American Journal of Computational and Applied Mathematics 2012, 2(2): 1-9
DOI: 10.5923/j.ajcam.20120202.01

Two Different Regimes of Fingerprint Identification – a Comparison

Terje Kristensen

Departement of Computer Engineering, Bergen University College, Nygårdsgaten 112, N-5020 Bergen, Norway

Abstract In this work a hybrid technique for classification of fingerprint identification has been developed to decrease the matching time. For classification, a Support Vector Machine (SVM) and a Multi-Layered Perceptron (MLP) network are described and used. Automatic Fingerprint Identification Systems (AFIS) are widely used today, and it is therefore necessary to find a classification system that is less time-consuming. The fingerprint patterns generated are based on minutiae extraction from a thinned fingerprint image. The given fingerprint database is decomposed into four different subclasses. Two different classification regimes are used to train the systems to do correct classification. The classification rate has been estimated to about 87.0 % and 88.8% of unseen fingerprints for SVM and MLP classification respectively. The classification rate of both systems is only differing marginally.A benchmark test has been done for both systems. The matching time is estimated to decrease with a factor of about 3.7 compared to a brute force approach.

Keywords Automatic Fingerprint Identification System, Support Vector Machine, Multi-Layered Perceptron, Back-propagation algorithm, Benchmark Test

1. Introduction

The first Automatic Fingerprint Identification System (AFIS) was developed in 1991, and since then, there has been an enormous progress in the field. Due to the ever-growing capabilities of computers and great achievements in research, the recognition rate has improved significantly. There is nevertheless a huge amount of work still to be done.

The current work in this field concentrates on reducing the computation time for feature extraction and matching. Embedded fingerprint systems supporting instant identification or verification are increasingly used, and the computation time for these processes is thus an important research field. One way to decrease this time is to divide the fingerprint database into different subclasses based on specific properties, such that only a part of the fingerprints needs to be considered for matching.

The uniqueness of fingerprints has been widely tested, and two identical fingerprints have still not been found (Pankanti et al., 2002). However, current fingerprint identification systems do not use all the discriminating information present in a fingerprint, and the probability of finding two identical fingerprints using the systems therefore increases. A lot of work is being done today to decide which information in fingerprints should be used to keep the uniqueness. In addition to this work, the current work in this field concentrates on reducing the computation time for feature extraction and matching.

Various methods have been used for classification. In this paper, both an Artificial Neural Network (ANN) and a Support Vector Machine (SVM) approach have been used for classification. Both networks are given the same feature vector as input, based on Computation of the Poincare index. This method was first proposed by Jain, Prabhakar and Hong in 1999 (Jain et al., 1999).

Among all the biometric techniques fingerprint identification is today the most widely used biometric identification form. It has been used in numerous applications. The problem is to develop algorithms which are robust to noise in the fingerprints and are able to deliver accuracy in real time.

Different matching techniques can be used for fingerprint identification. In this paper a matching technique based on minutiae extraction from the thinned fingerprint image is used. Such a technique is based on that the ridges of a fingerprint carries certain kind of features. If one considers the ridges as lines, there will be several occurrences in the fingerprint where the line ends or splits into two parts. These features are referred as minutiae.All the minutiae in a fingerprint form a minutiae structure that can be used to distinguish one fingerprint from another. In this paper we are concentrating on bifurcation and ridge endings minutiae. By examining a minutiae structure extracted from the fingerprint one may matched it against structures in other fingerprints.

* Corresponding author:
tkr@hib.no (Terje Kristensen)
Published online at http://journal.sapub.org/ajcam

Fingerprint matching algorithms vary greatly in terms of false positive and false negative errors.They may also vary with respect to features such as image rotation invariance and independence from a reference point given as the center or the core of the fingerprint pattern.

2. Background

In an automatic fingerprint identification system (AFIS), the fingerprint databases can be huge, often tens of thousands of fingerprints. If you were supposed to check for similarity between the query fingerprint and every other fingerprint in the database, it can take an enormous amount of time.

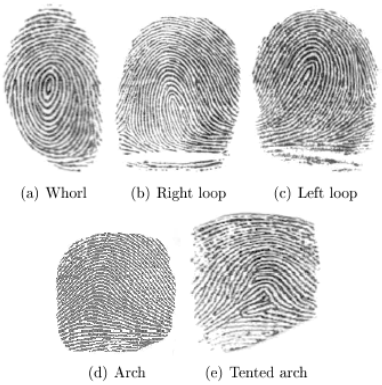

Figure 1. Five major fingerprint classes

With this in mind, fingerprint classification is an important step that has to be implemented in every fingerprint identification system. The classification criteria most widely used are the modification or extension of the standard Galton-Henrys classification system (Henry, 1990). Here, fingerprints are divided into 5 subclasses: whorl (W), right loop (RL), left loop (LL), arch (A) and tented arch (TA).

2.1. Singularities

Three singularities can be found in a fingerprint to distinguish between the classes. These three singularities are the *loop*, the *delta* and the *whorl*. A whorl fingerprint contains one or more ridges that make a complete 360-degree path around the centre of the fingerprint. Two loops (or one whorl) and two deltas are present. The deltas are placed under the whorl, one at the right and one at the left side. A loop fingerprint has one or more ridges that enter from one side, curves back and exits at the same side as they entered. In a left and right loop, the ridges enter from the left side and the right side, respectively. A loop and a delta singularity are present, with the delta under the loop, at the left in a right loop fingerprint and at the right in a left loop fingerprint. An arch fingerprint has ridges that enter from one side, rises to a small bump and exits at the opposite side.

When no singularities are present, this will make the classification of the class rather difficult. A tented arch fingerprint contains one or more ridges that enter from one side, loops in a high curvature and exits at the opposite site. When one loop and one delta singularity are present, the delta is typically placed right under the loop.

It is possible to further subdivide each class into more subclasses, but this is hardly of any practical importance. In poor quality fingerprints it is really difficult to even classify it to the five main classes, and further classification would probably increase the rejection rate. In addition, the complexity in the end renders the classification incapable of improving the identification time anymore.

2.2. Different Methods

Many fingerprint classification methods have been proposed in literature. In general, these methods can be categorised into five approaches (Maltoni, et al., 2005):

- rulebased
- syntacticbased
- structure based
- statistical based
- neural network based

In this work we have concentrated on the statisticalbased and neural network based approach and see how the performance of an AFIS comes out, based on a SVM and an ANN network as classifiers.

The current work in this field concentrates on reducing the computation time for feature extraction and matching. Embedded fingerprint systems supporting instant identification or verification are increasingly used, and the computation time for these processes is thus an important research field. One way to decrease this time is to divide the fingerprint database into different subclasses based on specific properties, such that only a part of the fingerprints needs to be considered for matching.

3. Extracting Features

3.1. Extracting Classification Features

The SVM network needs an input vector to be able to classify the fingerprints. This vector can be made by extracting features of the fingerprint, and then represent these features in a suitable way. We have chosen to create this vector based on a technique proposed by (Maltoni et al., 2005). Here, the authors present a feature vector called FingerCode, which is a vector consisting of 640 feature values.

First, a reference point in the fingerprint is be found. We set the core of the fingerprint as the reference point. Then, the image is filtered in eight different directions using different Gabor filters, each enhancing ridges oriented in different angles. Each of these eight images are divided into 80 sectors according to specific rules. The standard deviation of each sector is finally calculated, and these values represent the feature values. The total number of feature values is 640 (8 x 80), and a vector containing these values is used as input vector to both the SVM and the ANN network.

3.2. Reference Point Detection

We have chosen to use the core point as the reference point. This core point is defined as the most northern loop singularity in a fingerprint. A loop singularity can be detected by a method based on the Poincare index proposed in (Jain et al., 1999). Let G be a vector field and C be a curve immersed in G. Then, the Poincare index is defined as the total rotation of the vectors of G along C. Here, G is the vector field associated with an orientation image of the fingerprint. The curve C is a closed path defined as an ordered sequence of the neighbour elements dk of position (i,j) in the orientation image. Then, the Poincare index at position (i,j) is defined as the sum of the orientation differences between adjacent elements of C:

$$PG,C = \sum_{K=1}^{7} angle(d_k, d((k+1) \bmod 8)) \quad (1)$$

where dk is the neighbouring elements as shown in Figure 3.

$$P_{G,C} = \begin{cases} 0 \text{ deg. if (i,j) does not belong to any SR} \\ 360 \text{ deg. if (i,j) belongs to a whorl type SR} \\ 180 \text{ deg. if (i,j) belongs to a loop type SR} \\ -180 \text{ deg. if (i,j) belongs to a delta type SR} \end{cases} \quad (2)$$

As mentioned, the core point is the most northernloop. We assume that the fingerprints are captured with the finger in an approximately normal position, but tolerate a rotation of up to 45 degrees either clockwise or counter clockwise. The core point is used as reference point in the extraction of classification features.

Figure 2. A loop and a delta singularity in a right loop fingerprint

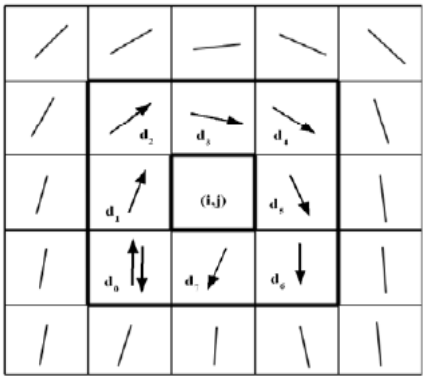

Figure 3. Computation of the Poincare index in the eight-neighbourhood of pixel (i, j)

3.3. Gabor Filtering

After the reference point is detected, the image is filtered using eight different Gabor filters. The image needs to be divided into 80 sectors, as illustrated in Figure 4. Here, the reference point is marked with a cross. Note that the innermost band is not divided into sectors, as it contains very few pixels, and the standard deviation will then become very unreliable. Before filtering, each of the 80 sectors has to be normalized, setting the mean and variance to desired values.

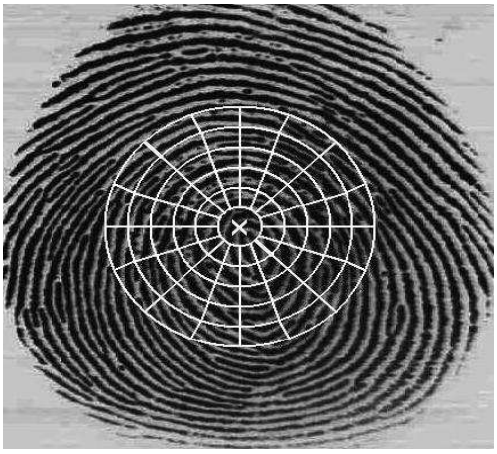

Figure 4. Computation of the Poincare index in the eight-neighbourhood of pixel (i, j)

Each sector is normalized locally, so the mean and variance have to be calculated for each sector. The desired mean and variance value for each sector are both set to 100, as recommended by the authors in (Delima, Yen, 2003).

Let I(i, j) denote the gray level at pixel (i, j) in a fingerprint image of size m x n and let (i_c, j_c) denote the reference point or core point. The region of interest is defined by a collection of sectors Sp, where the sector Sp is computed in terms of parameters (r,θ) defined by (Burges, 1998):

$$Sp = \{i,j) | b(T_p + 1) \le r \le b(T_p + 2)\} \quad (3)$$
$$1 \le i \le n,\ 1 \le j \le m,$$
$$\theta_p \le \theta < \theta_{p+1},$$

where

$$Tp = p \text{ div } k, \quad (4)$$
$$\theta_p = (p \bmod k)(2\Pi/k) \quad (5)$$
$$r = \sqrt{(i-i_c)^2 + (j-j_c)^2} \quad (6)$$
$$\theta = \tan^{-1}((j-j_c)/(i-i_c)) \quad (7)$$

b is the width of each band and k is the number of sectors considered in each band. The Gabor filters use eight fixed angle values; θε {0 degrees, 22.5 degrees, 45 degrees, … 157.5 degrees}. The frequency is fixed, to limit computation, and set as the average ridge frequency in the region of interest. The region of interest is here the union of all the sectors defined above. The Gabor filtering results in eight images that each enhance the ridges in a specific direction and also remove noise, thereby emphasizing the relevant information.

3.4. The Feature Vector

The feature vector is, as earlier mentioned, a vector consisting of 80 values for each of the eight Gabor filtered images. In each of these images, a section of the fingerprint containing ridges that are parallel to the corresponding filter

direction exhibits a higher variation. A section containing ridges that are not parallel to the corresponding filter tends to be smoothed by the filter, which results in a lower variation. The spatial distribution of the variations in the different sectors of the component images can thereby be a good characterisation of the global ridge structure. With this in mind, the feature vector is defined as a vector containing the standard deviation of all 80 sectors in the filtered image for all angles θ (Jain, Farrokhnia, 1999):

Let $F_{p\theta}(i,j)$ be the θ-direction filtered image for section Sp. For p ε {0, 1..,79} andθε {0 degrees, 22.5 degrees.., 157.5 degrees}, the feature value is the standard deviation Vp_{θ}, defined as:

$$V_{p\theta} = \sqrt{\frac{1}{K_p}\sum_{Kp}(F_{p\theta}(i,j) - P_{p\theta})} \quad (8)$$

where K_p is the number of pixels in S_p and $P_{p\theta}$ is the mean value of pixels in S_p in image $F_{p\theta}$ (i, j). Now, we have a 640-dimensional feature vector that can be used as input vector to both the ANN and the SVM network.

3.5. Scaling

To improve the classification rate, all feature vectors are scaled before training. The main advantage of scaling is to avoid attributes with big values which dominate those with small values.

Another advantage is to avoid numerical difficulties during the calculation. For instance, since the kernel values in Support Vector Machines depend on the inner products of the feature vectors, large attribute values might cause numerical problems. Scaling also makes the training run faster, and decreases the chance of getting stuck in local optima. The feature vectors are linearly scaled to the range [-1,+1]. Each value j in a feature vector i is scaled individually by:

$$S_{i,j} = -1 + 2* \frac{FV(i,j) - \min i}{\max j - \min j} \quad (9)$$

where FV(i,j) and S_{ij} are the feature value and scaled feature value at position j for feature vector i, respectively. minj and maxj are the minimum and maximum feature value at position j for all feature vectors, respectively. The table must appear inside the designated margins or it may span the two columns.

3.6. Minutiae Extraction

The ridges of a fingerprint carry certain features. Ridges can end up in ends or splitting of two parts. Such features are referred as minutiae. All the minutiae form a minutiae structure that is used to distinguish one fingerprint from another. To extract minutiae from a fingerprint this can most easily be done by using binarisation technique. Binarisation is the process of converting a gray-scale image into a binary black and white image. The Gabor filtered image is binarised by introducing a threshold value. A thinning process makes the ridges one pixel wide.

When the Gabor filtered image is of poor quality some problems may occur which may lead to spurious minutiae. Such minutiae have to be removed from the image. When the fingerprint image is thinned, the minutiae can more easily be extracted from the fingerprint. Only the bifurcation and ridge ending minutiae illustrated in Figure 6 will be handled here.

a) b)

Figure 5. a) A binarised Gabor filtered image. b) A corresponding thinned image

To extract the minutiae, we use a method proposed by Hung (1993). By looking at the thinned image where the ridge pixels are eight-connected, we are able to extract ridge endings and bifurcations. Examples of these minutiae points are shown in Figure 6. For each ridge pixel, a 3×3 local neighbourhood is examined. A concept called Crossing Numbers, as proposed by Arcelli and Baja (1985), is used. The Crossing Number of a local neighbourhood is defined as:

$$CN = 0.5 \sum_{i=1}^{8} |P_i - P_{i+1}|, P_9 = P_1 \quad (10)$$

where $P_i$is a pixel in the local neigbourhood ofP.

The local neighbourhood is scanned in a counterclockwise direction. Thus, the Crossing Number represents half the sum of differences between pair of adjacent pixels in the eight-connected neighbourhood. Now, each pixel in the image can be labeled according to the properties listed in Table 1. By use of this table one is able to extract the different kind of minutiae from the fingerprint image.

Table 1. Properties of Crossing Numbers (CN)

CN	Property
0	Isolated point
1	Ridge ending point
2	Continuous ridge point
3	Bifurcation point
4	Crossing point

From Table 1 we notice that the centre of a local pixel with a local neighbourhood that leads to a Crossing Number of 1, leads to a ridge labelled a ridge ending point. A Crossing point of 3 implies that the center pixel is labeled a bifurcation point.

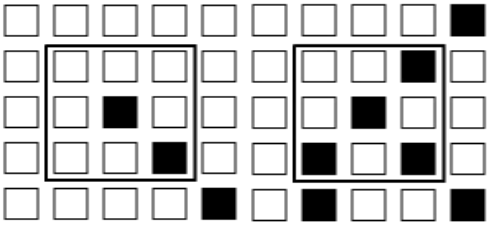

Figure 6. Examples of ridge an ending point and a bifurcation point

4. SVM Classification

Support Vector Machines is a computationally efficient learning technique that is now being widely used in pattern recognition and classification problems (Burges, 1998). This approach has been derived from some of the ideas of the statistical learning theory regarding controlling the generalization abilities of a learning machine (Vapnik, 1999, 1998).

In this approach the machine learns an optimum hyper plane that classifies the given pattern. By use of kernel functions, the input feature space may be transformed into a higher dimensional space where the optimum hyper plane can be learnt. By changing the kernel functions many different learning models can be established which makes SVM a very flexible formalism.

4.1. The SVM Classifier

The basic idea of an SVM classifier is illustrated in Figure 7. This figure shows the simplest case in which the data vectors (marked by 'x's and 'o's) can be separated by a hyper plane. In such a case there may exist many separating hyper planes. Among them, the SVM classifier seeks the separating hyper plane that produces the largest separation margin.

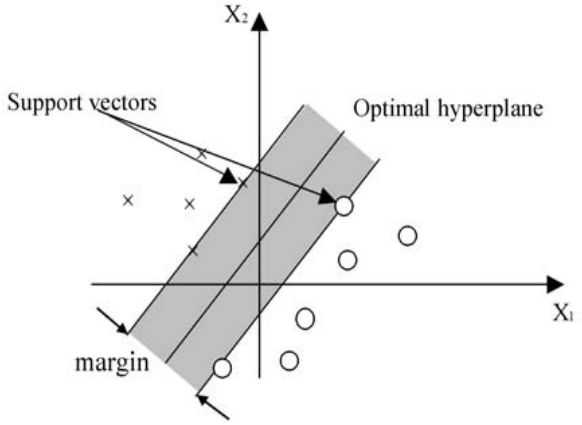

Figure 7. A Support Vector Machine classification defined by a linear hyper plane that maximizes the separating margins between the classes

In the more general case in which the data points are not linearly separable in the input space, a non-linear transformation is used to map the data vectors into a high- dimensional space (called feature space) prior to applying the linear maximum margin classifier. To avoid the potential pitfall of over-fitting in this higher dimensional space, SVM uses a kernel function in which the non-linear mapping is implicitly embedded. A function qualifies as a kernel function if it satisfies the Mercer's condition (Vapnik, 1998).

By use of a kernel function, the discriminant function in a SVM classifier has the following form

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \, K(\mathbf{x}_i, \mathbf{x}) + \alpha_0 \qquad (11)$$

where K(-,-) is the kernel function, $\mathbf{x_i}$ are the support vectors determined from the training data, y_i is the class indicator e.g. +1 and -1 for a two class problem associated with each $\mathbf{x_i}$, Nis the number of supporting vectorsdetermined during training.

Support vectors are elements of the training set that lie either exactly on or inside the decision boundaries of the classifier. In essence, they consist of those training examples that are most difficult to classify. The SVM classifier uses these borderline examples to define its decision boundary between the two classes.

4.2. SVM Kernel Functions

The kernel function plays a central role of implicitly mapping the input vectors into a high dimensional feature space, in which linear separability is achieved. The most commonly used kernel functions are the polynomial kernel given by:

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^p \qquad (12)$$

where $p > 0$ is a constant, and the Gaussian radial basis function (RBF) kernel given by

$$K(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|/2\sigma^2) \qquad (13)$$

where $\sigma > 0$ is a constant that defines the kernel width. Both of these kernels satisfy the Mercer condition mentioned earlier.

4.3. SVM Kernel

The fingerprint database that is used for training and testing for both SVM and MLP network is given in table 2.

Table 2. The number of training and testing instances for each fingerprint

	Training	Testing
LL	127	63
RL	123	59
TA	10	6
W	83	41
Total	343	169

The feature vector created is used as an input pattern to the classifier. The SVM network needs to be trained well to classify unknown patterns correctly.

We have manually labelled the fingerprint images according to the Henry classification, to create a training set. Each classifier is trained using the feature vector extracted from these labeled images. We have then tested the SVM classifier by presenting it for unknown fingerprints to see if SVM is able to classify these unknown patterns correctly.

Fingerprint images with no core point or a core point too lose to an edge or segmented area, are not used for training, due to false feature values. If such an image occurs in the classification stage, the image cannot be classified and must be matched against every other fingerprint in the final matching stage.

5. MLPClassification

A Multi-Layered Perceptron (MLP) network generally contains three or more layers of processing units (Haykin, 2009). Figure 8 shows the topology of a network containing three layers. The bottom layer constitutes the input layer

which obtains its input from the environment. The middle layer contains hidden units. The hidden layer has de ability to solve non-linear separable problems.

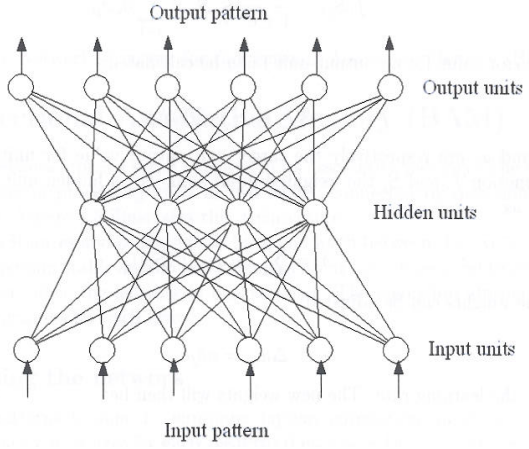

Figure 8. A MLP network consisting of three layers

The top layer constitutes the output layer which gives the produced results from the network. Every layer is here fully connected with the nearest layer, i.e. every unit in the first layer is connected to every unit in the second layer and every unit in the second layer is connected to every unit in the output layer. This is not a requirement, but most MLP networks are constructed like this.

5.1. Training of MLP

In the learning phase we present patterns to the network, and the weights are adjusted so that the produced outputs from the output nodes are equal to the target. In fact, we want the network to find a single set of weights that will satisfy all the (input, output) pairs presented to it. Neurons in one layer receive signals from neurons in the layer directly below and send signals to neurons in the layer directly above. Connections between neurons in the same layer are not allowed. Except for the input layer nodes, the network input to each node is the weighted sum of outputs of the nodes in the previous layer.

Each node is activated in accordance with the input to the node and the activation of the node. Then, the difference between the calculated output and the target output is calculated. The weights between the output layer, hidden layers and the input layer are adjusted by using this error function. The output of a node is calculated using the sigmoid function f(x) given by:

$$f(x) = 1 /(1+e^{-x}) \quad (14)$$

The weighted sum S_j is inserted into the sigmoid function, and the result is the output value from unit j:

$$f(S_j) = 1 /(1 + e^{-Sj}) \quad (15)$$

where

$$S_j = \sum w_{ji}\ a_i \quad (12)$$

The error value of an output unit j can be calculated by:

$$\delta_j = (t_j - a_j) f'(S_j) \quad (16)$$

where t_j and a_j are the target and output value of unit j, respectively. f' is the derivative of the sigmoid function f, and S_j the weighted input sum. The derivative of the sigmoid function is given by:

$$f'(S_j) = f(S_j) (1- f(S_j)) \quad (17)$$

We notice that it is expressed by the function itself which makes the derivative and the error more rapidly to calculate.

For a hidden node, the error value is calculated as:

$$\delta_j = \sum \delta_k w_{kj} (f'(S_j)) \quad (18)$$

From the formula we see that the error of a processing unit in the hidden layer is computed by the upper layer. Finally, the weights can be adjusted by:

$$\Delta w_{ji} = \alpha \delta_j \alpha_i \quad (19)$$

where α is the learning rate. The new weights will then be:

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t) \quad (20)$$

where t represents a processing step.

Usually, the momentum parameter β is introduced in the MLP network. It has been shown that the use of this additional parameter can be helpful in speeding up the convergence and avoiding local minima (Nigrin, 1993). Equation (20) can now be written as:

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t) + \beta \Delta w_{ji}(t) \quad (21)$$

where β is the momentum and $\Delta w_{ji}(t)$ is the weight change from the previous processing step.

6. Experiments and Results

6.1. SVM Experiments

The SVM is trained using a Radial Basis Function kernel $K(\mathbf{x_i},\mathbf{x_j})$, given by equation 13.

$$K(\mathbf{x_i},\mathbf{x_j}) = e^{-\gamma \| x_j - x_j \|}, \gamma > 0 \quad (22)$$

Before training, two parameters need to be found, Cand γ. C is the penalty parameter of the error term and γ is a kernel parameter. By cross-validation, the best C- and γ-values are found to be 8.0 and 0.001953125, respectively.

Arch images have no singularities present, and a reference point of these fingerprints is thereby not possible by our methods. However, loops constitute about 65 percent of the total fingerprint patterns, whorls about 30percent, and arches and tented arches together account for the other 5 percent (Karu, Jain, 1996). This decreases the problem of not being able to classify arches.

Table 3. Classification rate for each class using a SVM network

	Correct	Wrong	Percent Correct
LL	59	4	93.7
RL	55	4	93.2
TA	2	4	33.3
W	31	10	75.6
Total	147	22	87.0

We see that the SVM classifier is able to classify the loops very well (93.7% and 93.2%), but does not classify the whorls and tented arches at the same performance rate.

We have used a *one-vs-one* approach to a multi-class problem. This splits our classification problem into six separate problems that in the end are combined into one final set of support vectors. In total this set, which is used for classification of the unknown fingerprints, contains 232 support vectors. The classification results using SVM as a

classifier are shown in Table 3.

6.2. MLP Experiments

The MLP network was also trained by a database according to the numbers specified in table 1. The implementation of the MLP network is based on a network containing 50 hidden nodes. To classify the output from the network we used 4 output units, one for each classification. If a feature vector is classified as a left loop, the correct output is set to the vector {1, 0, 0, 0} and a right loop to {0, 1, 0, 0}. When deciding the parameters of the MLP network, we need to take into consideration the classification performance of unseen patterns, not only the training patterns.

Overtraining of an MLP network may be a problem, and the number of iterations and learning rate should be adjusted so that such a situation does not occur. We found that the best results were achieved by setting the learning rate to 0.1 and the momentum to 0.9.

Table 4 below shows the classification rate on the testing cases of the MLP network of the different fingerprint classes.

Table 4. Classification rate for each class using a MLP network

	Correct	Wrong	Percent Correct
LL	59	4	93.7
RL	55	4	93.2
TA	2	4	33.3
W	34	7	82.9
Total	150	19	88.8

The MLP network was trained with 5000 iterations. By using fewer iterations one may cause the network not to be properly trained, and a higher number would imply the network to be a little over-trained and then not able to classify unseen fingerprints so well.

In the experiments the javANN (java Artificial Neural Network) software package created by the company Pattern Solutions AS in Norway (Kristensen, 1997, 2007)] has been used for training and classification. All the experiments were performed on an AMD Athlon - 2.8 GHz machine with 1024 MB of RAM.

6.3. Benchmarking

A benchmark test was carried out on the total set of fingerprints to measure the average matching time of a fingerprint identification using different subclasses. Matching of two fingerprints (scaling, rotation, translation) takes about 0.268 seconds using the implementation environment defined above.

The fingerprint database consisted of 343 fingerprints which mean a total matching time against the entire database of 1 minute and 32 seconds without classification. Using a classification stage against the entire database the average matching time becomes 23 seconds. This time may vary as a tented arch fingerprint takes, for instance, less time since the database contains a smaller amount of fingerprints belonging to this class.

The classification time of a fingerprint is less than two seconds. Using a classification stage the matching thereby decrease the matching time by a factor of 92/(23+2) = 3.7.

7. Discussion

7.1. Classification by SVM

From table 3 we see that SVM failed to classify most of the tented arch fingerprints. However, this may be a result of too few training instances belonging to the tented arch class. SVM also performed better classification of left loops and right loops compared to the classification of whorls. This may be caused by the limited region-of-interest used to calculate the feature vector, causing many whorls to be wrongly classified as loops.

There are two main classes of whorls, classic whorl and double loop. The double loop causes often a classification problem. This is because the region-of-interest centered in the core looks quite similar in double loops and normal loops. Thereby, they can easily be misclassified as loops. A solution could be to increase the region-of-interest, but this would also increase the rejection rate, as more sectors would be outside the fingerprint area or even outside the entire image.

The experiments have shown that a SVM network is able to do a correct classification with a rate of 87.0% on a four-class classification problem.

In (Karu, Jain, 1996), a classification algorithm based on the number of cores and deltas, and their relative positions, is presented. The authors achieved a correct classification rate of 85.4% on a five-class classification task.

In (Cappelli, Lumini, Maio, Maltoni, 1999) one has partitioned the directional image into connected regions according to their fingerprint topology, thus giving a synthetic representation which can be exploited as basis for classification. This method achieved a correct classification rate of 92.1% on a four-class classification task. We see that our SVM classifier is not able to match these results at the moment.

However, training with a larger fingerprint database than we had available would probably also increase the performance rate of our classification, as the SVM network then is capable to better distinguish important differences between the fingerprint classes.

7.2. Classification by MLP

From table 4 we can notice that MLP failed in classifying most of the tented arch fingerprints, but again this may be a result of too few training instances belonging to the tented arch class. The MLP network also performed better in classification of left loops and right loops than classification of whorls, the same results as we obtained for SVM. And again this may be caused by the limited region-of-interest used to calculate the feature vector, causing many whorls to be wrongly classified as loops such as in the SVM case.

The experiments show us that a MLP network is able to classify correct with a classification rate of 88.8 % on a four-class classification problem. This is a slightly better result than obtained by using SVM as a classifier.

However, compared to results obtained in (Jain, Prabhakar, Hong, 1999) this is not so impressive. The authors have used an approach similar to the one used in this paper, with the FingerCode as feature vector and an equivalent feed-forward Multi-Layer Perceptron (MLP) network as the classifier. By using a MLP network one was able to achieve a correct classification rate of 86.4% on a five-class classification task and 92.1% on a four-class classification task. But, again the difference in performance between our approach and the approach used by these authors may be a result of that the training database is too small.

Another aspect is that our system is based on a matching process of fingerprint identification based on minutia matching where the minutiae have been extracted from the thinned image of a fingerprint. The two approaches are therefore not quite comparable.

8. Conclusions

To decrease the identification time of an AFIS, it is necessary to classify the fingerprints into different subclasses, such that a query fingerprint does not have to be tested against every fingerprint in the database. To solve this problem, we have implemented a classification stage in the AFIS by using either a SVM or a MLP network as a classifier.

A MLP classifier is able to classify different unseen fingerprints with a performance rate of approximately 89.0%. In addition, by introducing such a classification stage one is also able to reduce the average matching time of a fingerprint with a factor of about 3.7.

The main objection by the method used so far is that the number of training samples is too small compared to the number of features in the FingerCode vector. We believe that this is the main reason why we have not achieved so good results compared tothe results in the literature. However, by training the MLP with an extended training database we believe that the total performance rate will improve, and be comparable or better to the results achieved in the literature.

Compared to SVM a MLP network is able to do a slightly better classification (Kristensen, 2010). However, we also know that a SVM classifier becomes better when the dimension of the input space becomes higher. It is therefore difficult to predict which type of network is going to be used as the final classifier in a future developed AFIS.

REFERENCES

[1] Arcelli, C., Baja, G., 1985. A width independent fast thinning algorithm. IEEE Transactions on Pattrn Analysis and Machine Intelligence, 4(7).463-474

[2] Burges, C.J., 1998. A Tutorial on Support vector Machines for Pattern Recognition. *The book Knowledge Discovery and Data Mining 2*

[3] Cappelli, R., Lumini, A., Maio, D., Maltoni, D., 1999. Fingerprint Classification by Directional Image Partitioning *In IEEE Trans. Pattern Anal. Mach. Intell., vol. 21, nr.5, pp.402-421. IEEE Computer Society*

[4] DeLima, P.G., Yen, G.G., 2003. Multiple model fault tolerant control using globalized dual heuristic programming. *In IEEE Proceedings International Symposium on Intelligent Control, Houston, TX,pp. 523–528*

[5] Henry, E.R., 1990. Classification and Uses of Finger prints, *The book Pattern Recognition, George Routledge and sons: London*

[6] Jain, A.K., Prabhakar, S., Hong, L., 1999. A Multichannel Approach to Fingerprint Classification. *In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, pp. 348–359*

[7] Jain, A.K., Farrokhnia,F., 1999. Unsupervised texture segmentation using Gabor filters, *The book Pattern Recognition, vol. 24, nr. 12, pp.1167-1186. Elsevier Science Inc*

[8] Jain, A.K., Prabhakar, S., Hong, L., Pankanti, S., 1999. FingerCode: A Filterbank for Fingerprint Representation and Matching. *In Proc. IEEE Conference on CVPR, Colorado vol.2, pp. 187-195*

[9] Haykin, S. *Neural Networks and Learning Machines*. Third Edition. Pearson 2009

[10] Hung, D., 1993. Enhancement and feature purification of fingerprint images. *Pattern Recognition, 26(111):1661 – 1667*

[11] Kawagoe, M., Tojo A., 1984. Fingerprintpattern classification. *In the book Pattern Recognition, vol. 17(3), pp. 295-303*

[12] Karu, K., Jain, A.K., 1996. Fingerprint Classification. *The book Pattern Recognition, vol.29, nr. 3, pp. 389-404*

[13] Maltoni,D. Maio,D, Jain,A.J., Prabhakar,S., 2005. *The handbook of Fingerprint Recognition, Springer*

[14] Kristensen, T., 2010. Fingerprint Identification – a Support Vector Machine Apprach. *In proceedings of 2nd International Conference on Agents and Artificial Intelligence, ICAART 2010, Valencia, Spain*

[15] Kristensen,T., 2007. javANN:*Java Artificial Neural Networks. Pattern Solutions AS.*http://www.patternsolutions.no

[16] Kristensen, T. Neural Networks, Fuzzy Logic and Genetic Algorithms. *Cappelen Academic Publisher 1997 (in Norwegian)*

[17] Nigrin,A., 1993. Neural Networks for Pattern Recognition. *MIT Press, Cambridge, MA, USA*

[18] Pankanti, S., Prabhakar, S., Jain, A., 2002. On the Individuality of fingerprints. *In IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (8), pp. 1010-1025*

[19] Sarle, W., 1997. Neural network FAQ. *Periodic posting to*

the Usenet newsgroup com.ai.neural-nets

[20] Vapnik, V.N:, 1999. An overview of statistical learning theory. *In IEEE transactions on Neural Networks, 10, September, 1999*

[21] Vapnik, V.N., 1998. Statistical learning Theory. *Wiley, New York*