A Combination Fingerprint Classifier

Andrew Senior

IBM T.J.Watson Research Center,
P.O. Box 704, Yorktown Heights,
NY 10598-0218, USA.
E-mail: `aws@watson.ibm.com`

Abstract

Fingerprint classification is an important indexing method for any large scale fingerprint recognition system or database, as a method for reducing the number of fingerprints that need to be searched when looking for a matching print. Fingerprints are generally classified into broad categories based on global characteristics. This paper describes novel methods of classification using hidden Markov models (HMMs) and decision trees to recognize the ridge structure of the print, without needing to detect singular points. The methods are compared and combined with a standard fingerprint classification algorithm and results for the combination are presented using a standard database of fingerprint images. The paper also describes a method for achieving any level of accuracy required of the system, by sacrificing the efficiency of the classifier. The accuracy of the combination classifier is shown to be higher than that of two state-of-the-art systems tested under the same conditions.

Keywords

Henry fingerprint classification, hidden Markov models, decision trees, neural networks, NIST database.

1 Introduction

The classification of fingerprints has long been an important part of any fingerprinting system. A partition of fingerprints into groups of broadly similar patterns allows filing and retrieval of large databases of fingerprints for quick reference. Currently interest in fingerprint classification is stimulated by its use in automatic fingerprint identification systems (AFIS). In an AFIS, the goal is to find a match for a probe fingerprint in the database of enrolled prints, possibly numbering many millions. Classification is used in an AFIS to reduce the size of the search space to fingerprints of the same class before attempting exact matching.

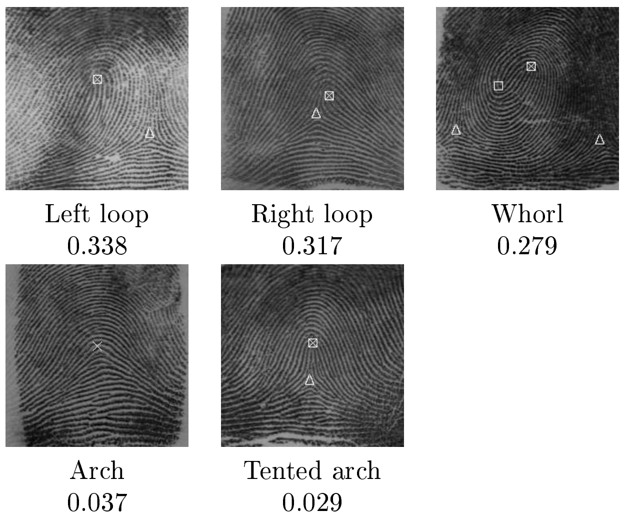

Figure 1: Examples of five fingerprint categories, marked with core and delta points, with their frequencies of occurrence.

The most widely-used system of fingerprint classification is the Henry system and its variants [1]. Examples from five of the main classes of the Henry system are shown in figure 1.

Many previous authors have developed automated systems to classify fingerprints, using a wide variety of techniques. Cappelli *et al.* [2] provide a recent review of a number of methods that have been used, and section 8 in this paper presents results from other authors.

Most automatic systems use the Henry classes, and these are important for existing AFIS databases and systems which require compatibility with human classifications, either because of legacy data or because some manual intervention is necessary in the process, requiring the use of human-interpretable classes. A variety of approaches to classification has been tried, the most fundamental being a syntactic analysis of the relative positions and number of core and delta points in the print. The core and delta points, shown in figure 1, are the singular points in the flow of the ridges. Finding these points in the image is a difficult image processing task, particularly with poor quality images, but if found reliably, the classification is simple [3, 4]. Maio and Maltoni [5] use a structural analysis of the direction of ridges in the print, without needing to find core and delta points. Blue *et al.* and Candela *et al.* [6, 7] use the core location to centre their representation scheme which is based on a principal components analysis (PCA) of ridge directions, and they then use a variety of classifiers. Halici and Ongun [8] similarly use PCA-projected, core-centred ridge directions, but classified with a self-organizing map; and Jain *et al.* [9] also use the core for translation invariance, using a Gabor filter representation and a k-Nearest Neighbour classifier.

For situations where there is no need to use existing classes, some researchers have developed systems which rely on machine-generated classes, or dispense with classes all together, and use 'continuous' classification [2, 10, 8, 11]. Here the criterion is not adherence to the Henry classes, but merely consistency among classifications of different prints from the same finger. Fingerprints are represented by points in a feature space on which some distance measure is defined. Test fingerprints are matched against all those in the database falling within some radius of the test print. By increasing the radius, classification can be made arbitrarily accurate, reducing errors by increasing the size of the search space and hence search time. Continuous classification holds the prospect of circumventing the difficult and restrictive Henry classification problem, and has produced the best results of recent years, but has disadvantages, besides the uninterpretability mentioned above. Using Henry classes, the portions of the database that must be searched are always the same, allowing for rigid segmentation of the database and *a priori* design of the search strategy. A continuous system presents an entirely different subset of the database for every matching operation, complicating and slowing the matching.

This paper describes a combination of novel approaches to fingerprint classification using the Henry system. The system described has been designed to operate on both rolled and 'dab' fingerprints, where some of the structural information used by other systems (such as the delta position) may not be available in the fingerprint image. The system described has been tested on the NIST Special database 4 [12] database of fingerprint images and results are presented. Further, a method of measuring the efficiency of a classification algorithm is described, allowing a principled comparison of this algorithm with previous published works. Finally a method for achieving arbitrary accuracy is described, allowing the Henry classifier to be used with the flexibility of continuous classifiers. This method trades off accuracy against classifier efficiency allowing an imperfect classifier to be used in a real-world system, while retaining all the advantages of a traditional Henry system.

The approach taken here is of a combination of classifiers each using different features and with different errors on test data. Two novel classifiers are described, using two-dimensional hidden Markov models (HMMs) and decision trees. In addition to showing that these classifiers perform well on the classification problem, and without the need for core/delta information, this paper shows that the combination of classifiers provides a way forward for the improvement of fingerprint classification, in the same way as recent improvements in isolated handwritten character recognition performance have been largely brought about not by better classifiers but by combinations of different classifiers. The classifiers are tested in isolation and in combination with the Probabilistic Neural Network classifier and pseudoridge tracer from the PCASYS system described by Candela *et al.* [7]. The experiments are all performed on discrete, Henry classification, but the system could be extended to continuous classification or classification with unsupervised clustering, using such techniques as unsupervised K-means HMM clustering [13].

The following sections describe the HMM classifier (previously described in [14]), the decision tree classifier, and PCASYS classifiers. In section 5, classification based upon the outputs of these classifiers is then described as well as combining the classifiers to improve accuracy. Section 6 describes a measure of efficiency of the classifier, and shows how arbitrary efficiency can be achieved. Section 7 presents results for the classifiers and their combinations, and results are compared with previously published results in section 8.

2 Classification by hidden Markov model

Hidden Markov models are a form of stochastic finite state automaton well suited to pattern recognition and successfully applied to speech recognition [15, 16] and other problems. They are appropriate to the problem posed here because of their ability to classify patterns based on a large quantity of features, whose number is variable and which have certain types of underlying structure, especially if that structure results in stationarity of the feature distributions over some spatial or temporal period. Such structure is found in fingerprints, where ridge orientations, spacings and curvatures are, for the most part, only slowly varying across the print. In a fingerprint the basic class information can be inferred from syntactic analysis of singular points, but can also be seen in the general pattern of the ridges — the way a non-expert human would classify prints. The HMM is able to statistically model the different structures of the ridge patterns by accumulations of evidence across the whole print, without relying on singular point extraction.

2.1 Ridge extraction

The system deals with fingerprint images stored as arrays of grey levels and obtained with a scanner or camera device — either from an inked fingerprint on paper, or as a 'live-scan' directly from the finger. For much of the work in this paper, the NIST-4 [12] database of rolled fingerprint images has been used, since this provides a large number (4000) of fingerprints with associated class labels. In addition part of the NIST-9 database has been used.

The features provided to the recognizer are based on the characteristics of the intersections of ridges with a set of fiducial lines that are lain across the fingerprint image. To find the ridge locations, a number of image processing techniques are used [17], summarized as follows:

1. initial segmentation; The PCASYS algorithm for extracting a central fingerprint region from a full rolled print is used on prints from the NIST-9 database. NIST-4 is already segmented at this level.
2. smoothing;
3. finding the predominant direction in each of an array of blocks covering the image;
4. segmenting the image into the area of the print (*foreground*) and the unwanted background, based on the strength of directionality found in each block;
5. applying directional filters to highlight the ridges and detect pixels that are parts of ridges;
6. thinning the ridge image so that each ridge is left represented by an eight-connected, one-pixel-wide line termed the *skeleton*.

2.2 Feature extraction

Given the skeleton image of the ridges, parallel fiducial lines are lain across the image at an angle ϕ, as shown in figure 2, and each one followed in turn. For

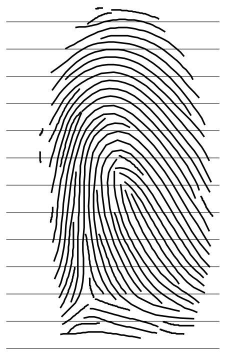
Figure 2: A sample fingerprint showing horizontal fiducial lines. ($\psi = 0$)

each intersection of a fiducial line with a ridge, a feature is generated. Each feature consists of a number of measurements, chosen to characterise the ridge behaviour and its development at the intersection point:

1. the distance since the last intersection;
2. the angle of intersection;
3. the change in angle since the last intersection;
4. the curvature of the ridge at the intersection.

The angle features (2) can be seen to contain similar information to the coarse direction field calculated in the preprocessing stages of this system and used by other systems as the feature set for classification [7]. However, this representation allows a higher resolution representation of the fingerprints, and allows more information to be represented (*e.g.* ridge spacing and curvature). Further measurements could also be taken at each point.

The measurements of each feature are termed a frame and the frames, R_{i_k} for the ith fiducial line are collectively termed a row, R_i, whose ordering is preserved. For each orientation ϕ of fiducial lines, a separate representation $\mathcal{R}^\phi = \{R_i, \forall i\}$ of the print is obtained. In this research, only horizontal and vertical lines have been used, giving features $\mathcal{R}^h$ and $\mathcal{R}^v$ respectively, but other angles may allow further information to be captured.

2.3 Hidden Markov models

Typically HMMs are one-dimensional structures suitable for analyzing temporal data. Here, the data are two dimensional, but the process of feature extraction can also be described as a one dimensional array of one-dimensional row processes. Thus we can apply a 'two dimensional hidden Markov model', similar to that of Agazzi *et al.* [18] which consists of a nesting of row models within whole-print models as shown in figure 3.

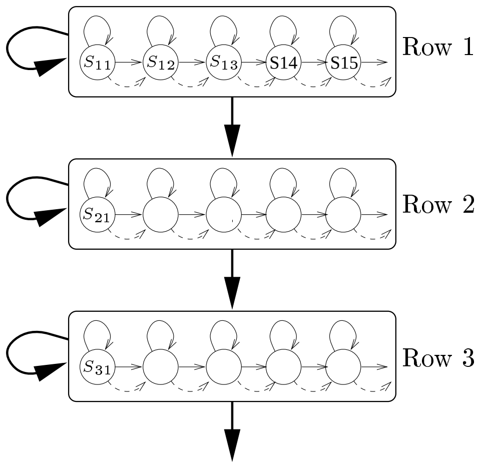

Figure 3: A schematic of the two-dimensional structure of the HMM, showing three row models of five states each, forming a global model for a single class.

For classification, a model $\mathcal{M}_c$ is constructed for each class, c, and the maximum likelihood class is chosen after calculating the probability of the data $\mathcal{R}$ given the model: $\text{argmax}_c P(\mathcal{R}|\mathcal{M}_c)$.

2.3.1 Row modelling

To simplify the analysis of the model, first consider a row model modelling a single row of fingerprint data. Each row model M_i is a conventional HMM and consists of a number of states, which model the small, stationary regions in a row. Any row R_i, is assumed to have been generated by the row automaton transiting from state to state, producing the frames in the observed order at each transition, with S_{ij_k} being the kth state in the sequence whereby M_i produces R_i (k corresponds to time in a temporal HMM). The state transitions are controlled by probabilities $P(S_{ij_k}|S_{ij_{k-1}})$ trained with certain constraints: the state must monotonically increase $S_{ij_k} \geq S_{ij_{k'}}$ for $k > k'$ and it is possible to skip states at the edge of the print. Because of the nature of the printing process whereby, especially for dabs, it is to be expected that edge regions of the fingerprint will be missing but the central regions will always be present, only states at the edge of the print may be skipped. This effectively constrains the initial state distribution $P(S_{ij_0})$.

The frame emission probabilities are modelled with mixtures of diagonal covariance, multivariate Gaussian distributions. Thus for any frame R_{i_k}, it is possible to calculate the likelihood $P(R_{i_k}|S_{ij_k})$ of it occurring in any state S_{ij_k}. With these likelihoods, for any row model, the likelihood of any row can be approximated by the maximum likelihood of any state sequence aligning the features and states calculated as a product of frame likelihoods and transition probabilities for the state sequence:

$$P(R_i|M_j) \approx \max_{S_{ij}} P(R_{i_0}|S_{ij_0})P(S_{ij_0}) \prod_k P(R_{i_k}|S_{ij_k})P(S_{ij_k}|S_{ij_{k-1}}) \tag{1}$$

The models are initialized by using an equal-length alignment with the frames evenly distributed across the states of the model. After estimating the initial parameter values, using smooth equal-length alignment [19], Viterbi alignment is used to find the maximum likelihood alignment of frames with states, which is used for retraining. Around two iterations of training are necessary to achieve good classification performance.

2.3.2 Global model

The global model is the same as a row model, except that its states are row models, and its frames are whole rows. Thus for each model c:

$$P(\mathcal{R}|\mathcal{M}_c) \approx \max_{S'} P(R_0|M_{S'_0})P(M_{S'_0}) \prod_k P(R_k|M_{S'_k})P(S'_k|S'_{k-1}), \tag{2}$$

where S' is an alignment specifying which row model $M_{S'_k}$ models the row of data R_k.

2.4 Multiple HMM classifiers

For each orientation of fiducial lines, a separate classifier can be made. Since the errors of the different classifiers will be different, a combination of their scores may yield a better accuracy. Denoting by $\mathcal{M}_c^h$, $\mathcal{M}_c^v$ the class c models trained with vertical and horizontal features respectively, and assuming independence, the likelihood of the data is written as:

$$P(\mathcal{R}^h, \mathcal{R}^v|\text{C}_c) \approx P(\mathcal{R}^h|\mathcal{M}_c^h)P(\mathcal{R}^v|\mathcal{M}_c^v). \tag{3}$$

Fusion of multiple classifiers is treated in more detail in section 5.3.

3 Decision tree classifiers

To provide a supplementary classification, hopefully giving uncorrellated errors, another type of features has been extracted and classified with a decision tree approach. Such decision trees are built using techniques based upon those of Amit *et al.* [20]. These authors tackled a number of problems including that of digit recognition — classifying images of the digits '0' to '9'.

The technique used by Amit *et al.* for constructing decision trees involves the generation of a large number of simple features. Each feature in isolation provides little information about the classification decision, for example, the existence of an edge at a particular location in an image may give little clue as to the digit's identity. However, combinations of such features can represent much important information needed to make an accurate classification decision. Amit *et al.* describe a procedure for making decision trees by growing questions based upon such combinations of simple features.

The procedure has been adopted here for fingerprint classification, and involves an initial feature extraction phase, followed by question building which creates informative questions assisting in classification. These complex questions are combined in a hierarchical manner to form decision trees which are used for classification. Because the trees are constructed stochastically, trees constructed for the same problem have different performances and as is common with decision tree classifiers, multiple trees are combined to give the final classification.

3.1 Feature extraction

This second classifier was designed to give a second opinion on the classification of a fingerprint image. For this purpose, the errors in classification should be as uncorrelated as possible with those made by the HMM, thus a different set of features was generated for this classification method. Again the motivation is to consider distributed information from across the fingerprint without extraction of singular points. Because the class information is implicit in the shapes of ridges, features that are easily and reliably extracted, and which encode the ridge shape in a simple, concise manner were chosen.

The initial preprocessing used is identical to that of the HMM classifier, up to the extraction of ridges (section 2.1), but instead of taking features at intersections with fiducial lines, features are generated at salient points on the ridges. The features consist of curvature maxima and four axis-parallel turning points ($\frac{dx}{ds} = 0$ or $\frac{dy}{ds} = 0$ for a ridge represented as the parametric curve $(x(s), y(s))$, and distinguished by the sign of the second derivative). Some example features are shown in figure 4. For each feature the feature type and location (in pixels at 500dpi) is recorded. These features are all based on local computations on the ridges, and again avoid the extraction of global features such as core and delta points. Again they are invariant to translation and to small amounts of rotation. These features are also appropriate for the classification of dabs, since the majority of features in a rolled print also occur in the region typically imaged in a dab.

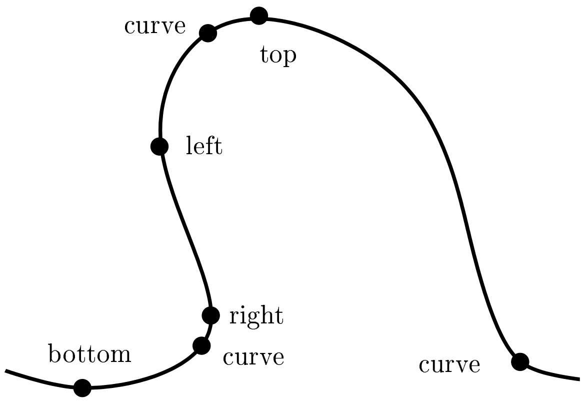

Figure 4: A single ridge showing the extracted features for the decision tree: curvature maximum and left, right, bottom and top turning points.

3.2 Decision trees

A binary decision tree is constructed as a hierarchy of binary questions [21]. Questions are logical statements about the features that may be present in the fingerprint, and about the relations between those features; for a given fingerprint a question is either true or false. At the 'top' level, each test sample is asked the first question. According to the test sample data, the question returns either true or false, and the branch to the second-level is determined. On which ever branch is chosen, a second level question is asked, and a further bifurcation is induced. In this way, each test sample descends the tree, by a route dependent on its features, and arrives at a leaf node. Leaf nodes are labelled according to the classes of the test data that arrived there. In a simple classification problem, leaf nodes will be pure — *i.e.* receive only training samples from a single class, and the unambiguous classification of any test sample arriving there would be the class of the training data at that node. For more complex problems, the leaf nodes contain mixed data, and the test data is labelled with a probability distribution across the classes.

Figure 5 shows a small decision tree with two levels. Each of the three nodes of the tree contains a question of the form specified in section 3.3. At each node, and at the leaves, a class histogram with four classes is shown, indicating the reduction in entropy as the tree is traversed. The root node has all classes equally likely, with no discrimination, and the other nodes have successively stronger discrimination between the classes.

3.3 Questions

A question consists of a list of required features and a set of relations between them, in the same manner as those of Amit *et al.* Each feature is specified as one of the five features described above (four turning points and curvature maximum). Relations are all of the form 'x is *direction* of y' where *direction* is one of North, South, East, West. Optionally a question can also impose a distance limit — that the feature must be within a certain distance. Experimentation led us to use two distance bands: features within 0.1" and features within 0.2". An example question may specify that there is a maximum of curvature East of a lower turning point which is itself East of, and within 0.2" of, a maximum of curvature. Other questions are shown in the nodes of figure 5. Questions are constructed in such a way that every feature is related to at least one feature in the feature list, and so that every pair of features can have at most one relation.

Given a new print, the print can be tested by a search that determines if the question can be fulfilled by the features of the print.

3.4 Question construction

During tree construction questions are constructed randomly as follows:

1. Select any feature class as the first feature
2. Test the data separation. If more than 2/3 of training data at this node reply yes, refine the question and repeat this step. If less than 1/3 reply yes, discard this question and construct a new question. Otherwise, evaluate the question.

Refining the question consists of adding extra restrictions, which inevitably make a 'yes' answer less likely. The proportion of samples answering 'yes' can be reduced in one of two ways. Firstly, a feature can be added. In this case, a random feature type is chosen and added to the list. A random relation is chosen to relate it to a randomly chosen feature already in the list. Secondly, if there are two or more features in the question, and some pair has no relation between them, then an additional relation can be added to the question between any pair of features that are as yet unrelated.

When adding a relation is not possible, a feature is added. Otherwise, a random choice is made, biased towards adding a relation, since this keeps the number of features lower, limiting the dimensionality of the search space for answering questions, and making testing faster.

Having arrived at a question which channels approximately half the data to each of the 'yes' and 'no' sides, the question is evaluated. The measure of the effectiveness of a question is the change of entropy in the distributions before and after applying the question. Classes at the root node have high entropy, but the leaf nodes should have very low entropy (be 'purer'). The relative entropy of the output distributions for a node is computed for many randomly constructed, candidate questions and the question with the highest entropy change is chosen.

A tree is recursively constructed until the leaves are pure or until a maximum depth (typically 7) is reached. Figure 6 shows the effect of varying the depth of the trees and the number of trees used. Multiple trees are merged by multiplying the leaf node distributions class-by-class as in section 2.4.

4 PCASYS

Candela *et al.* [7] have described a fingerprint classifier called PCASYS which is based upon a probabilistic neural network (PNN) classifying features consisting of principal-component projected orientation vectors. The orientations of the ridges taken in a 28×30 grid of points around the core are reduced in dimensionality with principal components analysis. The resulting 64-dimensional vectors are classified with the PNN. They have published results and made their software available, making possible a realistic comparison with this system. Lumini *et al.* [10] have used this software and extended it to provide continuous classification. To provide an alternative classification method and an enhanced combination classifier, PCASYS has been tested on the same testing data as classified by the HMM and decision tree classifiers. Results are presented for PCASYS alone and in combination with the other classifiers.

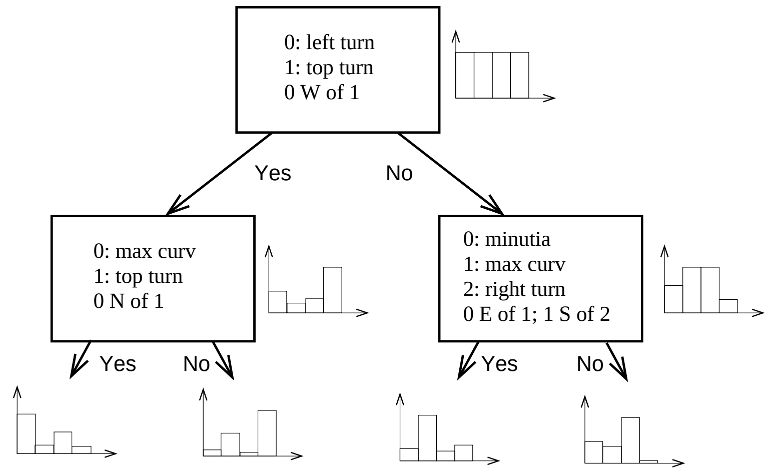

Figure 5: A two-level decision tree, showing hypothetical class distributions at each node. Each node has a question formed of a list of feature types that must be present and a list of relationships between them that must be true for the question to return 'yes'.

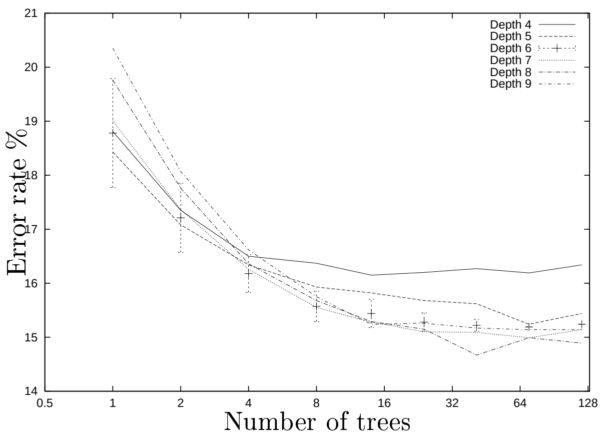

Figure 6: Raw error rates (no priors, unweighted) plotted against number of trees for the decision tree classifier tested on the NIST-4 test set, using averages of different numbers of trees, for trees built to different depths. Error bars in the error rate estimate are shown for the depth 6 case.

PCASYS incorporates a *pseudoridge tracer* which detects upward curving ridges and is able to correctly identify some whorl prints, but provides no information to distinguish among the other classes. This effectively penalizes the other classes when returning a 'yes' answer. PCASYS also exploits prior information to improve its accuracy (see section 5.1).

5 Postprocessing & classifier combination

Given the raw classifiers presented above, a number of steps must be taken to apply the classifiers to a test set. The following sections describe using class priors to enhance classifier accuracy, weighting the results to predict behaviour on true test sets and methods for the combination of multiple classifiers.

5.1 Class priors

Because the classes that are used are not equal in frequency of occurrence, calculating the posterior probability of a class given the data requires the product of the data likelihood given the class c and the prior probability of the class:

$$P(c|\mathcal{R}) \quad \propto \quad P(\mathcal{R}|\mathcal{M}_c)P(c). \qquad (4)$$

The class priors have been estimated by Wilson *et al.* [22] on 222 million fingerprints. (The proportions are 0.037, 0.338, 0.317, 0.029, 0.279 for arch, left loop, right loop, tented arch and whorl respectively).

5.2 Class weighting

Since the NIST-4 database (and correspondingly the test set used here) has equal numbers of prints from each class, to obtain a good estimate of the true test-condition accuracy, the results must be weighted according to the true frequencies of occurrence, using the same procedure as Wilson *et al.* [22]. Otherwise, a classifier good at recognizing arches, which are rare, would appear better on this test set than in the real world, or on a representative test set where this ability is rarely called upon.

5.3 Classifier combination

Four classifiers (counting the PCASYS pseudoridge classifier) are available in this work. Each by itself is capable of classifying fingerprints with a limited accuracy. However, each classifier uses different features and methods, so the errors produced by each classifier should be somewhat uncorrelated. In this situation, combining the results of the classifiers should produce an improved classifier with lower error rate. Many other authors have tackled the problems of decision fusion, and here we take two simple approaches.

5.3.1 Linear likelihood fusion

The first method of combination is a probabilistic approach since the output of each classifier is a probability distribution $P(c|\mathcal{R}^i)$ across the classes c.

Strictly speaking, if each classifier gave a true probability out, and with N independent classifiers operating on features $\mathcal{R}^i$, the posterior probability would be:

$$P(c|\mathcal{R}^1,\ldots,\mathcal{R}^N) \propto P(c)\prod_i P(\mathcal{R}^i|c). \tag{5}$$

However, in practice, the probabilities are correlated and have varying reliabilities. The HMM probabilities are the product of many correlated probabilities, and the PNN already incorporates prior information. To correct for these effects, weights w_i are introduced to balance the classifier combination. Working in the log domain, with normalization constant k:

$$\log P(c|\mathcal{R}^1,\ldots,\mathcal{R}^N) = k + \log P(c) + \sum_i w_i P(\mathcal{R}^i|c) \tag{6}$$

For simple classification, the class $\mathrm{argmax}_c \log P(c|\mathcal{R}^1,\ldots,\mathcal{R}^N)$ is chosen as the correct answer. Finding the weights w_i, however, is a difficult problem. Estimation of weights by line searches on the training set fails to generalize well to the test set, so the following trained approach was used, which is found to achieve accuracies close to those obtained when optimizing linear weights by line search on the *test* set.

5.3.2 Neural network fusion

The second fusion approach is to use a backpropagation neural network. Here the class probabilities for all the classifiers are combined in a neural network, trained to output the true class on the training set. Additionally four estimates of the fingerprint quality [23] are supplied to the network, though their effect is not significant. Training uses a momentum-based weight update scheme and Softmax outputs [24], giving an output class probability distribution. Training samples are weighted to simulate a uniform prior, and the output probabilities are multiplied by the class prior 5.1 when testing. Separate networks are trained to combine the HMM and decision tree, or to combine all four classifiers. To generate enough training data for the neural network, the first half of the NIST-4 database was supplemented with 5400 prints from the NIST-9 database (Volume 2, CDs 1,2,3).

6 Classifier efficiency

Since the purpose of a fingerprint classifier is to partition large fingerprint databases, in addition to the classification accuracy — the proportion of classifications that give the correct class — the classification efficiency must also be considered. Since many authors use different classes, a consistent measure of efficiency, as described here is essential for the comparison of results. An efficiency measure also permits the evaluation of rejection and backing-off strategies described in section 6.2.

The classification efficiency can be considered as a measure of reduction of search space. In practice, the proportion of the database to be searched will vary with each query, so over a test set, the average efficiency can be calculated as:

$$\frac{\text{Number of matches required with no classifier}}{\text{Number of matches required when classifier is used}}, \tag{7}$$

where an exhaustive 1:many match against a database of N prints is counted as N matches. If a perfect classifier is used to classify M prints prior to matching against a database of N prints, any of the MP_c test prints in class c (which occurs with probability P_c) need only be tested against the NP_c database prints of class c. Thus the total number of matches required is now $\sum_c NMP_c^2$ instead of NM. The efficiency of a perfect classifier using these classes is thus $\frac{1}{\sum_c P_c^2}$. Using the five NIST-4 classes and the frequencies of section 5.1, this gives an efficiency of 3.39. Merging arch and tented arch classes only reduces the efficiency to 3.37, since this distinction so rarely needs to be made. As can be seen, the imbalance of the class priors makes the efficiency significantly lower than would be obtained with five equally frequent classes (an efficiency of 5). In practice the efficiency of a fallible classifier will deviate from this value — for instance a classifier which consistently mistakes all prints for arches will have an efficiency of 15 (1/0.066).

6.1 Alternate classes

NIST-4 provides alternate class labels for 17.5% of the prints [12, p.8], but these are ignored in this work, a classification being deemed correct only if it matches the primary class label. Allowing matches with the alternate label too would increase the recognition rates but would lower the efficiency of the classifier, since such prints when enrolled would have to be stored twice (under both classes) in our database resulting in extra searches every time the secondary class was searched.

6.2 Backing off

Previous classification works have quoted error rates that would be unacceptable in real-world systems. It is clear that accuracy can be traded off for efficiency — searching more than just the top one class will give higher accuracy but lower efficiency [25]. If a reliable measure of confidence in the classifier's answer is available, it is possible to devise methods to adjust the reliance on the classifier answer, when that classification is uncertain, and thus reducing the number of errors made. Some classifiers [7] have used a rejection mechanism, which improves the accuracy at the cost of not pruning the search with those prints that are rejected. This section proposes a more complex scheme to allow graceful and efficient '*backing-off*' of classifier reliance based on a likelihood ratio confidence measure.

It is clear that if the likelihoods for the top two classes are very different, then the classifier can be considered to be more 'confident' in its answer than if the two likelihoods are similar (when it would only take a small perturbation to change the ranks of the answers). Thus, the likelihood ratio of the top two answers is examined. If this is less than an empirically determined threshold, then the top choice is deemed to be not confident and the top two classes are jointly returned as the answer (increasing the proportion of the database subsequently searched by the 1:many matcher). Similarly, the likelihood ratio of the second and third choices is compared to a threshold to allow backing off to three classes. Repeating the procedure, if all the likelihoods are similar, the classifier will return a "don't know" answer, and all classes must be searched. More traditional rejection strategies (*e.g.* [22]) use a criterion to back off directly from the 'top-choice only' to the 'don't know' answer without allowing as rich a classification.

The efficiency of the classifier when allowing backing-off is now:

$$\frac{MN}{\sum_{m=1}^{M} \pi_m N} \tag{8}$$

where $\pi_m, 1 \geq \pi_m \geq 0$ is the proportion of the database searched for query print m.

Adjusting the likelihood ratio threshold allows arbitrary accuracy to be obtained. A large threshold would give a null classifier with 100% accuracy but an efficiency of one. A threshold of zero would give the basic top-one accuracy and maximum efficiency (3.37 for the 4 class problem). Adjusting the threshold allows us to set the overall classifier accuracy to that deemed necessary for the whole system. However, it should be noted that this arbitrary classification accuracy is achieved within the context of a Henry classification system where the portions of the database to be searched will always conform to the Henry classifications, and thus allow the database partitioning and search to be designed to operate on prior knowledge, not having to cope with dynamically changing subsets as in continuous classification.

In fact the efficiency loss (*i.e.* extra search time) of searching the next class is dependent on the frequency of that class, so a more advanced backing-off algorithm should take this into account to achieve a better trade-off of accuracy for efficiency.

7 Results

Following the practice of Jain *et al.* [9], the system has been trained and tested on the NIST-4 database of fingerprint images. The training set was composed of the first 2000 prints and the test set was the remaining 2000 prints from NIST-4 (1000 pairs, so no test prints had corresponding prints in the training data). The primary class labels given with the databases were used, but since the efficiency is hardly affected, the classifier was only trained to distinguish four classes, treating arch and tented arch as identical. Table 1 shows the error rates for various combinations of classifiers. All results presented in this table are weighted, as in section 5.2, to simulate the natural frequencies of occurrence of the fingerprint classes.

Table 2 shows the confusion matrix for the four-classifier neural network fusion with priors, but without the class weighting, and shows the distribution of misclassifications and the error rate for each of the four classes used.

Classifier	Error (%)
HMM Horizontal ($\mathcal{R}^h$) features	20.8
HMM Vertical ($\mathcal{R}^v$) features	13.8
HMM Both with prior	12.8
DT with prior	14.4
HMM + DT + prior	9.0
PCASYS (PNN only)	8.5
PCASYS (PNN + Pseudoridge)	6.1
HMM +DT +PCASYS (no prior)	6.8
HMM +DT +PCASYS (prior)	5.1

Table 1: Error rates, testing different combinations of the classifiers described in this paper. The HMM classifier used here has 8 rows of 8 states each and uses a shared pool of 200 Gaussians with 4-dimensional features. The decision tree (DT) classifier is a mixture of 30 trees, each with seven levels. The three combination results use a neural network. The PCASYS PNN+Pseudoridge classifier result uses a heuristic combination [7] which incorporates the priors.

True	Assigned Class				Error
Class	A/T	W	L	R	Rate
A/T	637	5	66	112	22.3%
W	0	384	4	14	4.5%
L	10	2	370	4	4.2%
R	8	2	2	380	3.1%

Table 2: Confusion matrix for the NIST-4 test set (without reweighting). The table shows the assigned classes for fingerprints falling into each of the true classes, and a per-class error rate.

Figure 7 shows the trade-off between error rate and efficiency obtainable by varying the likelihood ratio used for backing off.

8 Previous results

Table 3 shows the results achieved by a number of fingerprint classification systems that have previously been published. These points are plotted in figure 8 along with the curve of possible operating points for the combination presented here and the continuous systems of Lumini *et al.* [2, 10] and Halici and Ongun [8]. For each system the efficiency of the classifier is shown with the corresponding error rate. In some cases, authors have used rejection strategies where a proportion π of prints are rejected, making the identification system search the whole database (with an efficiency of 1). Assuming these rejected prints are

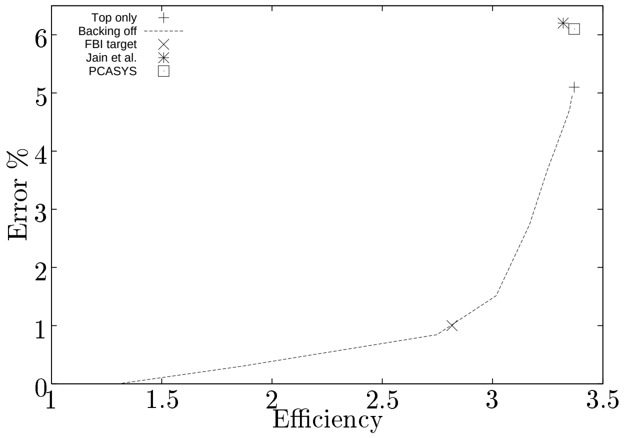

Figure 7: A graph of error rate against efficiency, for the combination described here. The curve shows the trade-off of efficiency against error rate for a variety of likelihood ratio thresholds. Other algorithms tested under the same conditions are also plotted, along with the FBI performance target.

uniformly distributed across the database, the efficiency of the combined system is

$$E_r = E(1-\pi) + \pi, \tag{9}$$

where E is the natural efficiency using a classifier returning a single class. (This is the value plotted in figure 8 where appropriate.)

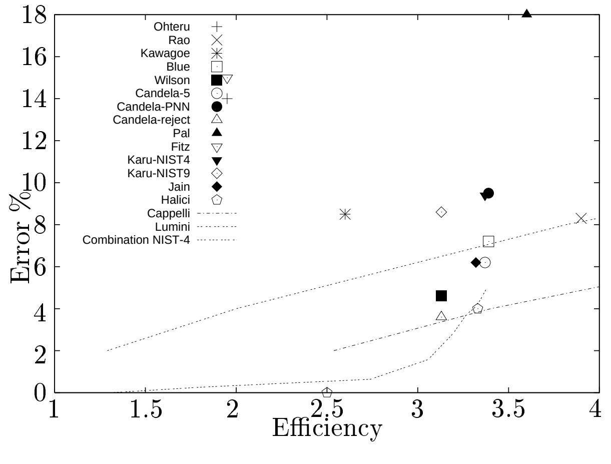

Figure 8: A graph of error rate against efficiency for a number of published algorithms.

Authors & Year	Classes	Efficiency	Error %	Reject	Test set
Ohteru *et al.* 1974 [3]	3	1.95	14		102 good quality
Rao & Balck 1980 [26]	6	3.9	8.3		60
Kawagoe & Tojo 1984 [27]	7	2.6	8.5		94
Wilson *et al.* 1993 [22]	5	3.39	4.6	10	weighted 2000 NIST-4
Blue *et al.* 1994 [6]	5	3.39	7.2		weighted 2000 NIST-4
Candela *et al.* 1995 [7]	5	3.39	7.8		2700 NIST-14
	5	3.39	3.6	10	2700 NIST-14
PNN only	5	3.39	9.5		2700 NIST-14
Pal & Mitra 1996 [28]	5	3.6	18+		45 (training set)
Fitz & Green 1996 [29]	3	1.95	15		40
Karu & Jain 1996 [4]	4	3.37	7.0	10%	4000 NIST-4 (priors)
	4	3.37	9.4		4000 NIST-4 (priors)
	4	3.37	6.1	10%	4000 NIST-4
	4	3.37	8.6	10%	4000 NIST-9
Halici & Ongun 1996 [8]	continuous	3.33	4.0		250 NIST
Jain *et al.* 1999 [9]	4	3.37	5.2	1.8 %	second half of NIST 4

Table 3: A comparison of published fingerprint classification methods

8.1 Comparison

Because of the estimation of efficiency in the case of rejection, and because of the wide range of testing conditions previously used, the figure and table present results which are not always directly comparable. In particular, the error rate of the PCASYS system is 7.8% under the conditions described by Candela *et al.* [7]. However, when their software is run on this test set and scoring in a manner consistent with the results presented here, with class weighting, the error rate was 6.1% (11.4% without weighting), a figure to which the results of the combined classifier here should be compared. Similarly Jain *et al.* quote an accuracy of 5.2% with 1.8% rejection, but if the data from the confusion matrix [9, table 3] are scored using the class frequencies found in real data (Figure 1), the accuracy is 6.2%[1]. These two are the only systems for which truly comparable results were available. Table 4 shows the error rates for the uniform testing conditions and these are plotted in figure 7.

One limitation of some previous works is simply that little can be inferred from the results presented when the test sets are so small or where the test set is not truly independent of the training set. For example [28] derive the test set from the training set using artificial noise processes. Although [22] and [6] use the NIST-4 database, they test on second imprints of the same fingers that were used for training, an unrealistic scenario for which classification by recognition of the fingerprints would result in much lower recognition rates, and efficiencies of the order of many thousands. [29] average over five samples of a fingerprint before attempting classification.

[1]A weighted average of the class error rates 7.4%, 7.3%, 5.0%, 1.4%, 6.0%, counting Arch/Tented Arch confusions as correct. For these results, however, where an alternate class label is given (cf. section 6.1), either answer is considered to be correct, giving a higher accuracy than would be obtained under the same conditions we have used.

System	Error (%)	Efficiency
Combination classifier	5.1	3.37
PCASYS	6.1	3.37
Jain *et al.*	6.2	3.32

Table 4: Comparative error rates and efficiencies for three systems on the second half of the NIST-4 data using true class frequency weightings.

A final problem with previous work is that the accuracies of the systems are simply not high enough. If one is to get the full filtering effect of the classification, only the top class must be chosen, and it has been seen that the classification accuracies for the top class (no paper presents any other accuracy, such as top 2 etc.) is never high enough to be used in a real system. The higher accuracies that are obtained are achieved by rejecting difficult prints, so the filtering achieved is even lower. Karu and Jain [4] quote the acceptable error rate for the FBI as being 1% at 20% rejection rate. With four classes, using equation 9, this is equivalent to a filtering efficiency of 2.816, a performance achieved by the combination classifier described here and by no previous system, as shown in figure 7.

9 Conclusions

This paper has proposed two new, effective methods for fingerprint classification which do not require core and delta information, and which have been designed to work on both dabs and rolled prints. The combination of classifiers described here produces significantly better results than any of the component classifiers. Existing Henry fingerprint classifier accuracies fall short of what is required to make a significant contribution to an AFI system.

This paper has proposed a method for comparing the efficiencies of different classification schemes and describes a system for achieving an arbitrary degree of accuracy from a classification system while evaluating the effect of the trade off. By this means, current fingerprint classifiers can be rendered of use in an AFI system. The new classification combination can achieve a filtering efficiency of 2.8, with an error rate of only 1.0%, meeting the FBI requirements for a classification system, and is the first system known to the authors to acheive this. Performance for this Henry system is comparable to the performance of continuous classifiers and extensions are envisaged to adapt the methods here for non-Henry and continuous classification

Acknowledgements

The authors would like to thank the reviewers for their suggestions, and Ruud Bolle, Sharath Pankanti and Nalini Ratha at IBM for assistance throughout this work.

References

[1] Federal Bureau of Investigations, *The Science of Fingerprints (Classification and Uses)*, US Department of Justice, Superintendant of Documents, U.S. Government Printing office, Washington D.C. 20402, 12–84 edition, 1984.

[2] Raffaele Cappelli, Alessandra Lumini, Dario Maio, and Davide Maltoni, "Fingerprint classification by directional image partitioning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 402–421, May 1999.

[3] S. Ohteru, H. Kobayashi, T. Kato, F. Noda, and H. Kimura, "Automated fingerprint classifier", in *International conference on pattern recognition*, 1974, pp. 185–189.

[4] Kalle Karu and Anil K. Jain, "Fingerprint classification", *Pattern Recognition*, vol. 29, no. 3, pp. 389–404, 1996.

[5] Dario Maio and Davide Maltoni, "A structural approach to fingerprint classification", in *Proceedings of the 13th International Conference on Pattern Recognition.* 1996, vol. 3, pp. 578–585, IEEE.

[6] J. L. Blue, G. T. Candela, P. J. Grother, R. Chellappa, C. L. Wilson, and J. D. Blue, "Evaluation of pattern classifiers for fingerprint and OCR application", *Pattern Recognition*, vol. 27, pp. 485–501, 1994.

[7] G. T. Candela, P. J. Grother, C. I. Watson, R. A. Wilkinson, and C. L. Wilson, "PCASYS — A pattern-level classification automation system for fingerprints", Tech. Rep. NISTIR 5647, NIST, April 1995.

[8] Ugur Halici and Güçlü Ongun, "Fingerprint classification through self-organizing feature maps modified to treat uncertainties", *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1497–1512, October 1996.

[9] Anil K. Jain, Salil Prabhakar, and Lin Hong, "A multichannel approach to fingerprint classification", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 348–359, April 1999.

[10] Alessandra Lumini, Dario Maio, and Davide Maltoni, "Continuous versus exclusive classification for fingerprint retrieval", *Pattern Recognition Letters*, vol. 18, pp. 1027–1034, 1997.

[11] Toshio Kamei and Masanori Mizoguchi, "Fingerprint preselection using eigenfeatures", in *Proceedings of Computer Vision and Pattern Recognition*, 1998, pp. 918–923.

[12] C.I. Watson and C.L. Wilson, "NIST special database 4: Fingerprint database", Tech. Rep., National Institute of Standards and Technology, March 1992.

[13] Michael P. Perrone and Scott D. Connell, "K-means clustering for hidden Markov models", in *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 2000, number 7, pp. 229–238.

[14] Andrew Senior, "A hidden Markov model fingerprint classifier", in *Asilomar conference on Signals, Systems and Computers*, 1997.

[15] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models", *IEEE ASSP magazine*, vol. 3, no. 1, pp. 4–16, January 1986.

[16] P. C. Woodland, J. J. Odell, V. V. Valtchev, and S. J. Young, "Large vocabulary continuous speech recognition using HTK", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Apr. 1994, vol. 2, pp. 125–128.

[17] N. K. Ratha, K. Karu, S. Chen, and A. K. Jain, "A real-time matching system for large fingerprint databases", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 799–813, August 1996.

[18] Oscar E. Agazzi, Shyh-shiaw Kuo, Esther Levin, and Roberto Pieraccini, "Connected and degraded text recognition using planar hidden markov models", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1993, vol. V, pp. 113–116.

[19] Krishna S. Nathan, Andrew Senior, and Jayashree Subrahmonia, "Initialization of hidden Markov models for unconstrained on-line handwriting recognition", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1996, vol. 6, pp. 3503–6.

[20] Y. Amit, D. Geman, and K. Wilder, "Joint induction of shape features and tree classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1300–1305, November 1997.

[21] Leo Breiman, *Classification and regression trees*, Wadsworth International Group, 1984.

[22] C. L. Wilson, G. T. Candela, and Watson C. I., "Neural network fingerprint classification", *Journal for Artificial Neural Networks*, vol. 1, no. 2, pp. 203–228, 1993.

[23] R.M. Bolle, S. Pankanti, and Y.-S. Yao, "System and method for determining the quality of fingerprint images", US Patent: 5963656, October 1999.

[24] Andrew Senior and Tony Robinson, "An off-line cursive handwriting recognition system", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 309–321, March 1998.

[25] Deba Prasad Mandal, C. A. Murthy, and Sankar K. Pal, "Formulation of a multivalued recognition system", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 4, pp. 607–620, July/August 1992.

[26] Kameswara Rao and Kenneth Balck, "Type classification of fingerprints: A syntactic approach", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 3, pp. 223–231, May 1980.

[27] Masahiro Kawagoe and Akio Tojo, "Fingerprint pattern classification", *Pattern Recognition*, vol. 17, no. 3, pp. 295–303, 1984.

[28] Sankar K. Pal and Sushmita Mitra, "Noisy fingerprint classification using multilayer perceptron with fuzzy geometrical and textural features", *Fuzzy sets and systems*, vol. 80, pp. 121–132, 1996.

[29] A. P. Fitz and R. J. Green, "Fingerprint classification using a hexagonal fast Fourier transform", *Pattern Recognition*, vol. 29, no. 10, pp. 1587–1597, 1996.