The use of Neural Networks in Fingerprint Analysis

Csomai András, Babeş–Bolyai Univesity, Intelligent Sistems

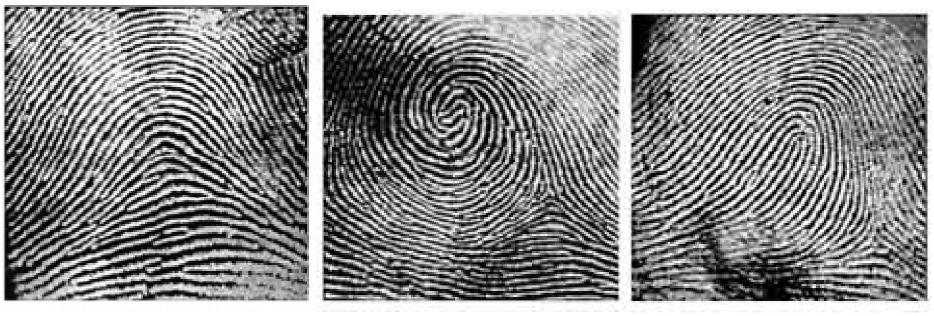
Introduction

Fingerprint Analysis is one of the most reliable biometric personal identification methods. It's been already used in criminology, for almost a hundred years. Nowadays its use has been extended to commercial tasks, such as personal identification for on-line banking, security systems, etc.

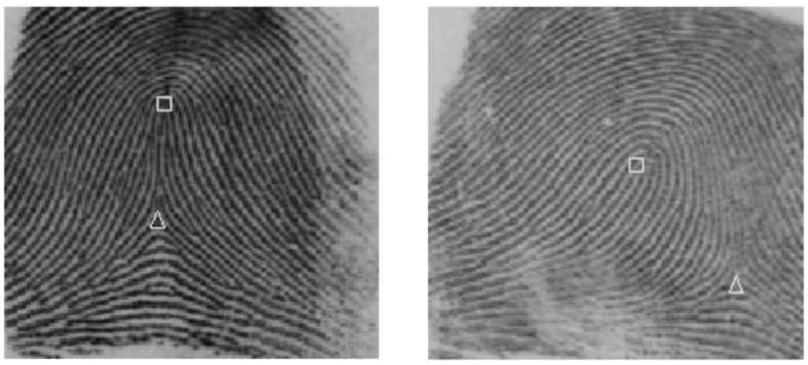
The large fingerprint databases brought the need for a fast, reliable automatised fingerprint identification system. The large database size implicates the need for dividing the database in certain classes, to ease the search. But the classification of a fingerprint image is difficult even for a human expert, therefore intelligent classification methods were introduced. In the followings we are trying to give an overview of the classification method, the possible intelligent approaches and a couple other possible uses of neural networks in the recognition process.

Fingerprint classes

As we already mentioned, the purpose of the classification is to divide the database, according to certain rules. The first complex classification system was introduced by Sir Edward Richard Henry, in 1901. This is still in use in many countries. There are other approaches also like the NCIC, which is a lot simpler. Most of the automatized fingerprint identification systems are based on the Henry system. The base of this system consists of five classes: the arch, whorl, left and right loops (see below). The arch and tented arch classes are often merged, because the difference between them is very slight (the presence of a separate core and delta point).


The basic classes: arch, whorl and loop.

The main property considered for classification is the shape of the ridges and furrows, more precisely the pattern they form. The other properties that could be considered are the ridge count, and the core and delta points. The ridge count (where available) is the number of ridges crossing the imaginary line between the core and delta point.


Delta and core points in a tented arch and left loop

In the first step, for a fingerprint classification system it would be enough, reliably distinguish the four basic classes already mentioned.

Feature generation

The most important issue in the intelligent classification is the choice of the feature representation. In the followings I will present two advanced techniques, and a quite simple one, developed and evaluated by me, which showed acceptable results.

Hierarchical graph.

In the first approach, a hierarchical graph is constructed, based on the regions of the fingerprint with the same ridge orientation. The process is the following (see in 1.):

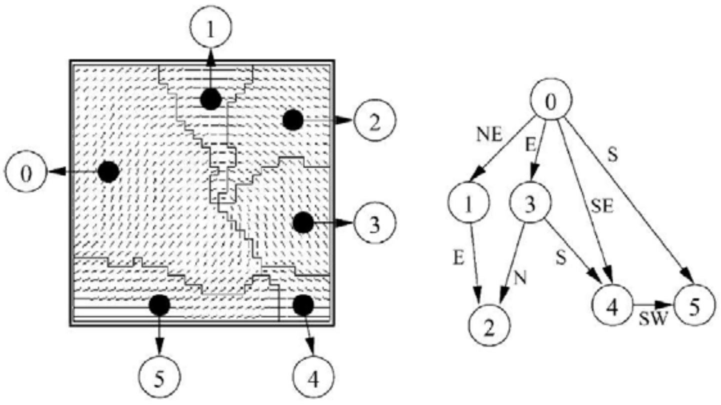
- Computation of the directional image of the fingerprint. This directional image is a 28 x 30 matrix. Each matrix element represents the ridge orientation within a given block of the input image.
- Segmentation of the directional image into regions containing ridges with similar orientations.
- Representation of the segmented image by a directed positional 4 acyclic graph (DPAG).The representation of fingerprint structure is completed by characterizing each graph node with a numerical feature vector containing local characteristics of the related image region (area, average directional value, etc.) and some geometrical and spectral relations with respect to adjacent regions (relative positions, differences among directional average values, etc.).
- Classification of the above DPAG by an RNN made up of two multilayer perceptrons (MLPs) neural nets. This neural network model is briefly described below.

The direction image can be computed in several different ways. The easiest would be the use of the local gradients. The segmentation can also be performed in several different ways, in (1.), there is a possible algorithm.

The construction of the relational graph, is more complex:

- The image region containing the “core” point is selected as the starting region for the graph construction, that is, a graph node associated to such a region is initially created;
- The regions that are adjacent to such a core region are then evaluated for the creation of new nodes. Nodes are created for adjacent regions which are located in one of the following spatial positions with respect to the core region: North, North East, East, South East, South, South West, West, and North West;
- The above process is repeated for each of the new nodes until all the regions of the segmented images have been considered;

- The graph nodes created by the above algorithm are finally characterized by a numerical feature vector containing local characteristics of the related image regions (area, average directional value, etc.) and some geometrical and spectral relations with respect to adjacent regions (relative positions,differences among directional average values, etc.).



Segmented fingerprint orientation image and relational graph.

The graph is then being fed to a Recursive Neural Network, which is capable of learning the classification, when the classification is dependent on a hierarchical data structure.

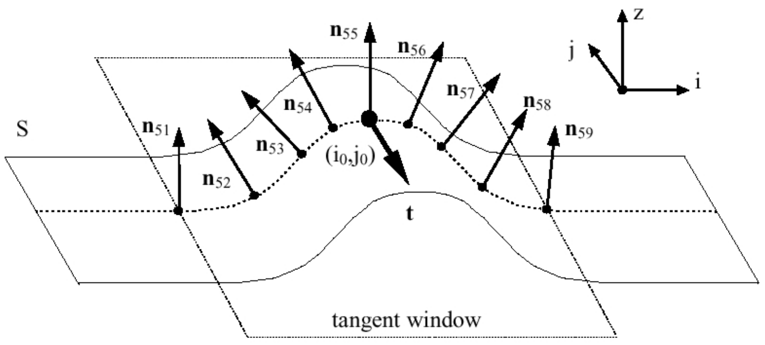
Orientation vector and MLP

The method implemented by me is a lot simpler. The features are nothing but the dimensionally reduced orientation image. This can be done by simply dividing the orientation image into blocks, assign to every block the average (dominant) orientation and translating the so obtained matrix into a vector (feature vector). Another possibility is to use an advanced dimensionality reduction method.

The obtained feature vector is then being fed to a MultiLayer Perceptron.

The calculation of the orientation field is quite simple (see 2.):

Let (i_0,j_0) be the pixel of the image **I** where the tangent direction φ_0 must be computed. Let the *tangent window* be a squared window centered in (i_0,j_0) with side length α pixels. For each pixel (i_h,j_k) belonging to the tangent window, a vector n_{hk} orthogonal to the surface $z=I(i,j)$ is defined. The tangent vector in each pixel (i_h,j_k) (when defined) lies on the *ij*-plane and is orthogonal to the corresponding vector n_{hk}. The average tangent vector *t*, which represents the required direction φ_0, is the unit vector lying on the ij-plane which is the "most orthogonal" to all the vectors n_{hk} computed.



Tangent window, tangent vector

The average tangent vector computation takes place as follows:

Let $a_1=T(i_{h+1},j_{k+1})$, $a_2=T(i_{h-1},j_{k+1})$, $a_3=T(i_{h-1},j_{k-1})$ and $a_4=T(i_{h+1},j_{k-1})$ be the pixels belonging to the *2 x 2* pixel neighborhood of each tangent-window pixel *h,k*. For each neighborhood the normal vector n_{hk} can be computed via least-squares minimization:

$n_{hk} = [a_{hk}, b_{hk}, 1]$, where

$$a_{hk} = \frac{-a_1 + a_2 + a_3 - a_4}{4}, \qquad b_{hk} = \frac{-a_1 - a_2 + a_3 + a_4}{4}$$

Let $v_{hk}=(a_{hk},b_{hk})$, $h=1,..\alpha$, $k=1,..\alpha$ be the vectors obtained by removing the *z* component from the corresponding normal vectors n_{hk} and let $t=(t_1,t_2)$. Formally it is a least-squares minimization:

$$\min \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} \left|\langle v_{hk}, t\rangle\right|^2 \quad \text{, subject to } \| t \| = 1$$

Neglecting the mathematical details:

$$A = \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} (a_{hk})^2 \quad , \quad B = \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} (b_{hk})^2 \quad , \quad C = \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} a_{hk} b_{hk}$$

$$t = \begin{cases} \left[1,\ \frac{B-A}{2C} - \operatorname{sgn}(C)\sqrt{\left(\frac{B-A}{2C}\right)^2 + 1} \right] \text{if C} \neq 0 \\ [1,0] \ \text{ ha C} = 0,\ \text{A} \leq \text{B} \\ [0,1] \ \text{ ha C} = 0,\ \text{A} > \text{B} \end{cases}$$

So the desired direction can be computed as:

$$\varphi_0 = \begin{cases} \arctan\left(\frac{t_2}{t_1}\right) \ \text{if } \mathrm{t}_1 \neq 0 \\ \frac{\pi}{2} \qquad \text{otherwise} \end{cases}$$

The so obtained orientation image can be used for feature generation.

We defined the length of the feature vector in 400. This means, that the orientation image is divided into 20 x 20 blocks, as we already explained.

The training dataset, we used, was generated with the Fingerprint Creator software, which is able to generate different fingerprints of the basic four classes. Of course this is not exactly like the original fingerprint (ex. there is no quality problem etc.), but for the evaluation is acceptable. The ideal would be a database with at least 400 real fingerprints per class. Disadvantage of the problem is that it is not rotation invariant, although the nature of the neural network accepts a certain level of deviation.

Fingercode

The algorithm separates the number of ridges present in four directions (0 degree, 45 degree, 90 degree, and 135 degree) by filtering the central part of a fingerprint with a bank of Gabor filters.

The general form of the even symmetric Gabor filter is

$$h(x, y : \phi, f) = \exp\left\{-\frac{1}{2}\left[\frac{(x\cos\phi)^2}{\delta_x^2} + \frac{(y\sin\phi)^2}{\delta_y^2}\right]\right\}\cos(2\pi f x\cos\phi)$$

where ϕ is the orientation, f the estimated frequency (can be empirical), δ_x , δ_y are constants. Practically, it can be computed by

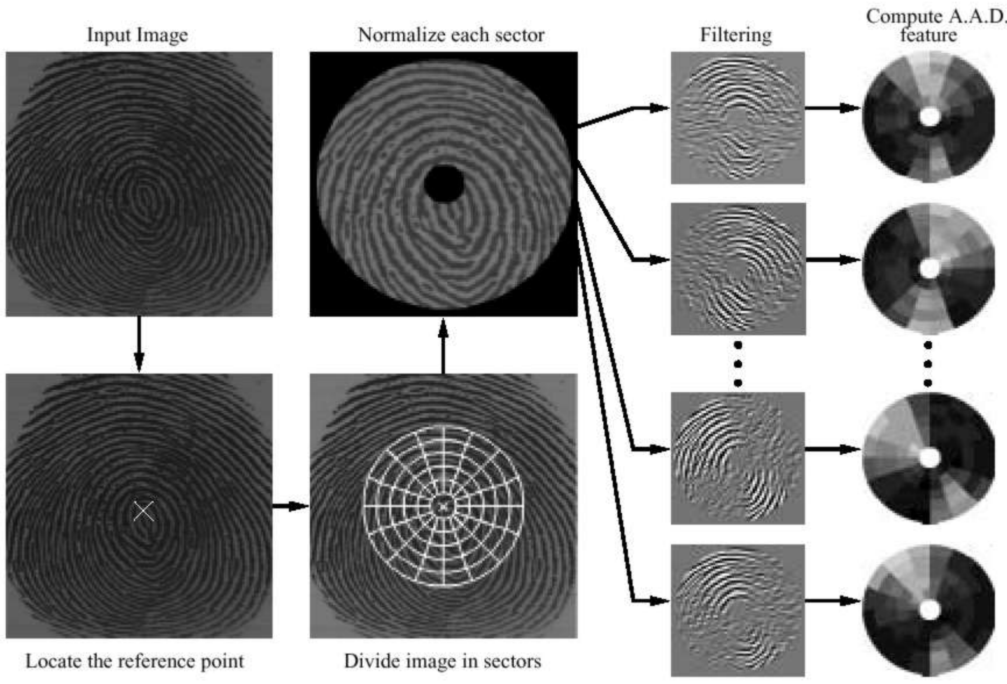
$$E(i, j) = \sum_{u=-w_g/2}^{w_g/2}\sum_{v=-w_g/2}^{w_g/2} h(u, v : O(i, j), F(i, j))G(i-u, j-v)$$

where O is the rientation image , F is the frequency image, G is the normalized input fingerprint image

This method was proposed in (3.), and is widely used. The four main steps in this feature extraction algorithm are:

- determine a reference point for the fingerprint image,
- tessellate the region around the reference point,
- filter the region of interest in eight different directions using a bank of Gabor filters, and
- compute the average absolute deviation from the mean (AAD) of gray values in individual sectors in filtered images to define the feature vector or the FingerCode.

The actual feature vector is constructed from the 80 AAD-s obtained from the individual sectors, and the ridge count information. These feature vectors are then being fed to a MLP. The system diagram is showed below.



System diagram for the fingercode generation.

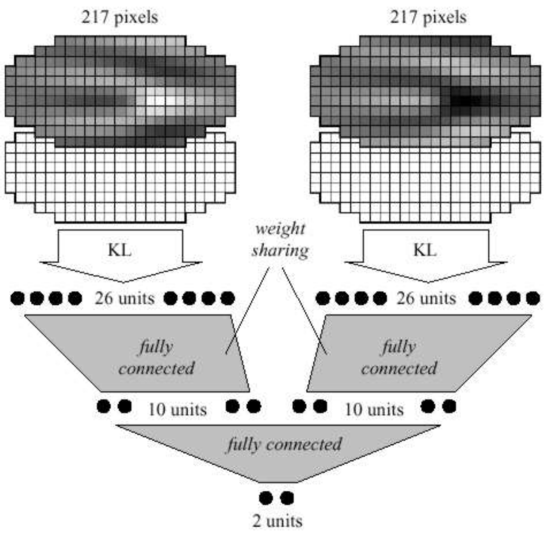
Other uses

The classification is not the only use of neural networks in fingerprint analysis. The two main other uses are the minutia filtering and the region of interest determination.

Minutia filtering

The minutia is a local feature of the fingerprint, more exactly is a feature of the ridge structure. There are two types of minutiae, the termination and the bifurcation. The termination is an end of a ridge. The actual fingerprint matching is performed trough comparing the minutia sets in two fingerprint images. The greatest problem in minutiae detection is the presence of false minutiae. A cut on a finger could result many false terminations, the bad quality of the image, or dirt stuck between ridges could result false bifurcation. The easiest method of classifying the minutiae trough neural networks, is to simply take a certain neighborhood of the minutia, and fed it to an MLP. This of course is not the best, but acceptable (it is also proposed by D. Dummitrescu Hariton Costin in 4.)

Another proposal is being made by Maio&Maltoni in 5. Which proposes another Neural Network, a three layer atypical partial weight sharing network, which's scheme is shown belov.

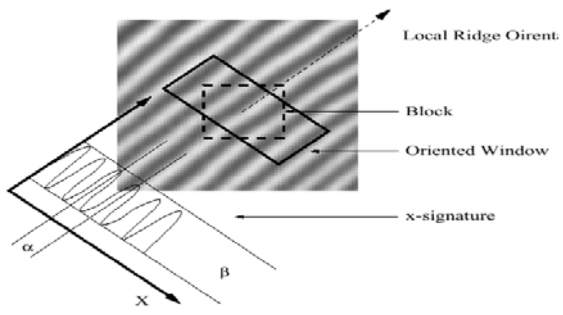


Atypical three layer partial weight sharing network

The first input is the true neighborhood, the second is the inverted image.

Region of interest determination

The region of interest determination is nothing but filtering out the noisy parts of a fingerprint image. Those regions, where a quasy sigmoid surface is present, are more likely correct. There are different approaches. In the first approach, the image is segmented, each segment is corrected according to the local dominant orientation. Then these segments are used as features. This is slow and unefficient. A method is proposed in 7. which constructs a so called x-signature for every block. The calculation of the x-signature is shown below:



As we might see, in regions, where no minutiae are present, the signature forms a sinusoidal wave. The classification of pixels into recoverable and unrecoverable categories can be performed based on the assessment of the shape of the wave. In their algorithm three features are used to characterize the sinusoidal shaped wave: amplitude, frequency and variance. Several fingerprint images were labeled with recoverable and unrecoverable regions, and assigned with the corresponding characteristics. They fed the so obtained patterns to a squared error clustering algorithm and identified six clusters Four of these clusters correspond to recoverable regions and the remaining two correspond to unrecoverable regions The six prototypes corresponding to cluster centers were used in an one nearest neighbor NN classifier to classify each *w x w* block in an input fingerprint image into a recoverable or an unrecoverable block.

Bibliography

1. Yuan Yao, Gian Luca Marsialis, Massimiliano Pontil, Paolo Frasconi: Combining flat and structured representations for fingerprint classification with recursive neural networks and support vector machines
2. Dario Maio,Davide Maltoni: Direct gray-scale minutiae detection in fingerprints
3. Anil K. Jain, Salil Prabhakar Lin Hong Sharath Pankanti: FingerCode: A Filterbank for Fingerprint Representation and Matching
4. Dumitrescu D. Hariton Costin : Retele Neuronale
5. Dario Maio Davide Maltoni :Neural Network Based Minutiae Filtering in Fingerprints
6. Salil Prabhakar, Anil K. Jain, Jianguo Wang, Sharath Pankanti, Ruud Bolle. *Minutia Verification and Classification for Fingerprint Matching*
7. Lin Hong ,Anil Jain ,Sharathcha Pankanti ,Ruud Bolle. *Fingerprint Enhancement.* IEEE WACV, Sarasota, Florida, 1996
8. Anil Jain, Sharath Pankanti: Fingerprint Classification and Matching
9. John M. Trenkle: Region of interest detection for fingerprint classification
10. www. Fingerprint Classification
11. wwww.bcso.