Perceptron một lớp và hai lớp.

4.2 THUẬT TOÁN HUẤN LUYỆN MẠNG NEURAL

Thuật toán huấn luyện được sử dụng là thuật toán lan truyền ngược suy giảm sai số gradient.

1. *Mạng Perceptron một lớp*



Hình 4.1: Mô hình mạng Perceptron một lớp

Trong đó:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \\ X_N \end{bmatrix}$$ Vector tín hiệu vào

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \\ W_H \end{bmatrix}$$ Vector trọng số

Y : tín hiệu ra

Thuật toán huấn luyện:

Bước 1:

Khởi động trị W(0)

Chọn hằng số học η

Bước 2: lan truyền thuận

Tính:

$Y(k) = net(k) = W^T X(k)$

Bước 3: lan truyền ngược

Tính:

$$W(k+1) = W(k) + \eta[D(k) - Y(k)]\frac{\partial a(net(k))}{\partial net(k)}$$ (a(.) là hàm kích hoạt)

Bước 4: lặp lại bước 3 K lần (một epoch), với K là số mẫu dữ liệu vào.

Bước 5: tính $J(K) = \frac{1}{2}[D(K) - Y(K)]^2$

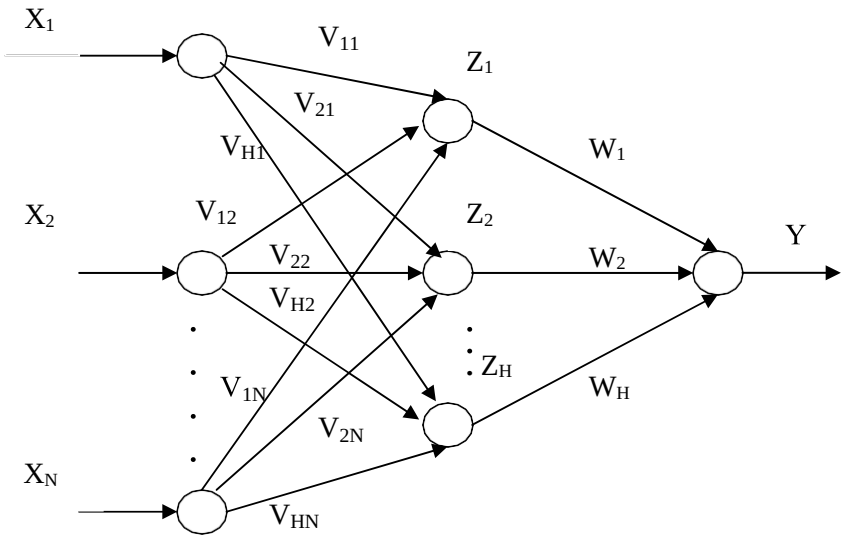
Bước 6: kiểm tra nếu J(K) đủ bé:

Đủ bé: kết thúc (lưu W)

Chưa: lặp lại bước 1 với các giá trị khởi động W(0) được cập nhật từ bước 4

2. Mạng Perceptron hai lớp (một lớp ẩn và một lớp ra)

Mạng Perceptron hai lớp (với một lớp ẩn và một lớp ra), được huấn luyện bằng giải thuật lan truyền ngược suy giảm sai số gradient. Hàm



kích hoạt lớp ẩn được chọn là hàm sigmoid, hàm kích hoạt lớp ra là hàm tuyến tính.

Hình 4.2: Mô hình mạng Perceptron hai lớp (một lớp ẩn và một lớp ra)

Trong đó:

$X = \begin{bmatrix} X_1 \\ X_2 \\ \\ X_N \end{bmatrix}$ Vector tín hiệu vào

$V_q = \begin{bmatrix} V_{q1} \\ V_{q2} \\ \\ V_{qN} \end{bmatrix}$ Các vector trọng số lớp ẩn ($q = \overline{1,...,H}$)

$W = \begin{bmatrix} W_1 \\ W_2 \\ \\ W_H \end{bmatrix}$ Vector trọng số lớp ra

$Z = \begin{bmatrix} Z_1 \\ Z_2 \\ \\ Z_H \end{bmatrix}$ Vector tín hiệu ra lớp ẩn.

Y : tín hiệu ra

Thuật toán huấn luyện:

Bước 1:

Khởi động trị W(0),

$V_q(0)$ Chọn hằng số học

Bước 2: lan truyền

thuận Tính:

$Z_q(k) = \text{sigmoid}(net_q) = \text{sigmoid}(V^T.X(k))$

$Y(k) = neto = W^T$

Z(k) Bước 3: lan truyền

ngược Tính:

$$d_O = D(k) - Y(k)$$

$$W(k+1) = W(k) + h.d_O.Z(k)$$

$$d_{hq} = d_O.W_q.\frac{e^{-netq}}{(1+e^{-netq})^2}$$

$$V_q(k+1) = V_q(k) + h.d_{hq}.X(k)$$

Bước 4: lặp lại bước 3 K lần (một epoch), với K là số mẫu dữ liệu vào.

Bước 5: tính $J(K) = \frac{1}{2}[D(K) - Y(K)]^2$

Bước 6: kiểm tra nếu J(K) đủ

bé: Đủ bé: kết thúc (lưu

W, V_q)

Chưa: lặp lại bước 1 với các giá trị khởi động $W(0)$, $V_q(0)$ được cập nhật từ bước 4

THỰC NGHIỆM VÀKẾTQUẢ

1. ***Chương trình***
2. ***Lưu đồ giải thuật***
3. ***Kết quả***
4. ***Đánh giá kết quả***

CHƯƠNG 5: THỰC NGHIỆM VÀ KẾT QUẢ

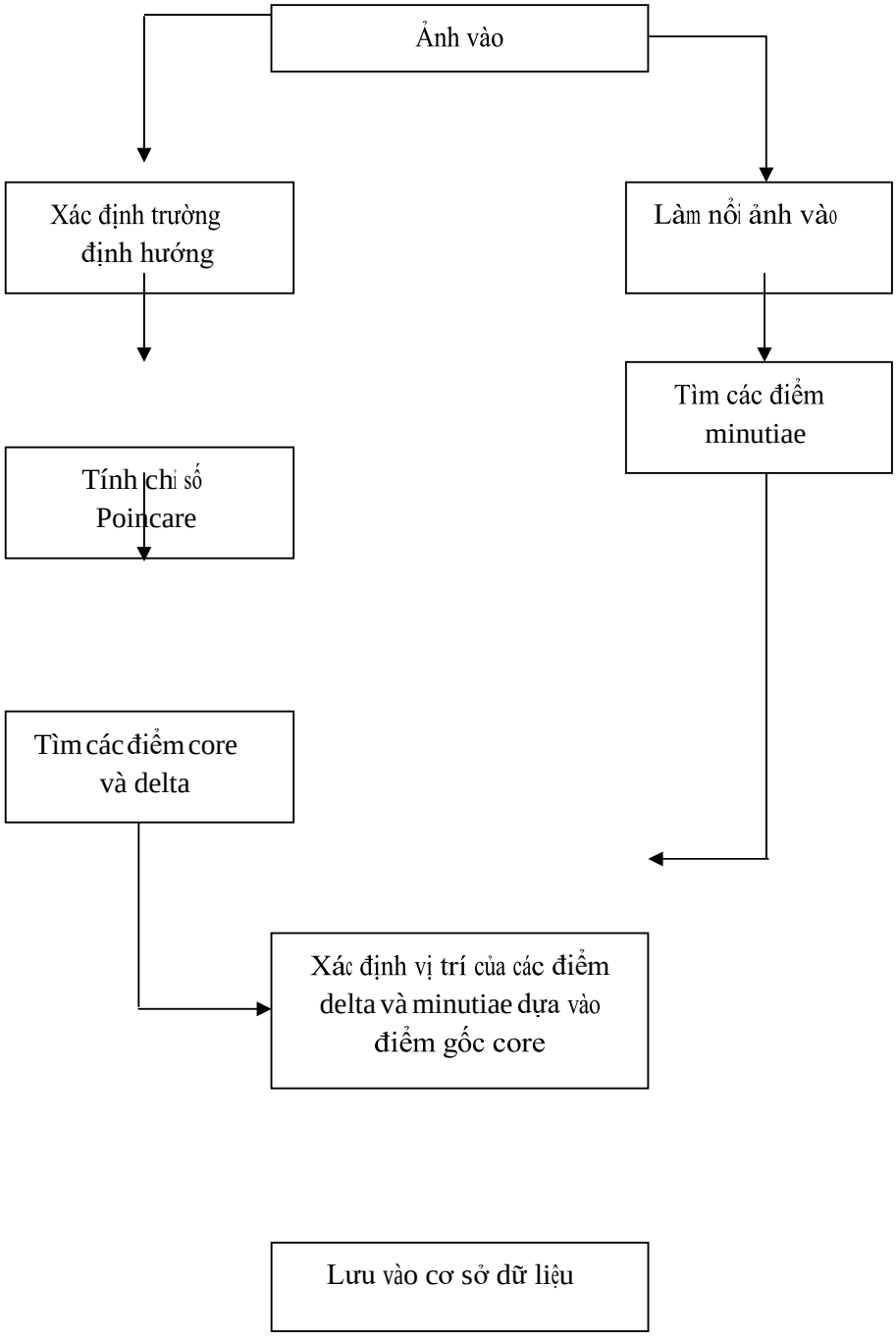
5.1 CHƯƠNG TRÌNH

Sử dụng:

- Ban đầu khi chưa có dữ liệu trong database, chọn "Lấy mẫu". Sau đó, click vào nút "mẫu" và chọn lần lượt ba ảnh vân tay của cùng một ngón tay để làm mẫu đối sánh. Tiếp theo, nhập các thông tin cá nhân của người có mẫu vân tay đó và click "OK".
- Sau đó, khi cần đối sánh một dấu vân tay nào đó chọn "đối sánh" và click vào nút "mẫu" để chọn ảnh vân tay cần đối sánh. Nếu mẫu này giống với mẫu nào có trong database thì thông tin cá nhân của người có dấu vân tay đó sẽ được thông báo.

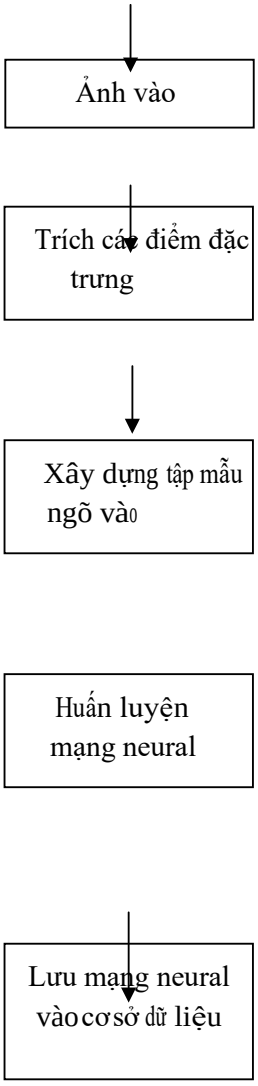
5.2 LƯU ĐỒ GIẢI THUẬT

1. Lưu đồ



Hình 5.2: Lưu đồ giải thuật trích các điểm đặc trưng

Tất cả thuật toán được nêu ở lưu đồ hình 5.2 được thực hiện bằng hàm *feature_extracting.m* (phụ lục).

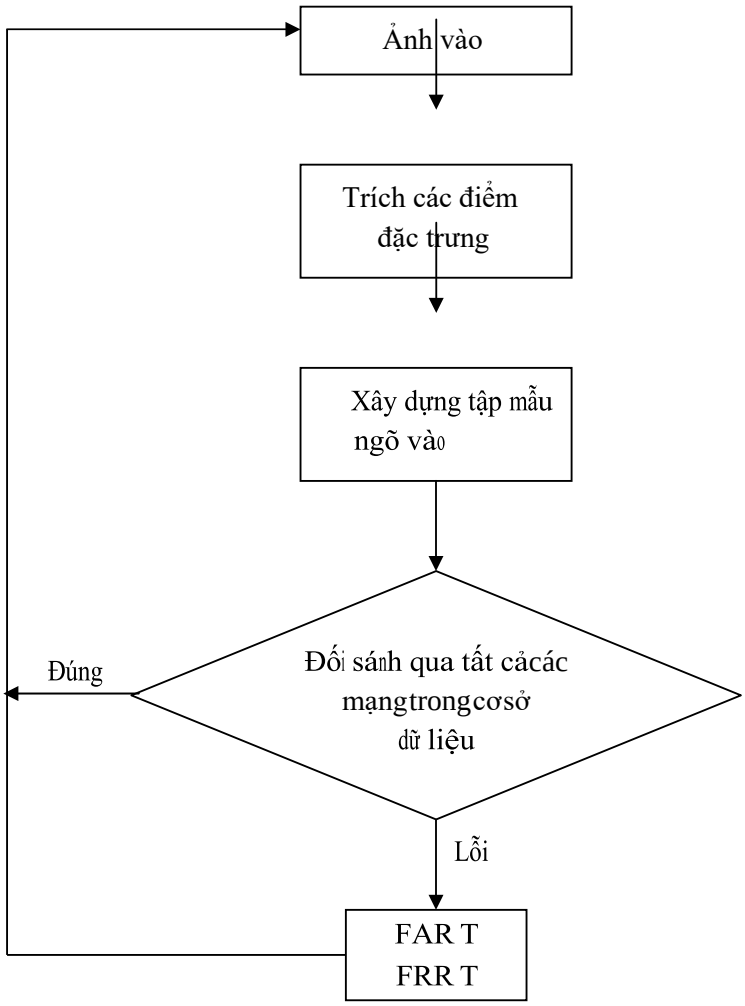


Hình 5.3: Lưu đồ giải thuật quá trình lấy mẫu

Các thuật toán ở lưu đồ trên được thực hiện bằng các hàm sau (phụ lục):

- *neural_template.m* : xây dựng mẫu ngõ vào cho mạng neural
- *training1.m* : huấn luyện mạng Perceptron một lớp

- *training2.m* : huấn luyện mạng Perceptron hai lớp



Hình 5.4: Lưu đồ giải thuật quá trình đối sánh

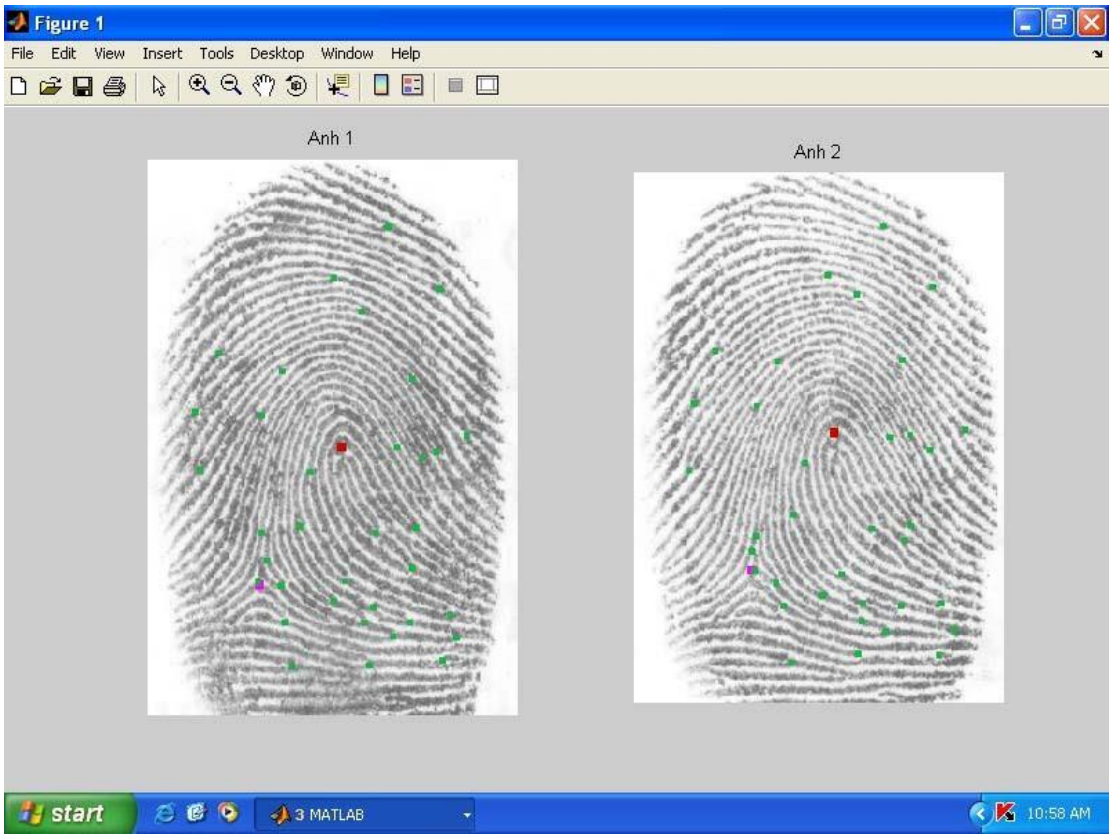
Việc đối sánh được thực hiện bằng cách cho mẫu cần đối sánh qua các mạng đã được huấn luyện trong cơ sở dữ liệu và so sánh ngõ ra Y với ngưỡng T:

- Y < T : hai mẫu giống nhau
- Y > T : hai mẫu khác nhau

Các thuật toán đối sánh và thu thập các lỗi FAR và FRR được thực hiện bằng các hàm *thuthap1.m* hoặc *thuthap2.m* (phụ lục).

2. Ví dụ minh họa

Hình 5.5 cho thấy các điểm đặc trưng (điểm màu đỏ là core, điểm màu tím làdelta và các điểm màu xanh là minutiae) của hai dấu vân tay khác nhau của cùng một ngón tay. Các điểm đặc trưng này được trích theo thuật toán được nêu ở lưu đồ hình 5.2 và bằng hàm feature_extracting.m (phụ lục).



Hình 5.5: Các điểm đặc trưng của hai dấu vân tay khác nhau của cùng một ngón tay

Sau đây là hai mẫu ngõ vào (một dùng để huấn luyện và một dùng để đối sánh) tương ứng với hai dấu vân tay trên. Hai mẫu này được xây dựng theo phương pháp được giới thiệu ở phần 4.2.

Theo quan sát ta thấy thường trên dấu vân tay chỉ có thể có một hoặc hai điểm delta (một bên phải và một bên trái) hoặc không có điểm delta

nào. Vì vậy, bốn giá trị đầu tiên của mẫu ngõ vào lần lượt là vị trí của hai điểm delta bên phải và bên trái nếu có (giá trị 300 hoặc -300 cho biết không có điểm delta ở vị trí đó). Các giá trị tiếp theo lần lượt là vị trí trung bình của các

điểm minutiae và số điểm minutiae ở bốn phần tư của mặt phẳng ảnh (như đãgiới thiệu ở phần 4.2). Như vậy, có tất cả 16 ngõ vào.

```
mau1 =
  300.00
  00
  300.0000
  109.0000
  -65.0000
  101.8667
   49.6667
   15.0000
   91.1000
   49.6000
   10.0000
   20.3733
    9.9333
    5.0000
   93.8000
   57.2000
    5.0000

mau2 =
  300.00
  00
  300.0000
  109.0000
  -65.0000
  105.8000
   50.2667
   15.0000
   97.1000
   48.9000
   10.0000
   21.1600
```

10.0533
5.0000
89.2000
58.6000
5.0000

Vector trọng số của mạng Perceptron một lớp tương ứng với mẫu 1, được
huấn luyện với hằng số học $h = 10^{-7}$ và sai số cho phép $e = 10^{-4}$ như sau:

W =

Columns 1 through 7

-0.2603 -0.2603 0.8691 0.1781 -0.0223
0.9404 0.0820

Columns 8 through 14

-0.0094 0.9404 0.0880 0.0755 0.9881 -0.1060 -0.2126

Columns 15 through 16

-0.1687 -0.1060

và giá trị ngõ ra khi cho lần lượt hai mẫu trên qua mạng được huấn luyện:

```
>> Y1 = W*mau1

Y1 =

   0.0139

>> Y2 = W*mau2

Y2 =

   0.6955
```

Như vậy, sai số đối sánh sẽ là Y2 – Y1 = 0.6816. Do đó, nếu chọn ngưỡng T>0.6816 thì hai mẫu này là giống nhau, ngược lại hai mẫu

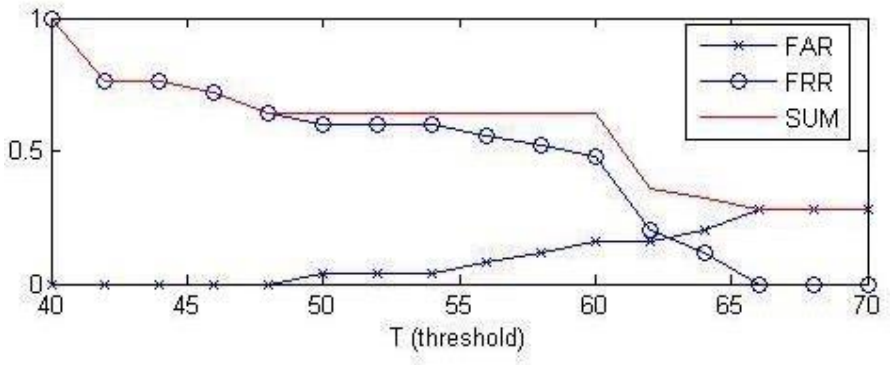
này là khác nhau.

5.3 KẾT QUẢ

Các kết quả được thu thập ở đây được thực hiện trên một cơ sở dữ liệu gồm 200 ảnh vân tay lấy từ 50 ngón tay khác nhau, mỗi ngón 4 ảnh (50 x 4 = 200). Độ phân giải của mỗi ảnh là 450 dpi. Từ 4 ảnh của mỗi vân tay, 1 ảnh được lấy làm mẫu huấn luyện mạng, 3 ảnh làm mẫu đối sánh.

Tất cả các thuật toán được thực hiện bằng ngôn ngữ MATLAB 7.0 trên máy PC Pentium IV 2.6 GHz 256 MB RAM.

Hình 5.6 cho thấy các kết quả đối sánh khi sử dụng mạng Perceptron một lớp. Đồ thị trên biểu diễn các tỷ lệ (%) lỗi chấp nhận nhầm FAR, từ chối nhầm FRR và tổng SUM = FAR + FRR theo ngưỡng T (giá trị sai lệch ở ngõ ra mạng neural so với giá trị mong muốn).

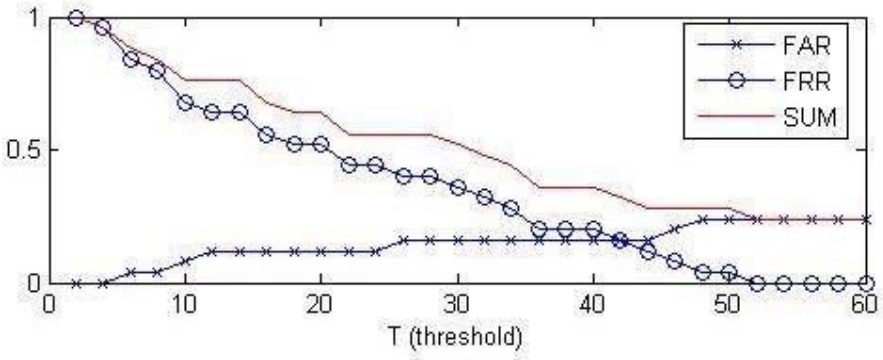


Hình 5.6 Kết quả đối sánh khi sử dụng mạng Perceptron một lớp.

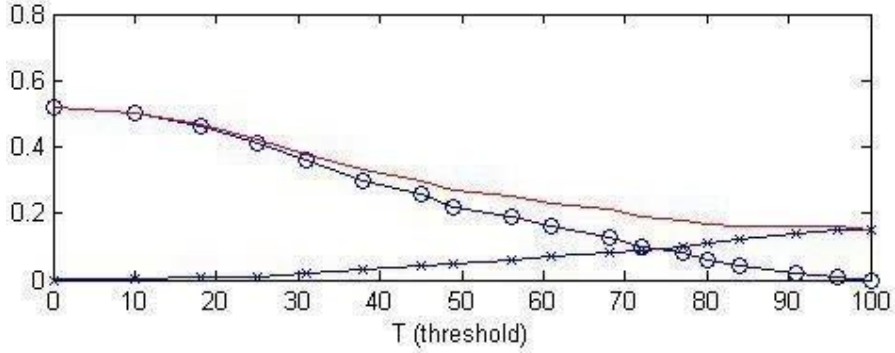
Ta thấy, khi tăng ngưỡng T thì FAR tăng, còn FRR giảm và ngược lại. Tùy theo yêu cầu về độ bảo mật của hệ thống ta có thể chọn giá trị ngưỡng T thích hợp. Ví dụ, khi hệ thống yêu cầu độ bảo mật cao ta chọn giá trị T nhỏ. Khi đó, lỗi nhận nhầm sẽ nhỏ nhưng lỗi từ chối nhầm sẽ lớn có thể gây phiền hà cho người sử dụng.

Để có đánh giá tổng quát hơn ta dùng tổng SUM = FAR + FRR, đây là tỷ lệ phần trăm lỗi xảy ra trên hệ thống. Rõ ràng, một hệ thống tốt phải có SUM nhỏ (trong trường hợp này SUM_{min} = 24%)

Hình 5.7 là các kết quả đối sánh khi sử dụng mạng Perceptron hai lớp, hình 5.8 là các kết quả đối sánh sử dụng phương pháp được giới thiệu ở phần 2.4



Hình 5.7 Kết quả đối sánh khi sử dụng mạng Perceptron hai lớp.



Hình 5.8: Kết quả đối sánh sử dụng phương pháp được giới thiệu ở phần 2.4

Bảng 5.1 tổng hợp một số kết quả thu thập được từ các phương pháp đối sánh bằng mạng Perceptron một lớp, hai lớp và phương pháp giới thiệu ở phần 2.4 (tạm gọi là phương pháp trực tiếp). Thời gian thu thập trong bảng dưới chưa tính thời gian trích các điểm đặc trưng (khoảng120s)

	Tổng lỗi nhỏ nhất (%)	Thời gian thu thập một mẫu dữ liệu (s)	Thời gian đối sánh một mẫu qua tập dữ liệu 50 mẫu (s)

Một lớp	24	0.143	0.1327
Hai lớp	22	2.175	0.2804
Trực tiếp	18	0	0.4764

Bảng 5.1 Một số kết quả tổng hợp từ các phương pháp đối sánh

5.4 ĐÁNH GIÁ KẾT QUẢ

- Phương pháp đối sánh bằng mạng neural được đề nghị có hiệu quả khi số điểm minutiae đủ lớn vì phương pháp cộng trung bình có tác dụng giảm sai số khi số phần tử cộng nhiều (lớn hơn 30) và số điểm lỗi nhỏ (nhỏ hơn 10%).
- Thời gian thu thập mẫu dữ liệu bằng phương pháp trực tiếp bằng 0 (không kể thời gian trích các điểm đặc trưng) bởi vì phương pháp này không cần phải huấn luyện mạng neural. Từ bảng 5.1 ta thấy thời gian huấn luyện mạng Perceptron hai lớp lớn hơn nhiều so với mạng một lớp, do thuật toán huấn luyện mạng hai lớp phức tạp hơn và thời gian hội tụ lâu hơn. Tuy nhiên, mạng hai lớp cho kết quả đối sánh tốt hơn và ổn định hơn mạng một lớp.
- Phương pháp trực tiếp có tác dụng sửa lỗi tốt hơn nên có tác dụng tốt khi
  chất lượng ảnh vân tay không đảm bảo (nhất là các ảnh được lấy từ mực). Nhưng việc đối sánh phức tạp hơn, do đó thời gian đối sánh lâu hơn.
- Các phương pháp đã được giới thiệu đều có ưu điểm và nhược điểm của nó. Tùy theo yêu cầu, điều kiện hiện có và mục đích sử dụng ta có thể chọn phương pháp phù hợp để sử dụng.

HƯỚNG PHÁT TRIỂN

CỦA ĐỀ TÀI

CHƯƠNG 6:

HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

- Hiện tại, đề tài chỉ nghiên cứu và thử nghiệm mạng Perceptron. Trong tương lai có thể nghiên cứu và thử nghiệm trên các mạng khác, chẳng hạn như mạng neural mờ .v.v.
- Có thể nghiên cứu kết hợp thêm các đặc điểm khác của ảnh vân tay ngoài các điểm đặc trưng như: Orientation field, Density map . . . để gia tăng hiệu quả đối sánh.
- Thuật toán làm nổi ảnh vân tay cần được nghiên cứu thêm phương pháp lựa chọn các thông số T, s_x, s_y để cải thiện hiệu quả của việc lọc.
- Các kết quả trên được thu thập trên một cơ sở dữ liệu còn khiêm tốn (50). Nếu có điều kiện có thể thử nghiệm trên các tập mẫu lớn hơn (100 – 200).

PHỤ LỤC

1. Hàm orientation.m

```
function O = orientation(I,R,Wo)
% Ham tim truong dinh huong cua anh I
% Wo (le): kich thuoc cua so xem xet;

I=double(I)
;
[m,n]=size(
I);
O(1:m,1:n)=pi;
[Gy,Gx]=gradient(I);
for i=(Wo+1)/2:m-
  (Wo+1)/2 for
  j=(Wo+1)/2:n-
  (Wo+1)/2
    if
      R(i,j)=
      =0
      A=0;
      B=0;
      for h=i-(Wo-1)/2:i+(Wo-1)/2
    for k=j-(Wo-1)/2:j+(Wo-
                    1)/2
      A=A-Gx(h,k)*Gy(h,k);
          B=B+Gx(h,k)^2-
        Gy(h,k)^2; end;
      end;
      O(i,j)=atan2(2*
      A,B)/2;
    en
  d;
  end;
end;
```

2. Hàm poincare.m

```
function P = poincare(O,N,i,j)
```

```
% Ham tinh chi so poincare de xet diem Singularity tai diem (i,j)
% O: Truong dinh huong cua anh
% N (le): Kich thuoc "duong cong so" (curve) xem xet

a=(N-
1)/2;
d=0;
do=O(i-
a,j-a); for
k=j-
a+1:j+a
   delta=O(i-
   a,k)-do; if
   delta<-pi/2
      delta=pi+d
   elta; elseif
   delta>pi/2
      delta=delt
   a-pi; end;
   d=d+del
   ta;
   do=O(i-
   a,k);
end;
for h=i-
   a+1:i+a
   delta=O(h,j+
   a)-do; if
   delta<-pi/2
      delta=pi+d
   elta; elseif
   delta>pi/2
      delta=delta-pi;
```

```
  end;
  d=d+delt
  a;
  do=O(h,j+
  a);
end;
for k=j+a-1:-
  1:j-a
  delta=O(i+a,
  k)-do; if
  delta<-pi/2
    delta=pi+d
  elta; elseif
  delta>pi/2
    delta=delt
  a-pi; end;
  d=d+delt
  a;
  do=O(i+a,
  k);
end;
for h=i+a-1:-
  1:i-a
  delta=O(h,j-
  a)-do; if
  delta<-pi/2
    delta=pi+d
  elta; elseif
  delta>pi/2
    delta=delt
  a-pi; end;
  d=d+del
  ta;
  do=O(h,j
  -a);
end;
P=round(d*18
0/pi);
```

3. Hàm singularity.m

```
function [core delta] = singularity(R,O,N1,N2)
% Ham tim cac diem Singularity tren anh van tay
% core: Vi tri cua diem core
% delta: Vi tri tuong doi cua diem delta so voi diem core (toa do decac)
% R: Mat na vung van tay (region mask)
% O: Truong dinh huong cua anh van tay
% N1 (le): Kich thuoc "duong cong so" (curve) xem xet (theo phuong phap
%      poincare) de tim diem core
% N2 (le): Kich thuoc "duong cong so" (curve) xem xet (theo phuong phap
%      poincare) de tim diem delta

[m n]=size(O);
S1(1:m,1:n)=0;
S2(1:m,1:n)=0;
%---------- Tim tat ca cac diem singularity (co the co diem loi) ----------
for
  i=N1+1:m-
  N1 for
  j=N1+1:n-
  N1
    if (R(i-N1,j)==0)&(R(i+N1,j)==0)&(R(i,j-N1)==0)&(R(i,j+N1)==0)
      P=poincare(O,
      N1,i,j); if
      (P==360)|(P==
      180)
        S1(i,j)=1; % Whorl
      or loop end;
    en
  d;
  end;
end;

for i=N2+1:m-N2
```

```
  for j=N2+1:n-N2
    if (R(i-N2,j)==0)&(R(i+N2,j)==0)&(R(i,j-N2)==0)&(R(i,j+N2)==0)
      P=poincare(O,
      N2,i,j); if P==-
      180
        S2(i,j)=1; %
      Delta end;
    en
  d;
  end;
end;

%.....Tim diem core .....
core=0.4;
i_core=[];
j_core=[];
for i=(N1+1)/2:m-
  (N1-1)/2 for
  j=(N1+1)/2:n-
  (N1-1)/2
    S1_sub=S1(i-(N1-1)/2:i+(N1-1)/2,j-(N1-1)/2:j+(N1-1)/2);
    if
      (mean(mean(S1_sub
      ))>=core)
      i_core=[i_core i];
      j_core=[j_core j];
    en
  d;
  end;
end;
i_core=round(mean(i
_core));
j_core=round(mean(j
_core)); core=[i_core
j_core];

%---------- Tim diem delta ben phai (neu co) ----------
delta=
0.5;
i_delta
```

```
=0;
j_delta
=0;
for
  i=i_core+(N2+1)/2:m-
  (N2-1)/2              for
  j=j_core+(N2+1)/2:n-
  (N2-1)/2
    S2_sub=S2(i-(N2-1)/2:i+(N2-1)/2,j-(N2-1)/2:j+(N2-1)/2);
    if
      (mean(mean(S2_su
      b))>delta)
      delta=mean(mean(
      S2_sub)); i_delta=i;
      j_delta=j
    ; end;
  end;
end;
delta_right1=300;
delta_right2=300;
if
  (i_delta~=0)&(j_de
  lta~=0)
  delta_right1=i_delt
  a-i_core;
  delta_right2=j_delt
  a-j_core;
end;
delta_right=[delta_right1;delta
_right2;0];

%--------------------- Tim diem delta ben trai (neu co)----------------------
delta=
0.5;
i_delta
=0;
j_delta
=0;
for
  i=i_core+(N2+1)/2:
  m-(N2-1)/2          for
```

```
j=(N2+1)/2:j_core-
(N2-1)/2
  S2_sub=S2(i-(N2-1)/2:i+(N2-1)/2,j-(N2-1)/2:j+(N2-1)/2);
```

```
    if
      (mean(mean(S2_su
      b))>delta)
      delta=mean(mean(
      S2_sub)); i_delta=i;
      j_delta=j
    ; end;
  end;
end;
delta_left1=
300;
delta_left2=
-300;
if
  (i_delta~=0)&(j_d
  elta~=0)
  delta_left1=i_delta
  -i_core;
  delta_left2=j_delta
  -j_core;
end;
delta_left=[delta_left1;delta
_left2;0];

%.....
delta=[delta_right delta_left];
```

4. Hàm minutiae.m

```
function R_minutiae = minutiae(I,R,core)
% Ham tim cac diem dac trung cua anh van tay I
% v: buoc nhay cua is, js
% sigma: 1/2 be rong van
% T: anh sau khi da giac hoa cua I
% T1: ma tran chua cac gia tri cua phuong tiep tuyen, tuong ung voi cac
% duong van sau khi da giac hoa.
% T2: ma tran chua cac gia tri cua phuong tiep tuyen tai cac diem dac
trung

v=4;
sigma=5;
I=255-
```

```
double(I);
[m
n]=size(I);
T(1:m,1:n)=0;
for
  is=26:v:m-
  25 for
  js=26:v:n-
  25
    if ((is>core(1)-20)&(is<core(1)+20)&(js>core(2)-
    15)&(js<core(2)+15))|(R(is,js)==255) else
      [ic
      jc]=ridgenearest(I,is,js,sig
      ma); label=true;
      if
        R(ic,jc)==255
        label=false;
      else
        for i=ic-
          4:ic+4 for
          j=jc-
          4:jc+4
            if T(i,j)==255
              label=false;
            en
          d;
          end;
        end;
      end;
      if label
        [T
        R]=ridgefollowing(I,T,R,ic,jc,core,
        sigma,2.4); [T
        R]=ridgefollowing(I,T,R,ic,jc,core,si
        gma,-2.4);
      end;
```

```
    en
  d;
  end;
end;
R_minutiae
=R;

%.....
function [T,R] =
ridgefollowing(I,T,R,ic,jc,core,sigma,mu
y)
% Ham do theo duong van tay
% muy: buoc do

[m
n]=size(I);
io=ic;
jo=jc;
if muy>0
  object=0;
  muy0=m
  uy;
else
  object=p
  i;
  muy0=-
  muy;
  muy=0;
end;
object1
=0;
phi00=tangentDir(I,
io,jo,9);
phi0=phi00+object;
esc=true;
while (esc)
  it=round(io+muy*sin(
  phi0));
  jt=round(jo+muy*cos(p
  hi0)); muy=muy0;
```

```
if ((it>26)&(it<m-25)&(jt>26)&(jt<n-25))&(R(it,jt)~=255)
  if (it>core(1)-20)&(it<core(1)+20)&(jt>core(2)-
  15)&(jt<core(2)+15)
    R(it,jt)=-3;     % Diem ngat do
    cham vao vung core esc=false;
  else
    phit=tangentDir(I,it,jt,9);
    [In in
    jn]=localmax(I,it,jt,phit,si
    gma); if
    (io~=ic)&(jo~=jc)&(in==
    io)&(jn==jo)
      esc=fal
    se; else
      phinn=tangentDir(I,in,jn,9);
      if abs(phinn-phiOO)>5*pi/6 % Duong van
        doi huong tu 90 do if phiO>pi/2    %
        sang -90 do hoac nguoc lai
          object1=0
        ; else
          object1
        =pi; end;
        object=0;
      end;
      phin=phinn+object+
      object1;
      [T R
      esc]=stopcriteria(I,T,R,In,in,jn,p
      hin); io=in;
      jo=jn;
      phiOO=phin
      n;
      phiO=phin;
    en
  d;
  end;
```

```
  else
    esc=fal
  se; end;
end;

%.....
function [ic,jc] =
ridgenearest(I,is,js,sigma)
% Ham xac dinh diem cuc dai cuc bo (ic,jc) gan (is,js) nhat

phiC=tangentDir(I,is,js,9);
i1=round(is-(sigma-
3)*cos(phiC+pi/2));
j1=round(js-(sigma-
3)*sin(phiC+pi/2)); [Ic1 ic1
jc1]=localmax(I,i1,j1,phiC,sig
ma); i2=round(is+(sigma-
3)*cos(phiC+pi/2));
j2=round(js+(sigma-
3)*sin(phiC+pi/2)); [Ic2 ic2
jc2]=localmax(I,i2,j2,phiC,sig
ma); d1=(ic1-is)^2+(jc1-
js)^2;
d2=(ic2-
is)^2+(jc2-js)^2;
if d1<d2
  ic=ic
  1;
  jc=jc
  1;
else
  ic=ic
  2;
  jc=jc
  2;
end;

%.....
function [T,R,esc] =
stopcriteria(I,T,R,In,in,jn,phin)
% Ham dieu kien dung cua thuat toan
```

```
% Ham nay se xet xem diem (in,jn) voi phuong tiep tuyen la phin co
thoa
% dieu kien dung cua thuat toan khong.
% phiO: phuong tiep tuyen cua diem (io,jo) lien truoc diem (in,jn).

esc=true;
phin=phin*18
0/pi;
I_nguong=20;
if ((phin>-90)&(phin<=-
   67.5))|((phin>247.5)&(phin<=270))
   case_phin=1;
elseif(phin>-
   67.5)&(phin<=-22.5)
   case_phin=2;
elseif(phin>-
   22.5)&(phin<=22.5)
   case_phin=3;
elseif
   (phin>22.5)&(phin<
   =67.5) case_phin=4;
elseif
   (phin>67.5)&(phin<
   =112.5)
   case_phin=5;
elseif
   (phin>112.5)&(phin<
   =157.5) case_phin=6;
elseif
   (phin>157.5)&(phin<
   =202.5) case_phin=7;
else
   case_phin
=8; end;
%.....
switch case_phin
   case 1
```

```
if T(in-
  1,jn)==255
  label=true;
  for i=in-
    6:in+4
    for j=jn-
    4:jn+4
      if (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==-2)|(R(i,j)==-3)
        R(i,j)=-2; % Mot diem minutae loi
        da duoc phat hien label=false;
      end;
    end;
  end;
  if (label)&(R(in-20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-
  20)~=255)&(R(in,jn+20)~=255)
    R(in,jn)=2; % Mot diem
  re nhanh end;
  esc=fals
e; else
  for i=in-
    1:in+1
    for j=jn-
    1:jn+1
      T(i,j)=255;
    end;
  end;
  if (In<I_nguong)&(R(in-
  20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-20)~=255)
  &(R(in,jn+20)~=255)
    label=tru
    e; for
    i=in-
    6:in+4
      for j=jn-4:jn+4
        if
          (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
          -2)|(R(i,j)==-3) R(i,j)=-2;
          label=fal
        se; end;
      end;
    end;
```

```
        if label
          R(in,jn)=1; % Mot diem
        ket thuc end;
        esc=fal
      se; end;
  en
d;
case
2
  if T(in-1,jn+1)==255
    label=true;
    for i=in-
      6:in+4
      for j=jn-
      4:jn+6
        if
          (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
          -2)|(R(i,j)==-3) R(i,j)=-2;
          label=fals
        e; end;
      end;
    end;
    if (label)&(R(in-20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-
      20)~=255)&(R(in,jn+20)~=255) R(in,jn)=2;
    end;
    esc=fals
    e;
  else
    for a=-
      1:1 for
      b=-1:1
        i=in-
        a+b;
        j=jn-
        a-b;
```

```
        T(i,j)=255;
      end;
    end;
    if (In<I_nguong)&(R(in-
    20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-20)~=255)
    &(R(in,jn+20)~=255)
      label=tru
      e; for
      i=in-
      6:in+4
        for j=jn-4:jn+6
          if
            (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
            -2)|(R(i,j)==-3) R(i,j)=-2;
            label=fal
          se; end;
        end;
      end;
      if label
        R(in,jn)
      =1; end;
      esc=false;
    en
  d;
  end;
case 3
  if
    T(in,jn+1)==25
    5 label=true;
    for i=in-
      4:in+4
      for j=jn-
      4:jn+6
        if
          (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
          -2)|(R(i,j)==-3) R(i,j)=-2;
          label=fals
        e; end;
      end;
    end;
```

```
    if (label)&(R(in-20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-
      20)~=255)&(R(in,jn+20)~=255) R(in,jn)=2;
    end;
    esc=fals
    e;
  else
    for i=in-
      1:in+1
      for j=jn-
      1:jn+1
        T(i,j)=255;
      end;
    end;
    if (In<I_nguong)&(R(in-
    20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-20)~=255)
    &(R(in,jn+20)~=255)
      label=tru
      e; for
      i=in-
      4:in+4
        for j=jn-4:jn+6
          if
            (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
            -2)|(R(i,j)==-3) R(i,j)=-2;
            label=fal
          se; end;
        end;
      end;
      if label
        R(in,jn)
      =1; end;
      esc=false;
    end;
```

```
  en
d;
case
4
  if
    T(in+1,jn+1)==255
    label=true;
    for i=in-
      4:in+6
      for j=jn-
      4:jn+6
        if
          (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
          -2)|(R(i,j)==-3) R(i,j)=-2;
          label=fals
        e; end;
      end;
    end;
    if (label)&(R(in-20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-
      20)~=255)&(R(in,jn+20)~=255) R(in,jn)=2;
    end;
    esc=fals
    e;
  else
    for a=-
      1:1 for
      b=-1:1
        i=in-
        a+b;
        j=jn-
        a-b;
        T(i,j)=255;
      end;
    end;
    if (In<I_nguong)&(R(in-
    20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-20)~=255)
    &(R(in,jn+20)~=255)
      label=tru
      e; for
```

```
        i=in-
        4:in+6
          for j=jn-4:jn+6
            if
              (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
              -2)|(R(i,j)==-3) R(i,j)=-2;
              label=fal
            se; end;
          end;
        end;
        if label
          R(in,jn)
        =1; end;
        esc=false;
      en
    d;
    end;
  case 5
    if
      T(in+1,jn)==25
      5 label=true;
      for i=in-
        4:in+6
        for j=jn-
        4:jn+4
          if
            (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
            -2)|(R(i,j)==-3) R(i,j)=-2;
            label=fals
          e; end;
        end;
      end;
      if (label)&(R(in-20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-
        20)~=255)&(R(in,jn+20)~=255) R(in,jn)=2;
      end;
      esc=fals
      e;
    else
```

```
      for i=in-
        1:in+1
        for j=jn-
        1:jn+1
          T(i,j)=255;
        end;
      end;
      if (In<I_nguong)&(R(in-
      20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-20)~=255)
      &(R(in,jn+20)~=255)
        label=tru
        e; for
        i=in-
        4:in+6
          for j=jn-4:jn+4
            if
              (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
              -2)|(R(i,j)==-3) R(i,j)=-2;
              label=fal
            se; end;
          end;
        end;
        if label
          R(in,jn)
        =1; end;
        esc=false;
      en
    d;
    end;
  case 6
    if T(in+1,jn-1)==255
      label=true;
      for i=in-
        4:in+6
        for j=jn-
        6:jn+4
          if
            (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
            -2)|(R(i,j)==-3) R(i,j)=-2;
            label=fals
```

```
        e; end;
      end;
    end;
    if (label)&(R(in-20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-
      20)~=255)&(R(in,jn+20)~=255) R(in,jn)=2;
    end;
    esc=fals
    e;
  else
    for a=-
      1:1 for
      b=-1:1
        i=in-
        a+b;
        j=jn-
        a-b;
        T(i,j)=255;
      end;
    end;
    if (In<I_nguong)&(R(in-
    20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-20)~=255)
    &(R(in,jn+20)~=255)
      label=tru
      e; for
      i=in-
      4:in+6
        for j=jn-6:jn+4
          if
            (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
            -2)|(R(i,j)==-3) R(i,j)=-2;
            label=fal
          se; end;
        end;
      end;
      if label
```

```
          R(in,jn)
        =1; end;
        esc=false;
      en
    d;
    end;
  case 7
    if T(in,jn-
      1)==255
      label=true;
      for i=in-
        4:in+4
        for j=jn-
        6:jn+4
          if
            (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
            -2)|(R(i,j)==-3) R(i,j)=-2;
            label=fals
          e; end;
        end;
      end;
      if (label)&(R(in-20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-
        20)~=255)&(R(in,jn+20)~=255) R(in,jn)=2;
      end;
      esc=fals
      e;
    else
      for i=in-
        1:in+1
        for j=jn-
        1:jn+1
          T(i,j)=255;
        end;
      end;
      if (In<I_nguong)&(R(in-
      20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-20)~=255)
      &(R(in,jn+20)~=255)
        label=tru
        e; for
        i=in-
```

```
        4:in+4
          for j=jn-6:jn+4
            if
              (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
              -2)|(R(i,j)==-3) R(i,j)=-2;
              label=fal
            se; end;
          end;
        end;
        if label
          R(in,jn)
        =1; end;
        esc=false;
      en
    d;
    end;
  case 8
    if T(in-1,jn+1)==255
      label=true;
      for i=in-
        6:in+4
        for j=jn-
        6:jn+4
          if
            (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
            -2)|(R(i,j)==-3) R(i,j)=-2;
            label=fals
          e; end;
        end;
      end;
      if (label)&(R(in-20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-
        20)~=255)&(R(in,jn+20)~=255) R(in,jn)=2;
      end;
```

```
      esc=false
    ; else
      for a=-
        1:1 for
        b=-1:1
          i=in-
          a+b;
          j=jn-
          a-b;
          T(i,j)=255;
        end;
      end;
      if (In<I_nguong)&(R(in-
      20,jn)~=255)&(R(in+20,jn)~=255)&(R(in,jn-20)~=255)
      &(R(in,jn+20)~=255)
        label=tru
        e; for
        i=in-
        6:in+4
          for j=jn-6:jn+4
            if
              (R(i,j)==1)|(R(i,j)==2)|(R(i,j)==
              -2)|(R(i,j)==-3) R(i,j)=-2;
              label=false
            ; end;
          end;
en
      end;
d;    if label
        R(in,jn)
        =1; end;
        esc=false
        ;
      en
    d;
    end;


%.....
function phit = tangentDir(I,it,jt,anpha)
% Ham xac dinh phuong tiep tuyen phit tai diem (it,jt)
```

```
I=255-
double(I);
A=0;
B=
0;
C=
0;
for h=it-(anpha-
  1)/2:it+(anpha-1)/2
  for k=jt-(anpha-
  1)/2:jt+(anpha-1)/2
    ahk=(-I(h-1,k-1)+I(h-1,k+1)+I(h+1,k+1)-I(h+1,k-1))/4;
    bhk=(-I(h-1,k-1)-I(h-1,k+1)+I(h+1,k+1)+I(h+1,k-1))/4;
    A=A+ahk^
    2;
    B=B+bhk^
    2;
    C=C+ahk*
    bhk;
  end;
end;
if C>0
  t1=1;
  t2=(B-A)/(2*C)-sqrt(((B-A)/(2*C))^2+1);
elseif C<0
  t1=1;
  t2=(B-A)/(2*C)+sqrt(((B-A)/(2*C))^2+1);
elseif
  A<=B
  t1=1;
  t2=0;
else
  t1=0;
```

```
   t2=1;
end;
if t1==0
   phit=
pi/2;
else
   phit=atan(
t2/t1); end;


%.....
function [In in jn] =
localmax(I,it,jt,phit,sigma)
% Ham xac dinh diem cua dai cuc bo (in,jn) theo phuong vuong goc voi
phit

In=0;
d=[1/23 2/23 5/23 7/23 5/23 2/23 1/23];
in=it;
jn=jt;
LM=[];
io=[];
jo=[];
for x=-sigma:sigma
   i=round(it+x*sin(phit+pi/2))
   ;
   j=round(jt+x*cos(phit+pi/2)
   );
   % Io la gia tri trung bình cua I(i,j) voi hai
   diem lan can cua no Io=(I(round(i-
   sin(phit)),round(j-cos(phit)))+I(i,j)+...
      I(round(i+sin(phit)),round(j+cos(phit))))/3;
   LM=[LM Io]; % LM la vetor chua cac gia tri trung
   binh tren doan sigma io=[io i];
   jo=[jo
j]; end;
LMn(1:2*sigma
+1)=0; for
x=4:2*sigma-2
   LMn(x)=LM(x-3)*d(1)+LM(x-2)*d(2)+LM(x-1)*d(3)+LM(x)*d(4)+...
        LM(x+1)*d(5)+LM(x+2)*d(6)+LM(x+3)*d
   (7);% Loc Gaussian if In<LMn(x);
      in=io(x);
      jn=jo(x
```

```
    );
    In=LMn
    (x);
  end;
end;
```

5. Hàm normalize.m

```
function [I_nor R] = normalize(I,N)
% Ham chuan hoa muc xam cua anh I, dong thoi tim vung co van tay
(region
% mask)
% I_nor: anh sau khi chuan hoa muc xam
% R: region mask

I=double
(I); [m
n]=size(I
);
R(1:m,1:n)=255;
for
  i=1:N:m-
  N for
  j=1:N:n-
  N
    I_sub=I(i:i+N-
    1,j:j+N-1); X=[];
```

```
    for x=1:N
      X=[X
    I_sub(x,:)]; end;
    M=mean(X);
    VAR=sqrt(var
    (X,1)); if
    VAR>6
      for
        x=1:N
        for
        y=1:N
          if I_sub(x,y)>M
            I_sub(x,y)=100+sqrt(100*(I_sub(x,
            y)-M)^2/VAR);
          else
            I_sub(x,y)=100-
          sqrt(100*(I_sub(x,y)-M)^2/VAR);
          end;
        end;
      end;
      I(i:i+N-1,j:j+N-1)=I_sub;
      R(i:i+N-1,j:j+N-1)=0;
    en
  d;
  end;
end;
I_nor=uint
8(I);
```

6. Hàm gabor_filter.m

```
function I_filted = gabor_filter(I,Wg,T,detax,detay)
% Ham loc anh I bang bo loc Gabor doi xung chan
% Wg: Kich thuoc cua bo loc Gabor
% T: Chu ky cua ham cos trong ham loc Gabor
% detax,detay: Cac he so deta cua ham loc Gabor
% I_filted: Anh sau khi loc
%I=normalize(I);

I=double
(I); [m
n]=size(I
```

```
);
for h=11:Wg:m-
  Wg-9 for
  k=11:Wg:n-
  Wg-9
    %------ Lay anh con tu anh chinh de loc --------
    p=0;
    for i=h-
      10:h+Wg+9
      p=p+1;
      q=0;
      for j=k-
        10:k+Wg+9
        q=q+1;
        I_sub(p,q)=I(i,
        j);
      en
    d;
    end;
    %--------- Doi ra mien tan so va loc ---------
    O=tangentDir(I_sub,(Wg+11)/2,(Wg+1
    1)/2,9); I_sub=fftshift(fft2(I_sub));
    G=fftshift(fft2(es_gabor(Wg+20,T,O,det
    ax,detay))); G=abs(G);
    I_sub=abs(ifft2(fftshift(I_sub.*G)));
    %------ Thay anh con da duoc loc tro lai anh chinh ------
    p=10;
    for i=h:h+Wg-1
```

```
        p=p+1
        ;
        q=10;
        for j=k:k+Wg-
          1 q=q+1;
          I(i,j)=I_sub(
          p,q);
        en
      d;
      end;
    end;
end;
I_filted=uint8(I);
%.....
function g =
es_gabor(Wg,T,O,detax,detay)
% Ham MTF (modulation transfer function) cua bo loc Gabor

for x=-(Wg-1)/2:(Wg-1)/2
  for          y=-(Wg-
    1)/2:(Wg-1)/2
    xO=x*cos(O)-
    y*sin(O);
    yO=x*sin(O)+y*
    cos(O);
    g(x+(Wg+1)/2,y+(Wg+1)/2)=exp(-
  ((xO/detax)^2+(yO/detay)^2)/2)*cos(2*pi*xO/T); end;
end;
%.....
function phit = tangentDir(I,it,jt,anpha)
% Ham xac dinh phuong tiep tuyen phit tai diem (it,jt)

A=
0;
B=
0;
C=
0;
for              h=it-(anpha-
  1)/2:it+(anpha-1)/2 for
```

```
  k=jt-(anpha-
  1)/2:jt+(anpha-1)/2
    ahk=(-I(h-1,k-1)+I(h-1,k+1)+I(h+1,k+1)-I(h+1,k-1))/4;
    bhk=(-I(h-1,k-1)-I(h-1,k+1)+I(h+1,k+1)+I(h+1,k-1))/4;
    A=A+ahk^
    2;
    B=B+bhk^
    2;
    C=C+ahk*
    bhk;
  end;
end;
if C>0
  t1=1;
  t2=(B-A)/(2*C)-sqrt(((B-A)/(2*C))^2+1);
elseif C<0
  t1=1;
  t2=(B-A)/(2*C)+sqrt(((B-A)/(2*C))^2+1);
elseif
  A<=B
  t1=1;
  t2=0;
else
  t1=0
  ;
  t2=1
  ;
end;
if t1==0
  phit=
pi/2;
else
  phit=atan(
t2/t1); end;
```

7. Hàm matching.m

```
function kq = matching(tem,db_tem,thres_P,thres_O)
% Ham doi sanh mau van tay "tem" (template) voi mot mau van tay
trong
%  database "db_tem" (database template). Mau phai co cau truc la mot
%  mang 2 hang (hang 1 chua cac gia tri bien do, hang 2 chua cac gia tri
%  goc cua cac diem dac trung) va N cot (N la so diem dac trung detect
%  duoc tren anh van tay). Trong do 2 cot dau tien la vi tri cua hai diem
%  delta (neu co).
% thres_po: nguong sai lech ve vi tri (position).
% thres_ph: nguong sai lech ve goc (orientation).
% kq: ket qua doi sanh - ty le phan tram su giong nhau

N_tem=length(tem(1,:));
N_dbtem=length(db_tem(1,:));

N=0; % So diem giong nhau

d_delta_right=sqrt((tem(1,1)-
db_tem(1,1))^2+(tem(2,1)-db_tem(2,1))^2);
d_delta_left =sqrt((tem(1,2)-
db_tem(1,2))^2+(tem(2,2)-db_tem(2,2))^2);

if d_delta_right<thres_P
  N=N+5; % Mot diem delta bang nam
lan diem minutiae end;
if d_delta_left<thres_P
   N=N+5; % Mot diem delta bang nam
lan diem minutiae end;

for
  i=3:N_tem
  dmin=50
  0; jmin=0;
  for j=3:N_dbtem
     d=sqrt((tem(1,i)-
     db_tem(1,j))^2+(tem(2,i)-
     db_tem(2,j))^2);    if dmin>d
        dmin
        =d;
        jmin
        =j;
```

```
    en
  d;
  end;
  if (dmin<thres_P)&(abs(tem(3,i)-
    db_tem(3,jmin))<thres_O) N=N+1; %
    Mot diem minutiae giong nhau
    db_tem(1,jmin)=0;
    db_tem(2,jmin
  )=0; end;
end;

kq=N/(N_dbtem+8

)*100;
```

8. Hàm feature_extracting.m

```
function [template core] = feature_extracting(I)
% Ham xay dung tap mau cho ham matching va ham sampling (vi tri cua cac
% diem dac trung co dang toa do decac)

[J R]=normalize(I(:,:,1),10);
O=orientation(I(:,:,1),R,9);
[core
delta]=singularity(R,O,4
5,25); J=(255-
double(J))/255;
J=sigmoid(J,25,0.4);
J=uint8((1-J)*255);
J=gabor_filter(J,21,6,1,2);
Rm1=minutiae(J,R,core);
%.....
[J
R]=normalize(I(:,:,2),10);
J=(255-
double(J))/255;
J=sigmoid(J,25,
0.4); J=uint8((1-
J)*255);
J=gabor_filter(J,21,6,1,2);
Rm2=minutiae(J,R,core);
%.....
[m n]=size(Rm1);
for
  i=1:m
  for
  j=1:n
    if
      (Rm1(i,j)==1)|(Rm1(i,j)=
      =2) label=false;
      for x=i-
        7:i+7
        for y=j-
        7:j+7
          if
            (Rm2(x,y)==1)|(Rm2(
```

```
                x,y)==2) label=true;
                Rm2(x,y)=0;
              end;
            end;
          end;
          if ~label
            Rm1(i,j)=0;
          en
        d;
        end;
    end;
end;
%.....
template=delta;
for
  i=1:m
  for
  j=1:n
    if
      (Rm1(i,j)==1)|(R
      m1(i,j)==2) mi1=i-
      core(1);
      mi2=j-
      core(2);i3=0;
      mi3=O(i,j)*180/pi;
      mi=[mi1;mi2;mi3];
      template=[templat
      e mi];
    en
  d;
  end;
end;
%.....
function y = sigmoid(x,a,b)
y=1./(1+exp(-a*(x-b)));
```

9. Hàm neural_template.m

```
function neural_tem = neural_template(tem)
% Xay dung mau cho mang neural

N1=
0;
N2=
0;
N3=
0;
N4=
0;
mi(1:3,1:4)=0;

m=length(tem
(1,:)); for
i=3:m
  if
    (tem(1,i)>=0)&(tem
    (2,i)>=0) N1=N1+1;
    mi(1,1)=mi(1,1)+tem(1,i);
    mi(2,1)=mi(2,1)+tem(2,i);
  elseif
    (tem(1,i)>=0)&(tem(2,i)<=
    0) N2=N2+1;
    mi(1,2)=mi(1,2)+tem(1,i);
    mi(2,2)=mi(2,2)+tem(2,i);
  elseif
    (tem(1,i)<=0)&(tem(2,i)<=
    0) N3=N3+1;
    mi(1,3)=mi(1,3)+tem(1,i);
    mi(2,3)=mi(2,3)+te
  m(2,i); else
    N4=N4+1;
    mi(1,4)=mi(1,4)+tem(1,i);
    mi(2,4)=mi(2,4)+te
  m(2,i); end;
end;
if N1~=0
  mi(:,1)=abs(mi(:,1
  )/N1);
```

```
   mi(3,1)=N1;
end;
if N2~=0
   mi(:,2)=abs(mi(:,2
   )/N2);
   mi(3,2)=N2;
end;
if N3~=0
   mi(:,3)=abs(mi(:,1
   )/N3);
   mi(3,3)=N3;
end;
if N4~=0
   mi(:,4)=abs(mi(:,4
   )/N4);
   mi(3,4)=N4;
end;
n_tem=[tem(1,1);tem(2,1);tem(1,2);tem(2,2);mi(:,1
);mi(:,2);mi(:,3);mi(:,4)];
```

10. Hàm training1.m

```
function W = training1(X,nuy,epsilon,D,n)
% Ham huan luyen mang neural perceptron don lop 16 ngo vao
% Ham tac dong la ham tuyen tinh
% X(N,K): ma tran cua cac vector mau ngo vao (K=1 mau, moi mau co
N=16 gia tri)
% nuy: hang so hoc
% epsilon: nguong huan luyen
% W: cac vector trong so duoc huan luyen

%-------------- Tao gia tri khoi dong -------------------------
W=[0.1 0.1 1 0.1 0.1 1 0.1 0.1 1 0.1 0.1 1 -0.1 -0.1 -0.1 -0.1];
%.....
epoch=0;
J=epsilon;
while (epoch<1000)&(J>=epsilon)
%----------Lan truyen thuan-------------------------
  net=W*X
  ; Y=(-
  1)^n*net;
%----------Lan truyen nguoc-------------------------
  W=W+(-1)^n*nuy*(D-Y)*X'; % Ham tac dong tuyen tinh
%.....
  .....
  Y=(-1)^n*W*X;
  J=((D-Y)^2)/2;
  epoch=epoc
h+1; end;
```

11. Hàm training2.m

```
function [V W] = training(X,H,nuy,epsilon)
% Ham huan luyen mang neural perceptron hai lop (mot lop an va mot
lop ra)
% Ham tac dong lop an la ham sigmoid, ham tac dong lop ra la ham
tuyen tinh
% X(N,K): ma tran cua cac vector mau ngo vao (K mau, moi mau co N
gia tri)
% nuy: hang so hoc
% epsilon: nguong huan luyen
% V,W: cac vector trong so duoc huan luyen cua lop an va lop ra

[N K]=size(X);
%-------------- Tao gia tri khoi dong -------------------------
V(1:H,1:N)=0.1/(N-
1); W(1:H)=1;
```

```
for
  i=1:H
  for
  j=1:N
    if j==i
      V(i,j)=-0.1;
    en
  d;
  end;
  if mod(i,2)==0
    % i chan
    W(i)=-1;
  end;
end;
%.....
epoch=0;
J=epsilon;
```

```
dnuy=0;
while
  (epoch<1000)&(J>=epsilon)
  nuy=nuy+dnuy;
  for k=1:K
  %----------Lan truyen thuan-------------------------
    for q=1:H
      netq=V(q,:)*X(:
      ,k);
      z(q)=sigmoid(
      netq);
    end;
  %----------Lan truyen nguoc-------------------------
    detaO=-W*z';   % Ham tac
    dong tuyen tinh
    W=W+nuy*detaO*z;
    for q=1:H
      netq=V(q,:)*
      X(:,k);
      detaHq=detaO*W(q)*5*exp(-0.25*netq)/(1+exp(-0.25*netq))^2;
      V(q,:)=V(q,:)+nuy*detaHq*X(:,k)';
    end;
  %.....
  end;
  for q=1:H
    netq=V(q,:)*X(:,k
    );
    z(q)=sigmoid(ne
    tq);
  end;
  y=W*z'
  Jo=J;
  J=(y^2
  )/2;
  dJ=J-
  Jo;
  if dJ<0
    dnuy=0.01*n
  uy; elseif dJ>10
    dnuy=-
  0.1*nuy;
```

```
  else
    dnuy=0;
  end;
  epoch=epoc
h+1; end;

%.....
function y = sigmoid(x)
y=20/(1+exp(-0.25*x));
```

12. Hàm thuthap1.m

```
function [FAR FRR] =
thuthap1(threshold) False_match=0;
False_non_match=0;
load('E:\neural
matching\database\NDB.mat'); %load
NDB for n=1:NDB
  filename=['E:\neural
  matching\template\matching\tem'
  num2str(n)]; load(filename,'matching_tem');
  % load matching_tem match_min=threshold;
  j_min=0
  ; for
  j=1:ND
  B
    filename=['E:\neural matching\database\1\db' num2str(j)];
```

```
    load(filename); %
    load W
    match=W*matching
    _tem(:,i); if
    match<match_min
      match_min=ma
      tch; j_min=j;
    en
  d;
  end;
  if j_min~=0
    if j_min~=n
      False_match=False_match
      +1;
    en
  d;
  else
    False_non_match=False_non_
  match+1; end;
end;
FAR=False_match/(
NDB);
FRR=False_non_match/(NDB);
```

13. Hàm thuthap2.m

```
function [FAR FRR] =
thuthap2(threshold) False_match=0;
False_non_match=0;
load('E:\neural
matching\database\NDB.mat'); %load
NDB for n=1:NDB
  filename=['E:\neural
  matching\template\matching\tem'
  num2str(n)]; load(filename,'matching_tem');
  % load matching_tem match_min=threshold;
  j_min=0
  ; for
  j=1:ND
```

```
  B
      filename=['E:\neural
      matching\database\2\db' num2str(j)];
      load(filename); % load V,W
      for q=1:16
        netq=V(q,:,1)*matching_t
        em(:,i);
        z(q)=20/(1+exp(-
      0.25*netq)); end;
      match=W*z';
      if
        match<match_mi
        n
        match_min=matc
        h; j_min=j;
      end;
  end;
  if j_min~=0
    if j_min~=n
      False_match=False_match
      +1;
    en
  d;
  else
    False_non_match=False_non_
  match+1; end;
end;
FAR=False_match/(
NDB);
FRR=False_non_match/(NDB);
```

TÀI LIỆU THAM KHẢO

[1] Dario Maio and Davide Maltoni,"Direct Gray-Scale Minutiae Detection In Fingerprints", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 1, January 1997.

[2] Li Xin Wang, "A course in Fuzzy system and control".

[3] D.Maltoni, D.Maio, A.K.Jain, S.Prabhakar, "Singularity and Core Detection" Extract from "Handbook of Fingerprint Recognition", Springer, New York, 2003.

[4] D.Maltoni, D.Maio, A.K.Jain, S.Prabhakar, "Minutiae-based Methods" Extract from "Handbook of Fingerprint Recognition", Springer, New York, 2003.

[5] Anil Jain, Lin Hong, Ruud Bolle,"On_line Fingerprint Verification", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 4, April 1997.

[6] Nguyễn Kim Sách, "Xử lý ảnh và video số", NXB Khoa học và kỹ thuật, 1997.

[7] Nguyễn Quang Thi, "Áp dụng mạng neuron trong kỹ thuật xử lý ảnh", Luận án cao học, 2000.

[8] Jianwei Yang, Lifeng Liu, Tianzi Jiang, Yong fan, "A modified Gabor filter design method for fingerprint image enhancement", Pattern Recognition Letters 24 (2003) 1805-1817.

[9] Jinwei Gu, Jie Zhou, Chunyu Yang, "Fingerprint Recognition by Combining Global Structure and Local Cues", IEEE Transactions on Image Processing, Vol. 15, No. 7, July 2006.

[10] Dingrui Wan, Jie Zhou, "Fingerprint Recognition Using Model-Based Density Map", IEEE Transactions on Image Processing, Vol. 15, No. 6, June 2006.

[11] Karthik Nandakumar, Anil K. Jain, "Local Correlation-based Fingerprint Matching", To Appear in Proceedings of ICVGIP,

Kolkata, December 2004.

[12] Anil Jain, Sharathcha Pankanti, "Fingerprint Classification and Matching".

[13] Anil Jain, Lin Hong, Yifei Wan, ”Fingerprint Image Enhancement: Algorithm and Performance Evaluation ”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, 1998.

[14] Anil Jain, Lin Hong, Sharathcha Pankanti, Ruud Bolle,”Fingerprint Enhancement”.

[15] D.Maltoni, D.Maio, A.K.Jain, S.Prabhakar, ”Fingerprint Scanners and their Features” Extract from “Handbook of Fingerprint Recognition”, Springer, New York, 2003.

[16] “Evaluation of Fingerprint Recognition Technologies – BioFinger”, www.bsi.bund.dc