Handwritten Signature Verification Using Complementary Statistical Models

Thesis submitted by
Alan McCabe
November, 2003

for the Degree of Doctor of Philosophy
in the School of Information Technology at
James Cook University of North Queensland.

Supervisor:
Doctor Bruce Litow

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text, and a list of references is given.

Alan McCabe

November 10, 2003

Statement on Access to this Thesis

I, the under-signed, the author of this thesis, understand that James Cook University of North Queensland will make it available for use within the University Library and, via the Australian Digital Thesis network, for use elsewhere. All users consulting this thesis will have to sign the following statement:

In consulting this thesis, I agree not to copy or closely paraphrase it, in whole or in part, without written consent of the author; and to make proper written acknowledgement for any assistance I have obtained from it.

Beyond this, I do not wish to place any restrictions on access to this thesis.

Alan McCabe

November 10, 2003

ELECTRONIC COPY

I, the undersigned, the author of this work, declare that the electronic copy of this thesis provided to the James Cook University Library, is an accurate copy of the print thesis submitted, within the limits of the technology available.

_____ _____

Signature Date

Acknowledgements

I would like to thank a number of people who have made it possible for me to complete this project. Firstly I would like to thank the School of Information Technology and in particular my supervisor Dr Bruce Litow. I appreciate Dr Litow's time, constructive advice and his words of wisdom, many of which I remain unable to comprehend.

I also need to thank a number of friends and work colleagues for providing an outlet for the many frustrations over the past few years, as well as always being a willing source of distraction, whether it was required or not. I would especially like to thank Gerry O'Reilly who has continually remained patient, supportive and understanding, regardless of the pressures or the situation. She has always managed to put a smile on my face.

Finally and most significantly, I would like to thank both of my parents, Frank and Vicky, whose generosity quite simply knows no bounds. None of this would have been possible without the unlimited support, in all its forms, offered by them both.

Abstract

There is considerable interest in computerised personal identification and in particular in *biometrics*, a branch of identification that deals with verifying physical or behavioural characteristics of human beings. This thesis is concerned with the development of the particular biometric of handwritten signature verification, which is superior in many ways to other biometric authentication techniques that may be reliable but are much more intrusive.

Specifically, this project involves the use of two complementary artificially intelligent systems in the form of neural networks and hidden Markov models. Five sample signatures are used to build a reference in each of the independent models and experimentation and testing is done using an extensive database of almost 4,000 genuine signatures and forgeries. The confidence levels from each model are then combined and tested on unseen signatures resulting in an equal error rate of 1.1%. Further experimentation is performed and includes analysis of different verification scenarios, error contribution and the importance of visual feedback when signing. Finally, experiments are conducted exploring the possibility of "signing" handwritten passwords, with the developed system resulting in an equal error rate of 0.7% in the worst case.

Major Contributions

- A new method of signature segmentation, detailed in Appendix A, which dramatically reduces "false" segments;
- The development of successful methods for selecting effective training forgeries from a body of other users' reference signatures;
- A method for comparing signatures via string edit distance, outlined in Appendix B;
- Examination and development of methods for combining the output of neural networks and hidden Markov models;
- The introduction of several new features not previously used in on-line handwritten signature verification;
- A study on the need for a signer to have visual feedback when performing their signature.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Handwritten signatures remain one of the most commonly used and widely accepted means of authenticating an individual's identity. Despite the abundance of computing technology available, almost all signature verification is currently done by (often untrained) humans through simple visual inspection. Consequently, there is considerable interest in development of an effective automatic signature verification system capable of quickly and accurately verifying an individual's handwritten signature.

Handwritten signature verification (HSV) is a fertile research area in the field of pattern recognition and document analysis. It is not only a challenging, theoretically interesting pattern recognition problem, but also an ongoing real-world problem and one with significant commercial implications.

There are two main types of HSV, classified according to the method of data acquisition. An *off-line* (sometimes known as *static*) system uses an optical scanner to obtain a digital representation of a signature written on paper or some official document. An *on-line* (sometimes known as *dynamic*) system requires specialised hardware, such as a pressure sensitive pen or a digitizing tablet to obtain the signature. Each method has its own advantages and disadvantages.

Off-line systems have a wider area of applicability in that any handwritten signature can simply be scanned into a computer and digitised. This means that signatures written on cheques, wills or other important documents can be processed at a later date with minimal hardware requirements. However, off-line systems require an extra, possibly significant, pre-processing stage to remove irrelevant data from the image (such as noise or, with more difficulty, the background pattern on a cheque). Additionally, little dynamic or temporal information such as pen-tip pressure or velocity is available when using off-line

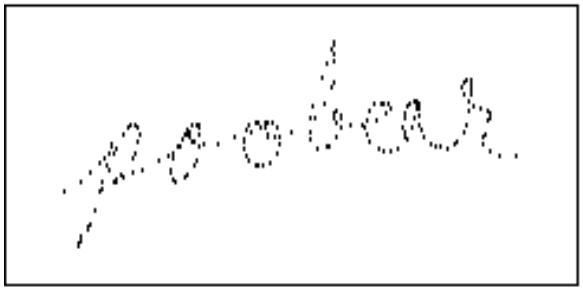

Figure 1.1: *A typical handwritten word, sampled at 205 points per second. Each dot in the handwriting represents the position of the pen tip for one of these samples.*

data (some researchers [30, 61] argue that it is possible to obtain basic velocity information from high quality images of a signature), which limits the insight that off-line data can provide as to the underlying nature of the handwriting. From a forgery standpoint, it is also easier to forge an off-line signature as the forger only needs to imitate the basic shape of the signature. For these reasons, the accuracy of purely off-line HSV system is generally lower that what is desirable.

On-line HSV systems use the rapidly sampled pen-tip position as input (see Figure 1.1), typically represented as x-y coordinate pairs. From this series of points it is a simple matter to obtain details related to timing information (such as overall duration and ratio of pen-down time to pen-up time) and dynamic information (such as velocity and acceleration). Some of the hardware tools used are also able to capture further details such as pen-tip pressure (how hard the user is pushing) and the pen orientation (such as the angles of the pen relative to the writing surface). This allows for a much richer feature set and a much greater insight into the underlying nature of the user's handwriting, and also allows the system to capture a significant array of details invisible to a potential forger. The biggest drawback of on-line systems is not the additional hardware requirements (digitizing tablets are becoming more commonplace), but rather the requirement that the signer has to be present at the time of signing and actively perform their signature on the tablet. That is, on-line systems are unable to verify signatures that are not produced on the tablet. The higher level of accuracy and difficulty in forging, it is believed, outweighs this drawback.

1.1 Handwriting

An understanding of the fundamental properties of handwriting is necessary in order to build an effective handwriting analysis system.

Natural variation in handwriting is a problem for signature verifiers (both human and automated) as it requires the verifier to judge whether the variation is the natural difference occurring between different instances of a signature, or whether the variation is significant enough to reject the instance as being an attempted forgery. The variations in handwriting are themselves habitual and can be used to aid verifiers as there is a marked individuality, even in the *way* the different instances of handwritten words or signatures vary. An effective signature verification system will take into account not only the expected *magnitude* of variation, but also the expected *type*.

The handwriting itself is the result of the inter-relationships between various forces applied by the writer's fingers, wrist and arm. These forces are broken down into three groups: pen pressure, point load and travel action [59] depicted in Figure 1.2. The characterisation of these inter-relationships into features (like velocity, pen-tip pressure, acceleration etc.) forms the basis of automated HSV [31]. Discussion and analysis of these features and their extraction continues throughout this dissertation.

1.2 Handwritten Signatures

There are many different forms of handwritten signatures and there is a great deal of variability even in signatures of people that use the same language. Some people simply write their name, while others have signatures that are only vaguely related to their name, if at all [18]. Some signatures are quite long and complex, while others are simple and appear as if they may be easily forged. It is also interesting to note that the signature style of an individual relates to the environment in which their signature was developed. For example, people in the United States tend to use their names as their signature, whereas Europeans tend away from directly using their names. Systems that rely directly on the "written-name" style of signing, such as [94], may not perform well when using signatures of Europeans, or signatures written in different languages.

It is well known that no two genuine signatures of a person are exactly alike and some signature experts note that if two signatures of the same per-

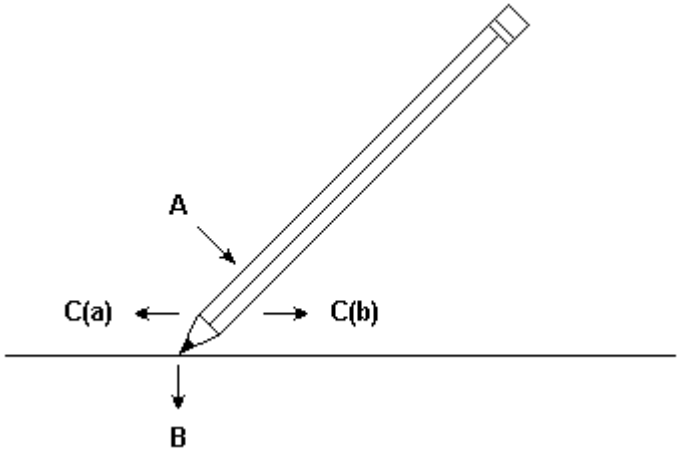

Figure 1.2: *The various forces at work in generation of handwriting.* `A` *is the "pen pressure" exerted by the writer perpendicular to the axis of the writing instrument;* `B` *is the "point load", which is the component of pressure exerted perpendicular to the writing surface, responsible for indentations, line thickness etc.; and C is the "travel action", the pressure component exerted in two dimensions (forward/sideward for upstrokes or crossings and drag/backward for downstrokes) across the writing surface, responsible for line generation. Together these forces contribute to features like velocity, acceleration, shape etc.*

son written on paper are identical they can be considered forgery by tracing. Successive signatures by the same person will differ, both globally and locally and may also differ in scale and orientation. In spite of these variations, it has been suggested that human experts are very good at identifying forgeries but perhaps not so good at verifying genuine signatures. For example, early work in [51] described experiments in which as high as 25% of genuine signatures were either rejected or classified as no-opinion by trained document examiners while no forgeries were accepted. Untrained personnel were found to accept up to 50% of forgeries.

Handwritten signatures, and especially handwriting in general, show great variation in speed and the author's muscular dexterity. Similarly, forgeries vary in perfection all the way from the clumsy effort that anyone can see is spurious, up to the finished work of the adept that few can detect. In a large majority of cases the work of a forger is not well done and in many instances is very clumsy. The process of forging a signature, if it is to be successful, must involve a double process requiring the forger to not only copy the features of the writing imitated but to also hide the forger's own personal writing characteristics. If the writing is free and rapid it will almost certainly show, when carefully analyzed, many of the characteristics of the natural writing of the author no matter what disguise may have been employed.

It should also be noted that varying conditions under which signatures are produced may have an effect on the signature. For example, hastily written, careless signatures, like those written in a delivery person's books may bear little resemblance to the same person's signature produced when seated comfortably. Furthermore, signatures written with a strange pen and in an unaccustomed place are likely to be different than the "normal" signatures of an individual. When a signature is being written to be used for comparison this can also produce a self-conscious, unnatural signature [99].

For HSV purposes a series of signatures is usually obtained from which a reference or model is built. Obviously, a reference built from five signatures should always provide a more satisfactory basis for an opinion than a reference built using one, and ten should be better than five.

[54] discusses what a signature is and how it is produced in a major study on the physiology of handwritten signatures. The author notes that the movement aspect of signatures is produced by the muscles of the fingers, wrist, and sometimes arm and that these muscles are controlled by nerve impulses. Once an individual is used to signing his or her signature, these nerve impulses are

controlled by the brain without any particular attention to detail or visual feedback from the signature being produced [54]. The implications of this for on-line HSV is that when an individual is signing on a tablet, it is not necessary to provide any visual feedback to the signer. This not only brings down the cost of hardware (as it is not necessary to use tablets capable of echoing the pen movements), but more importantly there is no physical legacy of the signature for a potential forger to access, even through casual observation of the signing process.

It is also noted in [54] that rarely does an individual's signature evolve over time, but rather once the signature style has been established, the modifications are usually slight. When such a change does occur, the earlier version is most often completely abandoned and replaced by the current one. Studies reported in [80] indicate variation occurs in about one percent of cases. When the change is immediate and permanent (for example due to name change after marriage), HSV systems would require that the user re-register and provide a new set of reference signatures.

1.3 Design Overview

The developed system that is described in this dissertation uses two fundamental pattern recognition models to learn different aspects of human handwriting. The result is a robust system, with a more complete representation of the nature of the handwriting.

The two models chosen were neural networks and hidden Markov models. These were chosen because of their inherent suitability to the various characteristics of handwriting.

Neural networks (NNs) are well-suited to being presented with a series of representative feature sets describing an entity (or class of entities) and then learning the relationship between the features and the class.

For example, a classical example of machine learning (including neural networks) is to teach a system to recognise a particular class of iris, given measurements of the sepal length and width and petal length and width. By closely examining the combinations of measurements of various samples it is possible for the NN to infer the relationship between the iris measurements and the class of iris to which it belongs. The NN is then presented with a *test* case (that is, a series of measurements with no associated class) and decides to which class the iris belongs. Neural networks (as well as many other machine

learning techniques) can perform this classification quite well.

The process of handwritten signature verification (at least, one of the possible approaches) parallels this learning mechanism. There are many ways to structure the training of a neural network, but a very simple approach is to firstly extract a feature set representing the signature (details like length, height, duration etc.), with several samples from several different signers. The second step then is for the neural network to learn the relationship between a signature and its class (which is either "genuine" or "forgery"). Once this relationship has been learned, the network can be presented with test signatures that can be classified as belonging to a particular signer. NNs therefore are highly suited to modelling *global* aspects of handwritten signatures. A detailed description of NNs is presented in Chapter 3.

Hidden Markov models (HMMs) are finite stochastic automata, with a powerful capability of modelling temporally-varying dynamic patterns. Traditionally used in speech recognition tasks [120] they have recently undergone a popularity increase in handwriting applications including character recognition (for example, [146] and [25]) and signature verification (for example, [33] and [85]). The training of HMMs is done in a similar fashion to the training of NNs in that a feature set is extracted from the signature and a series of these feature sets is presented to the HMM. Rather than a classification however, HMMs output a probability - the probability that a test signature was produced by a given signer.

HMMs are naturally suited to modelling "flowing" entities such as signatures and speech. They are made up of a series of states, with transitions between these states. Signatures can be split up into different sections through a process of *segmentation*, with the number of sections often related to the number of states in the HMM. It is then possible to progress through the states of the HMM in a corresponding fashion to progressing through the segments of the signature. At each state, the *local* features of the signature (such as the average velocity, acceleration or duration in the current segment) can be examined. *Local* features are therefore naturally modelled by HMMs. Further details of HMMs are presented in Chapter 4.

Several authors have used one of these models with varying levels of success, for example [105] did useful work with NNs and [32] did the same with HMMs in signature verification. Most work done in combining these two models has been to use NNs to provide more intelligent initialization of the HMMs [154, 155, 68], rather than combining the output of the two. The focus of the work

presented in this dissertation however was to "take the best of both worlds" and combine the capabilities of global modelling by NNs with the local modelling by HMMs to create a system with a better view of the "complete picture" when verifying handwritten signatures. This is analogous to getting two opinions on the validity of a signature from two different experts and combining their opinions. Chapter 5 discusses combining the output of NNs with the output of HMMs in the final system.

1.4 Thesis Structure

The thesis has the following structure: Chapter 2 presents a background and a review of the general, relevant signature verification literature (emphasis on specific approaches such as neural network and hidden Markov model systems are presented in the corresponding chapters). Chapter 3 presents a discussion of the theory of neural networks and discusses some of the different topological structures used for signature verification. In addition, the different features used in the NN are described as well as the different experiments conducted and discussions of these. A theoretical discussion of hidden Markov models forms the first half of Chapter 4, and the second half presents the different models configurations, experiments and discussions. Chapter 5 explains the method by which the NN and HMM output is combined and final discussions and conclusions are presented in Chapter 6. A new approach to avoiding local extrema in a specialised domain is also presented in Appendix A, and fast handwritten signature comparison via string edit distance is presented in Appendix B.

Chapter 2

Literature Review

Personal identification through the use of signatures or some other personal mark has existed for thousands of years. Handwritten signatures have been the customary method of this identification for hundreds of years, and societies increasing dependence on computerised storage and transmission of data and finance has prompted the need for automated handwritten signature verification.

The importance of handwritten signature verification (HSV) is discussed in [89] and [138] in terms of commercial development. Both discuss the growing availability of commercial products, as well as the fact that several hundred patents have been granted in the field of computer verification of handwritten signatures. This interest is at least partly due to the fact that HSV maintains public favouritism over many other biometric authentication techniques such as finger prints or retinal patterns, which are reliable but much more intrusive. In order to increase the acceptability of automated HSV in live authentication situations, the techniques must become more reliable.

It should be noted that the aims of HSV systems are going to be different for different types of applications. For example, the primary concern of verification in a credit card environment (where the card holder presents a card to make a purchase and signs on an electronic device that verifies the signature) must be to have a zero or near zero false rejection rate (FRR or Type I Error) so the genuine customer is not annoyed by unnecessary rejections. Fast verification is essential in this environment along with small storage requirements for the information involved (as it may need to be stored on a credit card strip or smart card memory). A high level of security against forgeries is not required in this environment and a false acceptance rate (FAR or Type II Error) of 10% or even 20% may be acceptable since even that is better than the minimal

checking that is done currently [51]. On the other hand, in a security sensitive environment that involved granting an authenticated user access to sensitive information or other valuable resources, it would be necessary to have a high level of security against intruders and a zero or near zero FAR. A FRR of 10% or higher would be a nuisance but might be acceptable. Of course an ideal HSV system should have both the FRR and the FAR close to zero. It should be noted that FRR and FAR are closely related and an attempt to reduce one invariably produces an increase in the other.

As this dissertation is concerned with the development of an on-line signature verification system (the differences between on-line and off-line systems are discussed at the beginning of Chapter 1), most of the systems discussed will be on-line.

An attempt is made to describe important techniques and achievements in HSV research and assess their performance based on published literature. Techniques specific to neural networks and hidden Markov models are presented in the corresponding chapters.

2.1 Applications of Handwritten Signature Verification

This section describes some of the various applications of HSV and how a typical system such as the one described in this dissertation can be used.

Credit cards deal with extraordinarily large amounts of funds on a daily basis, with billions of dollars worth of purchases charged to credit cards each day. It has been reported that credit card issuers lost over one billion dollars to credit card fraud in 2002 [21] and over $700 million in on-line transactions. Although the losses due to fraud are quite small when expressed as a percentage of total credit card purchases (just over one percent [103]), the small commission available to credit card issuers means that the losses are significant.

The minimal manual checking done by banking and point-of-sales staff does little to curb the rate of forgery. A reliable and practical HSV technique obviously has applications in reducing fraud, and after the initial registration there would be a dramatic reduction in forgery acceptance.

HSV also has applications in other areas of user authentication. An example is as a replacement to simple password access of computer terminals. The most common mechanism used in this area is the username/password

combination, which is widely regarded as ineffective in stopping a determined attacker [142, 144]. The addition of inexpensive handwriting capture hardware would result in a much more effective and reliable access system.

A more complicated authentication scheme could also be facilitated by HSV. For example, the existing password system could be used to allow initial access to the computer system with a signature required if the user wants to access more sensitive areas of the computer system. With the addition of networking capabilities, various remote user-authentication scenarios become possible where individuals can remotely establish their identity for the purposes of commerce, establishing intent or simply to ensure the identity of communicating parties.

2.2 Automated Handwritten Signature Verification

This section presents a discussion of signature writing dynamics as well as the basic methodology of automated (computerised) HSV.

2.2.1 Dynamics of Signature Production

In order to fully understand the process of signature verification and the research in the area, it is first necessary to examine the production and capture of the signature itself.

As discussed in Chapter 1 there are two classes of HSV systems, off-line (or static) and on-line (or dynamic). Off-line systems capture only the image of the signature, whereas on-line systems capture extra dynamic information. As the system described in this dissertation falls into the on-line class, further discussion of papers in this chapter will focus only on the on-line approach, unless the techniques used in the off-line system are relevant to the discussion.

The dynamic information in a signature is captured via a graphics tablet and/or instrumented pen, typically in the form:

$$S(t) = [x(t), y(t), p(t)]^T \quad t = 0, 1, 2, ..., n$$

that is, the dynamic information is a collection of x and y coordinate values sampling the position the pen tip, and usually the pen tip pressure values, at given times (generally, equal time intervals). Some devices capture additional information such as the pen-tilt angle. Most devices sample at the rate of

around 200 times per second (sometimes less) and the resolution of such devices is often about 100 pixels per millimetre (2,540 pixels per inch). Error rates on pixel position are typically around 0.25 to 1.25 millimetres (0.01 to 0.05 of an inch). Typical United States-style signatures are a writing of the persons name and therefore, for this style of signature, the x values typically grow linearly with time with small oscillations on the linear curve while the y values show a more oscillatory variation with time, becoming larger and smaller many times during a signature.

The signature data, after appropriate pre-processing may be used to compute the derivatives of x and y with respect to t. The first derivatives of x and y are the velocities in the two directions (which may be combined to compute total velocity) and the second derivatives are of course the two accelerations. It is also possible to compute the third derivative (rate of change of acceleration, referred to as *jerk*), however this is rarely used in HSV. Once these derivatives have been computed, the following signature data (assuming no pressure derivatives) is available:

$$S(t) = [x(t), y(t), p(t), xv(t), yv(t), xa(t), ya(t), xj(t), yj(t)]^T \quad t = 0, 1, 2, ..., n$$

It is clear that every time a person signs his or her signature the number of samples obtained will differ, that is, n will have a different value. This variation in genuine signatures of the same individual makes it very difficult to directly compare one set of values from one genuine signature with another set of values from another signature. Some basic methods for comparison of feature sets are explained in Section 2.2.2.

Instead of representing signatures as a collection of many individual data points, many authors consider them as a sequence of *strokes* (sometimes referred to as a *stroke segment*). In [29] a stroke is defined generally as a sequence of fundamental components delimited by abrupt interruptions. A number of interpretations of specifically how to extract strokes have been made by many different authors throughout the literature. The simplest approach is to use a pen-down to pen-up change as the delimiting interruption or to use equal components (in terms of duration) of the signature [51]. A more heuristically advanced method of segmentation is usually required however, such as using a change in direction as the delimiter. This was the approach taken in many articles, for example [86], and considers a stroke to be a continuous "pen-down" segment between a y-minimum and a successive y-maximum, or between a y-maximum and successive y-minimum. A more common method of stroke

segmentation is based on velocity values rather than displacement. A velocity based strokes (VBS) (used previously in [85]) is a continuous pen-down segment bounded by consecutive minima in the pen-tip velocity [135, 151]. There are also several variations to the VBS, including delimitation using minima only in the y-velocity or sign-changes instead of minima. Appendix A presents a new approach to the segmentation problem, developed specifically for HSV.

Once the stroke segmentation has been performed (irrespective of the specific technique used) most approaches follow the same basic principle. A characterisation of each stroke is performed (for example, extraction of a set of features) and comparisons between signatures are made via the corresponding strokes, with less requirement for warping of signatures than in point-to-point systems.

The velocity profile of a signature was further examined in [109]. The author stated that the shape of the velocity profile obtained from rapidly written strokes, such as those produced when a signature is written, is approximately bell-shaped but is asymmetric. The velocity tends to rise quickly to a peak then reduces to zero much more gradually. It is further claimed that the shape is almost preserved for movements that vary in duration, distance or peak velocity. It may be that this consistent shape is related to the way the central nervous system plans and controls movement. A skilled forger may manage to closely approximate the shape of many strokes of a signature, but it is unlikely that his/her velocity and acceleration profiles for those strokes will closely resemble those of the genuine signer. This is the main reason that on-line systems are so effective at detecting forgeries.

Other reported work involved an examination of the difficulty in forging signature dynamics. Obviously, a forger cannot imitate the shape *and* the ballistic motion of another person's signature without a great deal of practice, therefore producing effective forgeries is never going to be easy (although some signatures lend themselves more readily to forgery than do others). This forgery production complexity is studied in [17] and [18] in which the authors attempt to estimate the intrinsic risk of a signature being forged. They assert that a signature can be thought of as a sequence of triples, consisting of two curvilinear strokes and an angular stroke. Based on this, an expression is derived for an imitation difficulty coefficient for a signature. Simply stated, the expression gives the difficulty of a given signature to be a function of the variation rates in length and direction of the strokes. It is claimed that people with a small value for this coefficient and high variability between their signatures

are problematic signers. The authors also recommend personalised thresholds based on the forging difficulty of a person's signature.

2.2.2 The Basic Methodology

Irrespective of the final approach used to perform the verification, most HSV techniques follow five basic phases: data acquisition, pre-processing, feature extraction, comparison and performance evaluation.

During the first three phases of the signature verification process, most systems generate a reference (or a set of references) for each individual. The reference can be in many forms, including a mathematical model (for example, a neural network or hidden Markov model), an array of feature values or simply copies of the signatures themselves. The process of creating a reference normally requires several samples of the user's signature to be captured at enrolment or registration time (these signatures are called *sample* signatures or more commonly *reference* signatures) and processed. When a user claims to be a particular individual, they must provide a signature (labelled as the *test* signature) to be compared with the reference for that individual. The test signature undergoes any necessary pre-processing such as noise removal etc. prior to extraction of any features used. The test signature (or, more often, the extracted features) is then compared to the reference and a value is obtained indicating the likelihood that the test signature was produced by the same person that provided the reference signatures. If the similarity is above a predefined threshold value the signer is authenticated, otherwise they are rejected as attempted forgers.

A performance evaluation of developed techniques is obviously of great importance and researchers normally use a set of genuine signatures and forgery attempts either collected by them or by someone else, and determine the false rejection rate and the false acceptance rate for the technique given the signature database. Obtaining practical estimates of FAR is very difficult since actual forgeries are almost impossible to obtain. Performance evaluations therefore rely on two types of forged signatures. A forgery may be *skilled* if it is produced by a person who has had access to one or more genuine signatures for viewing and/or practice. A forgery is called *zero-effort* or *random* when either another person's genuine signature is used as a forgery or the forger has no access to the genuine signature and is either only given the name of the person whose signature is to be forged or just asked to sign any signa-

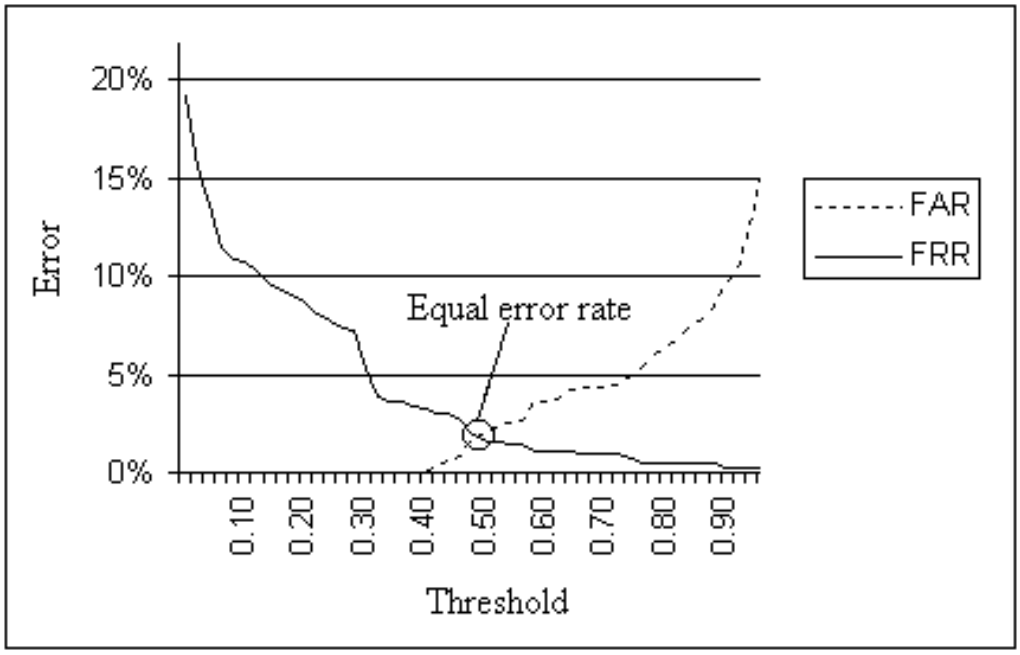

Figure 2.1: *The equal error rate is the error value at which the false acceptance and false rejection rates are the same.*

ture without even knowing the name. Tests on zero-effort forgeries generally lead to much smaller FAR than on skilled forgeries and are really a measure of how easily the system is confused, rather than of forgery defeating ability. In addition to FAR and FRR (and zero-effort FAR), many authors quote an *overall error rate* (OER) or an *equal error rate* (EER). The OER is simply the sum of FAR and FRR whereas the EER comes about because of the inverse relationship between FAR and FRR where one of these rates decreases as the other increases (and vice versa). The EER is the point at which these two error rates cross, as shown in Figure 2.1.

The value of performance evaluation in assessing a signature verification system is obvious, however it is not always a true indicator of the performance of the technique for a variety of reasons. The primary problem with testing procedures is that often the test signatures do not adequately represent the population at large. It is noted in [110] that there is a great deal of signature variability according to the author's nationality, age, time, habits, psychological or mental state, and physical and practical situations. Building a test database of signatures that is representative of real-world applications is quite a difficult task given the effort involved to find a large number of people that will willingly provide ten or twenty samples of their signature in several dif-

ferent sessions. Most people are reluctant to have their signatures stored in a computer or given to others to practice forging them. Additionally, few people can make themselves available to provide the signatures over many different days (as is required to capture the day-to-day variability of signatures).

Most test databases are built using signatures from volunteers from the research laboratory where the HSV research has been carried out and as a result have very few signatures from people that are old, disabled, suffering from a common disease (for example arthritis), or poor. Percentages of such people in the population is significant and these are the people whose signatures are likely to pose the greatest challenge for a HSV technique. The inconsistent signing environment in the real world poses further challenges for HSV as test databases are often created in very controlled situations and settings - something that is not the case in the real world.

The lack of common test databases also prevents any meaningful comparisons between different HSV systems. Most researchers have their own test signature databases with a varying number of genuine signatures, some have skilled forgeries while others do not, some have screened the signature database and removed several signatures that for some reason were not acceptable while others have done no screening, the number of signatures used in building a reference signature often varies, different tests and thresholds have been used and even different definitions of FAR and FRR have been used. Some studies use a different threshold for each individual while others use the same threshold for the entire database. This is a rather sad state of the art in HSV.

HSV systems concern themselves, at least on some level, with features extracted (either in the form of parameters or shape) from the handwritten signature. The primary issue then is: which features of a signature are most important? Despite the level of research into signature verification and handwriting in general (both computerised and non-computerised) there is no clear answer to this - nor is there ever likely to be. Despite this, the approaches can be broken down into two fundamental groups identified in [108] as *functional* and *parametric*.

In the functional approach, all of the collected position values (and/or velocity values and/or acceleration values) of a signature are assumed important and the test and reference signatures are compared point-to-point using one or more sets of these values. In this approach the major issue that arises is how the comparison is to be carried out, as the feature extraction phase is almost non-existent. The signatures *could* be compared by computing the correlation

coefficient between the reference signature functions and the corresponding test signature. Point-to-point comparison does not work well however, since the functions compared are of different duration and nonlinearly distorted with respect to each other [108]. In order to avoid the problems with direct point-to-point comparison, it is possible to perform a segmentation of the signature according to some boundary condition and then compare the corresponding segments. Some basic alignment of the segments may be necessary, but this is slight in comparison with the effort required to align signatures on a point-to-point basis. The segmentation approach seems to work significantly better and will be further discussed later in this dissertation.

The difference between the functional approach and the parametric approach is that the parametric does not use all of the available points. Instead of comparing the entire function (or segments of it), a collection of values is computed and comparison is done through these values. These values are generally known as *features* or *parameters* and some examples that have been used in the literature include:

- Total time taken in writing the signature;
- signature path length: displacement in the x and y directions and the total displacement;
- path tangent angles: profile of their variation and average or root mean square (RMS) values;
- signature velocity: profiles of variations in horizontal, vertical and total velocities as well as their average or RMS values;
- signature accelerations: variations in horizontal and vertical accelerations, centripetal accelerations, tangential accelerations, total accelerations, as well as their average or RMS values;
- pen-up time: total time spent with the pen off the page or the ratio of pen-up time to total time.

The above list is far from comprehensive. For example, the authors in [26] propose forty-four features that include some of the features listed above as well as several others. In a United States Patent, [102] proposes more than ninety features for consideration.

After a set of features has been selected and extracted, there is no real need to store the reference signature and only the feature values of the reference

signature need to be stored. Also, when a test signature is presented, only the feature values are needed, not the signature itself. This often saves on storage (which may be at premium if, for example, the reference signature needs to be stored on a card) and is the reason that signature representation using features is sometimes referred to as *compression* of the signature. In the methods that use feature-based comparison, selection of features and extracting the best subset once a set of features has been identified are major research tasks. Some of the problems related to these tasks are:

- How many features are sufficient?
- Do the features require some transformation to be applied to the signature data, for example resizing, deslanting, time-warping?
- How many sample signatures will be used in computing the reference?
- Would the reference signature be updated regularly? What would be the procedure for this?
- How is the distance between a test signature and the reference going to be computed?

Features, once decided upon and extracted from the test signature, then must be compared to the reference in some meaningful way. The focus of following discussion is on this comparison process.

Assuming that the reference is based on a set of sample signatures, and for each element of the set of selected features the mean and standard deviation of the feature values have been computed. The reference therefore consists of two vectors: a vector (R) of the means of the feature values and a vector (S) of the standard deviations. Clearly, to obtain accurate estimates of the mean and the standard deviations it is necessary to use several, perhaps a minimum of five, sample signatures. A larger set of sample signatures is likely to lead to a better reference and therefore better results but it is recognised that this is not always possible. The number of sample signatures needed is further discussed later in this chapter.

The distance between a test signature and the corresponding reference may be computed in several ways. Five different approaches are considered in [74], which involve direct comparison between the test signature and reference signatures themselves:

1. Linear discriminant function;
2. Euclidean distance classifier;
3. Dynamic programming matching;
4. Synthetic discriminant function;
5. Majority Classifier.

Linear discriminant function: a linear combination of the n components of feature vector x has the general form: $G(x) = \sum_i w_i x_i + w_0$ where w_i is the i^{th} element in the weight vector, x_i is the i^{th} in the feature vector and w_0 is a constant. Also called the *threshold weighting*, w_0 specifies the boundary between two classes. Two particular approaches to linear classification are proposed by the author. The first has each feature value t_i of the test signature normalised by the reference mean r_i and the second approach has feature value t_i normalised by the reference standard deviation s_i. Therefore, the first linear discrimination function becomes:

$$G(T) = (1/n) \sum_{i=1}^{n} \frac{t_i - r_i}{r_i}$$

where T denotes the test signature with feature set $(t_1, t_2, ..., t_n)$. The mean-value-normalised linear classifier produced and an equal error rate of about 17%. The calculation of the standard-deviation-normalised linear classifier is very similar and resulted in comparable error rates.

Euclidean distance classifier: The Euclidean distance discriminant function is used quite widely, for example in [26] and [96]. The Euclidean distance metric has the following form:

$$G(T) = (1/n) \sum_{i=1}^{n} \left(\frac{t_i - r_i}{s_i} \right)^2$$

where r_i and s_i are, respectively, the i^{th} feature's reference mean and reference standard deviation and t_i has the same definition as above. The performance evaluation of the Euclidean distance classifier using the 42 features described in [74] yields surprisingly poor results of an equal error rate of approximately 28%. Other researchers using the Euclidean distance classifier report much better results [96], and are discussed later in the chapter.

Synthetic discriminant matching: The use of synthetic discriminant matching (SDF) in HSV was introduced in [158] and consists of finding a filter impulse response w by solving a series of equations. Further details of the exact

nature of SDF can be found in [158] or more generally in [1]. SDF performed well in comparison to the earlier presented classifiers and an equal error rate of approximately 7% was obtained.

Dynamic programming matching: In simple terms, dynamic programming matching (DPM) involves minimizing the residual error between two functions (signatures) by finding a warping function to rescale one of the original functions' time axis. DPM is further discussed in [74] and is investigated in a number of other papers in the literature, for example [100]. Using the DPM technique to align the signatures resulted in an equal error rate of about 13%.

Majority Classifier: The main drawback of the linear classifier and Euclidean classifier is that a single feature can unduly influence the decision result when deviating far from the mean value, even if the other features have values close to their means for the genuine reference set. The majority classifier, based on the "majority rules" principal, is resistant to this problem. The basic technique is to examine the features individually and to declare the test signature to be genuine if the number of features passing a pre-determined test is larger than the number that fail. According to [74], the majority classifier achieves a very successful equal error rate of only 3.8% using the 42 features.

The issue of how many features a particular method needs to use to obtain a reliable verification result is a difficult one. The natural temptation is to include more and more features in the hope of improving performance and, as was noted earlier, some researches have proposed more than 90 features. Depending on the nature of the method being used, it is highly possible that the use of *too many* features may actually degrade the performance of a system. For example, at a basic level, the storage needed to record the reference details may exceed the available capacity (if the storage medium is some kind of "smart card" or any other device with limited memory). A large number of features also increases the complexity of many aspects of the problem (learning, comparison etc.) and will result in slower system execution. Additionally, in large feature sets where choosing of features has been less selective, it is likely that several of the features will be less representative of the natural properties of the signature. The effect of this is that the non-representative features will cause genuine signatures to appear further away from the reference than they should be and may well even cause forgeries to be closer to the reference than they should be (that is, "noise" is being introduced into the dataset). Obviously this is going to degrade the accuracy of the system. Ockham's razor, the maxim suggesting that when all other things are equal

a simplified approach will outperform a more complex one, is a principle that holds throughout signature verification.

It should be noted that some techniques like neural network learning (see Section 3.1.1) attempt to automatically associate a weighting or importance with the different features. The result is that, from a large feature set, those which are less important are given a lower weighting and thus and have less effect on the overall classification of that signature. Use of many features therefore will not of itself degrade the system, but if less relevant features are allowed to influence the classification process then a loss of performance will result.

As for the number of sample signatures required to build a reference, this remains an open question. There is a strong relationship between the quality of the reference and the number of sample signatures used to create this reference. Obviously, all other things being equal, a reference built using many sample signatures is going to be more representative of the signers natural variation than one built using few. Unfortunately, requiring that a user provide several (for example, more than ten) signatures would be of great annoyance to that user (in addition to larger computational and storage requirements). There is a degree of tradeoff between the level of reference quality and the level of user annoyance. It is argued that five sample signatures is the minimum necessary to produce a useful reference [46].

A further issue with regard to sample signatures is that of updating. Any real-world signature verification system needs to adapt to the changing signature of a user. Although most users will not undergo a notable change in their signature once the natural style has been established. A practical HSV system should make some attempt to update references for users in the database. The process of signature evolution is difficult to simulate in laboratory conditions and as a result systematic studies of adaptive systems are almost impossible. Techniques to handle the evolution include the re-taking of sample signatures at some regular interval (which is inconvenient for users) or incorporation of verified signatures into the reference set (at the risk of incorporating forgeries).

2.3 Review of Earlier Work

Given the potential for protection against financial loss and the commercial aspect of automated HSV, it has been a fertile ground for research in recent years. There is perhaps a great deal more research being done than is reported

in the literature due to the high perceived commercial value of innovations in the field and a reluctance of some industry groups to make details of their work public. Despite this fact, the volume of literature published is quite large, with approaches being as diverse and varied as any area in computer science.

One of the earliest cited works on the automation of HSV was published in 1965 and appears [84]. The author here reportedly took fifty signatures from each of forty subjects and used these to evaluate a method involving spectral density and zero-crossing features extracted from pen acceleration waveform measurements, resulting in a FRR of 37% (no forgeries were tested). Another early study from 1972 is presented in [39] where the authors use chain-encoded tablet data and test the method on signatures from ten subjects. A FRR of 27% and FAR of 27% are reported.

An early study of HSV using handwriting pressure was presented in [141] and cited in [51] and [166]. In this study, handwriting pressure was treated as analog z-axis information and included in HSV attempts. A FRR of 0.7% and a FAR of around 2% using zero-effort forgeries were reported. Later tests found the rates to be 6.8% FRR and 3.2% FAR for zero-effort forgeries and 17% FAR for skilled forgeries [51].

The remainder of this chapter is devoted to discussions of recent previous work. The focus of these discussions is on approaches that are relevant in some way to systems described in the later chapters of this dissertation. There is also a specific subsection devoted to systems making direct comparisons between extracted feature values.

Early work performed in [51] examined, among other details, timing and velocity features. The authors used two orthogonal accelerometers mounted on an experimental pen to sample the signatures. They comment that signature writing is predetermined by the brain, and as such, the total time taken for writing signatures by an individual is remarkably consistent. This relates to velocity in that consistent duration across signatures implies consistent velocity. Most signatures were of two to ten seconds in duration, with an average time of about five seconds. Not surprisingly, signatures that were longer in duration were found to be more difficult to forge.

The basic approach used in [51] was to perform a segmentation of the signature into a series of components and then use regional correlation to match the test signature to the reference. Limits were placed on how far the regions could be shifted to match up with corresponding regions, for example a test signature is automatically rejected if the duration differs by more than

20% from the expected value calculated from the reference signatures. A FRR of more than 20% was obtained with a FAR of around 1%. Interestingly, 59% of forged signatures were rejected on the basis of the signature duration being more than 20% different than expected, without undergoing the correlation test [166]. Duration therefore appears to be quite an important aspect of a signers natural style.

To overcome the limitations of the regional correlation technique used in [51], a nonlinear time alignment is used in [162] to align signatures, a technique borrowed from speech recognition. It was asserted that nonlinear alignment handles the natural variation in signatures better than the linear approach. The technique works by building a series of "time registration paths" by matching peaks in the pen force values, and selecting the path for which the Euclidean distance is minimal. Thresholds are placed on this distance, below which the signature is accepted, otherwise it is rejected. Error rates of less than 5% were reported when verifying the author of a keyword, using ten subjects.

In another attempt to improve the performance of the original regional correlation approach, pressure waveforms are considered in [80]. As part of the study, the authors noted that the pressure waveforms were reasonably consistent, varying slightly depending on the feel of the pen, how the pen is inking etc. Unfortunately though, the pressure waveform was found to be reasonably easy to imitate. Combining pressure correlations with acceleration correlations from [51] obtained results of close to 16% FRR and below 1% FAR using twenty-four subjects.

Further modification of the correlation technique is proposed by allowing more advanced transformations in the correlation process [78]. This algorithm allows warping of the signature to create a better fit between components, but little evaluation of the method was performed.

A different method for correlation involved extracting accelerations and their orientations (expressed as θ values), along with minimum, maximum, average and variance values [16]. Histograms were built for each feature extracted and a reference histogram was obtained by averaging the histograms from the five provided reference signatures. Comparison of test signatures involved performing several alignments of the histograms and choosing the one that produced the highest correlation. This same alignment is then used to align all of the remaining feature histograms, and an overall correlation was computed. Other user's genuine signatures are used as zero-effort forgeries and

error rates of 1.2% FRR and 1.0% FAR are reported. It should be noted that the accuracy of the system is significantly elevated due to the test/verification signatures also being used in the training set.

A large scale comparison of different correlation techniques was performed in [112] along with a comparison of three fundamental data types. Position, velocity and acceleration data are all considered using the techniques of regional correlation, dynamic time warping and skeletal tree matching. Segmentation of the signatures is performed by extracting fragments of 0.7 seconds, based on a claim that handwriting signals tend to fall out of phase beyond this time. The regional correlation method is similar to that mentioned above and used in [51]. Time warping is a nonlinear correlation technique borrowed from speech recognition and is used to map segments from a test signature to segments of the corresponding reference signature by removal of timing differences. The distance between two samples is then computed between the adjusted segments. Tree matching is a technique that involves building a "tree" of peaks and valleys of signature features such as position or velocity. Once the trees corresponding to the test and reference signatures are available, the distance between them may be computed in terms of minimum number of operations needed to transform one tree into the other.

Each of the three techniques uses all the values of position, velocity and acceleration in the x and y directions. A number of important results were reported in this study, which used fifty signatures from each of thirty-nine signers. The authors found that data in the y direction is more discriminating that data in the x direction. This could well be because, as noted earlier, most United States-style signatures tend to have much more variation in the y direction. Additional findings from the authors included a ranking of the utility of different types of data, placing velocity as the most discriminating (resulting in the lowest average error rates) followed by position and finally acceleration. The effect of permitted time lags in regional correlation is not found to be significant whereas the length of segments is. Best results are obtained for 1.5 second segments if $y(t)$ values are being compared. No correlation technique proves to be globally superior to the other two, however regional correlation is much faster and on average, slightly more accurate. Similar findings are reported elsewhere in the literature [9].

Further work by these authors was presented in [100] where they investigated the discriminating power of different types of writing samples. This study examined the utility of handwritten signatures compared with handwritten

passwords and handwritten initials from each of thirty-nine users. Comparisons were made as per the previous study, using position, velocity and acceleration data. As expected, signatures were found to be most user-specific and discriminatory due to the increased practice the users had in signing, with handwritten passwords next best, followed by initials. The lack of discriminatory power in initials is somewhat expected due to the fact that less information can be extracted from the smaller writings.

A more recent investigation into the use of general handwriting versus handwritten signatures as an authentication tool was described in [14]. This study found that there was a much higher variance between different writer's signatures (measured at 0.02481) than different writer's handwriting (0.00824). Additionally, there was a higher variance between different instances of the same writer's signature (0.00097) than different instances of the same writer's handwriting (0.00035).

Direct Feature Value Comparisons

A classical approach to feature value comparison is presented in [26], where a strain-gauge instrumented pen was used to sample x and y position values as well as pen-tip pressure. From this information, forty-four features were extracted for use in the system. After some experimentation and optimal subset of twenty-three of these features were chosen. During an enrolment/registration phase, ten sample signatures were collected and a reference formed by computing the mean and standard deviation of each feature. Test signatures were compared to the reference and the Euclidean norm of the distance array was computed and compared to a threshold in the usual way. The FAR and FRR each vary from 0.5% to about 3%, but the experimental setup leaves something to be desired in some areas (such as allowing three attempts at verification, and rejecting over 5% of the users as their signatures are inappropriate for use with the system).

Another basic feature-based system was reported in [81] which used seven geometric and dynamic features that are claimed to be invariant under rotation and scaling. The features include the number of handwriting components (that is, the number of pen-ups plus one), number of loops, total duration (it is not clear if this is invariant under magnification as claimed) and mean and maximum velocities. The Euclidean norm was used in the same way as described above to obtain the distance between test signatures and the reference. This distance was compared to a global threshold and resulted in a

6.4% FRR (no forgeries were used).

Further analysis of basic HSV features appears in [46] with the aim being to use a set of features that are easy to compute, invariant under most two-dimensional transformations (like rotation and scaling), global in nature (less susceptible to local variations), on-line/dynamic in nature (more difficult to forge) and few in number. Six features were used in the initial system including total duration, x and y components of velocity and acceleration, path length and total pen-up time (the amount of time the pen is *off* the paper during the signing process is immeasurable to a potential forger). Mean and standard deviations are computed for each feature extracted from signatures in the reference set. Test signatures were verified by finding the difference vector Z between test feature values and the reference means normalised with the standard deviations, and then taking the norm of vector Z. The test signature was rejected if the norm was larger than a global threshold, otherwise it was accepted as genuine. Building a reference using ten sample signatures resulted in a FRR of around 0.5% with a FAR of just above 10%.

This same technique was used to evaluate the discriminative power of individual features. The authors found that duration was the single best discriminator (2.6% FRR and 16.6% FAR), followed by pen-up time (6.1% FAR with 20.9% FRR) and mean velocity (7.2% FAR with 21.1% FRR). Other experiments were carried out to evaluate the number of sample signatures required to form a useful reference, with three being to be of little use, five producing acceptable results and ten samples signatures found to be very useful. Another interesting finding of the work includes the fact that a surprisingly large number of forged signatures had feature values that were more than twenty standard deviations away from the reference mean.

Spline smoothing is used to fit a smooth curve to irregular samples and is used frequently in signature verification. One of the earlier applications of spline smoothing in HSV was performed in [95] using the analysis described in [49]. Pen velocities and accelerations in x and y directions were computed using spline smoothing and further computations found the path velocities and accelerations as well as path tangent angles, tangential accelerations, centripetal accelerations, jerks and curvatures. Amongst their findings, the authors note that both pen-tip pressure and velocity have highly repeatable patterns in valid signatures of a signer. An interesting point is made that the shape and dynamics of a signature might play complementary roles in HSV since if a forger is trying to get the shape right, they are unlikely to get the dynamics

right and vice versa.

A major study in [74] attempts to design an effective on-line system for use in point-of-sale applications. A comprehensive database of 5,603 genuine signatures from 105 human subjects and 4,762 forgeries was developed. 240 of the genuine signatures were "fast signatures" in which nine subjects were asked to sign as fast as they could resulting in writing duration being 10% to 50% lower than their normal writing duration. 210 of the genuine signatures were obtained from seven subjects standing up, each of them providing 20 signatures when signing at a height of 90 centimetres and another 10 at a height of 60 centimetres. The forgeries consisted of zero-effort, skilled and timing forgeries.

A feature set of forty-two features is described that includes thirteen static features. A number of features related to the velocity of signature production are also present including mean and maximum velocity, duration of accelerations as well as x and y components of the velocity. A set of forty-two features was used initially, but this was reduced to an optimal thirty-four features during training. Among other results, a best-case equal error rate of 3.8% is reported using skilled forgeries in training (although obviously these will rarely be available).

A comparative study of 210 features for off-line HSV was performed in [27] in an attempt to find the best feature set for HSV of Arabic signatures. The methodology followed was to calculate the mean, standard deviation and the spread percentage for every feature for every user in their database and select those features that have a spread of less than 25% for each user. Many of the selected features were found to have very small interclass variations and were excluded. The feature set was ultimately reduced to twelve features and a fast back-propagation neural network method was used to perform the classification. A FRR of 1.4% is reported, however testing did not include any skilled or zero-effort forgeries.

A different discussion related to selection of best features is presented in [37]. The authors describe a simple approach of selecting the best features based on the individual performance of each feature (this makes the assumption that the features are independent which often is not true). A technique called sequential forward selection (SFS) was recommended in which one feature is added each "round" by determining which single feature, when added to the existing set, most significantly improves the results. The process can be terminated if no improvement occurs, if a pre-specified performance is achieved or if a pre-specified number of features is obtained. This technique is effective,

but doesn't account for the differences that would be experienced using a different ordering of feature presentation, or for inter-relationships between features.

A comparison of basic distance metrics was the focus of work in [96], which tested the Euclidean distance metric, the Mahalanobis distance method and the quadratic discriminant method in correlating a set of twenty-five timing, velocity and angular features. It is worth noting that the Mahalanobis method [121] is invariant under coordinate scaling, eliminating a peculiarity of the Euclidean method. The authors impose an ordering on the various features by ranking them using the ratio of standard deviation to mean for each. Smaller values for this ratio meant that features would be ranked higher. A variety of schemes are evaluated using the best eight, ten, twelve, fourteen and all twenty-five features and the different distance metrics. The affect of the different feature set sizes was minimal, with set sizes of eight and ten resulting in a slightly lower FRR. The quadratic distance model performed better than the Euclidean and the Mahalanobis methods with the major drawback of requiring forgery data during training. Realistically, forgeries are not always going to be available and of the two remaining techniques, the Euclidean measure provided slightly better results. The authors also examined a majority voting model described in [74], but found that this performed worse than using just the Euclidean distance. Error rates of 0.5% FRR and 14% FAR were obtained using the optimal system parameters.

2.3.1 Combining Local and Global Features

A technique involving both a local matching method and a global interpretation of the scene appears in [132]. Approaches such as this are perhaps expected to exhibit superior performance as they take into account a more complete picture of the signature than approaches concentrating on just global or just local features. This off-line system involved segmenting the signature image into a collection of arbitrarily shaped primitives and then template matching to perform the local comparison. The global comparison involved obtaining distance values using the signature image as a whole. Results of 1.5% FRR and 1.37% FAR are reported.

Another discussion of off-line HSV based on extracting global and local features of a signature image appears in [116]. The global features used in the study were the height and width of the signature's bounding box, width

of the signature excluding blank spacing between signature components, slant angle of the signature, vertical centre-of-gravity, maximum horizontal projection, area of black pixels and baseline shift of the signature. Local features were based on grid positions and included the structural information of image elements, for example curvature of arcs, intersection between line strokes, number of pixels within each grid position and angular information. Testing using 450 signatures from 25 users and some basic forgeries (produced by other users in the database) resulted in a FRR varying from 3% to 11.3% and a FAR varying from 0% to 15%.

Foundational studies on representing large amounts of handwritten text as a texture was described in [104]. These initial studies involved script and language identification and are extended to handwriting authentication in [133]. The basic approach is to convert an image of a large block of handwritten text into a texture and use texture analysis techniques to compare different handwriting samples. The method showed robustness with respect to noise and the presence of foreign characters or numerals, resulting in less than 5% error in language identification. The advantages of using this approach to handwriting authentication are that the written text is not fixed (for example to a signature or password), forgery is much more difficult and any standard texture recognition algorithm (for example, the multi-channel Gabor filter) may be used. The system correctly identified the author of the handwriting in 95.3% of cases, based on a testbed of 300 documents from twenty writers and using ten samples to build a reference. The drawback of this approach is that it is unable to perform verification/identification using small samples such as signatures, handwritten passwords or pass-phrases. It does seem quite an attractive approach for larger blocks of texts (handwritten letters, archived handwritten documents etc.) as the identification is independent of the actual text written.

In the last five to ten years, researchers have been doing more work on recovering on-line information from off-line image [30]. This process of recovery attempts to include information about the *process* that gave rise to the markings on the page, such as the time sequence of the strokes, pressure exerted by the instrument or basic indicators of velocity and acceleration. These attributes are parameters of the writing process, but are not clearly visible in the image. By closely examining local anomalies such as *striations* (small marks left by ball-point pen ink when writing), it is possible for the authors to obtain direction and timing information. It is not clear the extent to which this

information can be exploited for use in HSV, but is encouraging nonetheless.

The first in a series of writer identification papers, [167] discusses an approach to off-line writer identification using *moments* (in an attempt to measure force characteristics of the signature). The authors firstly pre-processed the handwriting image using thresholding and thinning and used a projection of the words onto the vertical axis to obtain a projection function $p(x)$, normalised to be the same length for each word. The next step was to use morphological openings to obtain a ten component feature vector to use in writer identification. The second feature vector was based on the simple first central moments obtained on a local basis. The one-dimensional moment function is derived from a moving window (left-to-right), and the resulting function is processed in a similar way to $p(x)$, resulting in a second feature vector. Handwriting specimens were collected from twenty writers, each writer contributing four given (Greek) words written eight times each on a pre-segmented form. Training was done using a simple neural network trained using the back-propagation algorithm. Unfortunately, few details are reported as to the actual error rates resulting from the system, other than the authors claim that the technique is able to separate and discriminate "completely" among the clusters.

An example of different interpretations of error rates can be seen in [157]. An interesting methodology for off-line handwriting analysis is described including signature alignment, investigations of algorithm susceptibility to different transformations and a method for estimating the level of distortion between two aligned signatures. Experiments are conducted on both signature verification and writer identification with impressive error rates reported (for example, under 6% error for content-independent writer identification) using a database of 45 signers and 2,384 signatures. It should be well noted however that the authors defined a "correct" match to occur when the actual writer was in the top nine of the candidates chosen by the system. This measure of success differs from most other error rates reported in the literature.

There are however other aspects of the study provoking interest. For example the authors use forty features in total, including area, height and width of character components, aspect ratio, area, height and width of the characters themselves, mean number of components per character etc. The features are designed to be dimensionless and independent of the resolution of the image. Some texture features are also used, specifically the power spectrum computed from the Fourier transform of each character. These are summed over six rings (for rotation independence) of equal area and four double wedges (for scale in-

dependence) of similar area. Nearest neighbour classification is used to predict the authors.

As a point of conclusion, it should be noted that it is very difficult to perform an accurate comparison of HSV techniques presented in the literature. Some of the main reasons for this are:

- Systems that use different individual threshold values for different users are generally better than systems that use the same threshold value for all users;
- Different systems reported in the literature use a different number of sample signatures to build a reference;
- Some researchers include the sample signatures (used to build the reference) in their test set for performance evaluation, giving an elevated impression of the accuracy of the system;
- Some researchers use skilled forgeries to train the system and create the reference, while others do not. The assumption of the availability of skilled forgeries is not valid as it is not possible to obtain these in a real-world application;
- The most significant and fundamental difference throughout the literature is that different researchers develop their own data sets, collected in different conditions using different writers, different forgers, different number of signers etc. The use of a common data set would go much of the way to facilitating accurate comparisons between systems (note that the database from [32] is available for academic use).

Chapter 3

Neural Networks

Neural networks (NNs) have been a fundamental part of computerised pattern recognition tasks for more than half a century and continue to be used in a very broad range of problem domains. The two main reasons for their widespread usage are power (the sophisticated techniques used in NNs allow a capability of modelling quite complex functions) and ease of use (as NNs learn by example it is only necessary for a user to gather a highly representative data set and then invoke training algorithms to learn the underlying structure of the data).

Neural networks (which have a number of aliases such as "artificial neural networks") have much of their origins based in their biological counterparts. Although the workings of biological neural networks and the human brain remain largely a mystery, it is known that the *neuron*, or nerve cell, serves as the fundamental functional unit in the system [130]. The brain is principally composed of a very large number of neurons (of the order of 10,000,000,000), massively interconnected, and each neuron is a specialised cell that can propagate an electrochemical signal. Each neuron also has a cell body, a branching input structure (known as dendrites) and a branching output structure (the axon). The axons of one cell connect to the dendrites of another via a synapse. Each neuron forms synapses with anywhere from one dozen to one hundred thousand other neurons.

One of the most significant findings in research into these biological neural networks is that the synaptic connections exhibit a property of *plasticity* - that is, there can be long-term changes in the strength of connections in response to patterns of stimulus. Neurons can form new connections with other neurons, even to the point of entire collections of neurons migrating from one place to another. These mechanisms are thought to form the basis for learning in the brain [130]. Thus, from a large number of extremely simple processing units

(each performing some kind of simple operation based on the received inputs), the brain manages to perform extremely complex tasks such as learning and thought. Of course, this is a gross oversimplification of the human neural structure, but it is interesting that artificial neural networks can achieve remarkable results using models based very much on this system.

The history of artificial neural networks can be traced back to initial work in developing individual neurons in the 1940s [87, 107]. The model developed consisted of two inputs with equal input weights and a single binary output. If the summed inputs exceeded a certain threshold level, the output would be set to one, or zero otherwise. This basic model as become known as a *logic circuit*. In the 1950s, Rosenblatt presented the *perceptron* as one of the first major neural models of computation [125]. The perceptron idea was augmented by the use of a weight space introduced in [136]. The weights here are adjusted by randomly choosing a direction vector. If the performance does not improve, the weights are returned to their original values and a new random direction vector is chosen. At around the same time, many other researchers were performing their own work in neural network systems such as linear associative networks, boolean networks and adaptive linear neurons [139, 159, 156]. In 1969 neural network research came to an almost complete (temporary) halt due to the Minsky and Paper book *Perceptrons* [90]. This work illustrated the limitations in the representative capabilities of one-layer perceptrons (these limitations however do not apply to more complex multi-layer networks).

The period of time between the mid-1970s and mid-1980s represented a mini-renaissance for neural networks with the several variations of the powerful back-propagation learning algorithm being independently developed by several researchers, for example [153, 45, 101, 129]. This algorithm remains one of the most used in neural network research and indeed in machine learning and is the algorithm used as part of the NN-based signature verification system presented in this thesis. Back-propagation is described in detail in Section 3.1.3.

3.1 The Theory of Neural Networks

A neural network is made up of a number of *nodes* (also known as *neurons* or *units*) connected by links. Each link has a numeric weight value associated with it, which is the primary means of long-term storage in NNs, and learning usually takes place by updating the weights. Some of the units are connected

to the external environment and can serve as input or output units. The basic learning algorithm of the NN involves modifying the weights so as to try to bring the network's input/output behaviour closely into line with that which is expected.

Each neuron has a set of input connections from other units and a set of output connections to other units, a current *activation level*, and a means of computing a new activation level based on the input values and the current weights. The idea is that each neuron has its own individual behaviour and contributes to the overall behaviour of the network.

The earliest decisions of a network designer are to decide on the number of neurons, what *kind* of neurons and topological network structure is to be used, and how the neurons are to be connected to form the final network. These decisions are not trivial to make and are often based more on experience and long-term trial-and-error than any definitive set of heuristics. The network designer must also decide on the training (or learning) algorithm to use to teach the network. To further complicate the job of the network designer, many of these decisions cannot be made independently and a change to one aspect of the network often necessitates changes elsewhere.

In addition to decisions made about the structure and training of the network itself, the selection of which of the available parameters or features to use as inputs to the NN poses a difficult problem. This difficulty arises for a number of reasons:

- In real problem domains, a neural network is typically employed where the user does not have a strong idea of the relationship between the available variables and the desired prediction. It is therefore likely that a variety of data is accumulated, with some suspected to be important and some that is of dubious (but unknown) value.

- In many problem domains there will be interdependencies and redundancies between parameters; for example, a pair of parameters may be of little value individually, but extremely useful in conjunction, or any one of a set of parameters may be useful. These aspects mean that it is often not possible to simply rank parameters in order of importance.

- The "curse of dimensionality" means that the size of the network takes on considerable importance and in some situations is it better to discard variables with small genuine information content (particular if they are closely coupled to other informative features) in order to limit the size

and complexity of the network. The lesser complexity in the network can often improve the network's generalization capabilities [13].

The only method that is guaranteed to select the best input set, is to train the network with all possible input sets and all possible architectures, and to select the best performer. In practice, this is impossible for any significant number of candidate inputs.

Later sections of this chapter present a discussion of the various design decisions, particularly with regard to networks topologies and their applicability to HSV systems. A detailed discussion is presented on the multi-layer perceptron - the network structure ultimately selected for use in the HSV system. The features used are also described in detail in Section 3.4.3.

3.1.1 Learning in Neural Networks

One of the key features of neural networks is that they learn to solve problems by example. Rather than setting the weights in the network directly (although this is possible), it is only necessary to provide a set of sample *training* data. A learning algorithm can be used to adjust the weights in the network so that it learns to recognise the training data and appropriately associate sample inputs with desired outputs.

As discussed earlier, the process of learning in a neural network consists of updating the numeric weight values associated with each of the links between neurons. While this procedure in biological neural networks is still only vaguely understood, the general learning procedure in artificial neural networks is well-defined and is discussed abstractly in this section (of course different network structures and approaches will use specialised versions of this general approach).

Each neuron has a set of input links, an associated set of weights and a set of output links. Some of the neurons are connected to the external environment, and can be designated as input or output units. Input units generally take data from the environment (for example, in the form of features describing an object - in the case of this work the object is a signature) and output units provide output from the network to the external environment (in this case the output is typically a confidence value indicating the believed validity of the input signature). Each neuron performs a local computation based on the inputs and without the need for any global control over the set of neurons as a whole. The result of this computation is then passed to other elements through

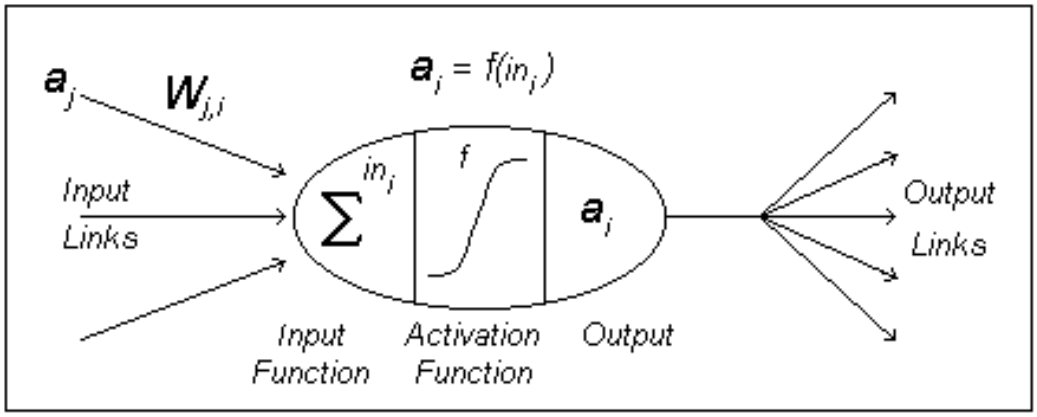

Figure 3.1: *The general structure of a neuron.*

the neurons output links and is often referred to as the *activation value* of the neuron.

The basic structure of a neuron in a NN is illustrated in Figure 3.1 (based on the figure in [130]). In this figure, a_j refers to the input activations linked from neuron j, $W_{j,i}$ refers to the weight associated with the link from neuron j to neuron i (the input to the node therefore is the product of the input activations and the associated weight, or $a_j W_{j,i}$). The input function is generally the sum of the product of the input values with their corresponding weight. The activation function f (sometimes known as the *post synaptic potential*, or PSP, function) is applied to this sum to produce the activation level a_i that serves as the output from the neuron.

The choice of the activation function used can have a marked effect on the operation of the network and the learning that goes on. It is the activation function that transforms the weighted sum into the neuron's activation value. There are a variety of activation functions that are commonly used in NNs and they each produce different model behaviour. Some useful work has taken place in developing adaptive activation functions [160] however these are relevant to higher order neural networks and aren't investigated here. Each of the functions used during experimentation reported in this thesis are briefly described below. Note that NNs with any continuous non-polynomial activation function are universal approximators [56, 57, 75].

Linear activation function: the activation level is passed on directly as the output. This is used in a variety of network types, including linear networks and sometimes in radial basis function networks (see below for explanations of these network types).

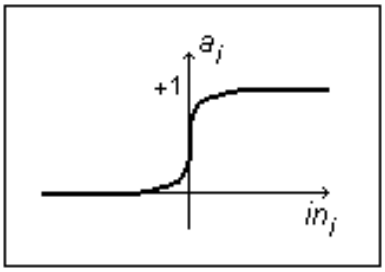

Figure 3.2: *The sigmoid activation function.*

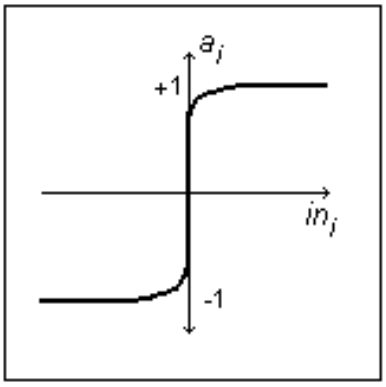

Figure 3.3: *The hyperbolic activation function.*

Sigmoid: also referred to as *logistic*, this is a sigmoid (S-shaped) curve, with output in the range [0,1]. This is the most commonly-used neural network activation function and can be seen graphically in Figure 3.2. It is represented by the function:

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

The sigmoid function is used in most of the NN structures explored in this chapter.

Hyperbolic: the hyperbolic tangent function (tanh) is identical to the sigmoid curve, except that output lies in the range [-1,+1]. This can often perform better than the sigmoid function because of its symmetry [13]. The function is illustrated in Figure 3.3 and the formula that implements it is:

$$hyperbolic(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Square root: used to transform the squared distance activation in a Kohonen network to the actual distance as an output, obviously given by the formula:

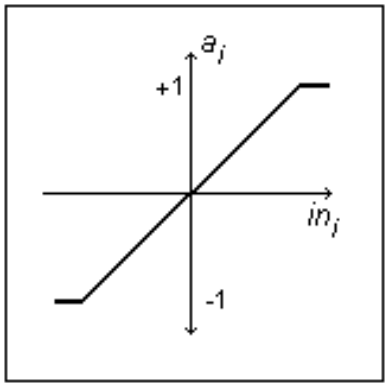

Figure 3.4: *The ramp activation function.*

$$square\ root(x) = \sqrt{x}$$

Ramp: using this approach, the value of the activation level is limited to be between -1 and +1 before being passed on as output. During experimentation, the *ramp* function did tend to execute slightly faster than others such as *sigmoid*, but resulted in poorer training performance. This function is shown graphically in Figure 3.4 and has the basic definition:

$$ramp(x) = \begin{cases} -1 & x \leq -1 \\ x & -1 < x < +1 \\ +1 & x \geq +1 \end{cases}$$

Step: this function includes a threshold t and outputs a 1 when the input is greater than the threshold and outputs a 0 otherwise. This threshold is therefore representative of the minimum total weighted input necessary to cause the neuron to fire. In practice, the threshold is often replaced with an extra input weight that allows for simpler learning as it is only necessary to adjust weights rather than adjusting both weights and thresholds. *Step* functions are used in many simple NNs. The function appears in Figure 3.5 and has the following definition:

$$step(x) = \begin{cases} 0 & x < t \\ 1 & x \geq t \end{cases}$$

Sign: this is a slight variation on the *step* function in which the threshold t is essentially set to 0 and the output range is adjusted to [-1,+1]. See Figure 3.6 and the definition:

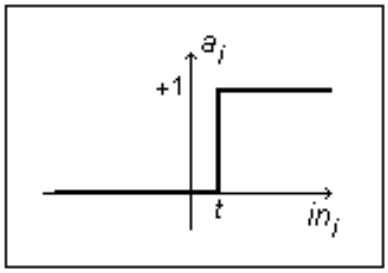

Figure 3.5: *The step activation function.*

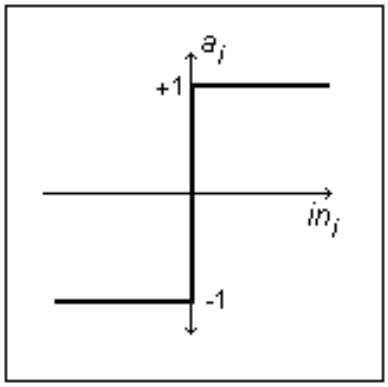

Figure 3.6: *The sign activation function.*

$$sign(x) = \begin{cases} -1 & x < 0 \\ +1 & x \geq 0 \end{cases}$$

The choice of the most appropriate activation function is usually based on a combination of knowledge of the application area, experience and trial-and-error. Once the structure of the individual neurons is fixed, the focus of effort for a network designer is on the specific structure, or topology, of the NN itself.

There are a variety of network structures, each resulting in very different computational properties and behaviours. The primary distinction to be made is between what are known as *feed-forward* networks and *recurrent* networks. Feed-forward networks are fundamentally acyclic - links are unidirectional and connect in a direction away from the input neurons towards the output neurons, with neurons able to be arranged into layers. In recurrent networks it is possible to create arbitrary topologies with cycles. All of the network structures considered in the NN experimentation described in this thesis are feed-forward networks rather than recurrent. This decision was made for a number of reasons:

- Computation in feed-forward networks is much more uniform and proceeds from input neurons to output neurons;
- Recurrent networks can have a tendency to become unstable, and obviously learning is made more difficult in this environment;
- Feed-forward networks alleviate the need to keep track of an internal state (other than the weights themselves) as is necessary in a network that contains cycles;
- Feed-forward networks are better understood than recurrent networks and still have significant representative power - certainly sufficient for use in signature verification.

For these reasons, there will be little further discussion on recurrent networks (although significant in their own right) and most of the NN discussion will centre on multi-layer feed-forward networks.

Within feed-forward networks there are several variations, the most fundamental of which is perhaps the number of *layers*. Regardless of the chosen topology, there are common elements to the layers in a multi-layer network. There is always a set of input units, the activation value of which is determined by the environment, and a set of output units (often this can be a single unit, such as in signature verification networks). In between these two sets there can be a number of *layers* of neurons that have no direct connection to the environment. These are called *hidden units* because they cannot be directly observed by noting the input/output behaviour of the NN. NNs with no hidden layers are sometimes referred to as *single-layer perceptrons* or simply *perceptrons* (depending on the terminology being used) and networks with one or more hidden layers are referred to as *multi-layer networks*, or often *multi-layer perceptrons* (MLPs). With one sufficiently large layer of hidden units, it is possible for NNs to approximate any continuous function of the inputs; with two layers, any discontinuous functions can also be approximated [130]. Figure 3.7 shows some examples of different kinds of NN topologies.

It turns out that $2^n/n$ hidden units are required to represent all Boolean functions of n inputs [50]. In practice, most problems can be solved with far fewer weights. The design of an effective topology is however not an exact science and often involves a great deal of trial-and-error experimentation.

The computational efficiency of a NN depends on the amount of processing required to train the network to fit a given set of examples. If there are m

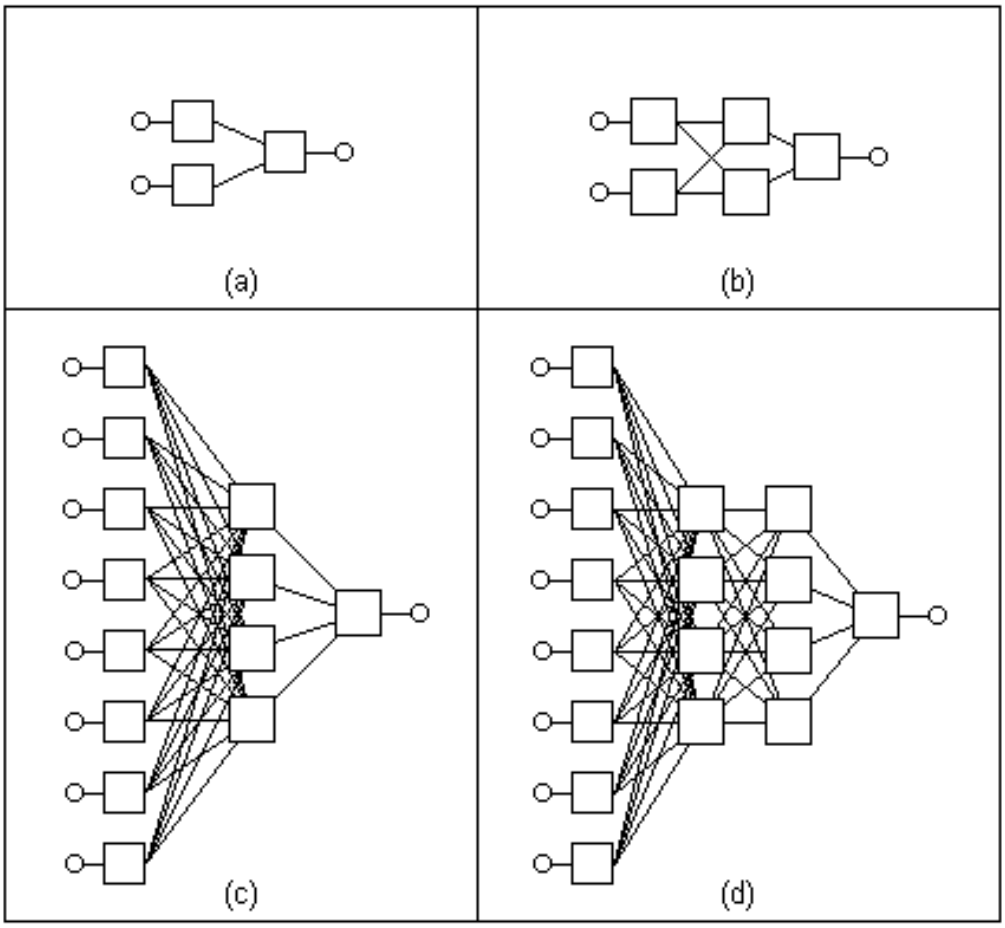

Figure 3.7: *Various feed-forward network topologies. (a) A simple two-input, one output network with no hidden layers. (b) A two-layer network with two inputs, two hidden nodes and one output node. (c) A more complicated network consisting of eight input nodes each connected to four nodes in the hidden layer and a single output node. (d) A network similar to that in (c) except with two hidden layers.*

examples, and $|W|$ weights, each epoch takes $O(m|W|)$ time. In practice, time to convergence is highly variable, and a large array of techniques have been developed to try to speed up the process using an assortment of tunable parameters.

Other researchers have recently discussed means of using prior knowledge to "prime" a network to better learn the given function [154, 68, 165]. Although much of this research is promising, there are still some fundamental obstacles with regard to use of prior knowledge. Primarily, because of the lack of transparency and difficulty in comprehending the underlying semantic meaning of weights and activations in complicated multi-layer NNs, it is quite difficult to attach any intelligent prior knowledge to the network. It is only in rare domains that this is possible.

Once a network has been designed and built, computing the output of feed-forward networks is not a difficult task, it is simply a matter of performing the basic operations within each neuron in the network. The training or learning phase is obviously far more challenging. The fundamental approach to learning in NNs is to modify the weight values so as to try to bring the network's input/output behaviour more into line with that of the environment providing the inputs. To put it another way, during training there is an expected set of outputs for each set of inputs and any algorithm used is responsible for aligning the actual output of the NN as closely as possible with the expected output.

For example, the back-propagation learning algorithm (see Section 3.1.3) works by iteratively training the network using the available data. On each iteration (or *epoch*), the inputs are presented to the network, which is executed to produce output values. The output values are compared with the desired outputs present in the data set, and the error between the desired and actual outcome is used to adjust the weights in the network so that the error is likely to be lower. The algorithm must (as with any learning algorithm) compromise between the various cases, attempting to alter the weights to minimise the error over the database as a whole.

A standard technique in neural networks is to train using one set of data (called the *training* or *reference* set), but to evaluate the performance against a *verification* data set not used in the training phase. The effect of this is that it provides an independent check that the network is actually learning something useful. This experimental setup is known as *cross verification* (and is the approach used wherever applicable throughout this thesis). Without

cross verification, a network with a large number of weights and a small amount of training data can *overfit* the training data - that is, it is likely to model the noise present in the data, as well as the underlying function. The ability of a network to not only learn the training data, but to perform well on previously unseen samples, is known as *generalization*. It is worth noting that testing of this sort is often be impractical due to the larger data requirements.

In *classification*, the input cases to the network are assumed to represent measurements of some features of an object. The aim is to decide to which of a number of classes the object belongs. Signature verification is a two-class pattern verification problem with one class denoting the genuine signatures and the other class denoting the forgeries.

One of the significant practical considerations when training any network is when to stop training. Ideally the network training is observed and the process continues until the performance starts to deteriorate. However it is often not practical to maintain this close observation so it is necessary to apply some sort of *stopping condition* that causes the training to cease when certain criteria are satisfied.

The simplest form of stopping condition is to train for a particular number of epochs (for example, "stop training after one hundred epochs"). Due to the fact that most domains contain training examples that are unpredictable, the number of epochs required to sufficiently train can vary greatly. Using a maximum epoch value for the stopping condition often results in over-training (this is when the learned function fits the training data too closely and as a result displays poor generalisation) or under-training (when the learned function does not fit the training data well enough and classification performance suffers).

Another simple stopping condition is to use a target error rate, for example stopping training when the error rate drops to 1%. There are obvious problems with this approach in that there is no guarantee that the training or verification error will ever drop to the desired level, and it is difficult to predict what the ultimate error rate will be.

A more useful stopping condition is based on a minimum improvement level. This states that if the classification error does not improve by at least some given amount over a set number of epochs (known as the *window*) then training should stop. This allows a tradeoff in the training process so that the network will continue training as long as there is a reasonable improvement and will stop when a plateau has been reached. One problem with using

minimum improvement level as a stopping condition is that often the error rate may fluctuate during training, sometimes rising slightly only to fall again afterwards (the classic "local minima" problem). In order to account for this it may be necessary to experiment with the size of the window or the size of the error deterioration. For example, setting the window size to ten epochs will only stop training if the error rate deteriorates and then fails to improve beyond the current minimum for ten epochs.

Often, it is not only important to obtain the best possible training or verification error - it is also important to create the smallest possible network (*Occam's razor* says that simpler models should be preferred over more complex ones and that this preference should be traded off against the error rates obtained). This factor can be taken into account when searching for the optimal model structure by associating a *unit penalty* with each unit in the network. Use of unit penalties in network design gives some preference to smaller networks.

The exact training procedure differs for different network structures and each of the structure options is now discussed in detail.

3.1.2 Linear Networks

Linear networks were originally studied in the early 1950s under the name *perceptrons* [125] and are based on the concept of *linear separability*, in which groups or clusters of data can be separated by a line (or more generally a hyperplane). For example, consider the classic ballet dancer/rugby player example shown in Figure 3.8. This figure portrays a geometric representation of a feature space consisting of weight and height of human beings, with the values for ballet dancers and rugby players plotted. As can be seen, these two groups form two distinct clusters and can be easily separated by a single line. That is, the two groups are *linearly separable*.

Linear networks attempt to solve classification problems such as the one described above by drawing a single hyperplane between classes. Structurally they are actually perceptrons (see Section 3.1.3) with only an input layer consisting of n neurons (one for each feature) and a single output neuron. Each of the input neurons is connected to the output unit and has an associated weight. The output neuron then performs a basic operation with the weighted inputs (generally summation then application of an activation function) and outputs a classification. Figure 3.9 depicts one way of looking at the basic

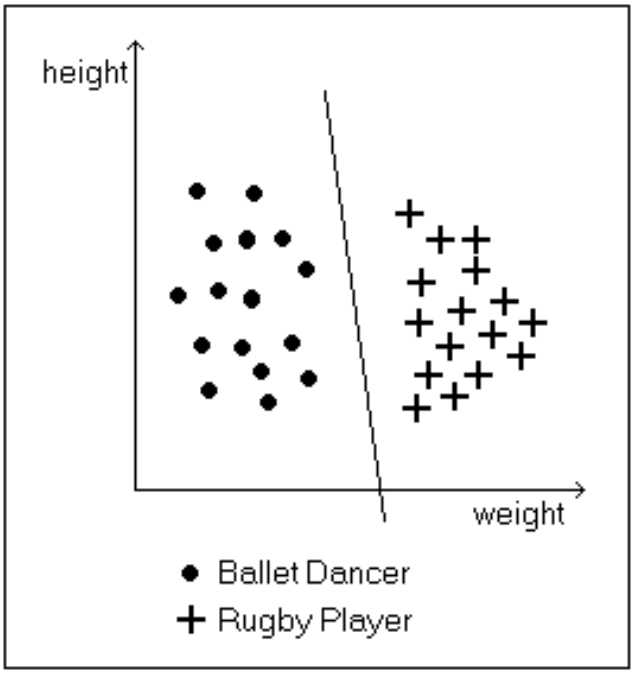

Figure 3.8: *A linearly-separable feature space.*

structure of a linear network.

There is a major limitation with linear networks in that they can only represent a limited range of functions (famously proven in [90]). For example, consider the illustrations in Figure 3.10. As can be seen in part (a) and part (b), the `AND` and `OR` functions are linearly separable but the `XOR` function is not. A more powerful representation is needed (such as a *multi-layer perceptron*) to perform classification in such an environment.

Despite these limitations, linear networks are useful for two reasons: firstly, it is not greatly uncommon to find that a problem, previously perceived to be difficult and non-linear, can actually be solved adequately by linear techniques. Secondly, a linear model provides a useful benchmark against which to judge more complex techniques. Results of HSV presented later in this thesis include linear approaches.

Learning in Linear Networks

Learning in linear networks (as with most other neural network learning algorithms) is done using a *current-best-hypothesis* search. The idea behind this is to maintain a single hypothesis, and to adjust it as new examples arrive in order to maintain consistency. In this case, the hypothesis is the set of weights

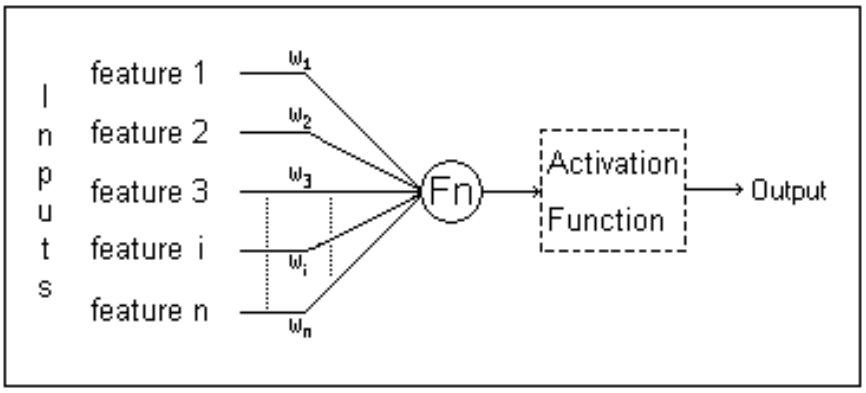

Figure 3.9: *A linear network.*

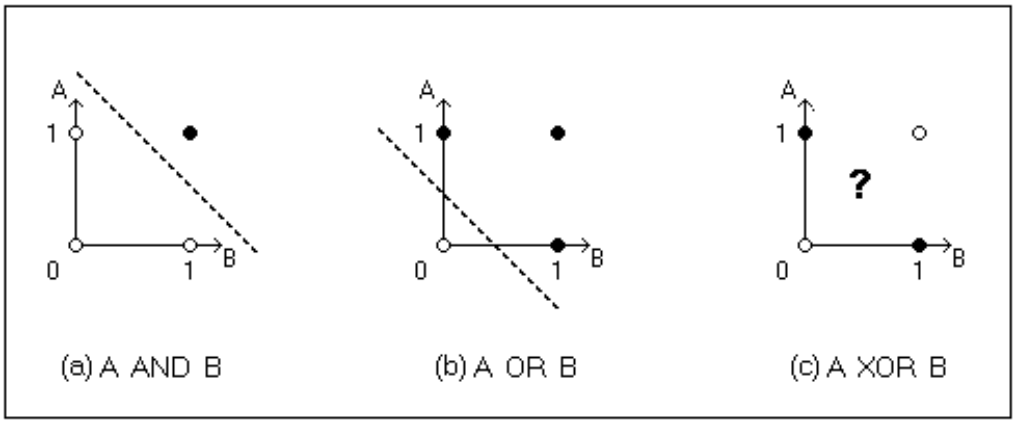

Figure 3.10: *Linear separability of Boolean functions - the axes represent the input values and the dots represent the output (a solid dot is a 1 and a hollow dot is a 0). (a) The* `AND` *function which is linearly separable. (b) The* `OR` *function which is linearly separable. (c) The* `XOR` *function - it is not possible to draw a single line to separate the classes.*

in the linear network.

The initial weights are typically randomly assigned and adjusted in an update phase when presented with examples. The update phase is repeated several times until the network achieves some kind of convergence.

In linear networks, the weight update rule is quite simple. The difference (error) between the predicted output O and the correct output T is given by:

$$Err = T - O$$

If the error is positive then O needs to be increased and if the error is negative then O should be decreased. Now, assuming that input j has the value I_j and is associated with weight W_j, then that input unit contributes a total of $W_j I_j$ to the overall input. Therefore, if I_j is positive, an increase in W_j will tend to increase O and if I_j is negative, an increase in W_j will tend to decrease O. The updating of weight W_j can then be performed as follows:

$$W_j = W_j + \alpha \times I_j \times Err$$

where α is a constant called the *learning rate* which can be used to control how quickly or slowly a network learns (more information on the learning rate appears in the next section).

This was proven to converge to a set of weights that correctly represents the examples for *any* linearly separable function [126].

3.1.3 Multi-Layer Perceptrons

There is some difference in terminology across different publications and different authors as to what constitutes a perceptron versus a multi-layer perceptron. To clear up any confusion, throughout this thesis, the term "multi-layer perceptron" refers to a feed-forward network with at least one hidden layer.

Multi-layer perceptrons are well suited to the task of recognising characteristic feature sets. This is highly relevant to the parametric version of the signature verification task (see Section 3.4) in that a characteristic feature set is extracted from the signature, and the MLP attempts to "learn" those characteristic features. Additionally, MLPs are known to be able to compute nonlinear mapping functions [79], so they can handle the nonlinear distortions found in human handwriting.

There are a number of different approaches to training MLPs but unfortunately none of the learning algorithms guarantee convergence to a global

minimum, and indeed do not guarantee efficiency or any level of accuracy. However, it is known that a feed-forward network with one hidden layer can approximate any continuous function of the inputs, and a network with two hidden layers can approximate any function at all [130].

The training of multi-layer networks proceeds in a similar manner to training a linear network, where examples are presented a single case at a time. The network computes the output value and if there is an error the weights are adjusted. The difficult part of this adjustment lies in deciding *which* particular weights to adjust. In multi-layer networks this is particularly difficult as there are many weights connecting each input to an output, and each of these weights can contribute to more than one output.

The following sections describe the different learning algorithms considered in this work, with a concentration of the most successful of them, the back-propagation learning algorithm.

The Back-Propagation Learning Algorithm

The most popular method for learning in a multi-layer network is the backward error propagation learning technique, or simply back-propagation. This algorithm is a sensible approach to dividing the contribution of each weight. The goal is still the same as in linear networks, that is, to minimise the error between each target output and the observed output generated by the network, but there are two main additions to the weight update process. Firstly, it is necessary to consider the activation value of the hidden unit a_j instead of the simply the input value when determining the weight update. Secondly, the formula also includes a term to incorporate the gradient of the activation function. So if Err_i is the error $(T_i - O_i)$ at the output node, then the weight update rule for the link from unit j to unit i is:

$$W_{j,i} = W_{j,i} + \alpha \times a_j \times Err_i \times g'(I_i)$$

where g' is the derivative of the activation function g with respect to the input I_i. For convenience a new error term δ_i is introduced, which is defined as $\delta_i = Err_i \times g'(I_i)$. The update rule then becomes:

$$W_{j,i} = W_{j,i} + \alpha \times a_j \times \delta_i$$

It is relatively simple to observe the output of the network, compare this to the target output and update weights in a network with no hidden layers.

In networks with hidden layers, this process is more difficult and involves the *back-propagation* of the output error back through the network. The process of updating the connections between the input units and the hidden units requires the definition of a quantity similar to that of the error term for output nodes. This is handled in back-propagation by the hidden node j being "responsible" for some fraction of the error δ_i in each of the output nodes to which it connects. To this end, the δ_i values are divided according to the strength of the connection between the hidden node and the output node, and propagated back to provide the δ_i values for the hidden layer. The propagation rule for the δ values is as follows:

$$\delta_j = g'(I_j) \sum_i W_{j,i} \delta_i$$

Now the update rule for the weights between the inputs and the hidden layer is almost identical to the update rule for the output layer:

$$W_{k,j} = W_{k,j} + \alpha \times I_k \times \delta_j$$

The back-propagation algorithm therefore works by iteratively training the network using the training data available. On each iteration (that is, epoch) the entire training set is presented to the network, one case at a time. The inputs are presented to the network, which is executed to produce output values.

The algorithm must compromise between the various cases, attempting to alter the weights so that the overall error across the whole training set is reduced. Because it works on a case-by-case basis, the overall error does not always decrease.

The back-propagation learning algorithm can be summarised as follows:

1. Compute the δ values for the output units using the observed error.
2. Starting with the output layer, repeat the following for each layer in the network, until the earliest hidden layer is reached:
   - Propagate the δ values back to the previous layer.
   - Update the weights between the two layers.

As an implementation issue, in the interests of efficiency, some of the intermediate values are saved to alleviate re-calculation. In particular it is useful

to cache the activation gradient $g'(I_i)$ in each unit, which speeds up the subsequent back-propagation phase considerably.

There are a number of other parameters included in variants of the back-propagation algorithm. Some of these are standard and some are non-standard parameters implemented to modify the learning process and hopefully improve the learning phase:

- *Learning Rate*: this has already been mentioned, but is defined explicitly here. It is usually represented as α, the "speed" at which the algorithm learns. A higher learning rate will cause the algorithm to converge faster with the trade-off being that this may introduce instabilities in performance for some problems, especially if data is noisy.

- *Momentum*: generally improves performance by speeding up training where there is little change in the error. A high learning rate should be complemented with a low momentum (and vice versa) to minimise the resulting instability.

- *Conversion Function*: this is the pre-processing function used in many network structures to normalise the data. *Minimax* is a simple and useful conversion function that assigns linear scaling coefficients for the data set. The minimum and maximum of the set are found, and scaling factors selected so that these are mapped to desired minimum and maximum values (usually scaled from 0 to 1).

In addition, when training the network it is often useful to change the order of presentation of training cases between each epoch. This adds some noise into the training process, which means that the error may oscillate slightly. As a result the algorithm is less likely to get stuck and the network's ability to generalise is usually improved.

Other aspects relevant to network performance, but not directly involved in the learning process:

- *Root Mean Squared Error*: abbreviated to RMS, this is calculated by summing the squares of all the individual errors, dividing by the number of errors and then taking the square root. This gives a single value that summarises the overall performance error.

- *Confidence Limits*: these are the accept and reject thresholds used in classification tasks to determine whether a pattern of outputs corresponds to a particular class. For example, in HSV it may be decided to

accept a signature as being genuine only if the network is more than 95% confident that it is genuine and reject only if it is less that 5% confident that it is genuine (values in between indicate the system is "undecided"). These confidence limits are applied according to the conversion function of the output variable.

In a multi-class classification problem, also known as *one-of-N* or *identification*, the confidence levels are still used. In a signature verification scenario, this may involve a single network with N units in the output layer being trained to identify the author of a particular signature. In this situation, in order for an output unit to be considered "on", the activation value must be greater than the *accept* threshold. Similarly, if a unit has an activation value of less than the *reject* threshold it is considered "off", irrespective of the relation of the activation to the other unit's activations. As is often the case in multi-class classification problems, it may be desirable to assign the output to the class corresponding to the "winning" unit (the unit with the highest activation), regardless of other settings in the network. In this case, the *accept* threshold would be set to 0.0.

As has been mentioned, computing the output of a MLP or any feed-forward network is not a difficult task, it is the training phase that is far more challenging and time consuming. This is an excellent fit for signature verification as the verification phase (computing the output) needs to be completed quickly (this is most often performed in real-time while the user is waiting), whereas training can be done off-line and can take much longer if necessary.

Most of the developmental effort for the HSV project went into the back-propagation algorithm as other learning algorithms did not prove as successful in experimentation (although error rates are reported using a number of these - see Section 3.4.4).

Conjugate Gradient Descent

Conjugate Gradient Descent (CGD) is an advanced method of training multi-layer perceptrons. Some researchers claim that CGD performs significantly better than back-propagation [137], but this largely depends on the problem, the data and the network topology. In the experimentation described in Section 3.4.4 the performance of conjugate gradient descent was inferior to back-propagation.

CGD has a different update philosophy to that of back-propagation in that

the network weights are updated at the end of each epoch rather than after each case in the dataset. This algorithm works by constructing a series of "line searches" across the error surface. The direction of the steepest descent is found (as in back-propagation), however the process of gradient calculation greatly differs. The gradient of the error surface is averaged across all cases in the training set before updating the weights at the end of each epoch. Once the gradient has been found, instead of taking a step proportional to the learning rate (as in back-propagation) a straight line is projected in that direction and the minimum is located along this line. Further line searches are conducted every epoch until no further minima are found (this is hopefully the global minimum of the error surface or at least close to that).

Although the line searches themselves are one-dimensional and are very quick, the gradient calculation is quite intensive and as a result a CGD epoch is substantially slower than a back-propagation epoch (in practice around three to ten times slower). The line search however, converges in much fewer epochs. As part of the experiments conducted and described in Section 3.4.4, back-propagation was often found to take twenty to thirty times the number of epochs required by CGD. As a result CGD converged much quicker, usually by around a factor of five.

CGD however is not ideally suited to the network structure for the problem of HSV (at least the parametric version). A typical parametric HSV network consists of a relatively small number of inputs (rarely more than fifty), rarely more than two or three hidden layers and a single output unit. CGD is more suited to networks with several hundred weights (or more) and/or multiple output units [13]. The Levenberg-Marquardt algorithm is usually recommended for smaller, single-output networks.

Levenberg-Marquardt

The Levenberg-Marquardt algorithm is the result of work by the two researchers in the early development of neural networks [76, 83]. This approach uses a combined strategy of linear approximation and gradient-descent to locate a minimum in the error surface. The resource requirements of the algorithm restrict its use to very small networks with a single output. The space requirements are significant (just over the square of the number of weights) and rapidly become unmanageable in large networks.

The algorithm makes the assumption that the underlying function being modelled is linear. The minimum of this line (or curve) is taken and if this is

lower than the current minimum the weights are adjusted to this new point. As the linear assumption is generally ill-founded, many times the point being tested will not be superior to the current minimum. Ideally, after enough successful steps are taken the current minimum approaches the global minimum.

The Levenberg-Marquardt algorithm works well in optimal circumstances, but the performance was empirically found to deteriorate in less-optimal circumstances. When the error surface is particularly jagged the algorithm has problems and in experimentation with HSV networks (see Section 3.4.4) convergence was impractically slow, taking hundreds of times longer than both back-propagation and conjugate gradient descent, with the error oscillating wildly at times.

3.1.4 Radial Basis Functions

Multi-layer perceptrons are perhaps the best known neural network architecture. The second most commonly used is the *radial basis function* (or RBF) network [50, 13]. The MLPs approach to dividing up the sample space is to use a series of *hyperplanes* (in two-dimensional space, a hyperplane is simply a line). In contrast a radial basis function network uses a *radial* function and works by dividing up the pattern space using *hyperspheres* (in two-dimensional space, a hypersphere is just a circle).

The radial approach is very localised (it basically draws circles around clusters of data) whereas the linear approach is active over the entire pattern space. Consequently, RBF networks tend to need more units that MLPs, but MLPs make extrapolations over the entire pattern space that may be unjustified if data is seen that is unlike any of the training data (whereas a RBF will always have a near-zero response in this case).

A RBF has a more rigid topological structure than MLPs, with exactly three layers: the input layer, one hidden layer that contains the radial units, and a linear output layer. The approach to RBF training is therefore quite different to that used in a MLP.

Training begins with the radial centres and their deviations (spreads) being set using unsupervised techniques (that is, the techniques consider only the input variables in the training data). Essentially, the idea is to pick centres that lie at the heart of clusters of training data and deviations selected to reflect the density of this data.

In the second phase of training, the linear output layer is optimised using

the *pseudo-inverse* technique [13, 114, 44]. Alternatively it is possible to alter the output unit activation functions to logistic and use back-propagation, conjugate gradient descent or the Levenberg-Marquardt algorithm to train this layer, thus combining the advantages of radial non-linearities and logistic stability, although at the expense of increased training time. In signature verification the training time is far less important as this is just a one-off occurrence and it is not necessary to do the training in real-time. Thus the user is not inconvenienced by having to wait for the training. While it is possible to train RBF networks to perform *identification* (trying to identify the author of a particular writing sample), they are less successful at verification because of the difficulty obtaining large amounts of both positive and negative training data.

3.1.5 Bayesian Networks

Bayesian networks are networks that are based on Bayes' theorem and are essentially models based directly on the input data.

MLP and RBF network architectures infer a parameterised model (the weights are the parameters) from available training data. The parameterised model (the network) is usually much smaller than the training data, and can be executed quite quickly, although the time taken to train the model may be long.

An alternative approach is to model the function directly from the training data. This type of architecture results in a very small and very fast training phase in which the classical perception of training is not performed, but rather a manipulation and re-organisation of the training data to a form where it can be used for classification. The disadvantage of this is that the resulting model is very large, consumes much memory and is relatively slow to execute. It would therefore seem that these types of models (such as *Probabilistic Neural Networks* and *Generalised Regression Neural Networks*) are less appropriate for real-time operations such as signature verification. Despite this fact, they are briefly discussed here for the sake of interest.

Probabilistic Neural Networks

This type of architecture has a first layer containing radial units that store every training case. These radial units output a Gaussian activation centred at the stored point (and hence acts as a probability density distributed around

that point), and subsequent layers combine these outputs into estimates of class probabilities.

There are not many parameters for training of probabilistic neural networks (PNNs): a *smoothing* factor, optional *prior probabilities* and, for four layer PNNs, an optional *loss matrix*.

The *smoothing* factor determines the widths of the Gaussian functions, centred at each training case and stored in the first hidden layer radial units. A smaller smoothing factor gives a sharp, "spiky" approximation to the underlying function, which may perform well on the training set but generalise poorly (and therefore perform badly on the verification set). A large smoothing factor gives a "blurred", smooth approximation, which may perform relatively poorly on the training set, but generalises well to the verification set.

Usually, the algorithm is not too sensitive to the precise choice of smoothing factor. A process of trial and error is normally used to obtain the best results typically experimenting with values between 0.1 and 10.

The *prior probabilities* are used in situations where the training set is known to be biased (for example, if in a two-class problem the population at large has only 1% of cases positive with the rest negative, but the training set has 50% positive and 50% negative). As the PNN estimates class probabilities by adding estimates of density centred at each case, the resulting probabilities will be highly biased unless adjusted to reflect the imbalance between population and training set representation. Ideally however, cases are drawn randomly so there is no need for adjustment. Additionally if the prior probabilities are unknown, the working assumption must be that the data set fairly represents the population distribution.

The *loss matrix* is an extra layer that can be added to the PNN (a fourth layer containing the same number of units as the third) in order to associate a penalty with a particular misclassification. The reason for this is that some misclassifications, in reality, are more expensive mistakes than others. For example, it may be thought of as more expensive to misclassify an orange traffic light as green than it is to misclassify it as red. The entries in the loss matrix represent the relative costs of various types of misclassification with columns representing the actual class of the case, and the rows representing the assigned class. The leading diagonal of the loss matrix always contains zeroes (as there is obviously no penalty involved in getting the right answer).

A useful aspect of PNNs is that the output unit activations are actually class probabilities (as opposed to arbitrarily-scaled confidence measures), so

a more rigorous interpretation of the output is possible than with standard architectures.

Generalised Regression Neural Networks

A generalised regression neural network (GRNN) is used for regression problems. Unlike a PNN, a GRNN can be specified with fewer units than training cases, and a clustering algorithm used to assign centres. Usually however, the number of centres cannot be significantly less than the number of training cases without sacrificing performance. The simplest (and most common) approach is to use the full set of training cases.

Like PNN's, GRNN's have an associated *smoothing* factor which, as with PNN's, controls the deviation of the Gaussian functions located at the radial centres.

3.1.6 Kohonen Self-Organising Maps

Kohonen networks [69] have been successfully applied to several handwriting analysis projects (often in handwritten character recognition [150]), sometimes in signature verification [23]. Kohonen networks (typically referred to as "self-organising maps") perform unsupervised learning: that is, they learn to recognise clusters within a set of unlabelled training data.

Kohonen networks always have two layers: an input layer and an output layer consisting of the *topological map*. The topological map is a radial layer of the Kohonen network, with units laid out in two-dimensions, and trained so that inter-related clusters tend to be situated close together in the layer.

Kohonen training is quite simple and the neighbourhood points play a critical role [40]. During training, the algorithm selects the unit whose centre is closest to the training case. That unit and its neighbours are then adjusted to be more like the training case. By updating the surrounding units in addition to the "winning" unit, Kohonen training assigns related data to contiguous areas of the topological map. During training the neighbourhood size is progressively reduced, together with the learning rate, so that a rough mapping is produced initially (with large clusters of units responding to similar cases), with finer detail being attained later (as individual units within a cluster respond to finer levels of discrimination amongst related cases).

3.1.7 Autoassociative Networks

It is worth mentioning autoassociative networks although they were not considered in detail in this work. Autoassociative networks have the same number of neurons in the input and the output layers, and fewer neurons in hidden layers [13]. These types of networks are designed to reproduce the input data as the output, that is, the training inputs also act as the desired outputs, with the hidden layers performing a kind of dimensionality reduction. The objective is to retain as much discriminative information as possible while reducing the number of input variables (in the sense of this network model, to be able to reproduce the input data as closely as possible after it has been "squeezed" through the lower-dimensionality hidden layer).

One of the more extensive studies of handwriting analysis using autoassociative networks can be found in [66]. In this work the authors compared several network structures for their ability to learn to recognise handwritten numerals. Specifically, autoassociative networks, mutual associate networks, the nearest neighbour method and the projection distance method were examined. The authors used 29,883 handwriting samples to train the networks and 14,979 in testing. Autoassociative networks were found to perform most successfully, resulting in 98.1% recognition (as compared with the second best of 97.4% recognition for the nearest neighbour method).

There are some minor inconveniences to the use of autoassociative networks such as the need to specify the dimensionality of the hidden layer in advance of training the network. However, the main disadvantage that precludes the use of these kinds of networks for use in handwritten signature verification is the requirement of very large training sets, when typically only a very small number of sample signatures are available.

3.2 Applications to Handwritten Signature Verification

Concentrated efforts at applying neural networks to the problem of HSV have been undertaken for ten to fifteen years with varying degrees of success and for around twice as long as hidden Markov models discussed in Chapter 4. The main attractions for the use of neural networks (NNs) in this field are:

1. *Expressiveness*: NNs are an attribute-based representation and are well-suited for continuous inputs and outputs. The class of multi-layer net-

works as a whole can represent any desired function of a set of attributes [130], and signatures can be readily modelled as a function of a set of attributes.

2. *Ability to generalise*: NNs are an excellent generalization tool (under normal conditions) and are a useful means of coping with the diversity and variations inherent in handwritten signatures [20].

3. *Sensitivity to noise*: The underlying basis of neural network computation is nonlinear regression, so NNs are very tolerant of noise in the input data. The networks are designed to simply find the best fit through the input points within the constraints of the network topology.

4. *Graceful degradation*: NNs tend to display graceful degradation rather than a sharp drop-off in performance as conditions worsen.

5. *Execution speed*: Depending on the nature of the training phase of the NN, training can take a large amount of time. In signature verification this training is a one-off cost and can be done off-line (that is, the training phase in HSV is rarely performed while a user waits for results). The execution of NNs corresponds to the verification of test signatures and this phase is extremely fast, which is a good match for HSV because it is during the verification where rapidity is necessary.

The following section contains a summary of the more significant studies into neural network based approaches to handwritten signature verification.

3.3 Previous Work

One of the more significant early studies into off-line HSV using NNs appeared initially in [88] and further in [158], which discusses a system for detecting casual forgeries (those in which the forger is given no opportunity to practice the signature being forged). The system is evaluated using eighty genuine signatures and sixty-six forgeries but most results are presented for only one person. The signatures were collected on cards which were then scanned into the computer, followed by thresholding to produce binary images that are centred and normalised. A back-propagation learning algorithm is employed using a training set of ten genuine signatures and ten forgeries for one person and the testing is done on seventy genuine signatures and fifty-six forgeries.

A false rejection rate (FRR) of 1% with a false acceptance rate (FAR) of 4% is reported and lowering the threshold resulted in 0% FRR and 7% FAR. Although the results for one person are encouraging, they cannot be assumed to apply to a larger population. In addition, the performance of the technique degraded considerably if no forgeries were available for training.

The above work forms part of a PhD thesis [158] in which the system was more thoroughly tested and some additional techniques implemented. The new test database consisted of 1,190 images including 590 genuine signatures from nine subjects. Each subject contributed fifty to seventy samples collected over a period of eighteen months, and 396 forgeries were contributed by forty-four volunteer forgers who were shown the name of the person whose signature was being forged, but not the actual signature. One of the techniques proposed involves building a slope histogram of a signature using twenty valid signatures and ten forgeries for each of the nine subjects. By building this slope histogram, the author is attempting to exploit the regularity of length and curve in the signature. Overall signature shape at various angles is evaluated to form a histogram. Histograms are then passed to a classifier that compares them to those constructed from a number of valid signatures. Equal error rates reported average about 7%.

The second approach presented in this paper is the synthetic discriminant function (SDF) approach. This technique involves selecting a linear filter that produces a specified output for each image of a training set. If forgeries are included in the training set, the error rate produced is 4%. The author then combines the histogram and SDF techniques to form an integrated system, which results in an error rate of approximately 1% on casual forgeries and 5-6% on skilled forgeries.

A more thorough investigation of the use of NNs to detect casual forgeries is found in [105]. The training of the NNs requires the presentation of forgeries as well as genuine signatures, but the work also showed that the genuine signatures of other signers may suffice as forgeries for training purposes. The signature database used consists of static signature features collected from five individuals over two years. There are 380 genuine signatures in total and the same five individuals contributed 265 forgeries where they knew the name of the person whose signature was being forged, but had not viewed a genuine signature. A variety of network structures are investigated, ranging from a single neuron to fully-connected multi-layer networks with thirty-two (the number of inputs) or fifty-three (using some additional overlapping inputs) units. The

best performing structure is found to be a single neuron connected to the input image (multi-layer networks had similar classification performances, but obviously required more weights and computation time). An equal error rate of 3% for genuine signatures and casual forgeries is reported when training the system using ten genuine signatures and ten forgeries.

A technique based on Bayesian neural networks is explored in [22] which involves dynamic HSV of Chinese signatures. A set of sixteen features is used including duration, average velocity, number of segments, average length, aspect ratio (width versus height) as well as two density ratios (left versus right and top versus bottom) in an attempt to extract global information of both the static and dynamic type. The database used consists of 800 genuine signatures from eighty signers and two hundred simple and two hundred skilled forgeries from ten forgers. Results show about 2% FRR, 2.5% skilled forgeries FAR and 0.1% zero-effort FAR. Unfortunately the paper does not present any details of the experiments, particularly regarding how the reference was computed and the sizes of the training and verification sets.

Another on-line system developed at around the same time appears in [91] using a totally different approach based on autoregressive (AR) models, in which the signature is treated as an ordering of curve types. Each signature is divided into eight segments with three features extracted from each segment, giving twenty-four features in total. A database consisting of fifty-eight sample signatures from each of sixteen subjects taken over six sessions is used for testing. No skilled forgeries were available and zero-effort forgeries were used instead. Overall error rates using user-specific thresholds varied from a low of 7.92% to a high of 21.83%.

An investigation of handwriting and handwriting-style as well as writer classification was described in [135] (similar ideas are investigated in a more recent, independent work in [143]). Although the authors are concerned with stroke-based classification rather than full-scale signature verification, the work done remains theoretically interesting for many reasons, such as the classical use of Kohonen Self-Organising Maps (SOMs) for handwriting analysis. In the study, handwriting data is captured using a digitizing tablet and strokes defined as the pen-tip trajectory between consecutive minima in the pen-tip velocity. A Kohonen SOM is used to obtain a finite list of prototypical strokes (PS) from a large body of normalised (for size and slant) handwriting from several different users. This "alphabet" of strokes approximates the handwriting in the training set with a minimised root mean squared error, and is shown

to generalise well to strokes in the handwriting of unseen writers. Sequences of these stroke codes (the typical sequence size was three) are then classified as specific letters.

The database used consists of isolated on-line handwritten words collected from thirty writers of different nationalities in various situations (during page, sentence and word copying task) and languages (Dutch, English and Italian). For each input stroke, the PS alphabet is searched for the closest match on the basis of Hamming distance using a fourteen-dimensional feature vector, and the frequency of usage for each PS is counted. The resulting histograms are considered as the Writer Feature Vector. Through the use of the SPSS program CLUSTER the authors are able to show that style and writer groupings can be obtained by clustering the PS alphabets for each writer. Success rates weren't emphasised in the study, rather the approach itself was considered novel.

Another use of Kohonen SOMs in handwriting analysis can be seen in [151] where the authors attempt to automatically characterise handwriting style as being handprinted, cursive or mixed. Two 20×20 Kohonen SOMs are trained using the handwriting of cursive and handprint writers (187 writers in total). Two distance measures, d_c and d_h are calculated based on these SOMs as the average Euclidean distance between fourteen stroke features (mainly angular and including loop area, pressure and total length of the stroke) and the "winning" neurons in each SOM. The value for d_h represents the distance to the handprint SOM and d_c the distance to the cursive SOM. That is, if d_c is small and d_h is large, then the writing is classified as cursive and inversely, a small d_h and large d_c implies handprint. If both of these values are similar then the handwriting is classified as mixed. The authors took this further and created a "cursivity index" using a formula based on the average number of pen-lifts per letter in the written words of one writer. Thresholds are set on the product of the cursivity index with the ratio of d_h to d_c in the final decision of whether the handwriting is cursive, handprint or mixed. Using samples from the UNIPEN database [134] annotated with writing style, the system is correct just over 60% of the time. However, as noted by the authors, the annotated style (performed by the people gathering the original data) is often dubious and the actual accuracy is probably much higher than this. The purpose of this research is to act as a "front end" for handwritten character recognition where specialised recognisers for particular writing styles would be invoked based on the detected writing style. The cursivity index also goes some way towards detecting an individual's natural writing style and has implications

for use in feature sets for HSV and writer identification.

3.4 Methodology

As discussed previously, the system described in this dissertation is a *on-line* one with respect to the method of data capture (see Chapter 1). The primary distinction that can be made between on-line HSV systems is the method of feature representation: those that use functions as features and those that use parameters [110, 108]. The system described in this chapter falls clearly into the parametric approach, with a number of parameters (or *features*) being extracted from the signature and analysis of the signature being based on the features themselves.

This section presents the methodology followed for the development of the neural network portion of this project. This includes a discussion of the pre-processing performed and the signature database used, along with a thorough examination of the features used in the neural network, the experimental setup and the training process.

3.4.1 Pre-processing

There is minimal pre-processing of the signature data used in this study. Many other areas of handwriting analysis (for example, handwriting recognition) require large amounts of pre-processing such as slant correction, rotation correction and size normalisation to reduce variations in the handwriting [19]. In handwritten signature verification however, most of the subtle nuances of the writing such as size, and particularly slant, are indicative of the signers natural style, removal of which would deny the HSV system of useful information. Additionally, the use of high quality tablet hardware to capture signatures prevents most of the noise that might be introduced through processes such as scanning.

The only pre-processing performed is rotation normalisation (necessary as the orientation of the tablet and resulting signatures is not always consistent). This procedure involves multiple stages, with the first being to extract the baseline points from the signature. The baseline points are defined as the bottoms of all non-descender characters (see Section 3.4.3 for further discussion of these and Figure 3.31 for an illustration) and linear regression is used to best fit a straight line through the baseline points. The signature is translated to

the origin and is rotated using the following formulae:

$$\begin{aligned} x' &= x \cdot cos(\theta) - y \cdot sin(\theta) \\ y' &= x \cdot sin(\theta) + y \cdot cos(\theta) \end{aligned}$$

where θ is the inverse tan of the gradient of the line found during linear regression and x' and y' are the new x and y coordinates.

3.4.2 Signature Database

A database of signatures was captured as part of this project in order to facilitate the HSV experimentation. One of the most unfortunate aspects of current HSV research is the lack of standard databases. This is a major inconvenience for researchers who need to spend a large amount of time capturing their own data, but more importantly, use of different databases makes it almost impossible to perform meaningful comparisons between different HSV systems.

Before any effort was made to capture sample signatures, the makings of an effective signature database were considered in detail:

- *Large sample size*: One of the most basic rules of scientific experimentation is that the experiments must be applied over a large dataset. Exactly what constitutes "large" depends on the application area and how expensive (in terms of both money and time) it is to accumulate the data. In HSV most researchers tend to work with dataset of well over one thousand signatures and anything under this size is probably insufficient. Obviously the more signatures used and the more diverse the dataset, the more reliable the obtained error rates are.

- *Diversity of samples*: It is desirable to have a high level of diversity in the signature samples. A system that is going to be used in the real world must be able to handle a large variety of signatures and signature types, so the testing process should simulate a realistic environment as closely as possible. This means using signers of different age groups, sexes, nationalities, backgrounds and left- and right-handedness. Many researchers use signatures provided by colleagues within their research group, which clearly doesn't satisfy the diversity requirement. Taking the diversity issue a step further, researchers have examined environmental effects such as seating position and time of day and their influence on signature generation [38].

- *No arbitrary exclusion*: For similar reasons of realism to that described above it is necessary to include *all* captured signatures in the database. Many databases referred to in the literature have been filtered to remove "undesirable" or "inappropriate" signers, or those that are not likely to work well with the proposed system (for example, [26, 74]), which generates biased results. In a realistic environment, a researcher is not going to be able to manually exclude individual signatures or signers from using the system. Some researchers include hypothetical information as to potential error rates obtained by ignoring signatures failing some objective criteria (for example, a threshold on the signature duration [32]), but this differs from the unacceptable removal of data without explanation.

- *Inclusion of skilled forgeries*: Although most recent HSV systems in the literature are tested using databases that include skilled forgeries, it is still worth noting this necessity. Obviously, it is more difficult to obtain skilled forgeries than genuine signatures, but without the inclusion of these, quoting false rejection rates is far less meaningful.

All samples used in the project described in this dissertation were captured using the Kurta XGT Serial Digitizing tablet [93]. The XGT consists of an opaque tablet and a cordless non-inking pressure-sensitive pen (experimentation and results obtained using a stylus to provide visual feedback are discussed in Section 5.3.6). The XGT has a 152×203 millimetre (6×8 inch) effective writing area and captures samples at the rate of 205 points per second. The resolution is 1,000 points per centimetre (2,540 points per inch) at an accuracy of 0.0127 centimetres (0.005 inches). In addition the tablet captures pen-tip pressure as one of 256 levels measured through the pressure sensitive tip of the cordless pen. The pressure and position values are translated into coordinates on the serial bus. The hardware interface is a Serial EIA Standard RS-232C port connected to a laptop computer running custom-written driver software.

The values in the output stream produced by the digitiser are equidistant in time and consist of tuples (x, y, p) that contain the following data:

- $x(t)$, the x-coordinate sampled at time t;
- $y(t)$, the y-coordinate sampled at time t;
- $p(t)$, the axial pen force at time t.

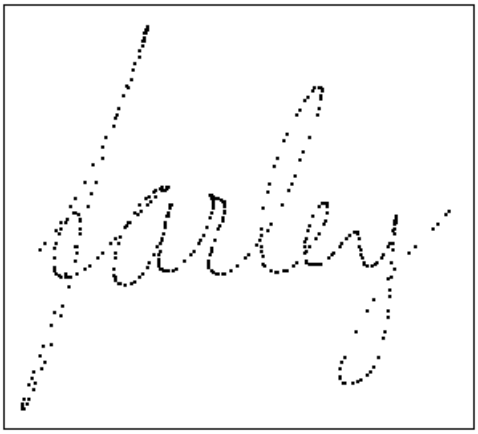

Figure 3.11: *The sampled coordinates captured from the handwritten word "farley".*

Figure 3.12: *Interpolation of the sampled coordinates produces the off-line, or static, image of the word.*

The sampled coordinates of a typical handwritten word can be plotted and displayed, an example of which can be seen in Figure 3.11 with the handwritten name "farley". The interpolation of the sampled coordinates (Figure 3.12) produces an image of the writing similar to the off-line version of the word.

Sampling of the pen-tip position is done by the XGT tablet even when the pen is not in contact with the writing surface, as long as it is within 2.5 centimetres (1 inch) of the tablet. The path of the pen while in a "pen-up" state is not detectable by potential forgers examining off-line copies of the signature and provides a useful source of extra information for a signature verification system.

The database used for the primary HSV experiments was captured using the above tablet with the signer being seated in a comfortable position with good lighting. The signers were orally prompted by the data collector to provide their signature sample in their own time. The signers generally provided five samples in one sitting, with this operation being repeated on two or three separate occasions (resulting in between ten and fifteen genuine signatures per person). Forgeries were obtained by allowing the forger to view a static image of the written password as well as being provided with basic information on the dynamics of the signature in the form of velocity and pressure profiles. Forgers were then allowed to practice their forgeries and to attempt the forgery in a similar environment to that in which the original signature was performed. Forgers were not professional or trained forgers but they were aware of the nature of the verification system and were aware of the nature of the writing they were attempting to forge.

Attempts were made to obtain a reasonably realistic database population, and to this end the set of genuine signatures contained writers of several different nationalities and backgrounds as well as different age groups, genders and left- and right-handedness (see Figure 3.13 for a statistical breakdown). In total there are 111 signers in the database contributing a total of 2,779 genuine signatures. In most experiments, five signatures are used to build each reference, leaving 2,779 - (111$\times$5) = 2,224 genuine signatures for testing. Ten skilled forgeries were captured for each signer, providing a total of 1,110 forgeries in the database. No captured signatures were excluded from use in the database (with the exception of a small number in which a device failure occurred and no signature data was obtained), and the database itself is quite large and diverse compared with others in the literature [47]. A summary of the database is presented in Table 3.1.

3.4.3 Extracted Features

The features that are extracted from signatures or handwriting obviously play a vital role in the success of any feature-based handwriting analysis system. They are arguably the most important aspect and exceed the choice of model or means of comparison. If a poorly constructed feature set is used with little insight into the writer's natural style, then no amount of modelling or analysis is going to result in a successful system. Further, it is necessary to have multiple, meaningful features in the input vector to guarantee useful learning

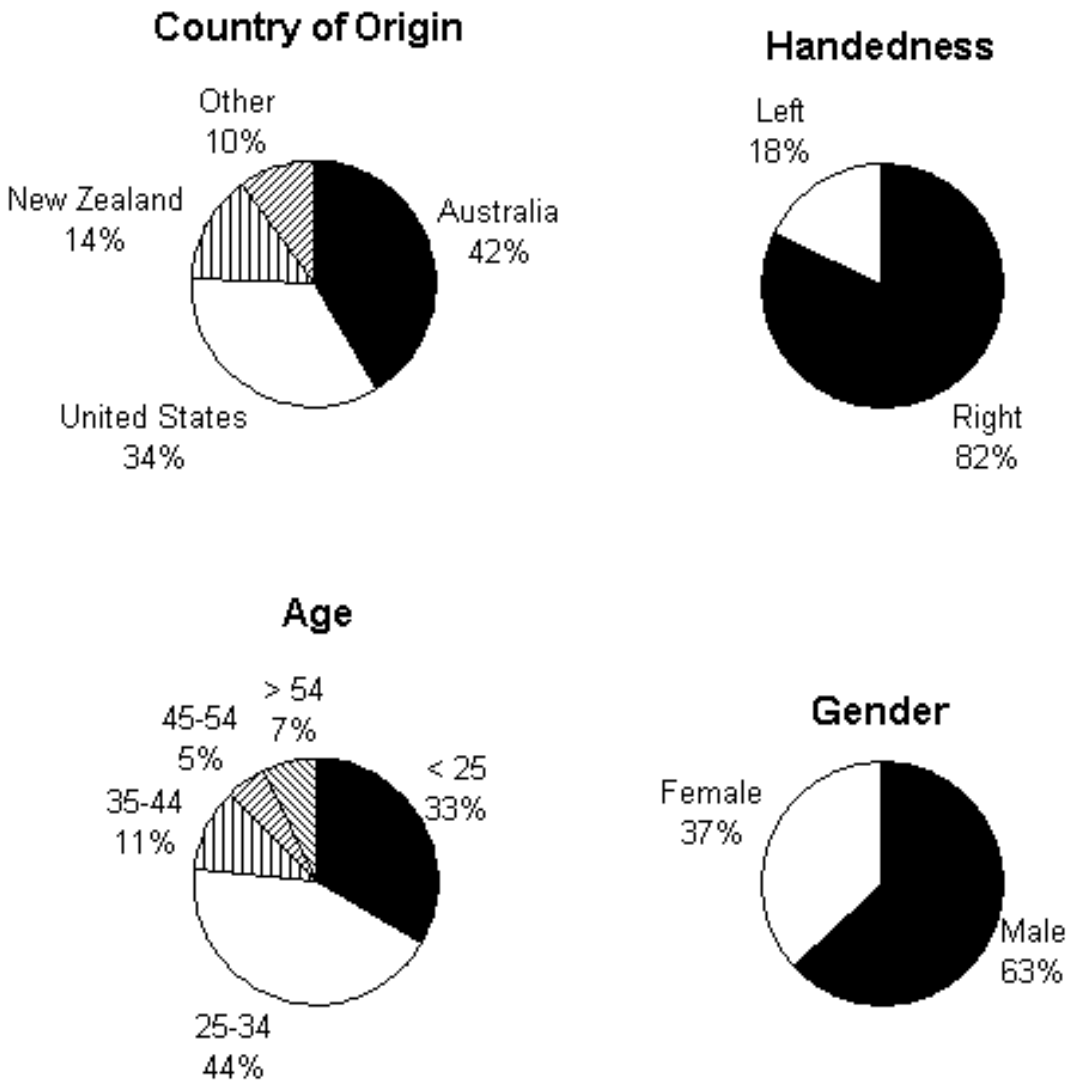

Figure 3.13: *Contributors to the database grouped according to (a) nationalities, (b) handedness (left or right), (c) age and (d) gender.*

Property	Value
Number of signers	111
Number of genuine signatures	2,779
Number of forgeries	1,110
Digitiser	Kurta XGT
Sample rate	205 pps
Resolution	1,000 ppcm (2,540 ppi)
Error	+/- 0.0127 cm (0.005 inches)
Pressure levels	256

Table 3.1: *A summary of the main database of signatures used in experimentation.*

by the neural network [43].

The initial decisions as to which features to incorporate, in order to maximise the accuracy of the approach, involved a combination of studying other publications in the area (finding out what other researchers have found useful or useless) and intuitively considering which other features might be most applicable. The intuitive approach was based on study of the handwriting process, forensic analysis of handwriting by humans and examination of features that are most useful to humans in deciding whether a particular handwriting sample is produced by some author.

The properties of "useful" features were examined in [167] and it was concluded that these must satisfy the following three requirements:

1. The writer must be able to write in a standard, consistent way (that is, the writer should not be required to write unnaturally fast or slow in order to produce a particular feature);

2. The writer must be somewhat separable from other writers based on the feature;

3. The features must be environment invariant. That is, they must remain somewhat consistent, irrespective of what is being written.

The third of these points is more relevant to the process of writer identification than HSV, as a person's signature is most often a fixed text. It is relevant to HSV however in the sense that the features should remain stable

irrespective of the environment in which the signature is being performed (the weight of the pen, the friction of the pen tip etc.).

These basic requirements are perhaps oversimplifying the feature selection process as there are often non-obvious combinations of features that lead to insightful information. They are however useful rules of thumb and the features selected for use in this study adhere to those rules wherever possible.

What follows now is a description of each of the features that are extracted from a given signature, as well as their significance and method of calculation. Each of these features acts as a single input to the neural network (note, many of these features are also used in some form in the model discussed in Chapter 4).

Signature Duration: The time taken to perform a signature is perhaps the single most discriminatory feature in handwritten signature verification. A study reported in [166] found that 59% of forgeries can be rejected on the basis of the signature duration being more than 20% different from the mean. In another study described in [46], simple feature values are examined and experimentation concludes that duration is the single best discriminator, resulting in an equal error rate of under 10% when used in isolation.

Pen-Down Ratio: This feature is the ratio of the pen-down time to the total writing time. It is a very personal feature and one that does not undergo a large amount of variation when signing, irrespective of the mood or emotions of the writer [96]. In addition it is a feature of handwriting that is very difficult for a forger to reproduce as there is no way of determining the pen-down ratio from an off-line copy of the writing (for example, see Figure 3.14 that has two samples with very different pen-down ratios). Calculation of this feature is done by firstly removing leading and trailing zeroes from the captured data then taking the ratio of the number of non-zero points to the total number of points.

Horizontal Length: The horizontal distance measured between the two most extreme points in the x direction (often simply the distance between the first point captured and the last point captured). Any fragments such as 't' crossings or 'i' dottings are excluded from this calculation (fragments such as these are far less stable and individual traits such as extravagant 't' crossings can cause high variability with this feature). The horizontal length tends to remain stable with a practiced word and particularly with a signature, irrespective of the presence of a bounding box, horizontal line or even with no line present. See Figure 3.15 for an example.

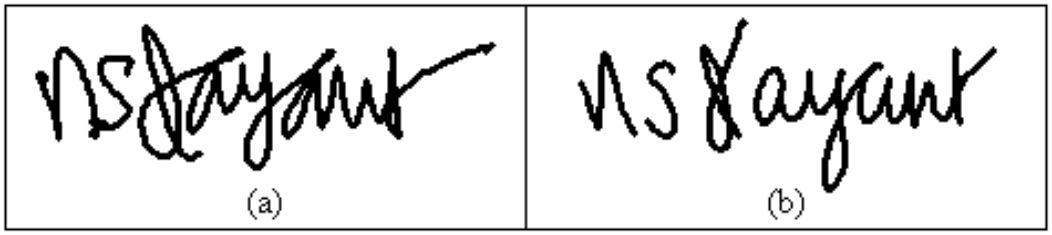

Figure 3.14: *This is an illustration of the difficulty that a potential forger has in trying to identify the pen-down ratio. The sample in (a) is a genuine signature and (b) is an attempted forgery based on the forger having seen an off-line version of the signature (both taken from the signature database used in this project). The pen-down ratio for the genuine signature is 0.992 and is 0.879 for the forgery (forgeries were typically found to have much lower pen-down ratios, presumably because of the extra attention to detail).*

Figure 3.15: *Horizontal length of a typical handwritten word is a simple feature to comprehend and calculate. The horizontal length of this sample is 1,345 pixels.*

Aspect Ratio: Aspect ratio is the ratio of the writing length to the writing height. This feature obviously remains invariant to scaling as if the user signs in a different size, both the height and the length will be altered proportionally and the aspect ratio will remain the same.

Number of "pen-ups": This is a basic feature indicating the number of times that a user lifts the pen while performing their signature after the first contact with the tablet and excluding the final pen-lift. This is highly stable and almost never changes in an established signature. Additionally, this can potentially be a difficult feature for a forger to discern from an off-line copy of the signature.

Cursivity: Cursivity is a number normalised to between zero and one that represents the degree to which a writer produces isolated handprint characters or fully-connected cursive script. The higher the value for cursivity, the more connected the word is. A value of one means that there were no pen-ups over the entire word and value closer to zero means that the writing was mostly printed rather than cursive. The term "cursivity" and basic definition were originally used in [151], however the calculation of this value by the original authors differs from that which is performed here. In [151] the formula for cursivity was:

$$C_n = \frac{1}{n} \sum_{w=1}^{n} \frac{N_{l,w} - N_{pd,w} + 1}{N_{l,w}}$$

where:

- C_n is the cursivity index;
- n is the number of words without letters 'i' or 'j';
- $N_{l,w}$ is the number of letters in a "non-ij" word;
- $N_{pd,w}$ is the number of pen-down streams in this word.

The main drawback with this approach is that it is necessary to have a priori knowledge of how many letters are in the word being written. While this is possible is some situations it is not a valid assumption with signatures and would defeat the purpose of making this authentication system entirely automated. The calculation of cursivity employed in this dissertation is done through the use of strokes (see Section 3.3 for more on strokes) instead of letters. Strokes are objective and easily calculable from the body of the signature itself. The formula for cursivity calculation here then is:

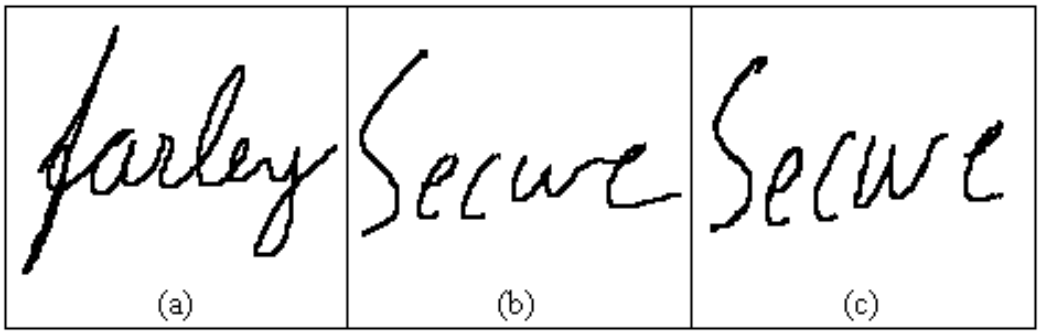

Figure 3.16: *Cursivity varies widely between different authors while tending to remain similar for different samples produced by the same author. For example consider parts (a) and (b) above that contain words written by different authors with very different cursivity values of 16.0 and 3.6 respectively. Part (c) is another sample of the same written word as (b), by the same author, and has a very similar cursivity value of 3.8.*

$$Cursivity = \frac{number\ of\ strokes}{number\ of\ pen-downs}$$

In this setup a high value indicates a lean towards a more cursive style and a low value implies a more handprinted style. The average level of cursivity is something that remains close to constant for an individual across a large body of handwriting [151]. The same can be said of the same word or small phrase being written by an individual, which is why this feature was considered for use. Different handwriting samples and their corresponding levels of cursivity can be seen in Figure 3.16.

Cursiveness: Cursiveness is a different measure of whether a particular handwriting sample is more cursive or more handprinted. This feature is one that remains close to constant across different productions of the same handwritten word by the same author. Cursiveness is a personal aspect of handwriting and can be very difficult for a forger to discern due to the dependence of writing style and pause characteristics.

Calculation of the value for cursiveness is quite simple and is the ratio of the horizontal length of the handwriting to the number of pen-downs. The value of cursiveness goes up when there are less pen-downs and goes down when there are more pen-downs. Figure 3.17 illustrates how difficult it can be to estimate values for this feature from just an off-line copy of the signature.

Top Heaviness: Technically, top heaviness is a measure of the proportion of

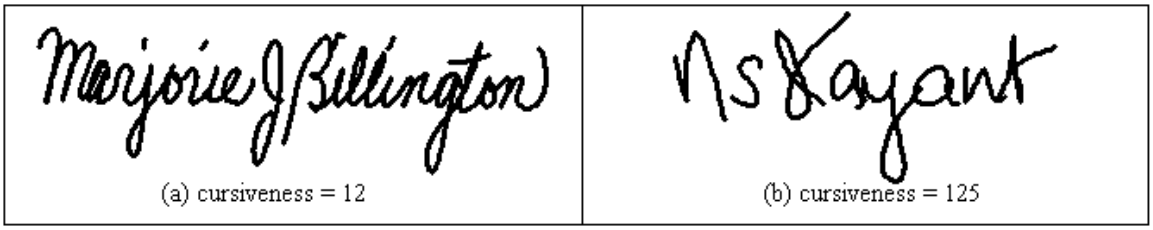

Figure 3.17: *Cursiveness varies somewhat between authors and is a feature that is highly indicative of natural handwriting style. (a) shows a signature with a seemingly high cursiveness, but the actual value for this is 12 which is significantly lower than the signature in (b) at 125. These signatures are examples of how visual inspection can be quite deceptive in estimation of cursiveness.*

the signature that lies above the vertical midpoint (that is, the ratio of point density at the top half of the signature versus the density at the bottom half). It is measuring the concentration of the handwriting about its midpoint or conversely, how widely spread the handwriting is. The only real issue in the calculation of top heaviness is to decide which measure of central tendency is most indicative of the true vertical midpoint. It would be pointless to use the median in this situation as, by definition, half of the points would lie above the median and half below. It is therefore a question as to which of the other two standard central measures (mean or mode) is more appropriate. Both of these options were investigated, examples of which can be seen in Figure 3.18 (mean is calculated in the standard way, while mode is found by creating a horizontal frequency histogram of the points in the sample and selecting the peak in that histogram). Although the mean is more likely to be affected by outliers in the sample, upon investigation with several different signatures from different users, it was found that the large number of points in the data minimised this effect and no measure is regularly more visually central than the other two. As such, both of the values are used to generate separate measures of top heaviness to be presented to the neural network.

Horizontal Dispersion: This is the same as top heaviness but with respect to the horizontal spread of the handwriting rather than the vertical spread. Calculation is done in a similar fashion.

Curvature: Curvature is a measure of how "flat" or how "curved" the handwriting is. A high value for curvature means that the writing is more dramatically curved, which is associated with more thorough or exaggerated completion

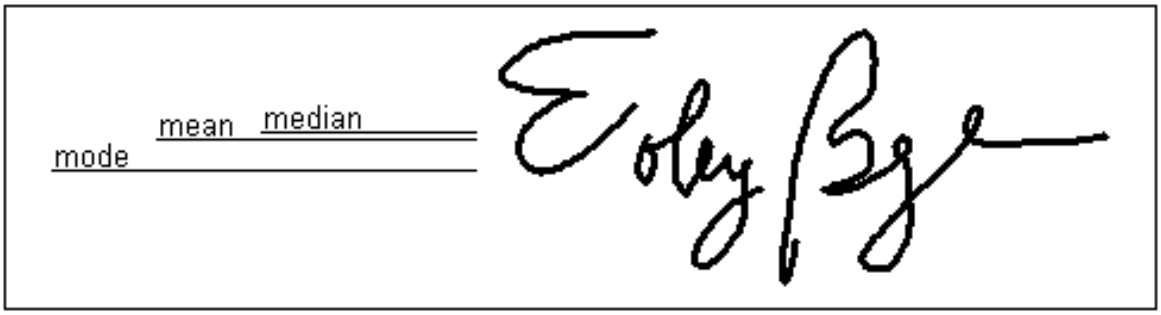

Figure 3.18: *This is an illustration of how the different measures of central tendency can give different midpoints for the calculation of top-heaviness. The figure has three horizontal lines drawn to illustrate the location of the calculated midpoint using the different central tendency measures.*

of handwritten characters. For example, in Figure 3.19(a) the letters are not well-formed and the writing is more "flat" or "lazy" (particularly towards the end), which is reflected in the low curvature value of 3.96. Conversely, the sample in Figure 3.19(b) has much more well-formed lettering resulting in a higher curvature value of 5.22.

Curvature, unlike most of the other features considered for use in this system is slightly susceptible to change depending on the mood or demeanor of the writer. If a user is trying to write quickly then they are more likely to write with a lower curvature, however this feature is still produced with sufficient consistency to be of use when the text is strongly fixed, as is the case with signature verification. In fact, if the mood of the writer remains stable this can be a highly effective feature in identifying the author as it is indicative of the writer's natural style.

Curvature is calculated as the ratio of the signature path length to the word length. The path length is the sum of distances between each consecutive point in the sample so is generally quite large, of the order of 10,000 pixels or more. The word length is the physical, or Euclidean, distance between the first point and the last point of the captured writing.

Average Curvature per Stroke: This is a feature based on the curvature value described above, except that the curvature value is calculated for each individual stroke in the handwriting sample, then averaged. The difference between this feature and the global curvature value is that by examining the curvature of the individual strokes, it is possible to obtain a more insightful measure of the depth of the local curves in the handwriting. For a graphical

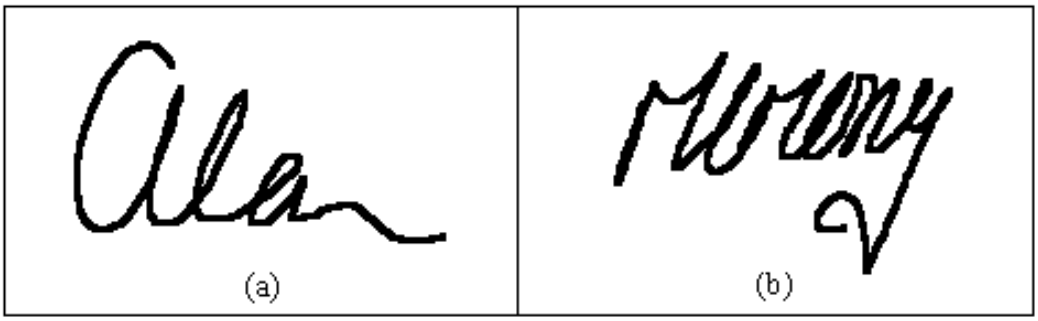

Figure 3.19: *Different handwriting samples can result in quite different curvature values. For example, (a) shows a sample in which the writing is quite flat and not well-formed, resulting in a curvature value of 3.96. Conversely (b) shows a sample with a much more pronounced forming of the handwritten characters resulting in the higher value of 5.22.*

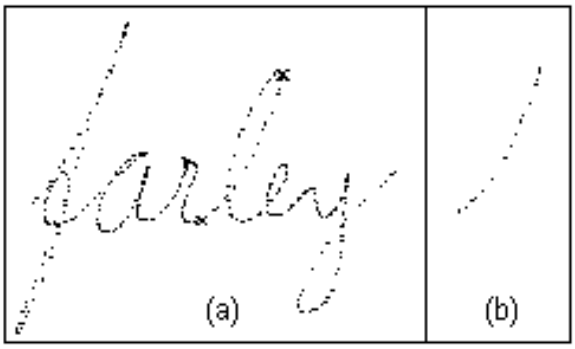

Figure 3.20: *The process of calculating the average curvature per stroke. (a) shows the entire handwritten word and (b) shows an isolated view of one of the extracted strokes.*

representation of an extracted stroke, refer to Figure 3.20.

Number of Strokes: The number of strokes is an easily understood feature representing the number of strokes in a signature. This is indicative of how many segments or states the handwriting goes through during the production of the signature. This feature remains quite stable over a user's various samples as even with the natural variations in a user's signature, the segmentation remains quite similar. The number of strokes is a non-trivial feature for a forger to reproduce as the segmentation is based purely on the pen-tip velocity, which is not visible with just a written version of the word. See Figure 3.21 for a pictoral representation of the stroke segmentation. Note that this figure depicts two different handwritten samples produced by the same author. The

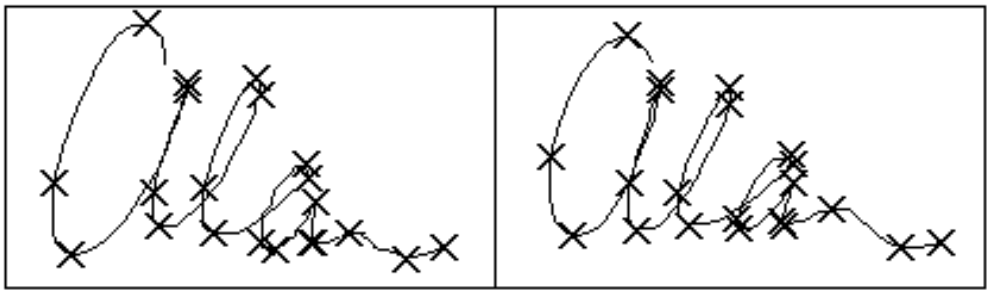

Figure 3.21: *Two signatures sections produced by the same author illustrating the consistency in the number of strokes. The crosses on the handwriting represent the stroke boundaries. Both of these samples have 21 strokes and as can be seen the segmentation is quite consistent.*

segmentation of both words is almost identical and the number of strokes is very similar in both samples.

Mean Ascender Height: This feature refers to the mean height of "ascenders" in the handwriting. Ascenders are letters such as 'd', 'f' and 't' in which a part of the letter extends above the main body of the sample [11]. Formal detection of ascenders in the body of a signature involves computing the mean of the data, as well as points at one quarter and three quarters of the maximum height. The peaks of ascenders then are defined to be the local maxima in the y direction that are above the three quarter mark. The distance between a local maximum and the y mean is found and this distance is taken as the height of that ascender. The mean height for all ascenders is then used as the value for this feature. Figure 3.22 shows a typical handwriting sample with ascenders, mean y value and ascender height (as well as some other details) clearly labelled.

Mean Descender Depth: Descenders here are essentially the opposite of ascenders. They are letters such as 'g', 'y' and 'j' that typically contain parts extending below the main body of the sample (note it is possible for an individual letter to be both an ascender and a descender - the letter 'f' is sometimes written in this way). Finding the descender extremities is done in a similar fashion to ascenders and uses the same frequency histogram. The descender extremities are the local minima in the y direction that fall below the lower quarter of the sample. The depth value for each extremity is measured as the distance between the local minimum and the y mean expressed as a positive integer. The depth values for all descenders are averaged to give the value for this feature. Figure 3.22 shows a typical handwriting sample with some

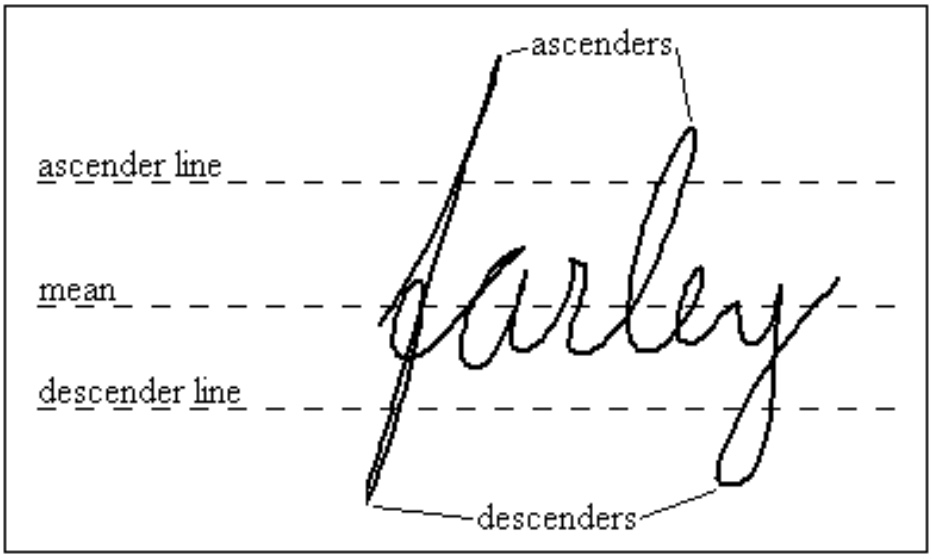

Figure 3.22: *A typical handwriting sample with labels indicating the ascenders, descenders, mean vertical displacement, ascender height and descender depth.*

examples of descenders.

Maximum Height: Maximum height is the distance between the lowest point in the handwritten word (the depth of the lowest descender) and the highest point in the word (the height of the highest ascender). Note that this calculation ignores 'i' dottings and 't' crossings or other such artifacts occurring in the handwriting. Also removed from consideration is the final trailing stroke in a signature - in examination of the trailing strokes in different signatures produced by the same signer, the height of this stroke was found to be by far the most variable. The maximum height feature using the remaining captured points reflects, to some extent, the "flair" with which the author writes and the maximum distance typically traversed by the pen tip. This feature remains reasonably stable across several written samples. Figure 3.23 illustrates maximum height.

Maximum Velocity: Maximum velocity is another purely dynamic feature of handwriting. There are two advantages in using a feature like this - firstly, it is something that a genuine writer is capable of easily reproducing and secondly, a potential forger will have great difficulty in reproducing it. The calculation of pen-tip velocity is done in terms of component velocities v_x and v_y, calculated as the first derivative of the x and y streams:

$$v = \sqrt{v_x^2 + v_y^2}$$

Figure 3.23: *The maximum height of a signature or handwritten word is defined as the distance from the top of the highest ascender to the bottom of the lowest descender. The vertical line seen here to the right of the writing sample is the maximum height, and in this case is calculated as 1,005 pixels.*

While maximum velocity is subject to some variability, it remains a valuable feature because it is unable to identified by a potential forger in an off-line copy of the signature.

Average Velocity: The average velocity is a personal measure of the how fast the pen-tip is travelling across the surface of the tablet. Velocity in handwriting has been widely studied, particularly in the field of signature verification, and is regarded as one of the most stable of all features and one of the best single features to use in HSV [51, 112, 109]. This feature is simply calculated as the mean of all individual velocity values (there is one velocity value for each pair of consecutive points).

Standard Deviation of the Velocity: This feature is calculated as the standard deviation of all the individual velocity values in the writing sample. It is a measure of the variation of the velocity values characteristic to the signer.

Average Absolute Acceleration: The absolute acceleration refers to the absolute value of the acceleration and deceleration measurements. It is computed as the second derivative of the data stream (or the derivative of the velocity values calculated earlier). The average absolute acceleration captures the mean rate of change in velocity in both positive and negative directions and was found to be another useful feature in verifying signers. The third derivative of the data stream (the derivative of the acceleration) is often referred to as "jerk". Jerk was examined as a potential feature, but did not prove to be either repeatable or individual and was abandoned as a result.

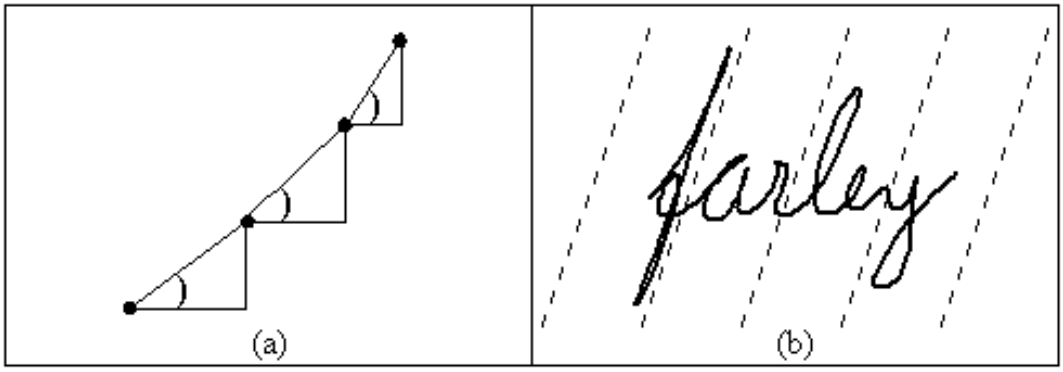

Figure 3.24: *The gradient of the line between each pair of consecutive points is determined (a sample of which is shown in part (a)), and the mean of those values found - this mean is the slant of the handwriting. Part (b) illustrates the computed slant value, drawn as a series of dotted lines laid over the handwriting sample.*

Standard Deviation of the Absolute Acceleration: This is a measure of dispersion of the absolute acceleration values. It captures the consistency (or lack thereof) with which a user's handwriting accelerates.

Maximum Acceleration: While this feature is less stable than some others, the purely dynamic nature and difficulty in forging still make it a useful characteristic.

Maximum Deceleration: Similar to maximum acceleration, this is a purely dynamic feature that measures the rate at which the pen-tip's velocity decreases as it approaches the end of a stroke.

Handwriting Slant Using All Points: The calculation of handwriting slant was a major focus during feature set development as it bears much importance in handwriting analysis [19, 59, 53]. The problem of slant calculation is not a trivial one and several different approaches were considered, each yielding their own insight. The first approach to calculating handwriting slant is a naive approach involving the use of all captured writing points. The points are spatially resampled and the angle (expressed as a gradient) between each pair of consecutive points in the signature is calculated, giving several gradient values (see Figure 3.24(a)). The slant using this first technique is then given by the mean of these gradient values. Note that 'i' dottings, 't' crossings and other such artifacts are removed from consideration here (as well as in other discussed forms of slant calculation).

Handwriting Slant Using "Long Stroke" End-points: This technique for

Figure 3.25: *"Long strokes" extracted from a typical handwriting sample. The long stroke is represented as bolded handwriting with the remainder of the handwriting appearing as a broken line in the background.*

calculating handwriting slant is based on the extraction of particular type of stroke, referred to here as a "long stroke". The definition of a long stroke is two-fold: firstly, the series of points between each vertical minimum and following vertical or horizontal maximum (whichever is encounted first) are extracted; secondly, the long stroke is retained if and only if the stroke path length is greater than some pre-defined threshold (experimentation was performed to find the threshold producing the most visually accurate slant and the final value was set at thirteen). Figure 3.25 illustrates a number of long strokes extracted from a handwriting sample. Once the long stroke start and end points have been identified, the gradient of that stroke is given by the gradient between just those two points. The mean of all such gradients then gives the handwriting slant.

Handwriting Slant Using All Points of "Long Strokes": This approach also uses long strokes and a technique very similar to that used to obtain the "Handwriting Slant Using All Points". After the long strokes are extracted and the constituent points spatially resampled, the gradient between each pair of consecutive points within the long stroke is calculated. The mean of these gradient values gives the handwriting slant. In experimentation, this method of slant calculation was found to be far less stable that the method using regression (described below) and was eventually removed from consideration in the neural network.

Handwriting Slant Through Regression of "Long Strokes": This approach is slightly different from slant calculation using long stroke end-points. The extraction of long strokes here is again done in the same fashion as described previously and illustrated in Figure 3.25. The difference comes in the actual

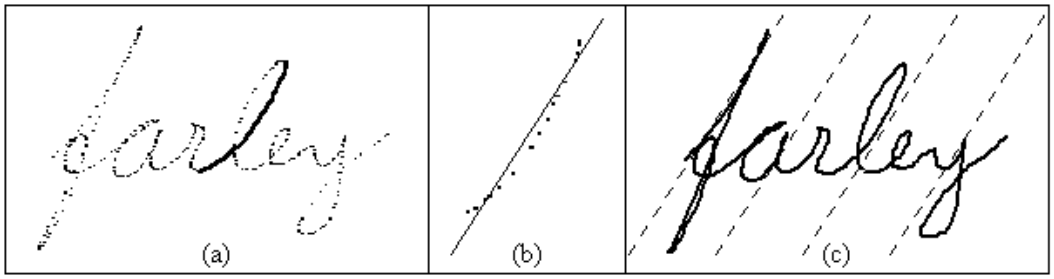

Figure 3.26: *Calculation of handwriting slant through regression of "long strokes". (a) shows one of the long strokes extracted from a typical handwriting sample. (b) shows a close-up view of that same stroke with the straight line being the line-of-best-fit as produced by simple linear regression. The gradient of this line is taken as the handwriting slant. (c) shows the same handwriting sample used in (a) and is overlayed with a series of straight lines parallel to the calculated slant using the regression of long strokes.*

calculation of the slant where linear regression is performed using all of the points in the long stroke. The mean of this value taken over all long strokes in the signature is used as the final slant. This procedure is illustrated in Figure 3.26.

Handwriting Slant Using Cai and Liu Technique: This method of determining slant is a technique borrowed from handwriting recognition [19]. This approach involves firstly calculating the centroid c_i for each row i in the signature within the vertical inter-quartile range (that is, ignoring ascenders and descenders that may skew the slant value if they are at one end of the signature), and obtaining h row centroid points, where h is the height of this range in pixels. Linear regression is then used to best fit a straight line through the centroid points. The estimated slant is the slope of the straight line, given by:

$$Cai - Liu\ slant = ctan^{-1}\left(\frac{S_{xy}}{S_{yy}}\right)$$

where:

- $S_{xy} = \sum_{i=0}^{h-1} G(i)(c_i - \overline{x})(i - \overline{y})$;
- $S_{yy} = \sum_{i=0}^{h-1} G(i)(i - \overline{y})^2$;
- $G(i)$ is the weight associated with the i^{th} row (the number of handwriting strokes crossed by a horizontal projection at i);

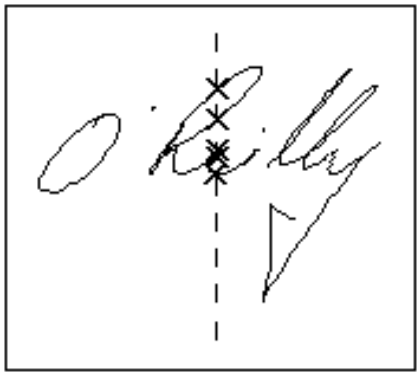

Figure 3.27: *The dotted line represents a single vertical projection, one of many used in the calculation of vertical overlap. The crosses are the points of intersection between the vertical projection and the handwriting stream (there are five in this instance). The average number of intersections is then a measure of the degree of handwriting slant (the higher the slant the higher the number of intersections).*

- $(\overline{x}, \overline{y})$ is the centroid of the signature;
- *ctan* is the complex circular tangent.

Handwriting Slant Based on Vertical Overlap: Vertical overlap measures the average number of handwriting strokes crossed by vertical projections through the handwritten sample. Unlike other methods of handwriting slant calculation, this method does not attempt to calculate a value for the gradient of the handwriting. The relationship between vertical overlap and slant is based on the fact that a more pronounced handwriting slant will result in a higher value for vertical overlap. In experimentation this feature was found to be highly stable (see Figure 3.27) and is not a feature obvious to potential forgers.

Calculation of vertical overlap is performed by making a series of vertical projections along the entire length of the writing sample. For each such vertical projection the number of strokes crossed is determined and averaged across all vertical projections (see Figure 3.27 for an example of a single vertical projection and associated overlap).

Stroke Concavity: Stroke concavity is a measure of how close the average stroke is to being a straight line. A stroke of high concavity does not closely follow the imaginary line drawn from the stroke start-point to the end-point. Calculation of the concavity value is done by firstly performing linear regression

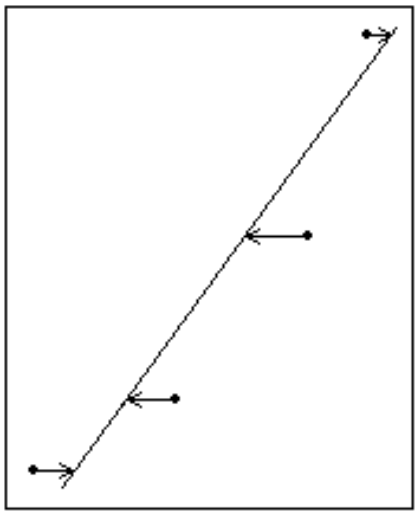

Figure 3.28: *Stroke concavity is depicted in this figure, showing a closeup of a stroke segment with a line-of-best-fit (found by simple linear regression) drawn through four points. The concavity is then found by taking the square root of the sum of squares of the minimum distance from each point in the stroke to the line-of-best-fit.*

using the points in the stroke to obtain the line-of-best-fit. This feature is then a measure of how well the points in the stroke "fit" that line, or how well the points are approximated by that line. The second step is to apply the following formula for each stroke:

$$Stroke\ Concavity = \sqrt{\sum_{i=1}^{n}(s_i - r_i)^2}$$

where:

- n is the number of points in the stroke;
- s_i is the i^{th} point in the stroke;
- r_i is the coordinate along the line-of-best-fit that is the least distance from s_i.

The stroke concavity is taken as the mean of the individual concavity values for each stroke in the sample. Figure 3.28 gives a graphical depiction of $s_i - r_i$ for a number of points.

Horizontal Velocity: Horizontal velocity is the average velocity over the x direction. It is a measure of how fast the signature moves horizontally and

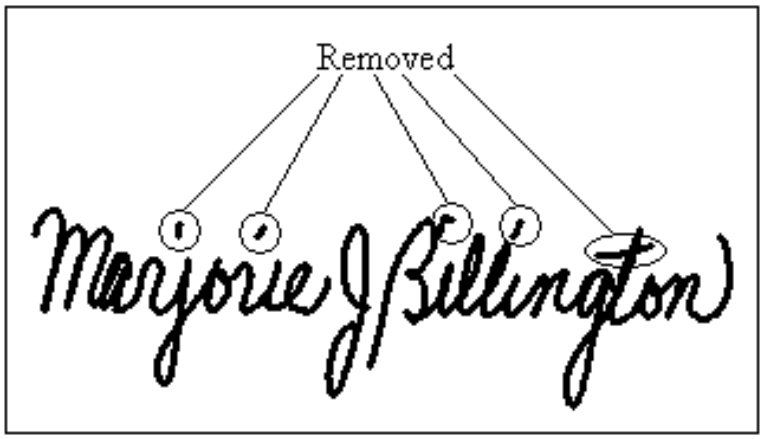

Figure 3.29: *Depending on the feature used, it may be necessary to remove certain pixels from calculation. Typically, fragmented information such as the dotting of 'i's and the crossing of 't's are removed.*

is related to pen-tip velocity, cursivity, horizontal length and acceleration. It is impossible for a potential forger to discern the horizontal velocity from an off-line copy of the writing. This feature is calculated as the ratio of horizontal distance to the duration in which the body of the sample was produced (artifacts and the duration associated with their production are removed from the calculation). Figure 3.29 illustrates which parts of a writing sample are removed from calculation.

Mean Pen-Tip Pressure: Pen-tip pressure is a measure of the amount of vertical pressure being applied by the pen to the top of the tablet. This is an option available on almost all current tablet and stylus hardware and is typically measured by an accurate sensor in the tip of the pen. Other verification software makes use of more complicated characteristics such as the breakdown into a horizontal and vertical component of the pen-tip pressure or the breakdown of the angle at which the pen is held. Because of the fact that angular pressure breakdowns are unavailable in much of the hardware, this verification system has been restricted to the assumption of a single *pressure* profile. Additionally, if pressure values are unavailable, all pen-down occurrences have pressure set to one and pen-up occurrences set to zero.

Pressure, like most features used in this system, is very difficult for a forger to discern from an off-line copy of the handwriting. Although pen-tip pressure is less stable than other dynamic features such as velocity, it is included because of the difficulty that a potential forger has in accurately simulating the pressure

profile. Mean pen-tip pressure used in this feature is simply the average of all non-zero values in the pressure profile.

Standard Deviation of Pen-Tip Pressure: The standard deviation of the pen-tip pressure is a gauge of how much a writer typically varies his or her pen-tip pressure during the course of the signature. Another purely dynamic feature, this is calculated as the regular standard deviation of the non-zero values in the pressure profile.

Maximum Pen-Tip Pressure: The maximum pen-tip pressure is the highest value in the pressure profile. Like all of the pressure features, the maximum pressure is not able to be extracted from an off-line copy of the writing and, while not as repeatable as velocity, is still useful in combination with other features.

Minimum Pen-Tip Pressure: The minimum pen-tip pressure is the lowest non-zero value in the pressure profile. This feature is not able to be extracted from off-line copies of the writing and previous researchers have found this feature to be quite useful [33].

Degree of Parallelism: The degree of "parallelism" refers to the extent to which slant remains consistent throughout the entire sample. This is a feature intrinsic to a writer's natural handwriting and is a characteristic naturally produced without conscious thought. The main problem with this feature is that users tend to write with higher parallelism if they are forcing themselves to write slower and more deliberately. However, in the natural flow of a user's handwriting (particular when performing a signature) the parallelism values are stable.

The calculation of the degree of parallelism is based on the calculated slant of the handwriting. Long strokes are extracted and the standard deviation of the slant of the long strokes is obtained (therefore, a higher value for this feature indicates a lower consistency of slant). Figure 3.30 depicts two handwriting samples, one of which has a high level of parallelism and one has a lower level.

Baseline Consistency: The baseline of a single handwritten word is the line-of-best-fit drawn through the bottom of all non-descender characters (see Figure 3.22 for an example of descenders). The baseline is analogous to the position of the line when a user is signing on a ruled or dotted line. This is another very personal feature and is particularly representative of a signer's natural tendency when no ruled line is present as a guide. Some writers are naturally more irregular or "sloppy" when forming their baseline than others.

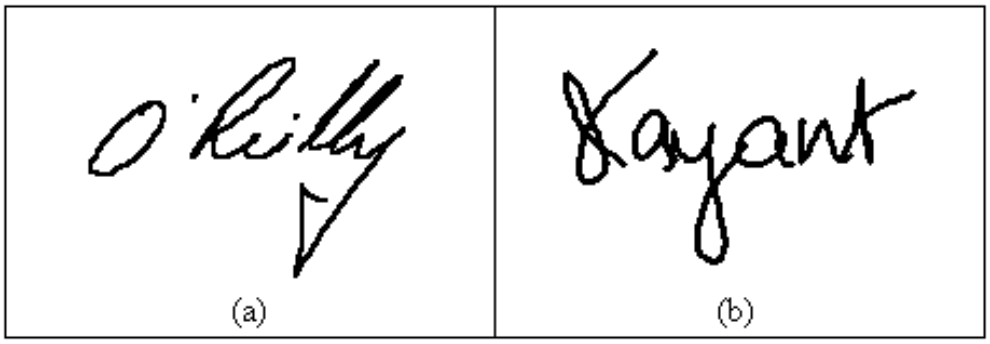

Figure 3.30: *Different degrees of parallelism. (a) and (b) are two sections of signatures taken from different authors with different values for parallelism. The sample in part (a) has a parallelism value of 0.10, whereas (b) has 0.34.*

Calculation of the baseline consistency is done by first extracting the set of minima from all non-descender characters. These minima are defined as y-minima that fall *below* the mean of the y data and *above* the lower quarter. Linear regression is then performed using these points and the line-of-best-fit is found. The baseline consistency is then given by the formula:

$$Consistency = \sqrt{\sum_{i=1}^{n}(b_i - r_i)^2}$$

where:

- n is the total number of baseline minima extracted;
- b_i is the i^{th} point in the set of baseline minima;
- r_i is the coordinate along the line-of-best-fit corresponding to the x value of b_i.

The process of baseline consistency calculation is illustrated in Figure 3.31.

Ascender-line Consistency: The ascender-line of a single handwritten word is slightly different to what would be expected given the previous definition of baseline. The ascender-line is the line-of-best-fit drawn through the upper extremity of ascender characters such as 't', 'b' and 'd' and ignoring fragments such as 't' crossings and 'i' dottings (as initially used in [11]). Given the ascender-extremity points, the calculation of the ascender-line consistency is done is the same fashion as the baseline consistency calculation.

Circularity: The circularity feature tries to capture how "round" or "distended" the handwritten characters are. It is measured as the ratio of the *area*

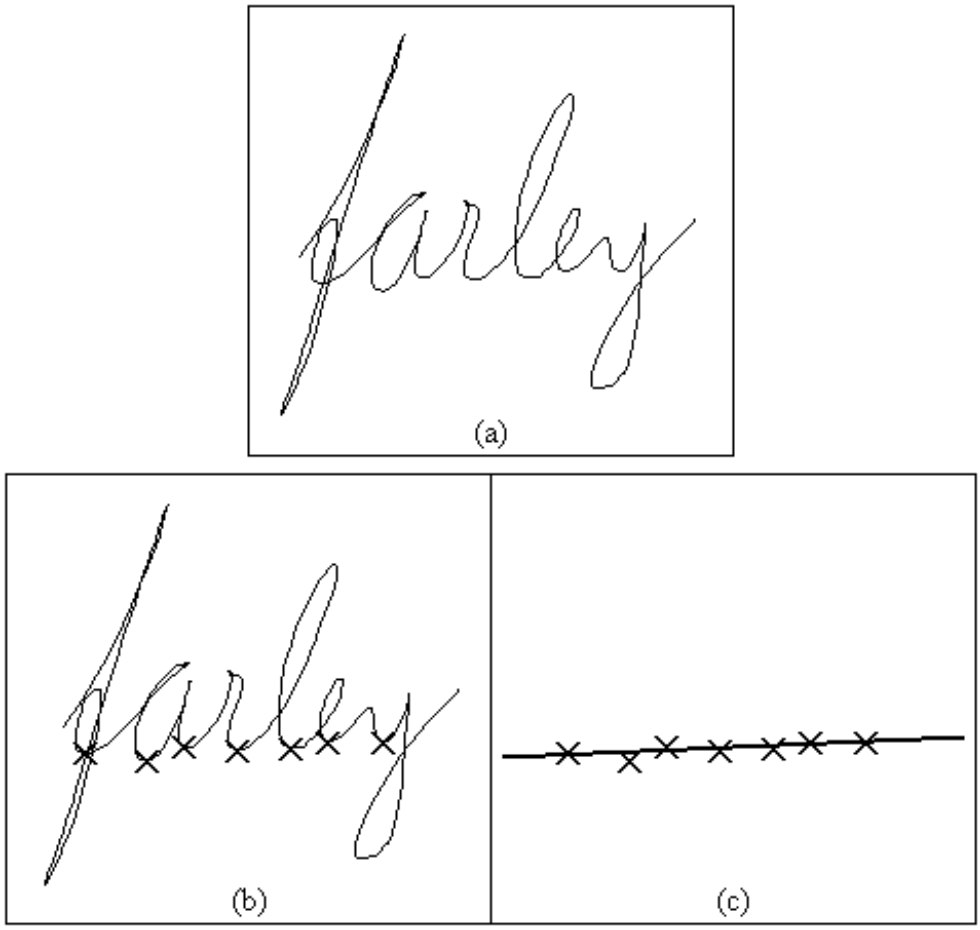

Figure 3.31: *The various stages in the calculation of baseline consistency. (a) shows the original handwritten word, (b) shows the extracted minima for non-descender characters and (c) shows the line-of-best-fit calculated for these points using linear regression. The baseline consistency is then the square root of the sum of the squares of the distances between the extracted minima and the line. The baseline consistency of this handwriting sample is 25.2.*

to the horizontal length. This feature is one that is quite difficult for a forger to judge so proves useful in preventing false acceptances. In addition, circularity was found to be quite a personalised feature and similar features were found to be useful in other studies. [55] described a system using something similar (under the name "sphericity") as one of only five discriminating features used in script, language and writer identification work.

Circularity is one of the most computationally expensive features in the feature set as it requires several iterations through the x and y profiles. The first step in the calculation is to extract the various components (areas of connected handwriting) within the sample. Calculation of circularity is then done separately on each of these components as described below.

For each x value in the component, a vertical projection is taken and the position of each of the points of intersection with the handwriting is found. It should be noted that one of the main difficulties in this calculation (and also with the vertical overlap feature) is in determining exactly when this intersection occurs. The problem lies mainly in the fact that with an on-line handwriting system there may not be an *actual* intersection between the projection and one of the recorded points (in fact, it is less likely that a direct intersection will take place). It is therefore necessary to perform an iteration through the recorded points in the current component and deem that an intersection has occurred if and only if there are two consecutive points for which:

$$(x[i] < x_p)\ and\ (x_p < x[i+1])$$

where:

- x_p is the x value of the vertical projection (the line with the equation $x = p$);
- $x[i]$ is the i^{th} point in the x data stream;
- $x[i+1]$ is the $(i+1)^{th}$ point in the x data stream.

The next step is to determine the height, or more specifically the y value, of the intersection. This is found as follows:

$$y_{int} = \left(\left(\frac{x_p - x[i]}{x[i+1] - x[i]}\right) \times (y[i+1] + y[i])\right) + y[i]$$

where y_{int} is the y coordinate of the point of intersection between the handwriting and the projection.

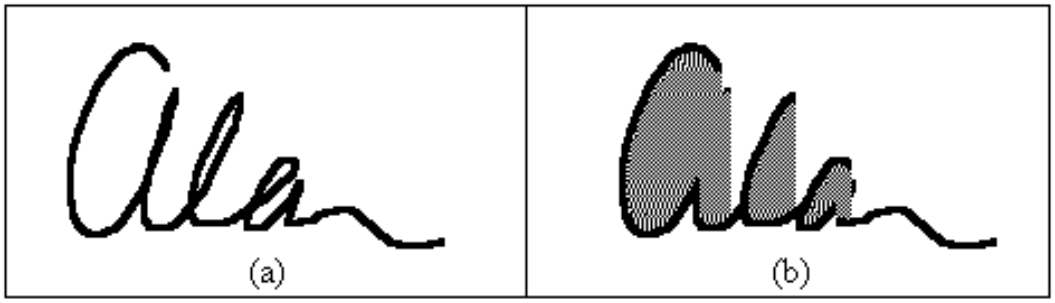

Figure 3.32: *The area of a signature. (a) shows the original sample and (b) shows the calculated area. In (b) the black lines represent the vertical extremities (the maximum and minimum intersections with vertical projections) and the shading shows the area of the signature segment. The area of signatures with multiple components is found by summing each of the independently calculated component areas.*

For each projection the heights of the various intersections are found and the difference in height between the uppermost intersection and the lowermost intersection is found and added to the cumulative total for area. If there is only one intersection (that is, a joining or trailing stroke) then one is added to the area value for the component (the area of a single horizontal line is defined to be one). Once projections are made for all x values in the component the area of that component is known. The ratio of the summed area to the summed horizontal length across all components is then calculated and gives circularity. Figure 3.32 depicts the calculated area.

Area: The actual area of the handwritten word is used as another feature. Calculation of area is performed in exactly the same way as described as a part of *circularity* calculation and shown in Figure 3.32 (the value is retained from the circularity calculations).

Middle-Heaviness: Middle-heaviness is a term used to describe the percentage of the handwritten samples bounding box that is interior to the signature itself. It was called middle-heaviness because it is a measure of the concentration of the handwriting around the midpoint as opposed to featuring high ascenders and low descenders. Calculation of this feature is done by first finding the area of the signature (as performed previously) and dividing this value by the area of the bounding box. The bounding box is a rectangle drawn around the sample using the the two extremities in each of the x and y data streams (with artifacts removed). This feature is illustrated in Figure 3.33.

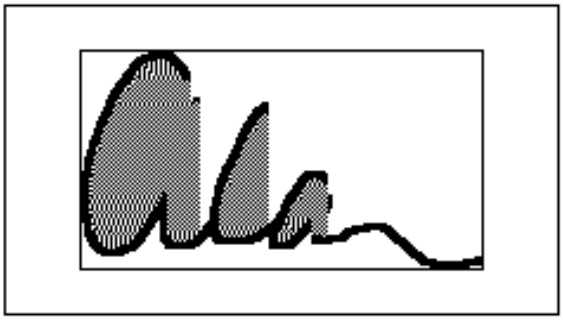

Figure 3.33: *This figure illustrates "middle-heaviness", which is defined as the percentage of the bounding box of a signature that is interior to the sample itself. The bounding box is shown in the figure and all shaded pixels are points interior to (or part of) the sample. The area of the shaded pixels is then divided by the area of the bounding box to give middle-heaviness.*

Component Physical Spacing: The average spacing between the components is again indicative of a writer's natural style and is very stable across multiple instances of the signature. The calculation of this feature involves simply taking the Euclidean distance between the last point sampled in a component and the first point sampled in the following component (if any). This value is calculated for each pen-up instance and averaged to obtain the final feature value. See Figure 3.34 for an illustration of the component physical spacing. When there is only one component a default value of zero is returned.

Component Time Spacing: This feature refers to the average duration of a pen-up instance in a signature and is often referred to as pen-up time [46]. It is slightly less stable than physical component spacing but this is made up for by the fact that it is impossible for a forger to copy this feature from an off-line signature image.

3.4.4 Experimental Setup

This section details the experimentation performed as part of the neural network HSV system development. It is split into three parts: a discussion of the benchmark linear network results, the multi-layer perceptron development (including the structure and network parameters resulting in the most robust and accurate system) and finally the different training approaches (including the classification error rates and timing results using different training algorithms and different compositions of the training set). In all experiments five genuine

Figure 3.34: *The physical spacing between components is a measure of the average distance between the last point sampled in a component and the first point sampled in the immediately following component (if any). This distance is illustrated in the figure and defaults to zero if there is only one component.*

signatures are used to train the network unless otherwise stated.

Linear Network Development

As discussed earlier in the chapter, linear networks are neural networks with no hidden layers or nodes. Despite the limitations of these networks, they provide a useful benchmark against which to measure more complex techniques. It is not uncommon to find that problems perceived to be quite difficult are handled well by linear approaches. To this end a linear network was developed and trained to verify the handwritten signatures in the database. Training was done using the pseudo-inverse technique (as recommended in most texts), five genuine signatures and various combinations of negative examples (see below for a complete discussion of negative examples in the training set). Typical training convergence behaviour of a linear network is illustrated in Figure 3.35(a).

The performance of this network is actually quite reasonable, with the lowest overall error rate achieved being 6.1% (3.4% FAR and 2.7% FRR) using a set of ten other user's genuine signatures as negative examples. These performance rates imply that the handwritten signatures used in this study are somewhat linearly separable via the extracted features. As expected, the stability of the network is quite poor however in relation to that displayed by networks with hidden layers.

Multi-Layer Perceptron Development

Most of the experimentation done in the neural network development involved MLPs as this particular structure is well suited to the parametric HSV problem. The structural experimentation is constrained to fully-connected multi-layer feed-forward networks. That is, the networks have a distinctly layered structure, all nodes in layer i have connections to all nodes in layer $i+1$ (see Figure 3.7) and no connections are made to previous layers.

The input layer always consists of n nodes where n is the number of features in the set and the output layer consists of a single node that calculates the weighted sum of the connections coming into it. The final output of the network is a confidence value indicating the likelihood that the test signature was performed by the same person that provided the reference signatures used in training. The confidence value is compared to a threshold and the test signature is verified if the confidence exceeds this threshold or rejected otherwise.

Training via the back-propagation algorithm (other training algorithms are considered below), the MLPs are able to fit genuine signatures much better than linear networks. The result is a lower overall error rate along with a much more pronounced convergence than was achieved with the linear network (in terms of the training the network, see Figure 3.35) and a clearer class separation between the two classes.

What follows now is a discussion of the different network parameters and architectural issues explored during experimentation, followed by an examination of MLPs with two hidden layers and the different training scenarios.

- *Number of Nodes in the Hidden Layer*: The most influential adjustable parameter within the model constraints is the number of nodes (or units) in the hidden layer. In theory, architecture determination can be treated as a kind of optimization problem exploring various possible designs looking for the most suitable structure. In a MLP with one hidden layer, once the training features are decided upon, the optimization problem reduces to a decision over the number of units in the hidden layer. This is essentially a linear search of a noisy function (each time a network is trained a slightly different error rate results). In the NN HSV system, the search involved imposing a minimum of two and a maximum of 120 (roughly three times the number of input units) on the number of hidden units and experimenting exhaustively within this range. Values between fifteen and thirty proved to be the most successful (in terms of overall error

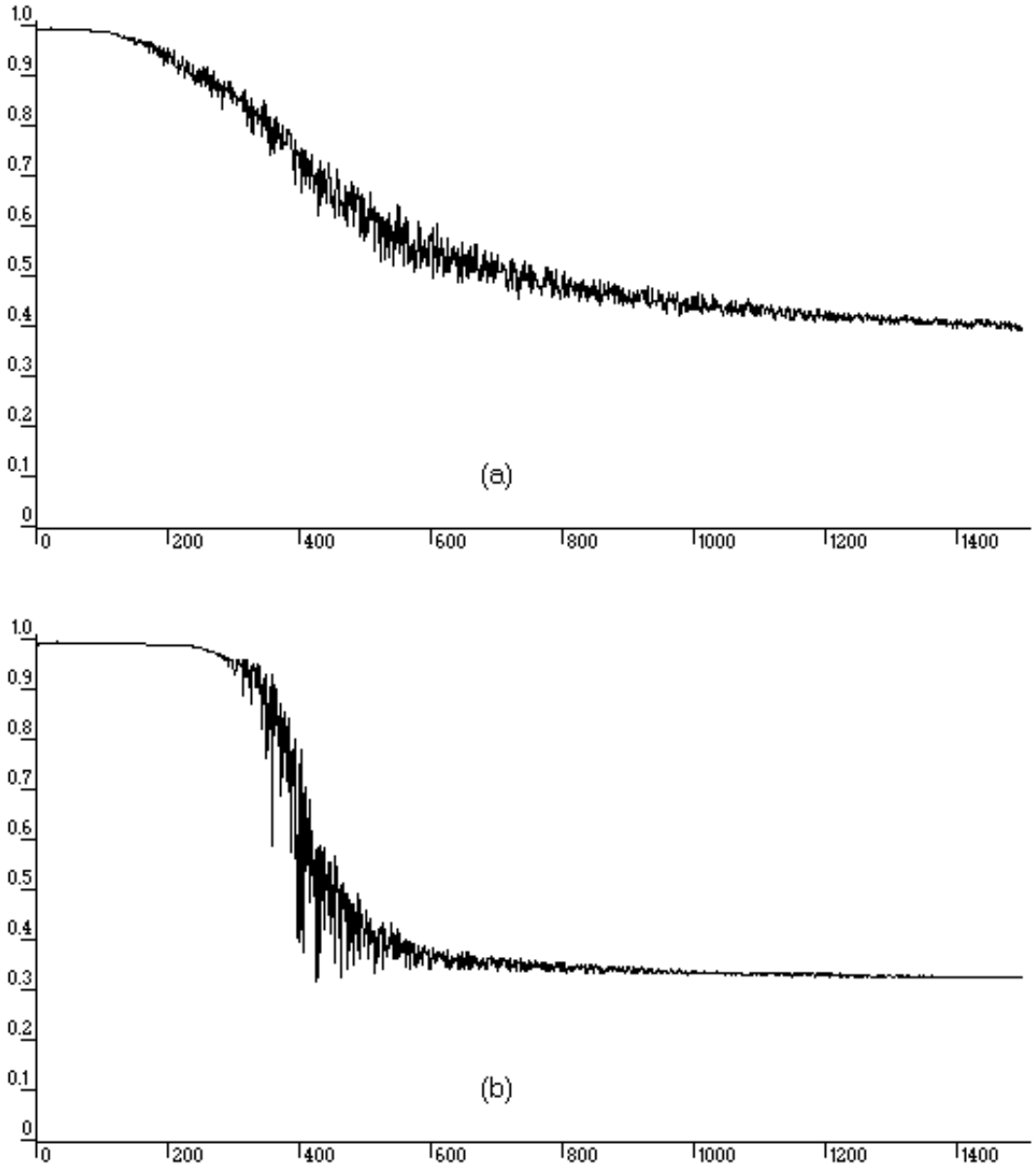

Figure 3.35: *Convergence of training and verification errors. (a) In a linear network. (b) In a multi-layer perceptron with a single hidden layer.*

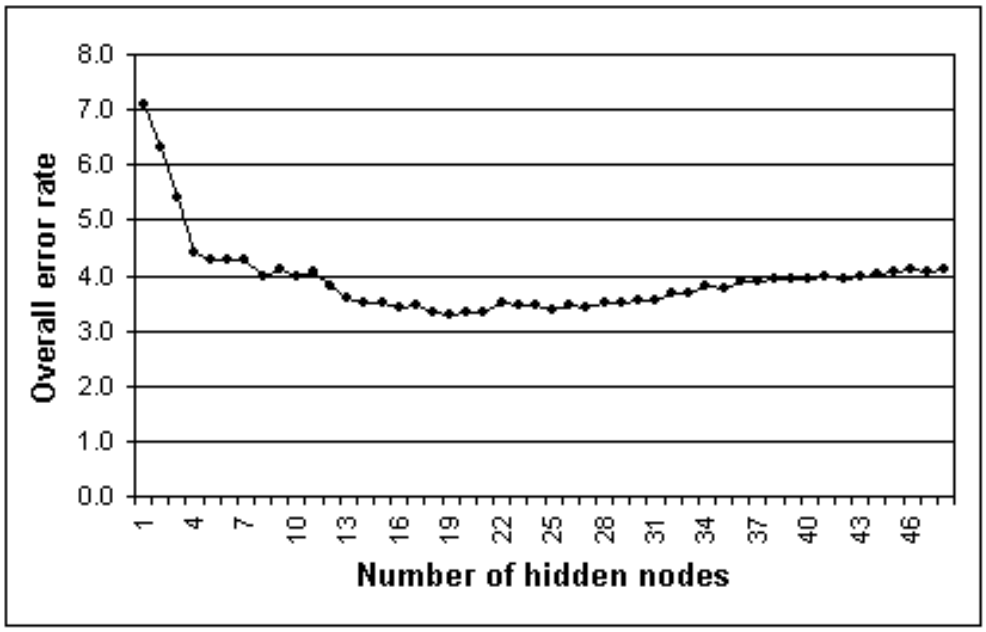

Figure 3.36: *Error rates resulting from varying the number of hidden nodes in a MLP with one hidden layer.*

rate) with nineteen hidden nodes used in any further NN experimentation. A plot of the number of hidden nodes (from two up to sixty) versus the resulting overall error rate is presented in Figure 3.36.

- *Learning Rates*: As discussed earlier in the chapter, the learning rate in a neural network controls the speed with which the network learns. A higher learning rate causes the algorithm to converge faster but may introduce instabilities, especially if the data is noisy. Experimentation involved exhaustively searching all learning rates from 0.05 to 0.95 with increments of 0.5. Small changes in the learning rate were not found to have a significant influence on the final error rates and a value of 0.6 produced consistently acceptable results. Degradation occurred when the learning rate moved much closer to zero or one.

- *Momentum*: Similar to findings regarding the learning rate, small changes in the value for momentum did not have a large impact on system accuracy. A value of 0.3 was used in the final version of the system.

- *Activation Functions*: Sigmoidal activation functions were used in each of the nodes in the NN. It is generally accepted that sigmoidal activation functions are appropriate in scenarios such as HSV and experiments

conducted versus other activation functions supported this.

The final overall error rate using the above structure and network parameters and the most successful structure was 3.3% (2.0% FAR and 1.3% FRR), which represents a clear improvement over linear techniques. The training scenarios are presented following the discussion of MLPs with more than one hidden layer.

MLP With Two Hidden Layers: Further experiments were conducted using MLPs with two hidden layers in an attempt to better fit the signatures. The RMS error for the genuine signatures was lower than for the single-hidden-layer MLP as the extra hidden layer enabled the underlying function to be more closely approximated. Experiments were conducted using various combinations of unit numbers in the two hidden layers. It was found that the results are best when the number of units is equal to around half the number of input units (similar to the network depicted in Figure 3.7(d)). The momentum value and the activation functions are the same as for the MLP with one hidden layer. The learning rate differed slightly and a setting of 0.3 resulted in the most stable network using the same training set contents (training sets are described below).

This more complex model unfortunately has a detrimental effect on the performance of the network (both in terms of classification error and processing speed). The execution of the network (classifying a test signature) is slightly slower and the training procedure takes on average almost double the amount of time to converge. A typical example of the time spent training the three network structures can be found in Table 3.2. In this table, the "number of epochs" is a measure of how many iterations each of the structures requires for convergence. This is less relevant that the overall time requirements (for example, a training epoch for a linear network is very different to a training epoch for a multi-layer perceptron with two hidden layers). The relative time is the average amount of time taken to converge, expressed as a percentage of the time taken by the two-layer MLP. Finally, the overall classification accuracy of the network deteriorated slightly as a result of the initial hidden layer. The network is able to fit the genuine signatures in the training set very closely, but overfitting seemed to occur as the FRR was increased. The final overall error rate using two hidden layers is 3.8% (1.3% FAR and 2.5% FRR).

A description of the different training approaches used is presented below. The experimentation and error rates apply to a MLP structure with one hidden layer unless otherwise specified.

Structure	Number of Epochs	Relative Time
Linear Network	1350	7.2%
MLP with 1 Hidden Layer	1000	60.1%
MLP with 2 Hidden Layer	1200	100.0%

Table 3.2: *The training performance of three network structures.*

Training Approaches

All of the different training approaches used in experimentation are described here. Experimentation included different training algorithms, different training sets and the use of forgeries (both skilled and zero-effort) in the training set. All discussions apply to the optimal network structure described above unless otherwise specified.

The first training aspect under consideration is the algorithms to perform the actual weight adjustments. The technical details and functionality of the MLP training algorithms have already been explained in Section 3.1.3. The performance of each of the three implemented training algorithms is discussed below.

- *Back-propagation*: This is the most widely used algorithm in neural network training due to its efficiency, simplicity and performance. Back-propagation has been used successfully in many different environments (including HSV [105, 88]) and results in the highest classification accuracy compared to the other implemented algorithms. The performance details using back-propagation are presented in Table 3.3 (error rates) and Table 3.4 (training time).

- *Conjugate Gradient Descent*: This is the major alternative to back-propagation but is not as widely used in HSV. The conjugate gradient descent algorithm converged much faster during training than back-propagation. Unfortunately, due to the fact that this training algorithm is suited to more complex networks (with several hundred weights) [13], the resulting error rate was somewhat worse than networks trained via back-propagation. The performance details can be found in Table 3.3 and Table 3.4.

- *Levenberg-Marquardt*: Experimentation was also done with the Levenberg-Marquardt training algorithm. The classification accuracy was slightly

Training Algorithm	FAR	FRR	Overall Error Rate
Back-propagation	2.0%	1.3%	3.3%
Conjugate Gradient Descent	1.9%	2.4%	4.3%
Levenberg-Marquardt	2.4%	1.6%	4.0%

Table 3.3: *The classification accuracy of the three implemented neural network training algorithms.*

Training Algorithm	Number of Epochs	Relative Time
Back-propagation	1000	4.6%
Conjugate Gradient Descent	147	0.8%
Levenberg-Marquardt	1280	100.0%

Table 3.4: *The convergence speed of the three implemented neural network training algorithms.*

better than that obtained using conjugate gradient descent but not as good as back-propagation. The training process however was exceedingly slow, which may be a problem in a large, dynamic database. Performance details can be found in Table 3.3 and Table 3.4.

The back-propagation algorithm seems to be superior in this HSV environment. Although it didn't converge as quickly as conjugate gradient descent, the classification accuracy was superior to the other two approaches and training error rates continue to improve after several hundred epochs. The actual "wall clock time" required for training to complete using back-propagation is typically between thirty seconds and two minutes. This training time is not prohibitively slow as it is generally a one-off cost and is not performed while the user waits.

Given that the training is unsupervised, a stopping condition (defined in Section 3.1.1) is needed to ensure that training time is finite and that the resulting network is not under-trained or over-trained. Experiments were conducted using heuristic approaches with different limitations and slightly different rules. The most successful method (and the method used in all training scenarios) is a two-fold rule of stopping if the RMS error reduces to below 0.05 or if this error fails to improve over a span of ten epochs. The first half of the condition ensure the training stops when error rate is low enough to be considered effective, and in a finite number of epochs. On some training runs, the RMS

error continued to reduce slowly for hundreds of thousands of epochs, so this element of the stopping condition protects against this occurrence. The second half of the condition is necessary as sometimes the RMS error reached a point beyond which it did not improve, irrespective of further time spent training. Ten epochs was experimentally found to be a large enough period of time to wait for at least *some* improvement. Note that if there is a deterioration of the error rate after a minimum has been reached, training will eventually stop due to the "n epochs without improvement" condition. Rather than use the now slightly deteriorated network, the attributes (weights) of the network revert to those which produced the lowest error rate.

One other point to note is that no limit is placed on the number of epochs spent training - in experimentation, the above stopping condition caused the training of the network to cease in a reasonable amount of time (a maximum of a number of minutes) in every instance. In practice it may be necessary for some (extremely large) maximum value to be placed on the number of epochs to prevent an exceedingly long training phase for a network that is regularly slightly improved. This was not found to be the case in any of the experiments conducted during the work described in this thesis, but this is by no means proof that the scenario will never happen.

The above stopping condition was used in all of the training scenarios described below unless otherwise specified. What follows now is a discussion of the different scenarios with respect to the makeup of the training set.

Training One Network with the Entire Database

This scenario involved training on the signature database as a whole, using forty-one input units (one for each feature), various different experiments with the size of the hidden layer and a single output unit (attempting to identify the author of the signature). The target output value was the identifier for the signer who provided the signature. That is, a single network was trained to identify the most closely matching user in the database for a given signature. This strategy was not greatly successful (it makes the assumption that the entire database of signatures is separable using a single set of weights) and the network displayed a large amount of confusion, resulting in an overall error rate of over 47%. Additionally, this approach is impractical in that the training takes a very large amount of time and re-training is necessary each time the population of the database changes.

Training One Network with the Entire Database and Multiple Outputs

This scenario works as above, except the number of output nodes is equal

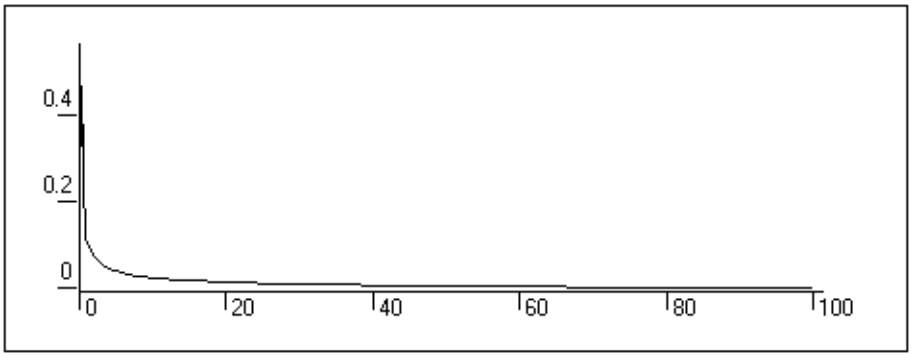

Figure 3.37: *Convergence of error rate using back-propagation in a typical MLP with no negative examples.*

to the number of users. When the network is presented with a test signature, the output node with the highest activation value is deemed the "winning" node and the corresponding user deemed to the be the author of the test signature. This is a multi-class classification problem and is often referred to as *one-of-N* classification. Slight improvements are achieved using this approach (an overall error rate of just under 45%), but the accuracy is still significantly inadequate for use in signature verification. In addition there is still the problem of re-training the network when a new user registers. The conclusion drawn from this is that both of the first two scenarios are impractical for signature verification.

Training Individual Networks for Each Signer

This scenario involved creating and training a single network for each individual, training using a randomly selected set of five genuine signatures for each user, with the remaining genuine signatures used in the verification set. This approach is very simple, fast and easily achievable (it is not necessary to obtain any forgeries). The problem is that the lack of negative examples in the form of forgeries (either skilled or zero-effort) makes the training process a somewhat haphazard approach as it is difficult to know with any certainty when to stop training (the most successful stopping condition was found to be simply to place a limit on the number of epochs). Error convergence occurs very quickly (see Figure 3.37) but the typical under-fitting causes problems when the network is tested using forgeries.

Classification of previously unseen genuine signatures was performed well (that is, there were very few false rejections), but classification of forgeries was quite poor (false acceptances were commonplace). This is not unexpected

however, as it's difficult for a network to recognise forgeries if it hasn't seen any before. The overall error rate using this approach is 42.5%, made up largely from false acceptances (31.2% FAR and 11.3% FRR).

Training Individual Networks for Each Signer using Negative Examples

This approach to training was found to produce the lowest overall error rate. As in the above scenario, a separate network was trained for each individual signer in the database. The difference is that here, in an attempt to improve the accuracy of the resulting system, negative examples (in the form of forgeries) are introduced into the training set (five reference signatures per signer are still used as the positive examples). Without the use of negative examples it is difficult to obtain effective threshold values on the neural network output, and the network is obviously less likely to recognise a forgery. Ideally the negative examples would be skilled forgeries, but these are rarely available in practical situations so an alternative must be found. Each of the alternatives explored with respect to negative examples is presented below and classification accuracy for all approaches is presented in Table 3.5.

1. The brute force approach is to use the entire set of reference signatures from other users in the database. This necessitates a change in the approach to training as genuine signature misclassifications can become insignificant in the makeup of the overall RMS error (as there are many more forgeries than genuine signatures). The modification involves applying a similar concept to the more powerful "loss matrix" (see Section 3.1.5) and specifically associating a more severe penalty with misclassifying genuine signatures. The results obtained using this approach were quite good, but this is impractical due the increased training time (training time increases linearly with respect to the number of users in the database) and the need to re-train the system every time a new user is added to the database. An overall error rate of 3.2% is obtained using this approach.

2. The impracticalities of the above approach render it unusable in any sort of real-world, dynamic environment. In order to cut down on the training time and the necessity to re-train, experiments were conducted using reference signatures from other randomly selected users as forgeries. Sets of ten, fifteen, twenty, twenty-five and thirty users were considered with the best results occurring with set sizes of twenty and twenty-five. When a new user is added to the database their negative training examples are

extracted from the users that are already present. Existing networks are not re-trained when a new user is introduced. The overall error rate, averaged over ten applications of the random approach, is 7.6%.

3. The random approach described above relieves the impracticalities of the brute force approach, but unfortunately the resulting error rate is too high and a compromise must be found. One of the problems with the random approach is that many of the selected signatures bear little or no resemblance to the reference signature they are trying to forge. As a result the negative examples are much less effective. In an attempt to create more effective negative examples, random noise (in the form of spatial resampling and geometric translation) is applied to the reference signatures and these are used as negative examples. The amount of noise was varied during experimentation, but no combination made this approach successful. The most successful classification rates are obtained when noise is applied to 25% of the sampled points, however the network still exhibits some confusion in this case. An overall error rate of 16.1% is the result.

4. The final (and most successful) attempt at selection of more appropriate signatures to use as negative examples is handled using a basic, very fast similarity measure described in previous work [86]. This similarity measure is based on string edit distance and is described in Appendix B. The nearest one hundred signatures according to this similarity measure are included as negative examples (the hundred signatures typically came from around twenty-five different signers). Due to the large number of forgeries a misclassification penalty (doubling the RMS error) is applied to false rejections. As the entire database needs to be searched when adding a new user, the training process is obviously going to take longer. Fortunately the total cost to the training process is not great and effectively lengthens training by around thirty seconds per hundred users in the database (training on Sun SparcStation 20) to find the similar signatures. This approach is realistic in that the training time, while increased, is not excessive and re-training of existing networks is not required when enrolling new users. The performance using this approach rivals that of the brute force approach at 3.3% overall error rate.

5. For the sake of interest and for comparative purposes, training was done using three skilled forgeries (this leaves either two or seven skilled forger-

Approach	FAR	FRR	Overall
1. All other reference signatures	2.4%	1.8%	3.2%
2. Random reference signatures	5.2%	2.4%	7.6%
3. Introduced noise	7.3%	8.8%	16.1%
4. Similarity measure	1.1%	2.2%	3.3%
5. Skilled forgeries	0.7%	1.6%	2.3%

Table 3.5: *The performance of the HSV system using different approaches to obtaining negative examples.*

ies per user for testing, depending on how many forgeries were originally captured). Use of skilled forgeries in the training set, as expected, results in a large improvement in error rates. The overall error rate using three skilled forgeries in training is 2.3%.

The results using the above approaches to the creation of negative training examples are summarised in Table 3.5. As can be seen from these results, approach five (using skilled forgeries) is the most successful, followed by using the entire set of genuine signatures from the database. Unfortunately these approaches are both impractical, but fortunately the performance of approach four (using other signatures that are "close" according to a similarity measure) still performs very well and is does not suffer from practicality issues. This approach is used to train the final version of the neural network based HSV system.

In summary, the final (most successful) version of the neural network based HSV system uses a single MLP with one hidden layer to model each user's signature and is trained using five genuine signatures and one hundred zero-effort forgeries (selected using a similarity measure). A summary of results using various MLP structures is presented in Table 3.6. Overall results deteriorated slightly when using two hidden layers in the MLP, which indicates that perhaps over-fitting is occurring as the training results do not generalise as well to the unseen signatures. Another aspect that can be seen in the summary is that the performance using a single MLP trained to *identify* any signer within the database results in quite a high overall error rate. The poor results indicate that a single, global weight vector is not sufficient for class separation in HSV systems with a large number of users. What is surprising is the low error rate obtained when a linear network is used. This suggests that the handwritten signatures in this database (through the extracted features) are somewhat

Structure	FAR	FRR	Overall
One MLP, single output unit	19.3%	27.9%	47.2%
One MLP, one output unit per signer	25.4%	19.5%	44.9%
One MLP per signer, no forgeries	31.2%	11.3%	42.5%
One MLP per signer, with forgeries	1.1%	2.2%	3.3%
One MLP per signer, two hidden layers, forgeries	1.2%	2.5%	3.7%
One linear network per signer, best case	3.4%	2.7%	6.1%

Table 3.6: *The performance of the HSV system using different MLP structures. Unless otherwise stated, the structure made use of a single hidden layer.*

linearly separable.

Figure 3.38 shows the performance of the final version of the HSV system using different threshold values. Included in the figure are the false acceptance rate (FAR), false rejection rate (FRR) and total error rate (the sum of FAR and FRR). Not included in the figure is the rate that some researchers report known as *equal error rate*. The equal error rate is the point at which the FAR and FRR are of equal value (that is, when the lines on the graph cross) - this value being 1.9%, occurring at a threshold of 0.51 (using one MLP per signer, trained with zero-effort forgeries). The lowest overall error rate was 3.3% (1.1% FAR and 2.2% FRR) occurring at a threshold of 0.42.

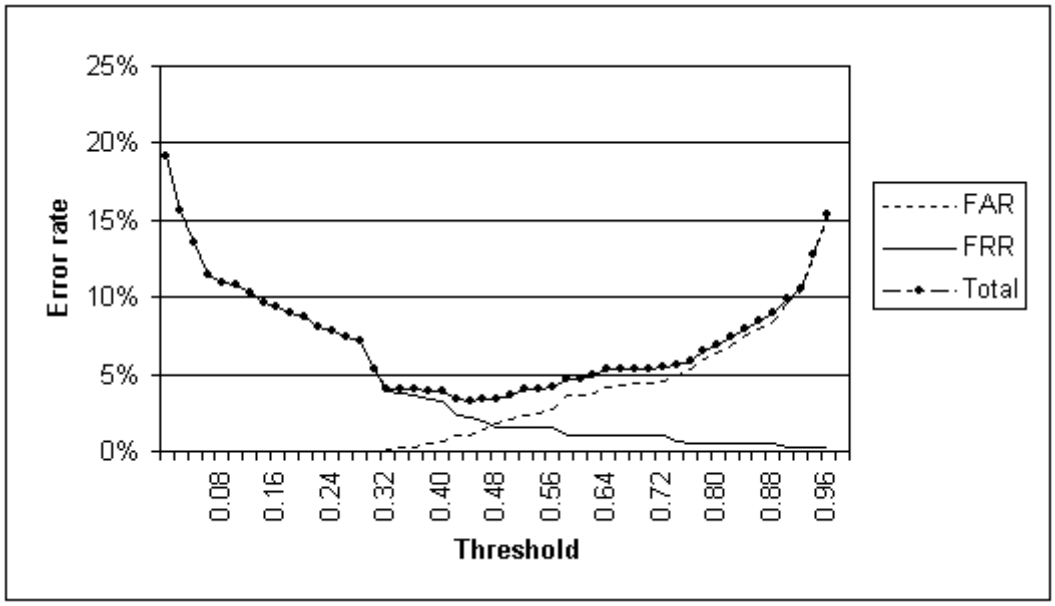

Figure 3.38: *The performance of the optimal network structure using different threshold values.*

Chapter 4

Hidden Markov Models

The previous chapter dealt with analysis of handwritten signatures through the machine learning technique of neural networks (NNs). The NN approach described is based on *global* features - that is, features that were extracted from the signature as a whole. No attempt was made to compare patterns that may exhibit temporal variations, that is, features that are *local* in nature.

Although the standalone system described in Chapter 3 performed quite well, it is clear that there is still a large amount of local information that is not being captured. This chapter describes another system that focusses on characterising that local information through the use of local features and a different modelling technique. The local features are described in Section 4.4.2 and the structure used in this chapter is a statistical signal modelling technique know as a hidden Markov model (HMM). It should be noted that there are NN structures, specifically time delay neural networks (TDNNs), that are capable of modelling sequences of data, however they do not have the representative power of HMMs in this scenario [10].

Hidden Markov models (also referred to as "Markov sources" or "probabilistic functions of Markov chains" [120]) are finite stochastic automata, with a powerful capability of modelling time-varying dynamic patterns. The original theory of HMMs appeared in a series of classic papers by Baum and his colleagues [4, 5, 6, 7, 8] in the late 1960s and early 1970s. Researchers have been using them for various modelling problems since as early as 1975 [2], particularly in speech recognition tasks [73, 120]. More recently they have undergone a popularity increase in handwriting analysis tasks including character recognition [146, 25] and signature verification [33, 32, 85].

The training of HMMs is done in a fashion similar to the training of NNs in the sense that a feature set is extracted from the signature and a series of

these sets is presented to the HMM. The HMM then tries to model the input data in some way. When testing a new signature (in signature verification tasks), instead of outputting a classification, the HMM outputs a *probability* indicating the likelihood that the test signature was produced by a given signer (or more specifically, the likelihood that the signature features could have been generated by the given HMM).

HMMs are naturally suited to modelling "flowing" entities such as speech and handwriting. They are made up of a series of states with transitions between these states. Signatures can be split up into different sections through a process of *segmentation*, with a similar number of sections to the number of states in the HMM. It is then possible to progress through the states of the HMM in a corresponding fashion to progressing through the segments of the signature. At each state, the *local* features of the signature (such as the average velocity, acceleration or duration in the current segment) are examined. *Local* features therefore can be naturally modelled by HMMs. A more thorough technical explanation of HMMs is presented in Section 4.1.

The remainder of this chapter is devoted to discussions of various design aspects of HMMs, the feature set used as well as all experimentation conducted and results obtained in the development of the HMM system.

4.1 The Theory of Hidden Markov Models

Hidden Markov models (HMMs) are a statistical method of characterising properties of segments of a pattern. They are quite good at separating a signal (in this case a stream of handwriting) into a series of *frames* or *states*, and then comparing the properties of these with corresponding states extracted from other signals (other handwriting samples). The underlying assumption of the HMM is that the signal can be well characterised as a parametric random process, and that the parameters of the stochastic process can be estimated in a precise, well-defined manner [120]. Signatures satisfy this assumption and later discussions support this claim.

Before discussing the mathematics behind hidden Markov models it is first necessary to understand the workings of Markov models (with no "hidden layer").

4.1.1 Markov Models

A Markov Model (MM), also known as a Markov chain [118], is a stochastic model that describes the probability of moving from one state to another based on some set of transition probabilities and the previously visited states. The *order* of the model describes how many previous states contribute to the current state's transition probability distribution. All models discussed throughout the remainder of this chapter are *first-order* models (often described as obeying the *Markov assumption* or *first-order assumption*), which means the transitions from state i to state j have the following property:

$$a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$$

where:

- a_{ij} is the probability of transiting from state i to state j;
- q_t is the state the model is in at time t;
- s_i refers to state i.

Put another way, the probability of being in state j at time t depends *only* upon the state occupied at time $t-1$ (say, state i) and the probability of transiting from state i to state j in a single step. Some argue that this is a limiting feature of HMMs [154] but it can also be argued that the first-order independence limits propagation of errors and generally simplifies the modelling process.

A Markov model therefore is defined as a 3-tuple $\lambda = (A, \pi, S)$ with the following properties:

- S is a set of states $S = s_1, s_2, ..., s_N$, where N is the total number of states. The state at time t is called q_t;
- A is an $N \times N$ state transition matrix that contains the transition probabilities a_{ij} where $i = 1...N, j = 1...N$. The transition probability from state s_i to state s_j is $a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$ (the probability that the state at time t is j given that the state at time $t-1$ was i), while the self transition probability is denoted as a_{ii}. The sum of the outgoing transition probabilities is $\sum_{j=1}^{N} a_{ij} = 1$;
- π is the set of *initial* probabilities $\pi = \{\pi_i | i = 1, ..., N\}$ where $\pi_i = P(q_1 = s_i)$ (that is, the probability that the start state is s_i);

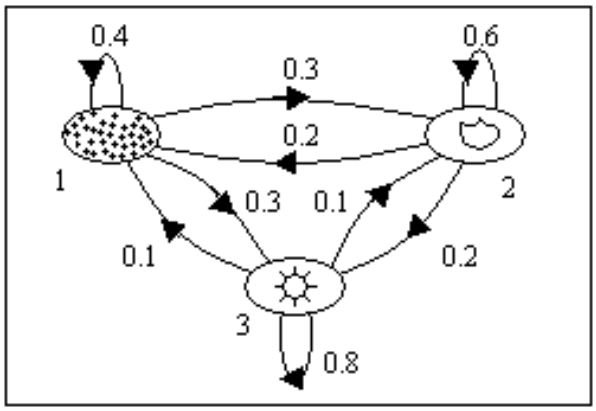

Figure 4.1: *A Markov model of the weather.*

- The probability of a particular state sequence $Q = q_1, q_2, ..., q_T$ given a model λ is computed as:

$$P(Q|\lambda) = \pi_{q_1} \prod_{t=1}^{T-1} (a_{q_t q_{t+1}})$$

MMs are often used to describe chains of events where every event corresponds to a *clearly observable* state in the model. One example of this is given in [58] in which character transitions are modelled using twenty-six states $a, ..., z$ and the transition probabilities of the MM are determined by a given vocabulary.

Another classic example is the three-state MM that models the weather [118] as shown in Figure 4.1. In this model it is assumed that once a day (for example, at noon) the weather is observed as being one of the following:

$$\begin{aligned} State\ 1: &\ raining \\ State\ 2: &\ cloudy \\ State\ 3: &\ sunny \end{aligned}$$

It is further assumed that the weather on a given day t is characterised by a single one of the above states, and that the matrix A of state transition probabilities is:

$$A = a_{ij} = \begin{vmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{vmatrix}$$

Given that the weather on day one (t=1) is sunny (state 3), it is then possible to answer simple questions such as "what is the probability (according

to the model) of the weather being sunny tomorrow?", or more complicated questions such as "what is the probability (according to the model) that the weather for the next seven days will be 'sunny-sunny-raining-raining-sunny-cloudy-sunny'?" Using a more formal notation, the *observation sequence* O is defined as $O = \{s_3, s_3, s_3, s_1, s_1, s_3, s_2, s_3\}$ corresponding to $t = 1, 2, ..., 8$, and the problem is to determine the probability of O given the model. This probability can be calculated as:

$$\begin{aligned} P(O|\lambda) &= P[s_3, s_3, s_3, s_1, s_1, s_2, s_3|Model] \\ &= P[s_3] \cdot P[s_3|s_3] \cdot P[s_3|s_3] \cdot P[s_1|s_3] \cdot P[s_1|s_1] \cdot P[s_3|s_1] \cdot \\ &\quad P[s_2|s_3] \cdot P[s_3|s_2] \\ &= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\ &= 1 \cdot (0.8) \cdot (0.8) \cdot (0.1) \cdot (0.4) \cdot (0.3) \cdot (0.1) \cdot (0.2) \\ &= 1.536 \times 10^{-4} \end{aligned}$$

Another interesting question (although not greatly relevant to the process of signature verification) that can be answered similarly is "given that the model is in a known state, what is the probability that it stays in that state for exactly d days?" More formally, the problem is to find the probability of the observation sequence:

$$\begin{array}{rcccccccc} O & = & \{ & s_i, & s_i, & s_i, & ..., & s_i, & s_j \neq s_i\}, \\ & & & 1 & 2 & 3 & & d & d+1 \end{array}$$

given the model, which is:

$$P(O|Model, q_1 = s_i) = (a_{ii})^{d-1}(1 - a_{ii})$$

It is also possible, based on the above formula, to calculate the expected duration in a particular state, conditioned on starting in that state as:

$$\begin{aligned} Expected\, duration &= \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1 - a_{ii}) \\ &= \frac{1}{1-a_{ii}} \end{aligned}$$

In the weather model, the expected number of consecutive sunny days equates to 1/(1-0.8), or 1/(0.2) = 5; for cloudy days it is 2.5; for rainy days it is 1.67.

4.1.2 The Hidden Layer

The Markov models discussed in the previous section are based on the assumption that the series of states directly corresponds to to an easily observable series of events. This assumption is highly restrictive and in reality many

interesting problems do not exhibit the observable state property. *Hidden* Markov models (HMMs) are an extension of Markov models that are capable of modelling entities in which the states are not observable (that is, they are *hidden*). The observations modelled by HMMs are probabilistic functions of the state - this probability density function models a hidden stochastic process.

These correspond to scenarios where the states in the model are not directly associated with an observable event. One of the classical examples of this type of environment is the "coin toss model" [120]. The coin toss model consists of an individual with n coins tossing them in some (unobservable) sequence. The only information available from this series of *hidden* coin tosses is the outcome of each instance - these outcomes are collectively known as an *observation sequence*. A typical observation sequence might be:

$$\begin{aligned} O &= o_1 o_2 o_3 o_4 o_5 ... \\ &= HHTHT... \end{aligned}$$

where H stands for heads and T stands for tails.

Given the above environment, the problem is how to build a model (known as a *hidden Markov model*) to explain the observation sequence. There are a number of different structures that may be appropriate for modelling this environment depending upon what is known or what is unknown (for example, the number n of coins being used). The most straightforward structure would consist of n states where each state corresponds to a different coin (with its own bias) being tossed. Each of these states has an individual probability distribution that indicates the likelihood of the corresponding coin landing on heads or tails (that is, it describes the bias). Transition between states might correspond to selecting a different coin and is characterised by a state transition matrix based on its own probabilistic event (perhaps some action like rolling a die to determine which coin to use). Models where $n = 2$ and $n = 3$ are presented in Figure 4.2.

If the number of coins being used in the experiment is known then the choice of model is greatly simplified. This choice becomes quite difficult if the number of coins is unknown. As larger HMMs have more parameters that can be adjusted, they are theoretically more capable of modelling an entity [118], although the limitations to this rule depend on the environment. In the problem of handwritten signature verification (HSV) using HMMs (discussed thoroughly later in this chapter) the selection of the most appropriate number of states is based more on experience and trial-and-error than any heuristic

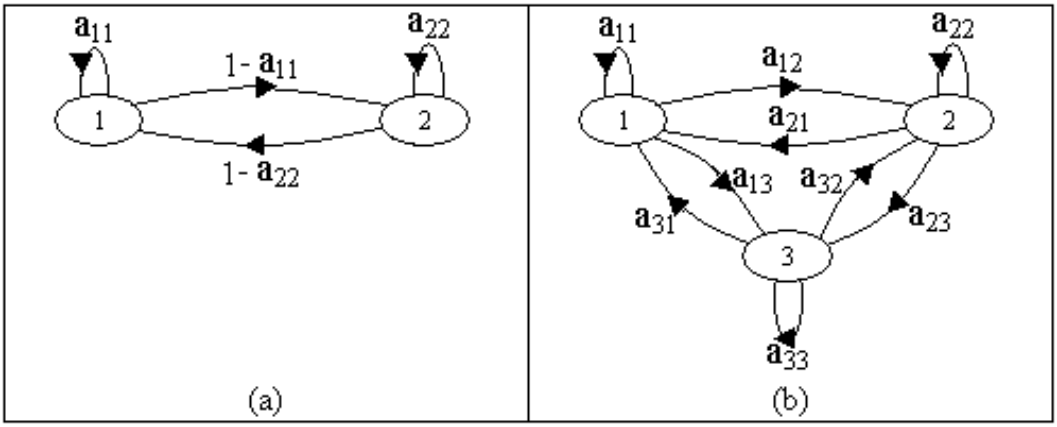

Figure 4.2: *A HMM that models coin tosses using (a) 2 and (b) 3 states.*

approach (similar to selection of the number of hidden nodes in a neural network).

Formally, a HMM is characterised as a 4-tuple $\lambda = (A, B, \pi, S)$ with the following properties:

- A, π and S all have the same definition as for MMs;
- B is a set of probability density functions $\{b_j(o_t)\}$ that models, for each state, the probability of observing symbol o_t while in that state. More formally $b_j(o_t) = P(o_t|q_t = s_j)$ (that is, the probability of observing o_t given that the current state is s_j);
- The probability of observing a particular observation sequence O and a particular state sequence Q for a given model λ is computed as:

$$P(O, Q|\lambda) = \pi_{q_1} \cdot \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \cdot \prod_{t=1}^{T} b_{q_t}(o_t)$$

The Three Basic Problems for HMMs

The HMM architecture described above allows for three basic problems to be solved and applied to real-world domains. The three problems are as follows:

Problem One: Given the observation sequence $O = o_1 o_2 ... o_T$ and the model $\lambda = (A, B, \pi)$, efficiently compute $P(O|\lambda)$, the probability of the observation sequence given the model.

Problem Two: Given the observation sequence $O = o_1 o_2 ... o_T$ and the model λ, estimate the corresponding state sequence $Q = q_1 q_2 ... q_r$ that best "explains" the observations.

Problem Three: Adjust the model parameters $\lambda = (A, B, \pi)$ to maximise $P(O|\lambda)$.

Problem one is also referred to as the *evaluation* problem and involves computing the probability that a given observation sequence was produced by a given model. An alternative way of looking at the problem is of rating how well a given model matches the given observation sequence. The first interpretation relates, from a signature verification point of view, to the likelihood that a particular signature was produced by a particular individual. The second interpretation is particularly useful in determining the "best" model from among a group of candidates. In HSV, this second interpretation corresponds to the *identification* task (identifying an author given only a signature) and is quite computationally expensive in a large database.

Problem two involves attempting to find the "correct" state sequence given a model and observation sequence. In most realistic environments there is no single "correct" state sequence that can be found. Instead it is often necessary to use some measure of optimality based closely on the environment.

Problem three involves attempting to adjust the model parameters so as to "fit" the observation sequence as closely as possible. In other words, this is the "training" or "learning" problem. An observation sequence used to train the HMM is usually referred to as a "training sequence". In HSV the training sequence is a set of observations extracted from a reference signature for a particular individual. The training process then involves updating the model parameters to create the best model for that individual's signature. This training process is typically repeated a number of times using several training sequences.

Solutions to the Three Basic Problems of HMMs

Solution to Problem One: In problem one the task is to calculate $P(O|\lambda)$, the probability of an observation sequence given a particular model. The simplest approach to this calculation is to enumerate every possible state sequence of length T. The probability of an observation sequence O for a state sequence Q then is:

$$P(O|Q, \lambda) = \prod_{t=1}^{T} P(o_t|q_t, \lambda)$$

With the first-order assumption (discussed in Section 4.1.1) this becomes:

$$P(O|Q, \lambda) = b_{q_1}(o_1)b_{q_2}(o_2)...b_{q_T}(o_T)$$

The probability of the particular state sequence Q can be written as:

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} ... a_{q_{r-1} q_r}$$

The probability then of both O and Q occurring at the same time is the product of $P(O|Q,\lambda)$ and $P(Q|\lambda)$:

$$P(O,Q|\lambda) = P(O|Q,\lambda) \cdot P(Q|\lambda)$$

The probability then of O given the model (that is, independent of a specific state sequence) is the sum of this joint probability over all possible state sequences:

$$\begin{aligned} P(O|\lambda) &= \textstyle\sum_{all\, Q} P(O|Q,\lambda) \cdot P(Q|\lambda) \\ &= \textstyle\sum_{q_1,q_2,...,q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) ... a_{q_{T-1} q_T} b_{q_T}(o_T) \end{aligned}$$

The result of this computation is the probability of the observation sequence occurring given the model, that is $P(O|\lambda)$, which solves problem one. At every $t = 1, 2, ..., T$ there are N possible states that can be reached (that is, a total of N^T possible state sequences) along with approximately $2T$ calculations for each term in the summation. The overall computational complexity of this approach therefore is $2TN^T$, rendering the approach unfeasible even for small values of N and T. For example in a simple three state model with one hundred observations, $2 \cdot 100 \cdot 3^{100}$ (more than 10^{50}) computations are required. Clearly a more efficient approach to solving this problem is required, and fortunately exists in what is known as the "forward-backward" procedure [5, 6], described below.

Consider the "forward" variable $\alpha_t(i)$ defined as:

$$\alpha_t(i) = P(o_1 o_2 ... o_t, q_t = s_i|\lambda)$$

That is, the probability of the first t elements of the observation sequence occurring and being in state s_i at time t, given the model. It is possible to solve for $\alpha_t(i)$ inductively as follows:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N$$

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}), \quad 1 \leq t \leq T-1, 1 \leq j \leq N$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The initialization step sets the forward probabilities to the joint probability of starting in state s_i and the initial observation o_1. The induction step involves the sum of multiple probability values. State s_j can be reached at time $t+1$ from any of N possible states at time t. $\alpha_t(i)$ is the probability of the observation sequence $o_1 o_2 ... o_t$ occurring and that the state at time t is s_i, therefore the product $\alpha_t(i)a_{ij}$ is the joint probability of the observation sequence $o_1 o_2 ... o_t$, and that state s_j is reached at time $t+1$ via state s_i at time t. This product is summed over all N possible states at time t giving the probability of s_j at time $t+1$. Once j is known, $\alpha_{t+1}(j)$ is found by including the probability of observing o_{t+1} in state s_j, that is, multiplying the summed value by $b_j(o_{t+1})$. This computation is performed for all states j for each given t from 1 to $T-1$. The termination step involves the calculation of $P(O|\lambda)$ as the sum of the forward variables $\alpha_T(i)$.

The calculation of $\alpha_t(j)$, $1 \leq t \leq T$, $1 \leq j \leq N$ requires of the order of $N^2 T$ computations, which is much lower than the $2TN^T$ computations required for the brute force approach. As an example, in a three state model with one hundred observations, $3^2 \cdot 100$ or 900 computations are required for the forward method as opposed to more than 10^{50} for the brute force approach.

A similar approach can also be taken to calculate the "backward" variable $\beta_t(i)$, defined as:

$$\beta_t(i) = P(o_{t+1} o_{t+2} ... o_T | q_t = s_i, \lambda)$$

That is, the probability of the partial observation sequence from $t+1$ to the sequence end, given the state s_i at time t and the model λ. The backward part of the forward-backward procedure is not needed for the evaluation problem (the forward part has already solved this), but it is used in other solutions. Like the forward approach, it is possible to solve for $\beta_t(i)$ inductively, as follows:

1. Initialization:

$$\beta_T(i) = 1, \quad 1 \leq i \leq N$$

2. Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad t = T-1, T-2, ..., 1, 1 \leq i \leq N$$

The calculation of $\beta_t(i)$, $1 \leq t \leq T$, $1 \leq i \leq N$ requires of the order of N^2T computations.

Solution to Problem Two: Recall that problem two involves attempting to find the "optimal" state sequence according to some criterion. In general the goal is to determine the state sequence Q such that the probability of the observation sequence O occurring from Q is greater than that from any other state sequence. That is, find the Q that maximises $P(Q|O,\lambda)$, which is equivalent to maximizing $P(Q,O|\lambda)$. There exists a well-known technique for finding this single best state sequence called the *Viterbi Algorithm* [149].

The Viterbi algorithm is an inductive algorithm in which, at time t, the best intermediate state sequence (that is, the one giving the maximum probability) is found. At the conclusion of the computation, the best path through the available states is known. A more formal specification of the Viterbi algorithm requires the definition of the following quantity:

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P[q_1 q_2 ... q_{t-1}, q_t = s_i, o_1 o_2 ... o_t|\lambda]$$

That is, $\delta_t(i)$ is the highest probability, through a state sequence, at time t, which accounts for the first t observations and ends in state s_i. By induction then:

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] \cdot b_j(o_{t+1})$$

In order to retrieve the optimal state sequence it is necessary to keep track of the argument that maximises the above equation for each t and j. This is done using an array denoted as $\psi_t(j)$. After the most likely state for time T is found, the best states from previous iterations are traversed backwards to obtain the overall "best" state sequence q^* (note that p^* denotes an approximation of the probability density $P(O|\lambda)$).

The entire algorithm can now be expressed inductively as follows:

1. Initialization:

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(o_1), \quad 1 \leq i \leq N \\ \psi_1(i) &= 0. \end{aligned}$$

2. Recursion

$$\begin{aligned} \delta_t(j) &= \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t), \quad 2 \leq t \leq T, 1 \leq j \leq N \\ \psi_t(j) &= arg\max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right], \quad 2 \leq t \leq T, 1 \leq j \leq N \end{aligned}$$

3. Termination

$$\begin{aligned} p^* &= \max_{1 \leq i \leq N} [\delta_T(i)] \\ q_T^* &= arg \max_{1 \leq i \leq N} [\delta_T(i)] . \end{aligned}$$

4. Backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad 1 \leq t \leq T-1$$

where *arg* refers to the argument(s) producing the particular maximum.

Note that in practical situations the high number of multiplications of probability values can result in a floating point underflow. In this case, the probability density P is usually replaced with the loglikelihood $\tilde{P} = -log(P)$.

The complexity of this algorithm is very similar to the forward algorithm described above. The operation of the algorithm in the worst case requires $O(N^2T)$ calculations, however in practical implementations many of the processed values can be performed once and copied, reducing the overall cost.

Solution to Problem Three: As with neural networks discussed in Chapter 3, one of the most important features of hidden Markov models is the capability of the model to automatically adjust to the sample data presented to it. This process is commonly known as "learning" or "training" and is by far the most difficult of the three HMM problems to solve.

The goal of the learning process is to arrive at a set of HMM parameters (A, B and π) that maximises the probability $P(O|\lambda)$ given a particular observation sequence. The major difficulty with learning in HMMs is that there is no way to systematically solve for the model parameters to bring about this maximum. There are however iterative procedures that are capable of performing accurate estimations of the parameters to locally maximise $P(O|\lambda)$.

The process of learning, or training, in HMMs is summarised in Figure 4.3. The first step is initialisation, followed by the two-step iterative process of best path determination and re-estimation of parameters based on the best path. This iteration continues until some stopping condition is met (for example, negative loglikelihood crossing some threshold).

There are several different methods for generating the initial parameter values such as randomisation, setting all values to some constant (for example, one) or more complex approaches such as linear time-alignment of the observation sequence with the model states [98, 32] or using a neural network to bias the HMM parameters before the training phase commences [154, 155, 68].

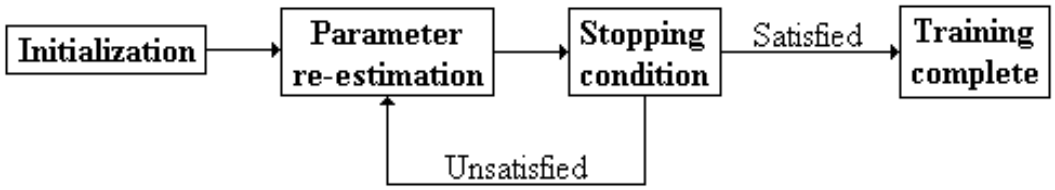

Figure 4.3: *A summary of Expectation Maximization (EM) training in hidden Markov models.*

Although the learning mechanism will result in convergence to an optimum, this is a local optimum only so the choice of the starting point can be quite important.

The iterative learning is done via best path determination (essentially performing a new alignment of the observation sequence O with the states S) and parameter re-estimation. The Forward-Backward and Viterbi algorithms are used to perform the time alignment (described formally below). The parameter re-estimation phase can then be accomplished in a number of ways and the two most popular and effective techniques, the "Baum-Welch" approach [120] and the "Segmental K-Means" approach [119], are considered in Section 4.1.4.

4.1.3 Bakis Models

Throughout previous discussions it has been assumed that it is possible to transit, in a single step, from any state to any other state (including self-transitions) in the HMM in a single step (all elements in the transition matrix are greater than zero). That is, the model is fully connected or *ergodic*. In some domains this is not the most desirable structure and better modelling can be achieved by applying restrictions to the transitions. For example, the left-right, or *Bakis* model [3], is depicted in Figure 4.4 and has the property that the state sequence proceeds from left to right. This type of structure is appropriate for modelling signals with properties that vary temporally, for example speech or handwriting. The fundamental property of all left-right HMMs is that:

$$a_{ij} = 0, \quad j < i$$

That is, it is not possible to transit to any states with indices lower than the current state. In a pure left-right model the start state is always 1 ($\pi_1 = 1$), but this condition is slightly relaxed in many implementations (however it is

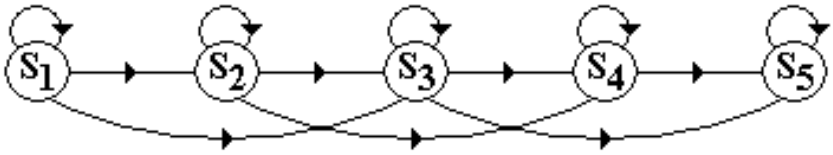

Figure 4.4: *A 5-state left-right, or Bakis, model. This example features loop (for example, 1 to 1), forward (1 to 2) and skip (1 to 3) transitions.*

usually still necessary to begin very close to the leftmost state of the model). In many implementations there are also limitations placed on the state transition coefficients to make sure that large "jumps" are not made (these limitations may be either included directly as rules, or indirectly due to the low probability of such large jumps). For example, in Figure 4.4, no jumps of more that two states ahead are permitted.

There are many other variations of HMM structure besides ergodic and left-right models and these are usually application specific (left-right models are quite common in HSV). Whatever constraints are applied to (or removed from) a model, the essential elements of the re-estimation process remain the same.

4.1.4 Learning in Hidden Markov Models

The Baum-Welch Training Algorithm

In order to describe the Baum-Welch procedure, it is convenient to define a value $\xi_t(i,j)$ as the probability of being in state s_i at time t, and state s_j at time $t+1$, given the observation sequence and the model. That is:

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda)$$

From the definitions of the forward and backward variables, $\xi_t(i,j)$ can be written as:

$$\begin{aligned} \xi_t(i,j) &= \frac{P(q_t=s_i, q_{t+1}=s_j, O|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \end{aligned}$$

As a further convenience the value $\gamma_t(i)$ is defined as the probability of being in state s_i at time t, given the model and the observation sequence. It is then possible to express $\gamma_t(i)$ in terms $\xi_t(i,j)$ by summing over j:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

If $\gamma_t(i)$ is summed over the time index t, this will compute the expected number of times that the state s_i is visited, or alternatively, the expected number of transitions out of state s_i (ignoring t=T). Similarly, if $\xi_t(i,j)$ is summed over t, the quantity computed is the expected number of transitions from state s_i to state s_j. That is:

$$\begin{aligned} \textstyle\sum_{t=1}^{T-1} \gamma_t(i) &= \textit{expected number of transitions from state } s_i \textit{ in } O \\ \textstyle\sum_{t=1}^{T-1} \xi_t(i,j) &= \textit{expected number of transitions from state } i \textit{ to state } s_j \textit{ in } O \end{aligned}$$

Using the above formulae it is possible to compute a re-estimated model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ as follows:

$$\begin{aligned} \overline{\pi}_j &= \textit{expected number of times in state } s_i \textit{ at time (t= 1)} \\ &= \gamma_1(i) \end{aligned}$$

$$\begin{aligned} \overline{a}_{ij} &= \frac{\textit{expected number of transitions from state } s_i \textit{ to state } s_j}{\textit{expected number of transitions from state } s_i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned}$$

$$\begin{aligned} \overline{b}_j(k) &= \frac{\textit{expected number of times in state } s_j \textit{ and observing symbol } k}{\textit{expected number of times in state } s_j} \\ &= \frac{\sum_{t=1, s.t. o_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \end{aligned}$$

The Baum-Welch approach is popular partly due to its intuitive nature but mainly because it has been proven [77] that either (1) the initial model λ results in a maximum in the likelihood function, in which case $\overline{\lambda} = \lambda$; or (2) the re-estimated model $\overline{\lambda}$ is *more* likely than λ to have generated the observation sequence, that is, $P(O|\overline{\lambda}) > P(O|\lambda)$. At the completion of training, problem three has been solved.

The Segmental K-Means Training Algorithm

The major alternative to the Baum-Welch algorithm for training HMMs is known as the "Segmental K-Means" algorithm, or sometimes referred to as segmental maximum likelihood. The Segmental K-Means (SKM) approach uses the state-optimised joint likelihood and the underlying state sequence as the objective function for parameter estimation [119]. Many researchers find

this approach an attractive alternative because in most situations it converges faster than the Baum-Welch approach, sometimes significantly so.

The major initial difference between the two approaches comes in the form of the optimization criterion for parameter estimation. Where the Baum-Welch algorithm focuses on summing probabilities over all possible state sequences, the SKM algorithm focuses only on the *most likely* state sequence. More formally:

$$\max_{Q} P(O, Q|\lambda) = \max_{Q} \pi_{s_0} \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(o_t)$$

where O is a single observation sequence and Q is a state sequence.

This is often referred to as the "state-optimised likelihood". Intuitively, an approach that considers only the most likely sequence would seem to operate much faster, requiring significantly less computation as it does not involve all state transition paths being examined.

The SKM algorithm works by embedding the K-means method in a Markov chain modelling algorithm [119] and consists of two major steps: segmentation and optimization. Starting from the initial model λ, the segmentation step is optimally performed using the Viterbi algorithm discussed in Section 4.1.2. The segmentation step calculates the most likely state sequence.

Given a state sequence Q and an observation sequence O, the optimization step computes a re-estimated model $\overline{\lambda}$ that maximises the above state-optimised likelihood. That is:

$$\begin{aligned} \overline{\lambda} &= arg\max_{\lambda} \{\max_{Q} P(O, Q|\lambda)\} \\ &= arg\max_{\lambda} \{\max_{Q} [logP(O|Q, \lambda) + logP(Q|\lambda)]\} \end{aligned}$$

The original model λ is then replaced with the new $\overline{\lambda}$ and the steps are repeated until the state-optimised likelihood converges to a pre-set threshold. It has been proven in [119] that the SKM algorithm converges in all situations.

The SKM algorithm can easily be extended to the case of multiple independent observation sequences $O = \{o^i\}$. The optimization criterion then becomes:

$$\max_{Q} P(O, Q|\lambda) = \max_{Q} \prod_{i} P(O^i, Q^i|\lambda)$$

where O^i is the set of multiple independent observation sequences and Q^i is the set of state sequences.

4.2 Applications to Handwritten Signature Verification

The application of HMMs to the problem of HSV is a relatively new area of research, having been undertaken for less than ten years. The main attractions for use of HMM in HSV are:

1. *Learning by example*: Many of the parameter values of HMMs are learned by the model, rather than being manually set by the designer. Training algorithms such as Baum-Welch and Segmental K-Means effectively infer a model from the available data.

2. *Structural suitability*: HMMs are highly suited to recognition of signals with varying time-flow.

3. *Applicability in similar fields*: HMMs are used extensively in speech recognition [58, 120], which bears similarity to handwriting in that they are both mostly continuous in time. The use in speech recognition means that HMMs are well-understood and many of the practical and implementation solutions are portable to HSV.

4. *Execution speed*: The learning phase of HMMs is the most computationally expensive (although even training often proceeds quite quickly). In signature verification this training is a one-off cost and can be done off-line (in the sense that the training phase in HSV is rarely performed while the user waits for results). The execution, or evaluation, of HMMs corresponds to the verification of test signatures and this phase is quite fast, which is a good match for HSV because it is the verification phase where response time is most important.

The following section contains a summary of the more significant studies into HMM based approaches to handwritten signature verification.

4.3 Previous Work

Handwritten signature verification is still a relatively new application area for HMMs, with few truly intensive investigations being reported in the literature. Some of the more landmark studies with HMMs in HSV are presented here.

The basic approach HMMs involves performing stochastic matching of a model and a signature using a sequence of probability distributions of the features along the signature [64]. The signing process is usually modelled with several states that constitute a Markov chain, each of them corresponding to a signature segment. The states are generally not directly observable (that is, they are hidden) and only the signature's local features (such as tangent angles) can be observed.

A system that combined local and global information in multiple models is presented in [64]. Methods of combination such as the one described are of significant interest in this dissertation and this particular approach is further discussed in Chapter 5. The local feature comparison used in the work though is of interest here and involves using HMMs to compare feature vectors. Six sample signatures were used to train the HMM, which modelled the user's signature as a series of states with probabilistic transitions between them. Given a test signature, the Viterbi algorithm [149, 120] is used to search for the most likely state sequence corresponding to the given observation sequence and give the accumulated likelihood score along that best path. The difference between this score and the mean likelihood obtained during training is then used as an error measure to classify a test signature as valid or a forgery. This approach resulted in an equal error rate (EER) of around 5% and the combination techniques (discussed later) reduce this to an error rate of 2.5% EER.

Another early study into HMMs for HSV is reported in [161], which uses absolute angular direction along the trajectory of the signature, encoded as a sequence of angles. Each signature is divided into a fixed number of segments in order to obtain sequences of the same length, and the normalised angles then quantised into sixteen levels. Experiments are conducted using several model structures including left-to-right and parallel models. Training is done using the forward-backward algorithm and probabilities estimated using the Baum-Welch algorithm. Thirty-one writers contributed sixteen signatures each for use in evaluation in which eight signatures are used for training and eight used in testing (no skilled forgeries were used). The left-to-right model with arbitrary state skips is found to result in the lowest error rates. Additionally, the study found that increasing the number of states and decreasing the observation length leads to a decrease in the false rejections and an increase in the false acceptances (presumably because less detail is captured when the observation length is decreased). The best results reported are a false acceptance

rate (FAR) of 4.4% and a false rejection rate (FRR) of 1.75%, although no skilled forgeries are used (only zero-effort).

The Markov model engine forming part of the HMM described in this chapter appears in related work in [85]. In this study, experimentation is performed using simple directional features for verification of handwritten passwords. The approach is based on segmenting the signature into conceptually significant parts (that is, extraction of strokes) and then extracting a single directional feature from the stroke. The net displacement direction of each stroke is characterised as one of n discrete directions and represented as a single integer (one integer value per stroke). The result of this processing is that a signature can be represented as a series of integer labels that characterises the basic directional changes undergone in the handwriting stream. This representation is invariant under scaling and the slight rotation that may occur in signature production. In the work, the optimal value for n was found to be four (corresponding to quadrants in a Cartesian grid) and the signatures are modelled using a five-state Markov model (one state for each direction, plus a "pen-up" state). The work also attempts to incorporate the duration of a stroke by repeating the integer symbol every twenty-fifth of a second that the stoke was maintained. A particularly long strong in direction "1" for example results in a series of 1's being placed into the integer string. The approach therefore records the basic shape of the handwriting as well as incorporating some sense of timing and velocity.

Five signature samples are used as the reference set to train the model, which involved re-estimation of the state transition probabilities based on the five extracted integer strings. For each test sample (both genuines and forgeries) corresponding to the current writer, the *forward* algorithm is used to find the probability that the test writing came from the same individual as the reference writings. The test signature is only accepted as genuine if this probability is higher than a pre-determined threshold. The approach is tested on a database consisting of 720 genuine handwritten passwords from forty-seven writers, and 265 forgeries. The equal error rate found is 11.25% for skilled forgeries and 0.64% for zero-effort forgeries (when forgers hadn't seen a password sample). The error rate needs to be made much lower for the system to be of any significant value, although results remain interesting because of the simplicity of the features used.

A thorough examination of HMMs for signature verification is presented in [32] and in more compact form in [33]. A formal definition of the problem

of HSV using HMMs was presented as: given an individual i, their signature is described with a model λ_i, a threshold T_i, and a sequence of input items $X = x_1, x_2, ..., x_n$, accept the signature if the condition $-log(P(\lambda_i|X)) \leq T_i$ holds. The approach used involved firstly segmenting the signature input into strokes based on zero crossings in the y velocity. A feature vector of thirty-two components (a mix of spatial, dynamic and contextual features) is then extracted for each segment, including angular features and seldom-used pen-tilt angles in the x and y direction. While experimentation showed that pen-tilt angles are highly discriminatory for HSV (at the most basic level it is impossible for a left-handed forger to forge the signature of a right-handed person), few digitizing tablets have this capability.

Each signature was modelled by a single left-to-right HMM with loop, forward and skip transitions with probabilities re-estimated during training. The number of states in the model was set to 0.8 times the average number of feature vectors (that is, strokes) per signature in the training data. The value of 0.8 was arrived at through trial-and-error experimentation and similar values for the number of states are reported elsewhere in the literature [34].

An extensive database of almost 5,000 signatures is used including genuine signatures, home-improved forgeries, over-the-shoulder forgeries and forgeries produced by forensic document examiners. Reported error rates include 2.58% and 1.11% FAR for over-the-shoulder forgeries and home-improved forgeries respectively (this corresponds to an equal error rate of 1.9% using the same threshold). Experiments conducted with "professional" forgeries produced by forensic document examiners provided with paper copies of the genuine signatures did not result in any degradation of these error rates. The speculated justification for this was that the professional forgers were not able to observe the dynamics of the original signers and were less focussed on forging dynamics than forging signature shape (highlighting the advantage of on-line HSV).

This study also describes linear discriminant analysis performed in examining the discriminative value of the features. Features are ranked for each writer and these individual rankings combined to provide an overall perspective. In general, dynamic features are found to result in much more accurate verification than spatial or static features. The five most discriminative individual features include velocity and pressure information, as well as both pen-tilt features.

A comparison of on-line and off-line signature verification using HMMs is made in [122]. Seven different features are investigated in the on-line system for

there discriminative capabilities including pressure, angular information, velocity, acceleration and the Fourier transform of the signal (although no global features are used in the study). The off-line system divides the signature image into a fixed number of squares of approximately ten pixels in diameter, with each column then represented as a vector. Few details are given as to the structure of the HMMs used, other than that they are discrete models. The Viterbi algorithm is used to compute the probability that the feature observation sequence extracted from the test signature was generated by the corresponding HMM (trained using the reference signatures for the same signer). The database consisted of fourteen writers, each contributing twenty signatures, and a total of sixty forgeries were available. Combining velocity, Fourier transforms, pressure and bitmap features (from the off-line system) yields the best on-line error rates of 1.0% equal error, with 1.9% the best equal error rate obtained using the purely off-line system. This study supports the claim that on-line signatures are more information rich than off-line, and also that HMMs are capable of taking advantage of this extra information.

4.4 Methodology

This section presents the methodology followed for the development of the hidden Markov model portion of this project. As previously discussed the system described in this chapter is a parameter-based dynamic system, and as such the explanation of methodology starts with the features used in the HMM and is followed by the experimental setup and the training process.

4.4.1 Signature Segmentation

The actual segmentation of handwriting at conceptually significant points (these segments are known in the literature as "strokes") is very important to many handwriting processing applications. Segmentation is of particular importance to this form of handwritten signature verification where a HMM is used to model the individual segments. The traditional approach to this is typically to delimit the segments using minima in the pen-tip velocity, or alternatively the y velocity [135, 151].

A new approach to this segmentation, known as the *Extremum Consistency* method, is proposed in this dissertation. This method involves attempting to reduce the number of "false" minima occurring due to anomalies such as shaky

Figure 4.5: *This figure presents a signature from the signature database. Using the velocity based stroke (VBS) technique for segmentation results in 290 strokes, whereas the extremum consistency approach results in 217 strokes.*

hands, muscle fatigue or device/rounding errors etc. Details of the extremum consistency (EC) approach are presented in Appendix A and an illustration of the segment reduction is presented in Figure 4.5. This figure presents a typical signature from the signature database that contains 290 strokes according to the traditional velocity based stroke (VBS) method. Using the EC approach the number of strokes reduces to 217.

Appendix A also presents the resulting improvement in error rates using the EC approach over other forms of segmentation and in one case the accuracy of a HSV system is reduced from 6.9% overall error (for a basic VBS implementation using gradient descent) to 2.3% overall error. There is a slight overhead associated with the EC calculations, however the number of strokes extracted using this approach is typically reduced by around 25%, so the overall signature processing procedure is generally faster.

4.4.2 Extracted Features

A general discussion of features and feature extraction is presented in the corresponding section in the neural network chapter (Section 3.4.3). Many of the HMM features are similar to those used as input to the NN (modified to apply to local strokes).

What follows now is a discussion of each of the features extracted for use in the HMM, as well as their significance and method of calculation. Each of these features is included in a feature array (often referred to as a feature vector) that acts as a single observation in the HMM.

Horizontal Length: The horizontal distance measured between the stroke's two most extreme points in the x direction. See Figure 4.6 for an example.

Aspect Ratio: Aspect ratio is the ratio of the stroke length to the stroke

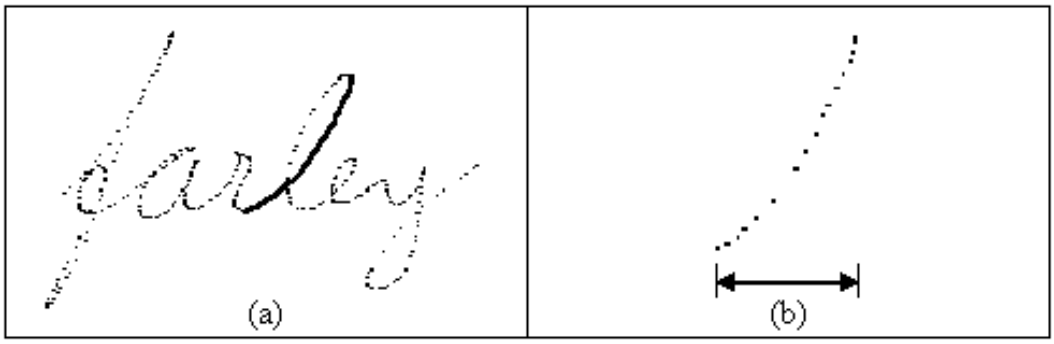

Figure 4.6: *This figure represents the horizontal length of a typical stroke. (a) contains the original handwritten sample with an extracted stroke in bold. (b) shows an expanded view of that same stroke with the horizontal length marked.*

height. This feature obviously remains invariant to scaling as if the user signs in a different size, both the length and the height will be altered proportionally and the aspect ratio will remain the same.

Curvature: Curvature is a measure of how "flat" or how "curved" the stroke is. A high value for curvature means that the stroke is highly curved, which is associated with a turning point in the handwriting. For example, in Figure 4.7(a) the stroke is almost a straight line (representing a straight section of handwriting), which is reflected in the low curvature value of 0.01. Conversely, the sample in Figure 4.7(b) has a much more pronounced curvature resulting in a much higher value of 0.40.

Curvature is calculated as the ratio of the strokes path length to the overall length minus one. The path length is the sum of distances between each consecutive point in the sampled stroke. The overall length is the physical, or Euclidean, distance between the stroke start-point and the stroke end-point.

Maximum Velocity: Maximum velocity is a purely dynamic feature of the stroke meaning that it is very difficult for a potential forger to copy. In addition, velocity is a highly stable feature, easily repeatable by a genuine user [112, 51]. The calculation of pen-tip velocity is done in terms of components velocities, v_x and v_y, (calculated as the first derivative of the x and y streams):

$$v = \sqrt{{v_x}^2 + {v_y}^2}$$

Average Velocity: The average velocity is a personal measure of the how fast the pen-tip is travelling across the surface of the tablet. This is calculated as the average of all velocity values extracted from the stroke.

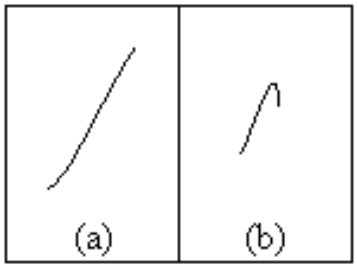

Figure 4.7: *Different strokes can result in quite different curvature values. For example, (a) shows a sample stroke that is quite flat, resulting in a curvature value of 0.01. Conversely (b) shows a sample with a much more pronounced curve that results in the higher curvature value of 0.40.*

Average Absolute Acceleration: This feature refers to the average absolute value of the acceleration and deceleration measurements. It is computed as the second derivative of the data stream (or the derivative of the velocity values calculated earlier). The average absolute acceleration captures the mean rate of change in velocity in both positive and negative directions.

Maximum Acceleration: While this feature is slightly less stable than some others, the purely dynamic nature and difficulty in forging still make it a useful characteristic.

Maximum Deceleration: Similar to maximum acceleration, this is a purely dynamic feature that measures the rate at which the pen-tip's velocity decreases as it approaches the end of a stroke.

Stroke Slant Using All Points: The handwriting slant is of significant importance and it is a major factor in determining the type of stroke. As with slant calculation in the previous chapter there are a number of techniques employed here that measure various aspects of the slant. The first approach to calculating the stroke slant is to take the mean of all gradient values between the spatially resampled neighbouring points. This process is identical to that depicted in Figure 3.24 in Chapter 3, except using points in the stroke as opposed to the signature as a whole.

Handwriting Slant Using Stroke End-points: The gradient of the stroke is taken as simply the gradient between the stroke start-point and the stroke end-point. This was found to be less stable than the previous method for slant calculation and eventually removed from the feature set due to excessive variability. An example slant calculation using this method appears in Figure 4.8, which shows a calculated slant that does not seem to accurately match the

Figure 4.8: *Handwriting slant calculated using stroke end-points. This is the same stroke as shown in Figure 4.7, depicted here as the series of sampled points. The solid line to the immediate right is the calculated slant.*

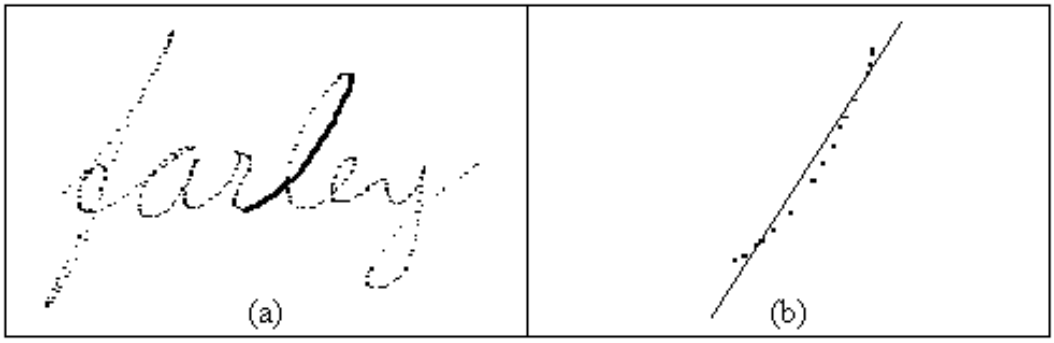

Figure 4.9: *An illustration of handwriting slant calculated through regression. (a) shows the original word as a series of sampled points with the extracted stroke in bold and (b) shows slant calculated via regression.*

slant of the stroke.

Handwriting Slant Through Regression: This approach involves performing linear regression using all the points in the stroke. The resulting line is taken as the gradient for the stroke and is illustrated in Figure 4.9.

Handwriting Slant Using Cai and Liu Technique: This method of determining slant is based on handwriting recognition techniques [19]. Calculation of this feature is done in an identical manner to the way described in Section 3.4.3.

Stroke Concavity: Stroke concavity is a measure of how far the stroke is from being a straight line. A stroke of high concavity does not closely follow the imaginary line drawn from the stroke's start-point to the end-point. Calculation of the concavity value is done by firstly performing linear regression using the points in the stroke to obtain the line-of-best-fit. This feature then becomes a measure of how well the points in the stroke "fit" that line, or how well the points are approximated by that line. The second step is to apply the following formula for each stroke:

$$Stroke\ Concavity = \sqrt{\sum_{i=1}^{n}(s_i - r_i)^2}$$

where:

- n is the number of points in the stroke;
- s_i is the i^{th} point in the stroke;
- r_i is the coordinate along the line-of-best-fit that is the least distance from s_i.

A graphical depiction of this calculation can be found in Figure 3.28 in Chapter 3.

Stroke Duration: This is a simple feature that records the amount of elapsed time between the first sampled point and the last sampled point in the extracted stroke. This feature remains quite stable between corresponding strokes and is not obvious to a potential forger.

Horizontal Velocity: Horizontal velocity is the average absolute velocity in the x direction. It is a measure of how fast the pen-tip moves horizontally when producing the stroke and is related to pen-tip velocity, horizontal length and acceleration. It is impossible for a potential forger to discern the horizontal velocity from an off-line copy of the writing. This feature is calculated as the ratio of absolute horizontal distance to the duration in which the stroke was produced.

Mean Pen-Tip Pressure: Pen-tip pressure is a measure of the amount of vertical pressure being applied by the pen to the top of the tablet. This is an option available on most existing tablet and stylus hardware devices and is typically measured by an accurate sensor in the tip of the pen. Pressure, like many features used in this system, is very difficult for a forger to discern from an off-line copy of the handwriting. Although pen-tip pressure is less stable than other dynamic features such as velocity, it is included because of the difficulty that a potential forger has in accurately simulating the pressure profile. Mean pen-tip pressure used in this feature is simply the average of all values in the pressure profile for the stroke.

Standard Deviation of Pen-Tip Pressure: The standard deviation of the pen-tip pressure is a gauge of how much a writer typically varies his or her pen-tip pressure during the course of the stroke. Another purely dynamic

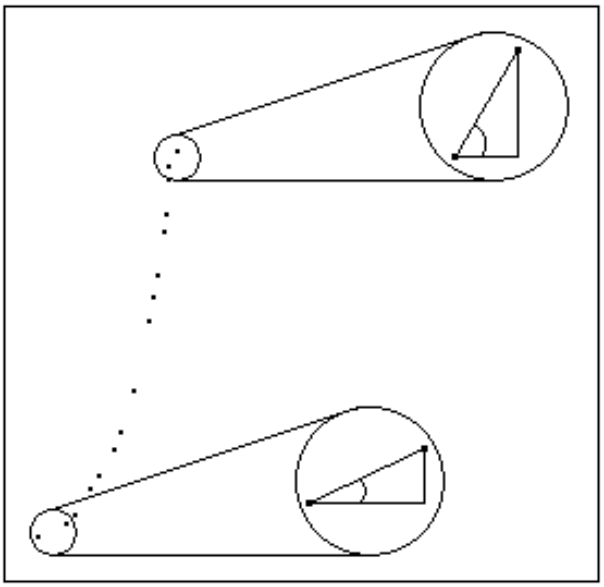

Figure 4.10: *A graphical representation of the beginning and ending gradient values within the stroke.*

feature, this is calculated as the regular standard deviation of the pressure profile.

Maximum Pen-Tip Pressure: The maximum pen-tip pressure is the highest value in the pressure profile. Like all of the pressure features, the maximum pressure is not able to be extracted from an off-line copy of the writing and, while not as repeatable as velocity, is still useful in combination with other features.

Minimum Pen-Tip Pressure: This refers to the minimum non-zero value in the pressure profile. This is another personalised, invisible feature that previous researchers have also found useful [33].

Gradient of the Start and End Points: These two features capture the angle or gradient at the beginning and the end of the stroke. These values are calculated using the first two points and last two points of the stroke respectively. A graphical representation of these features is presented in Figure 4.10.

Gradient of Intermediate Tangents: The tangent of the stroke is obtained at three intermediate points equidistantly spaced within the stroke. These gradients (along with those in the previous feature) try to capture some element of the shape of the stroke and the direction changes within it.

This complete set of features was used in the initial experimentation in building the HSV system, but it became evident through further experiment-

ation and examination of the inter- and intra-writer variability that some of these features were not insightful and the overall performance (in terms of both execution speed and classification error) suffered as a result of their inclusion. The removed features were as follows:

All acceleration and deceleration measurements: although these were initially thought to be useful (and are useful when taken over the signature as a whole), the values calculated using only the data points in the signature segment were not stable enough to be usable. The presumption here is that there are not enough points in an average stroke to get a useful measure.

Standard Deviation of Pen-Tip Pressure: again this feature simply lacked the stability to have any use in the HMM based system and was removed from the feature set.

Horizontal Velocity: there were typically not enough points dispersed in the horizontal direction to get use out of this feature.

Gradient of the Start and End Points: similar to the previous features it was found that this feature did not exhibit enough stability to be considered useful and was therefore removed from the feature set.

Handwriting Slant Using Stroke End Points: the instability of this feature also rendered it unusable.

This leaves a total of fourteen discriminative features in the feature set used in the HMM.

4.4.3 Experimental Setup

This section describes the experimental setup used in the development and testing of the HMM-based HSV system, including a discussion of the different structures and setups used in an attempt to obtain optimum results. The signature database and pre-processing have already been described in Section 3.4.2 and the feature space presented in the previous section. The different sets of experiments performed are now considered in detail and grouped according to category.

Model Structure

The signature of each individual was modelled with a single hidden Markov model with the basic structure being a left-to-right (Bakis) model with self, forward and skip transitions between the states. The number of states in the model depends on the number of observations, a point further discussed

below. The initial state probabilities and state transition matrix for each HMM is estimated during the training phase. The observation probabilities density function is approximated by a discrete distribution of vector quantiser indices. A discrete implementation was chosen over a continuous modelling approach using Gaussian mixtures partly because discrete models cope very well with the features typically used in handwriting [123]. Additionally, a vector quantiser can be derived during training for each individual signer, adding another level of personalisation to the HSV system. Finally, use of discrete distributions alleviates the need to make assumptions about the form of the underlying distribution.

Number of States

The selection of the most appropriate number of states to use in a HMM is not a trivial one (for example, see the coin toss model in Section 4.1.2). There is no heuristic approach to deciding on the number of states to use and the approach of other HSV researchers in the literature has been more one of trial-and-error and experience.

Other researchers used values of 0.75 or 0.8 times the number of observations to determine the number of model states [32, 34]. Experimentation with this HSV system included searches of constants (5, 10, 20, 50, 100) and multiplication factors of 0.5 to 1.5 in increments of 0.05. Ultimately the multiplication factor of 0.8 was found to perform slightly better than 0.75 or any other values. The factor of 0.8 is used as a basis for further experimentation presented in the chapter.

Training Approaches

The approach to training the model is the next aspect in the developmental process to be considered. A single HMM was used to model the signature for each signer in the database. Training of the HMM was done using five randomly selected samples of the user's genuine signature. A conscious decision was made to attempt to minimise the number of genuine signatures to use in building the reference to minimise the inconvenience to a potential user. Many researchers use more than five reference signatures when training HMMs [64, 92, 163], which would be expected to produce better results (use of more reference signatures gives the HMM better insight into the signers natural style), but the extra overhead and inconvenience to users may make

Training Approach	Number of Epochs	Relative Time
Baum-Welch	236.0 (138.5)	100%
Segmental K-Means	5.2 (1.4)	2.1%

Table 4.1: *The training performance of the two HMM training algorithms used in experimentation. The "Number of Epochs" is the mean number of epochs required for convergence to occur (with standard deviation in brackets). The relative time compares the elapsed time prior to convergence.*

the system less attractive. An examination of the effect of different reference sets is presented in Section 5.3.4.

The actual process of training the HMM involved experiments using both the Baum-Welch (BW) and Segmental K-Means (SKM) algorithms described above. Each takes a different approach to the training process and results in different training characteristics. Table 4.1 summarises the training performance details in terms of the number of iterations and the overall time required for the algorithm to converge. As can be seen from the table, the SKM approach converged, on average, in far fewer iterations than the BW approach (SKM actually converged in fewer epochs for every signer in the database). In addition it can be seen that SKM takes far less time to build the model than does the BW approach. Irrespective of the training approach used, the end result of the training process was a single trained HMM for each signer. It is then possible to determine the likelihood that a given test signature was generated by a particular signer's HMM.

For each remaining signature (both forgeries and genuine signatures not used in training the model) corresponding to the current signer, the *forward* algorithm (see Section 4.1.2) was used to find the likelihood that the test signature was generated by the current signers HMM (that is, the likelihood that the test signature was produced by the same individual that performed the reference signatures). If this likelihood value was greater than an applied threshold value T, then the test signature was accepted as genuine, otherwise rejected as an attempted forgery. Error rates can be determined based on how many forgeries are accepted and how many genuine samples are rejected.

The performances of the two training algorithms are also compared in terms of the resulting error rates. Essentially this was measuring which of the two approaches best models the signers natural writing style. The error plots for the SKM and BW approaches are presented in Figures 4.11 and 4.12 respect-

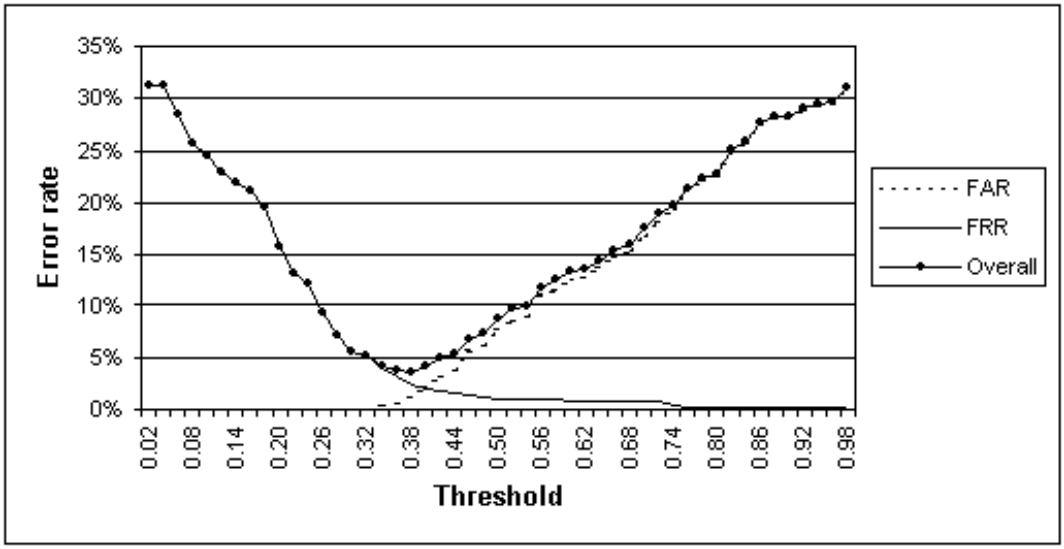

Figure 4.11: *A plot of the HSV results using the Segmental K-Means learning algorithm and different threshold values.*

Training Approach	FAR	FRR	OER	EER	ZEFAR
Baum-Welch	2.6%	2.4%	5.0%	2.5%	0.5%
Segmental K-Means	1.2%	2.3%	3.5%	1.9%	0.4%

Table 4.2: *A comparison of the modelling accuracy of the Segmental K-Means and Baum-Welch HMM training algorithms. All significant HSV-related error rates are reported in this table.*

ively, which show that in this instance the SKM approach more successfully models the signatures.

The significant error rates that measure how well the HSV performed (the error rates most often reported in the literature) for each learning approach appear in Table 4.2. The values in this table more clearly show the superior performance of the SKM approach that resulted in a 3.5% overall error rate (OER) compared to 5.0% for BW. Note that the value for zero-effort FAR (or ZEFAR, see Section 2.2.2 for more information on zero-effort forgeries) is calculated by using the genuine signatures of all other signers in the database as attempted forgeries. The threshold used in zero-effort FAR calculation is the same as that which produces the lowest overall error rate.

There was some expectation that the two sets of error rates resulting from the two training approaches would differ, however the magnitude of this dif-

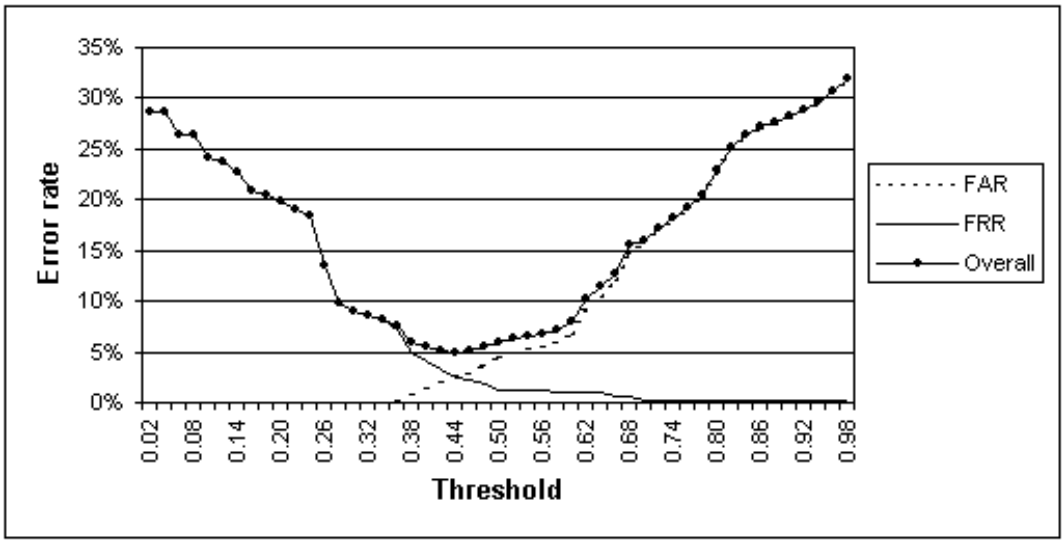

Figure 4.12: *A plot of the HSV results using the Baum-Welch learning algorithm and different threshold values.*

Training Approach	**FAR**	**FRR**	**OER**	**EER**	**ZEFAR**
Baum-Welch	1.3%	1.9%	3.2%	1.7%	0.2%
Segmental K-Means	1.0%	1.7%	2.7%	1.4%	0.1%

Table 4.3: *A comparison of the modelling accuracy of the Segmental K-Means and Baum-Welch HMM training algorithms when ten reference signatures are used to train the model.*

ference was unexpected. Further experimentation using different sizes for the reference signature set reveals that the disparity is reduced when more reference signatures are used. Error rates resulting from the use of ten reference signatures to train the model are presented in Table 4.3. This table shows that the BW approach responded well to an increase in reference set size and doesn't seem to cope as well as SKM when this set size is small. This table also shows how dramatically the error rates can improve when more reference signatures are used. They may also be used to perform comparisons between this system and others developed using larger reference signature sets.

Chapter 5

Combining Multiple Models

The previous chapters deal with two complementary approaches to handwritten signature verification (HSV). The neural network (NN) approach from Chapter 3 is base largely on the analysis of global features of the handwriting whereas the hidden Markov model (HMM) approach from Chapter 4 examines the local aspects. This chapter deals with the process of combining, or "fusing", the two methods in order to improve the robustness and performance into a single, powerful HSV system.

Fusion of multiple classifiers is a sub-field of biometrics research that has recently gained in popularity (see Section 5.1). There are a number of advantages to be gained by combining the output of multiple biometric systems:

- *Improved performance*: If the underlying systems are complementary and the fusion is done well, the performance of the resulting system will be better than that of the constituent systems. This is analogous to consulting a group of experts and making a decision based on multiple opinions, rather than consulting just one expert, and is the primary advantage of combining classifiers;

- *Increased universality*: The resulting system is generally applicable in more situations. This comes about because if one classifier is confused, a decision may still be made using the other(s);

- *Compromises*: Use of multiple systems introduces the possibility of "compromises" if the classifiers disagree. For example, offering restricted access if one of the classifiers verifies a signature and the remainder do not.

The problem of fusing the output of multiple verifiers is not a simple one and is subject largely to the properties of the constituent systems. It is not valid to assume that the combination will always improve performance, for example it is known that a strong system is better used alone than in combination with a weak one [28]. The underlying classifiers should be complementary and redundance between classifiers may actually degrade accuracy [113]. It is worth noting that the two classifiers being combined in this chapter are complementary both in terms of the type of data captured (local versus global) and in terms of misclassifications. Many of the misclassifications by the individual systems occur on different signatures suggesting that the models capture some independent information that may be exploited.

There are three phases in which classifiers can be combined: the feature extraction phase, the confidence phase and the decision phase [127]. The feature extraction phase generally involves the use of multiple sensors and is not relevant to this chapter. The confidence phase is where most of the following discussion is based and involves combining the actual confidence values output by each of the classifiers. The decision phase involves combining the binary accept/reject decisions of each of the classifiers, and approaches dealing with this phase are also considered below.

5.1 Previous Work

There have been few studies into the combination of multiple handwritten signature classification algorithms. Most of the discussion in this section is centred around methods of classifier combination, or fusion, applied to biometrics other than handwritten signatures.

One of the better designed and developed systems in the HSV literature is an approach combining both local and global information in multiple models [64]. Twenty-three global features were used in the study including total duration, pen-down ratio and velocity and acceleration details. The global feature comparison is done through the Euclidean distance metric applied between the feature vector extracted from the test signature and the reference (a vector of mean feature values taken from six sample signatures).

The local feature comparison is much more complex in that HMMs were used to compare feature vectors. Six sample signatures are used to train the HMM to model the user's signature as a series of states with probabilistic transitions between them. Given a test signature, the Viterbi algorithm [149,

120] is used to search for the most likely state sequence corresponding to the given observation sequence and give the accumulated likelihood score along the best path. That is, the algorithm obtained the likelihood that the test signature can be modelled by the HMM of the particular user. The difference between this score and the mean likelihood obtained during training is then used as an error measure to classify a test signature as valid or a forgery.

The authors attempted to combine the output of the global feature comparisons made using the Euclidean distance metric and the local comparisons using the HMM. Weights were associated with each technique and the two were combined using another Euclidean method by computing their root mean-square weighted combination. The authors only attempted a naive combination using equal weights. The global feature comparison (4.5% equal error rate) had a better error rate that the local approach (about 5% equal error rate) and the combination method improved the result to a 2.5% equal error rate. This reduction in error rate is brought about due to the complementary information being captured by the two different models. A more thorough examination of the global features is desirable, perhaps using a more statistically enhanced model, along with further exploration of means of combining the output of two models.

A number of interesting concepts are introduced in [11] (some previously discussed in Chapter 3) that extends work from [12]. The main aspect of interest relating to this chapter is the method by which the authors combined NNs and HMMs into a single handwriting recognition system.

Specifically, they use a NN that spots and recognises characters and a HMM to interpret the network output by taking word-level constraints into account. The NN and the HMM are jointly trained to minimise an error measure defined at the word level. The HMM models the long-range sequential structure while the NN classifies characters using local spatial structure.

The NN used is a Multi-Layer Convolutional Neural Network (MLCNN) [70, 71], which is a feed-forward NN designed to minimise the sensitivity to image transformations (translation, rotation etc.). The training of the network is done using the back-propagation algorithm.

Three-state HMMs with left-right transitions are used the NN outputs observed for each character. The observation graph was obtained by connecting these character HMMs. The authors obtained some interesting, but not outstanding, results using this approach.

A later, independent system also attempted to combine the advantages of

NNs and HMMs for handwriting recognition and is discussed in [124]. The approach was based on the use of a NN to model probability density functions in a handwriting recognition system. More specifically, the authors replaced the K-Means vector quantizations in a discrete HMM by a NN trained using the maximum mutual information principle (see [97] for more details of this). This hybrid system was shown experimentally to outperform regular discrete HMMs in this environment, slightly increasing maximum recognition rates from 94.6% to 95.0%.

Another study appears in [168] where the authors tried to fuse N decisions made about the author of handwriting samples. This method of fusion differs slightly from others reported in the literature in that it involves fusing the decisions made by the same classifier on a series of N words, known to be produced by the same author (what is not known is whether the author is a forger). That is, each word is verified separately and a probability is generated representing the level of belief that the writing sample is genuine. These N decisions are then fused to make one final decision. A one-dimensional feature vector is derived from each word to make individual assessments of authorship, and a modified randomised Neyman-Pearson test (a method of combining multiple simple hypotheses with individual likelihoods) is used to fuse the separate decisions. Using three words, the results were improved from the one-word non-fusion system error rate of 10% to 1.45% using the fused decisions.

One of the problems with the previous study is the assumption that the decisions made on each word are independent, which in general is not true for several words written by the same individual. Further studies on decision fusion for writer identification were performed by the same group with a slightly different approach [169]. These individual decisions are fused using the Bahadur-Lazarsfeld expansion, which is a method of incorporating multiple probabilistic confidences where independence is not assumed [35]. As in the previous work, the writer identification was carried out using the words of a short sentence with each word being processed separately and used to verify the author of the writing. The average overall error rate is again improved over the individual error rates.

Cooperation of multiple classifiers for character recognition is explored in [115] where the authors attempt to combine two independent nearest-neighbour systems, one off-line and one on-line. There is an obvious advantage in using both data types in that the on-line data are richer, as they supply temporal

information, but more heterogenous too as the same character can be made of a variable number of strokes that may be differently ordered (defining many character allographs). Sixty-two different character classes are used with more than 75,000 examples in the dataset, although the style of the handwriting (cursive, handprinted or mixed) is not described. The authors attempted to use naive weighting, cascading architecture and neural strategies. The naive weighting and cascading architecture approaches both resulted in a 30% relative improvement over the most successful individual (dynamic) system and the neural approach (which involved using a MLP) returned a 50% relative improvement.

More recent work in fusion of handwriting recognition systems appears in [148]. The authors compared four different combination algorithms (Borda count and Choquet integral from [42] and majority rule and averaging from [147]) with their own approach called *modified* Borda count. They use these approaches to fuse three handwriting recognition techniques that use different segmentation and neural network algorithms.

When trying to recognise a single written string, conventional Borda count for a word in a lexicon (a dictionary of words the system is trying to recognise) is defined as the sum of the number of words with lower recognition scores in the different lexicons produced by the various techniques. A higher value for Borda count indicates a stronger belief that a particular word is correct. The problem with conventional Borda count is that it doesn't take into account the confidence values produced by the various techniques (only the rankings). The authors modify this in [148] by essentially summing the product of the ranking and confidence for each technique. It is also possible to assign a weighting to each technique based on the observed recognition accuracy for that approach.

The testing of the individual and combined techniques was performed using cursive handwritten words taken from the CEDAR database [60]. The most successful of the individual techniques resulted in a recognition rate of 88%. It is worth noting that, with the exception of the modified Borda count, none of the combination techniques improved on the most successful individual techniques (in fact, both averaging and the Choquet integral resulted in clearly worse recognition rates). When using the optimal weight values for the modified approach (calculated via brute force) the recognition rate improves to 91%.

It is possible to apply the modified Borda count approach to the problem of handwritten signature verification, but it would be highly computationally

expensive (prohibitively so). It would be necessary to obtain verification scores for each signer in the database every time a signature is verified (the equivalent of performing signature *identification* rather than *verification*). Obviously in a signature database of any realistic size this is impractical.

Perhaps the most popular area for fusion of multiple classifiers is that of *multibiometrics*, where the output values from multiple biometrics-based systems are combined to give a single result [15, 127, 63]. Systems that combine multiple different biometrics are very difficult for a forger to compromise. It is however generally accepted that combining multiple representations and matching algorithms for the same biometric signal (the approach described in Section 5.2) is the most cost-effective and convenient way of improving biometric performance [113].

The authors in [127] and [128] consider three separate approaches to combining information at the confidence level and report on the results obtained. Attempts are made at combining the output of three separate biometric verification systems in face verification, fingerprint verification and hand geometry verification. The three approaches considered are the *sum rule*, *decision trees* and *linear discriminant functions*. Fifty users contributed nine face images, nine fingerprint impressions (of the same finger) and hand geometry data. The best individual biometric is found to be fingerprinting at an overall error rate of around 11%.

The sum rule approach involves taking the weighted sum of the individual output values from each classifier. The authors applied this to each combination of inputs, however only applied equal weight values. Experimentation found consideration of two biometrics to be more successful than any individual biometric, and all three to be more successful than any two. This method resulted in a false rejection rate of 1.78% and a false acceptance rate of 0.03%.

Decision trees are structures that take a set of properties as input, and through a process of discrete decision making arrive at a binary output (see [130] for further details). The authors used the well known software $C5.0$ [117] to generate a decision tree from a training set of over 11,000 "imposters" and 225 genuine subjects. The result of this was a false rejection rate of almost 9.6% and a false acceptance rate of 0.04%.

The final method investigated was the linear discriminant function (LDF) approach. This involved transforming the three-dimensional output vectors (one dimension for each classifier) into a new subspace and maximizing the

between-class separation. The results using this approach were better than for decision trees but worse than the sum rule, leading the authors to conclude that overall, the sum rule was the best method for combining classifiers in their experimentation.

The above system was extended in [63], mainly by augmenting the sum rule to take into account user-specific classification thresholds and weights for individual classifiers. Two methods of obtaining user-specific parameters are investigated. The first method involves assigning equal weights to each biometric (face, hand and fingerprint) and obtaining a new score as the sum of these weighted outputs. User-specific thresholds are then found using the cumulative histogram of imposter scores for each of the three biometric traits for each user. The second method involves estimating user-specific weights by exhaustive search and using a common matching threshold. The authors found that their most successful approach consisted of using common thresholds, but user-specific weights for each classifier. Error rates are reduced by up to 3% of the overall error using this method.

Linear discriminant analysis was also explored in [34], along with linear methods and neural networks as means of combining confidence measures in a speech recognition system. The specific application area was a command-and-control style system and the confidence measures were generated using hidden Markov models. A database of 3,345 feature vectors (roughly half allocated to training and half to testing) was used, with the vectors including five different confidence measures computed from word hypotheses. The smallest error rate for speaker-independent classification using any of the individual confidence measures was 9.8%. Using linear discriminant analysis this was improved to 9.0% and further improved to 8.4% using a one-layer perceptron to combine the confidences. Slightly more substantial improvements were reported using data-dependent confidence measures. The authors also found that use of a non-linear network (one hidden layer) trained using the back-propagation algorithm did not generalise as well as the linear network.

The above findings with respect to the sum rule's superiority over other forms of related fusion techniques are supported in [67] where several similar approaches are explored. These are further verified independently in [82] in an approach used for fingerprint analysis (another popular target area for fusion research). The authors also extended the approach to examine the *product rule*, which involved multiplying the weighted output values from each of two classifiers. The product rule is reported to significantly improve the

performance over the best single algorithm (approximately halving the error rate), however there was no direct comparison with the results using the sum rule.

The authors in [62] used a logistic transform to integrate the output values from three different fingerprint matching algorithms into a single overall score. Given the discrete probability distribution functions obtained from each classifier, the author's algorithm computes the set of tunable parameters of the combination classifier for a set of specified false acceptance rates. Testing of the combination algorithm was done using fingerprint images captured from 167 subjects (the first 83 subjects were used to train the system and the remainder were used in testing). Small reductions in the overall error rates were reported under ideal circumstances, but nothing that justified the extra overhead.

This approach is augmented by a scheme proposed in [113] involving the fusion of four different fingerprint matching algorithms. A large amount of theory is also presented for selecting the most appropriate classifier (based on the "independence" of classifiers), however there are often limitations to the availability of effective systems (for example, there are very few easily available, effective HSV algorithms). The experimental database consisted of 2,672 impressions taken from 167 subjects (four impressions of each of four fingers), however 100 impressions were later removed from the database due to "rejection" by the matching algorithm or due to poor quality images.

Combinations of pairs of classifiers are made by estimating two dimensional genuine and imposter densities from the training data. The optimal setup was found to be the combination of the three most accurate individual algorithms (excluding the fourth and least accurate), and the authors claim an overall improvement of 3%. The weakness with this kind of approach for HSV however is that a large amount of training data is required to obtain useful estimates (of the order of several thousand), along with forgery data.

Complex Bayesian sampling approaches to the decision fusion problem have been proposed in a small number of papers, for example [24]. These approaches are more suited to higher dimensionality situations (that is, where more classifiers are involved) and also have a very high computational complexity less appropriate for use in the signature verification environment.

Finally, a similar approach to that described in this dissertation is presented in [41]. In this article the authors examine a method for combining the output of HMMs and NNs. The HMM system models a signature using a feature vector consisting of seventeen parameters (eight dynamic and nine static)

including velocity, pressure, pen-tilt and some spatial characteristics. The authors use a single discrete, left-right HMM for each user and train the models using fifteen genuine signatures. The HMM technique results in an overall error rate of 6.30% using a database of 1,530 genuine signatures and 3,200 forgeries of various types. FRR is the major contributor to the overall error.

The NN used is a MLP featuring twenty-six inputs (one for each extracted feature), a hidden layer with five units and two sigmoidal outputs (one representing genuine signatures and the other forgeries). The MLP is used to model the global parameters of the signature and includes aspects such as the number of strokes, signature length and a number of velocity, angular and directional features. Training is done using fifteen genuine signatures and one random forgery from each of fifteen other users in the database. The MLP here is used mainly to combat the high FRR exhibited by the HMM, and far less effort is put into the MLP development.

Fusion of the NN and MLP scores is done using an approach known as a Support Vector Machine (SVM), which is a non-linear classification algorithm based on risk minimisation. The authors selected this technique due to its suitability to small training sets. The input to the SVM is the normalized log-likelihood computed by the HMM, smoothed by a sigmoidal function, along with one of the outputs of the MLP (the output that represents the confidence that the signature is genuine). The SVM tries to insert a "decision frontier" into the feature space to separate the confidence scores of genuine signatures and forgeries (the forgeries used are the same as those used in the MLP training). The SVM tries to maximize the "margin" between the separator hyperplane and the data. The final result of the fusion is an improvement from a previous best case of 6.30% to 5.32% overall error rate. This could likely be further reduced by improving the MLP model as well as examining alternative methods for fusing the two scores.

5.2 Methodology

This section presents the methodology and the experimentation performed in the combination of the models described in Chapters 3 and 4. The combination of the two classifiers essentially involves the development of a whole new classifier that has two input values: the confidence value output by the NN system and the normalized log-likelihood output by the HMM system. Figure 5.1 illustrates the basic approach.

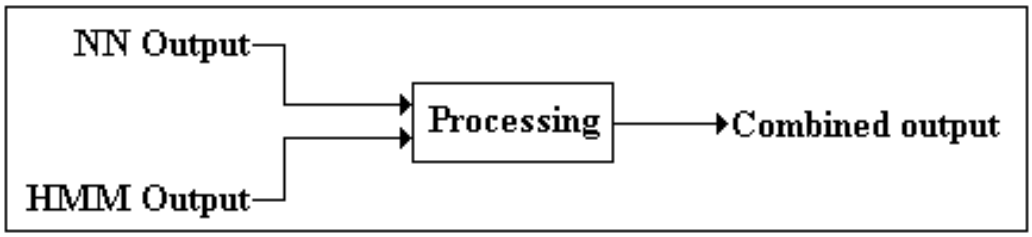

Figure 5.1: *The combination of models described in previous chapters.*

5.2.1 Experimental Setup

This section describes the experimental setup used in developing the combination model. The signature database used in experimentation is described in Section 3.4.2. The type of the two input values differs in that the value provided from the NN is a confidence measure and the HMM value is a probability. The domains of the input values are similar however in that they are both in the zero-to-one range with the same interpretation: a low score implies a low degree of confidence that a given signature is genuine, while a high score implies a high degree of confidence. What follows is a description of the different methods of combining the two models, grouped according to category.

Voting Schemes

These are the simplest methods of combining the HMM and NN output and involve the combination being performed at the decision level. That is, the only input taken in from the two models is the binary decision rather than the confidence values. With two different classifiers as input there are two meaningful schemes or verification scenarios than can be employed:

1. *Unanimous acceptance*: This means that the combined classifier should accept a test signature only if both constituent classifiers accept it. Put another way, the test signature should be rejected if *either* system classifies it as non-genuine. The ideal effect of this is that fewer forgeries will be accepted (as it is less likely that they will deceive both classifiers) without rejecting many more genuine signatures (as well-performed genuine signatures should be accepted by both classifiers). Results obtained using this approach are presented in Table 5.1, which include the false acceptance rate (FAR), false rejection rate (FRR) and overall error rate.

Acceptance Mechanism	FAR	FRR	Overall Error Rate
Unanimous	0.9%	2.1%	3.0%
Disputed	2.2%	1.6%	3.8%

Table 5.1: *The results using the two different voting mechanisms to combine the classifiers.*

2. *Disputed acceptance*: This setup results in a signature being accepted if it is accepted by *either* classifier (or both). Ideally, forgeries are still rejected (as both of the constituent classifiers are quite adept at detecting forgeries) while slightly lower quality signatures from genuine users are still accepted (as they may still contain enough characteristic information to be accepted by at least one of the classifiers). The results obtained using this approach are also presented in Table 5.1. Unanimous rejection by both classifiers should obviously result in the signature being rejected.

When discussing the results presented in Table 5.1 it is useful to recall that the best overall error rates for the NN and HMM systems are 3.3% and 3.5% respectively. Relative to the individual system error rates the "Unanimous Acceptance" results were an improvement over the most successful individual system, whereas the "Disputed Acceptance" approach actually degraded the overall performance. As expected, the unanimous approach resulted in a much lower false acceptance rate as it was far more difficult for a forgery to deceive both classifiers. Fortunately, there was not a greatly adverse affect on the false rejection rate, which meant the overall error rate was improved. The disputed approach results in the false rejection rate slightly improving, however the false acceptance rate increases substantially and the overall error rate suffers as a result. Inspection of the rejected genuine signatures offers an explanation for the lack of significant improvement in the FRR in that these signatures are either poorly written or differed greatly from the signatures given as a reference for that user. As a result they tend to be rejected by both classifiers.

The results above however do not give a definitive answer as to which acceptance mechanism is most suited to combining HSV systems. The approach to use depends largely on the environment in which the signatures will typically be provided. If the environment is casual as in a general purpose system, it is likely that the test signatures will be of slightly lower quality and the disputed approach is more forgiving and more appropriate. If the environment is a formal, high security one, then the added security of the unanimous ap-

proach is likely to result in more desirable performance. It is believed that the formal environment in which the handwritten signature database was captured contributed to the unanimous approaches superior performance.

A voting mechanism may also be used in granting different levels of access depending on the level of signature acceptance. That is, if both classifiers reject the test signature then no access is granted, if both accept the test signature then full access is granted or if the classifiers are in disagreement then partial or restricted access is granted.

Confidence-based Approaches

This section discusses the various techniques used to combine the confidence outputs from each of the classifiers. All explored methods and resulting error rates are presented below.

- *Weighted sum*: The weighted sum rule (sometimes referred to as simply the sum rule) involves taking the weighted sum of the individual scores from the classifiers to achieve the overall score for the combined system:

$$S_{combined} = W_{NN} \cdot S_{NN} + W_{HMM} \cdot S_{HMM}$$

  where $S_{combined}$ is the final score for the combined model, W_{NN} and S_{NN} are the weight and score for the neural network model and W_{HMM} and S_{HMM} are the weight and score for the hidden Markov model. The score values from each of the models are obtained independently and are a measure of the confidence that each model has in the test signature being genuine.

  The weight and threshold values are the same for every user and are obtained in a joint training phase. This phase involves an exhaustive search that tests all weight values in the range [0,1] (with increments of 0.01) for each classifier, with the constraint that the sum of the weights is always 1. The combined scores are obtained for each weight combination and compared to a threshold. The test signature is accepted if the score is above this threshold and rejected otherwise. The threshold value and weight pairing that gave the lowest global (that is, over all users in the database) overall error rate are used for all further experiments. These values are 0.42 for the threshold and 0.62 and 0.38 for the HMM and NN weights respectively.

This approach worked quite well and improved the overall error rate to 2.7% (1.5% FAR and 1.2% FRR).

- *Product rule*: The product rule is quite similar to the sum rule and involves taking the weighted product of classifier scores:

$$S_{combined} = W_{NN}.S_{NN} \times W_{HMM}.S_{HMM}$$

Weight values and thresholds are obtained in the same way as they were for the sum rule approach. The product rule represents the joint probability distribution of the values extracted by the classifiers. This approach produces very similar results to that of the sum rule and obtained an overall error rate of 2.8% (1.3% FAR and 1.5% FRR).

- *Mean transformation*: The mean transformation is essentially a special case of the weighted sum with the weights set to 0.5 and is used in [82] to combine classifiers for fingerprint verification. This transformation simply takes the mean of the two classifier scores for a test signature:

$$S_{combined} = \frac{S_{NN} + S_{HMM}}{2}$$

The mean approach did not perform well in this instance, returning an overall error rate of 3.9% (1.8% FAR and 2.1% FRR), which is worse than both individual classifiers.

- *Decision trees*: Results were obtained using the $C5.0$ program [117] to generate a decision tree from a training set of classifier score pairs. Five genuine score pairs were used as a reference and the score pairs from genuine signatures of twenty-five other users in the database were used as negative examples (the twenty-five users were selected in a similar way to training forgeries selected in Chapter 3). The data seemed to be insufficient for accurate construction of decision trees and $C5.0$ did not perform well using only five genuine references, resulting in an overall error rate of 8.4% (5.5% FAR and 2.9% FRR).

- *Multi-layer perceptrons*: Multi-layer Perceptrons (MLPs) are used in Chapter 3 to build one of the constituent signature verification systems. It was theorised that a non-linear classifier such as a MLP may be able to achieve better classification by capturing a more insightful relationship between the two confidence measures. The basic model structure

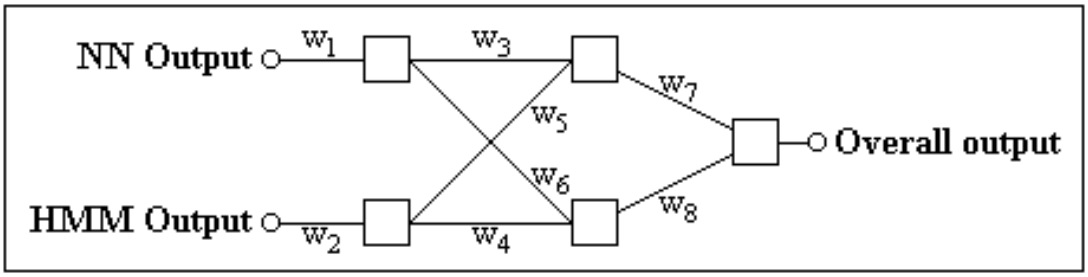

Figure 5.2: *The MLP structure that produced the lowest overall error rate when combining the constituent systems. Each of the weight values W_i is optimised via a learning algorithm.*

consists of a three-layer network with two input units and one output unit. The back-propagation algorithm was used to train the model and experimentation was done with different numbers of nodes in the hidden layer, but the most successful structure found contained two hidden nodes (the structure is illustrated in Figure 5.2).

The makeup of the training set (in terms of positive and negative examples) was identical to that used for the decision tree approach described previously. The performance of the MLP over the training set was very good but did not generalise as well to the test set, most probably because of the small amount of input data. The overall error rate using this approach was 3.0% (1.3% FAR and 1.7% FRR).

- *User-specific weighted sum*: This is the most successful of the approaches to combining model scores. The method here is similar to that used in the "weighted sum" approach, modified to apply to individual users. The basic algorithm for adjusting the user-specific weights is as follows:

  1. For user i, vary weights $W_{NN,i}$ and $W_{HMM,i}$ over the range [0,1] (with increments of 0.01) with the constraint that $W_{NN,i} + W_{HMM,i}$ equals 1;
  2. The overall score used for verification is then:

  $$S_i = W_{NN,i} \times S_{NN,i} + W_{HMM,i} \times S_{HMM,i}$$

  3. S_i is compared to a user-specific threshold T_i for each user and the test signatures is accepted if $S_i > T_i$ and rejected otherwise;

4. Choose the set of weights and thresholds that minimises the *total error rate* associated with the overall scores.

Here $W_{NN,i}$ refers to the user-specific weight associated with the NN output for user i and $W_{HMM,i}$ refers to the user-specific weight associated with the HMM output for user i. Similarly $S_{NN,i}$ refers to the score (that is, output or confidence) from user i's NN and $S_{HMM,i}$ the score from user i's HMM. S_i refers to the overall score value for user i. The *total error rate* referred to in step 3 of the algorithm (not to be confused with the *overall* error rate) is the sum of the individual error rates calculated during the training phase. Details of the weight and threshold selection appear below.

The overall score S_i is obtained for each weight combination in the range [0,1] (with increments of 0.01), with the constraint that the sum of the weights is always 1. In an extended training phase, error rates are calculated using the original reference signatures as genuine attempts (that is, the training and testing databases remain separate) and a set of thirty-five forgeries (obtained from the other users in the database in the same manner as described in Chapter 3). These error rates are calculated by exhaustive experimentation with threshold values, varying the threshold in the range [0,1] (with increments of 0.01) for each weight pairing. The threshold and weight values triple that produces the lowest overall error rate is then fixed for that user. Often a range of weight values results in an overall error rate of zero for a particular user - in this case the median of each weight range is used. Similarly, the median threshold is used when there is a range of threshold values resulting in the equal lowest error rate.

Overall error rates for each user are then calculated using the fixed weight and threshold values with the previously unseen genuine signatures and skilled forgeries being used as test signatures. This approach improved the overall error rate to 2.1% (1.1% FAR and 1.0% FRR) and returned an equal error rate of 1.1%. All error rates quoted in following sections will be based on this user-specific weighted sum approach unless otherwise specified.

The results of all of the confidence-based methods of combination are summarised in Table 5.2.

Combination Method	FAR	FRR	Overall Error Rate
Weighted sum	1.5%	1.2%	2.7%
Product rule	1.3%	1.5%	2.8%
Mean transformation	1.8%	2.1%	3.9%
Decision trees	5.5%	2.9%	8.4%
Multi-layer perceptrons	1.3%	1.7%	3.0%
User-specific weighted sum	1.1%	1.0%	2.1%

Table 5.2: *The resulting error rates using the different confidence-based approaches to combining the classifiers.*

Model	FAR	FRR	Overall Error Rate
Neural network	1.1%	2.2%	3.3%
Hidden Markov model	1.2%	2.3%	3.5%
Weighted sum	1.5%	1.2%	2.7%
User-specific weighted sum	1.1%	1.0%	2.1%

Table 5.3: *The most successful results for each of the different model scenarios used during development.*

A summary of all of the developmental models appears in Table 5.3, including the neural network alone, the hidden Markov model alone, both models combined via the weighted sum rule and both models combined via the user-specific weighted sum approach. As can be seen, the combination of models resulted in an increased performance over both individual models, with the user-specific weighted sum approach being most successful.

5.3 Further Results

The lowest error rate obtained for any system described in this thesis (the "User-specific weighted sum" version of the combined network) produced an overall error rate of 2.1% when tested over the entire database. This section presents some further results of interest obtained throughout testing of this final system.

5.3.1 Removal of "Short Signatures"

Any methodology that includes the removal of "unsuitable" signatures from a database is dubious, but is done here for comparative purposes. Other researchers (for example, [32]) have examined this sort of thing, purely for the sake of interest not through any attempt to artificially reduce the reported error rates of their systems. Short signatures (those with a duration less than some time t) are going to contain less information and be more variable than signatures of more significant length. Due to this lack of consistent information content it is more difficult to verify genuine short signatures and more difficult to reject attempted forgeries of these. As such, although these signatures constitute a small percentage of the database, they have a disproportionately large affect on the error rates. It can be argued that removing these signatures from the database gives a fairer description of the accuracy of the system in question.

Figure 5.3 shows the breakdown of average signature duration for each user sorted in order of ascending duration. Figure 5.4 shows the error rates that result from excluding all signers whose average signature duration is less than t seconds. As can be seen from the figure, as more short signatures are removed the error rate tends to reduce proportionally. A similar study in [32] found that the error rate was almost halved when a duration limit of 1.25 seconds was set. The error rate for the user-specific weighted sum system reduces from 2.1% to 1.6% using this same limit (which excludes a total of seven users from consideration).

5.3.2 Contribution to Overall Error

For various reasons it is typical for a small proportion of signers to be responsible for a large proportion of the error rate. Reasons for this include small signature duration as discussed in the previous section, unusually large variation in signature style or in extreme cases signers may have more than one version of their signature (as was the case with one signer in the database used throughout this thesis). These types of users are always going to be present in a large-scale realistic database, so it is not valid to exclude these users from calculations of error rate. However it is insightful to examine individual user's contributions to the overall error rate. Figure 5.5 is a plot of these individual contributions and illustrates that the majority of the error is contributed by a minority of the users. The entire error is contributed by eleven users, over 35%

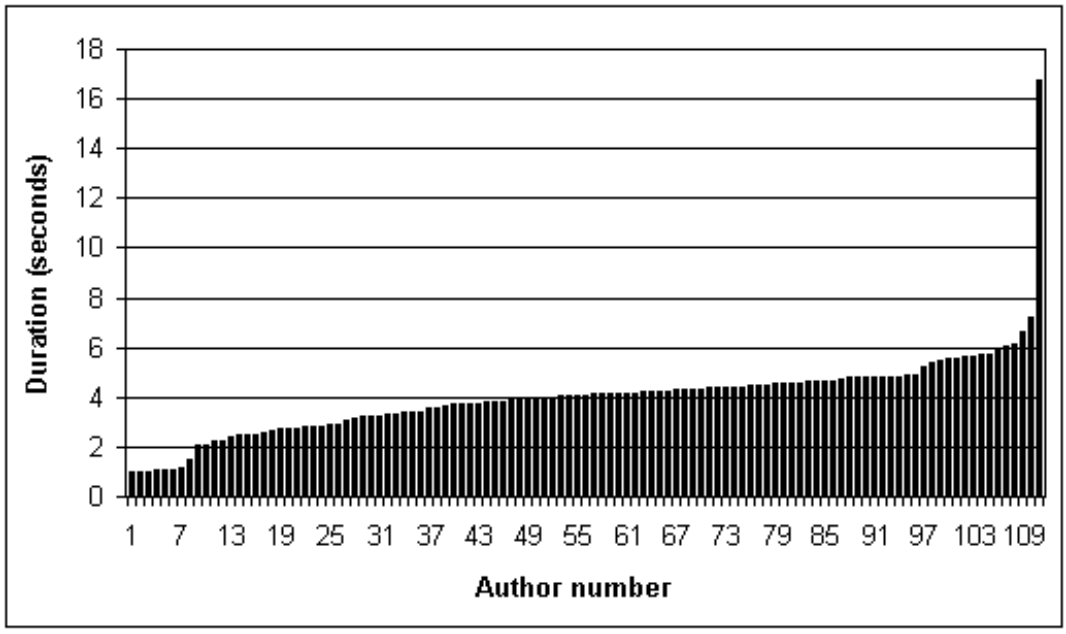

Figure 5.3: *The average signature duration (in seconds) per signer.*

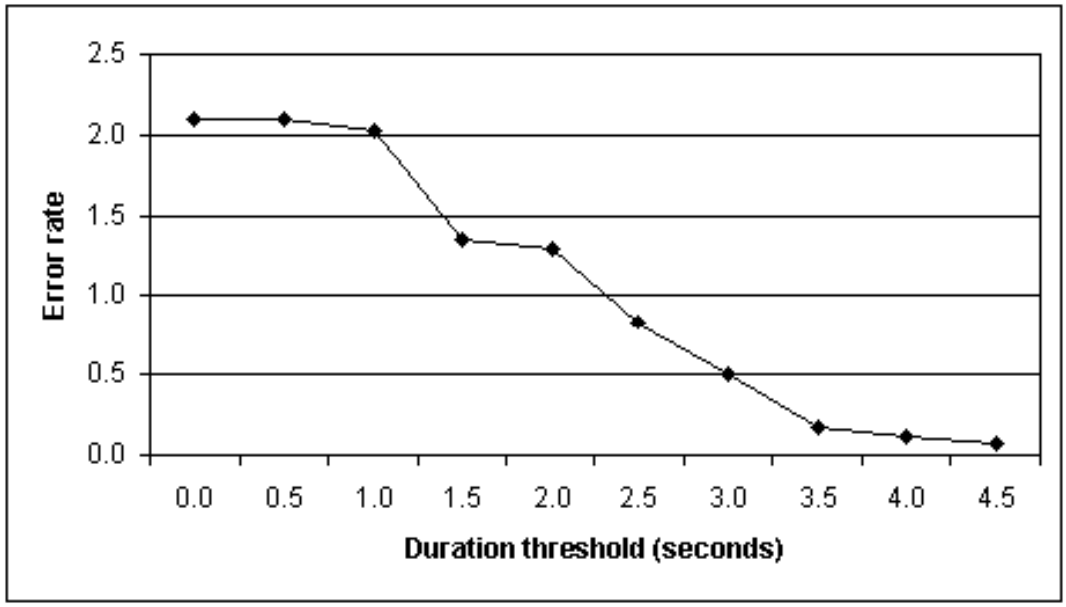

Figure 5.4: *The overall error rate versus the duration threshold. Signers with an average signature duration less than* t *seconds were removed from consideration. As can be seen, error rates generally improve as signature duration increases.*

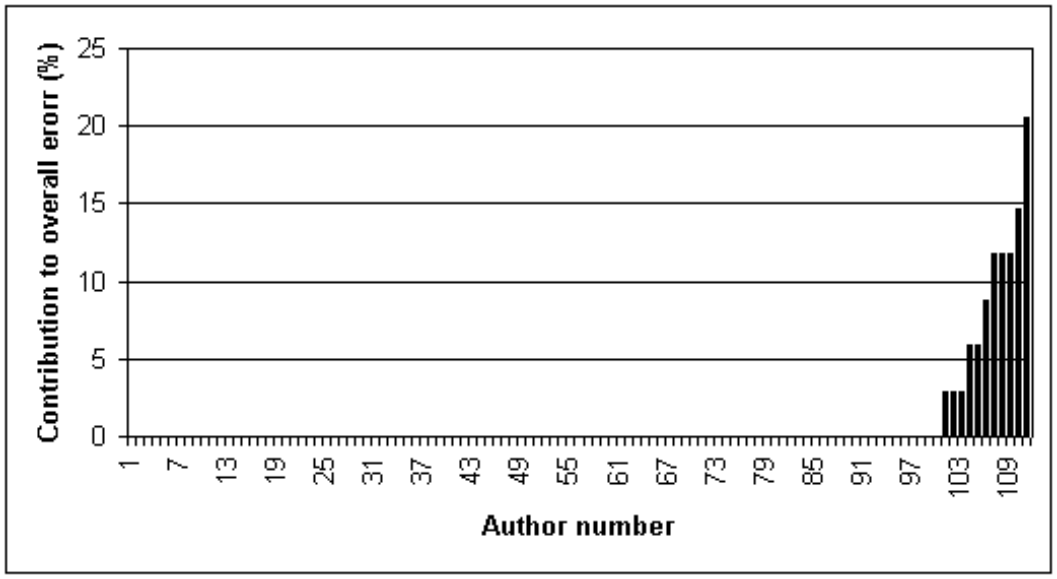

Figure 5.5: *A plot of individual contributions to overall error rate, sorted in order of increasing contribution.*

of the error is contributed by two users and over 45% is contributed by three. The verification is perfect for over 90% of the users (the exact weight and threshold values selected for these users is less important as small variations had no effect on the verification error rates).

5.3.3 Allowing Users Another Chance When Rejected

This approach has been taken by other authors in the literature [51, 80] and is perhaps a realistic approach to how HSV systems would function in a general usage scenario. If a test signature is rejected by the particular HSV system then this approach allows the signer to attempt verification again (typically a maximum of three attempts). This would be acceptable in a situation where more importance is placed on lowering the FRR and lessening the inconvenience to genuine users, rather than the emphasis being placed on security. Less false rejections will occur through this method, however there will most likely be an increase in false acceptances.

Experiments conducted to evaluate the success of this kind of approach involve firstly training the system, then presenting test signatures to the system for each user (as with other testing approaches). The difference here is that if the test signature fails, the following signature is tested, and a third if the second also fails. Acceptance is deemed to occur as soon as one of the test

Reference Set Size	FAR	FRR	Overall Error Rate
5	1.1%	1.0%	2.1%
8	1.0%	0.8%	1.8%
10	0.8%	0.8%	1.6%

Table 5.4: *A breakdown of the error rates for various reference set sizes, optimised to give the lowest overall error rate.*

signatures is verified, but if all three signatures fail the test signer is rejected. As expected, the FRR is improved dramatically using this approach to 0.1% with FAR suffering, rising to 1.5%.

There are a number of modifications that can be made in this sort of system, such as only allowing users to re-try if one of the constituent classifiers verifies the test signature.

5.3.4 Varying the Size of the Reference Set

All results discussed so far involve the use of just five reference signatures. Many systems presented in the literature require more than five samples (typically ten) in order to build a reference (for example, [26, 133]). Using more reference signatures will most likely lead to lower error rates as it allows further information to be extracted and more effective models to be built. The cost of using more reference signatures is extra computation, extra memory storage, greater inconvenience to users and the risk of them getting frustrated and refusing to use the system at all. There is also an issue with fatigue if a user is required to provide too many signatures in one sitting and a resulting loss of reference signature quality.

Experiments were conducted using more than five reference signatures in order to facilitate a more accurate comparison with other systems in the literature that use more sample signatures to build a reference. Table 5.4 and Figure 5.6 show the affect of increasing the number of reference signatures on error rates.

Note that there are approaches that can be used to obtain larger reference sets without the inconvenience. One example is to take five reference signatures during the enrolment phase and adding any later verified signatures to the reference set and re-training the models. The obvious disadvantage with this kind of approach is that if a forgery is incorrectly verified then it will be added

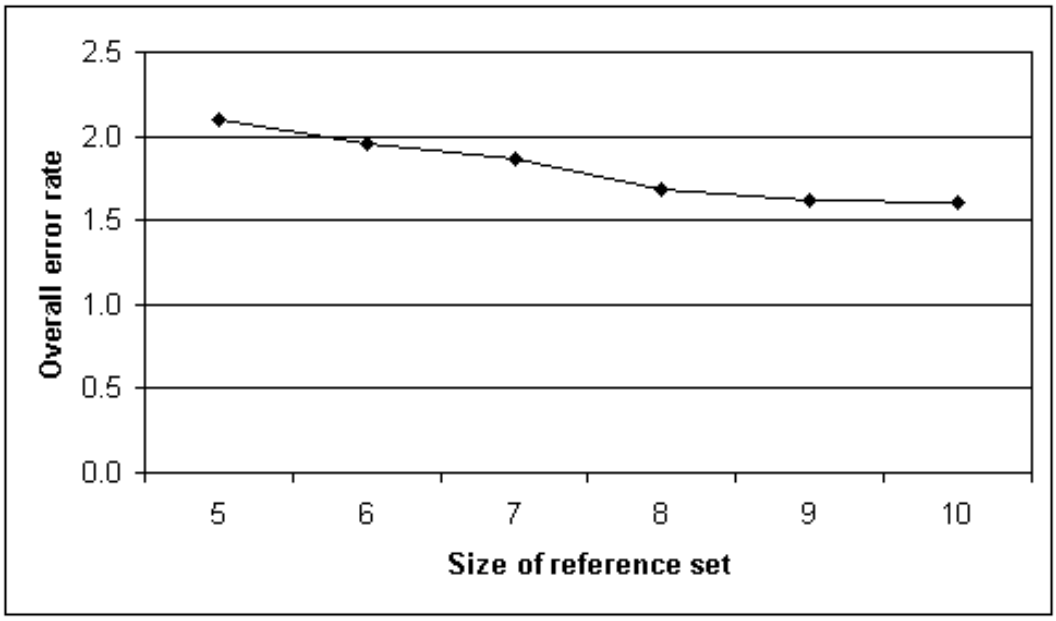

Figure 5.6: *The overall error rate versus the number of reference signatures used.*

to the reference set, corrupting the model.

5.3.5 Zero-Effort False Acceptance Rate

The zero-effort false acceptance rate (ZEFAR) has been discussed in previous chapters and is a measure of the "confusion" the system exhibits or the likelihood that a forger with no knowledge of the genuine signature will provide a successful forgery. ZEFAR is quoted in many HSV articles in the literature and is a useful measure as it illustrates the class separation obtained by the developed system. It is also useful as most successful forgeries in general signature verification environments are either zero-effort forgeries or are very poor attempts.

In order to arrive at a figure for ZEFAR, all signatures (both genuine and provided forgeries) from other users in the database are used as test signatures. This means that there are approximately 3,850 (approximately 35 signatures from each of the 110 other users) zero-effort forgeries for each user, totalling over 430,000 for the entire database. When testing the trained system, a total of 388 of these test signatures were accepted as genuine, resulting in a ZEFAR of 0.09%.

Note that no re-training occurred specifically for these tests and weight and

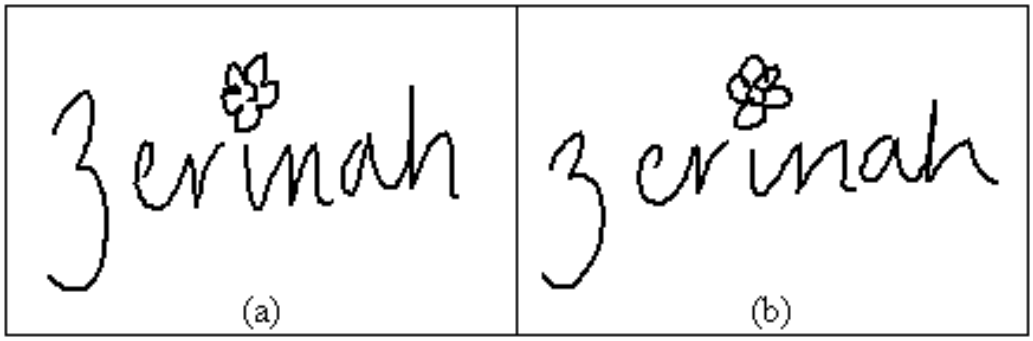

Figure 5.7: *(a) A signature sample captured using a stylus to provide visual feedback to the signer. (b) A signature sample from the same author captured without the use of the stylus.*

threshold values were the same as those used in the "User-specific weighted sum" approach.

5.3.6 The Importance of Visual Feedback When Signing

As part of the HSV system development a number of experiments were conducted to assess the importance of visual feedback when performing a signature. It is generally accepted that the movements controlling signature production are stored in some kind of muscle or nerve "memory". Further, once a signature has been practiced sufficiently the nerve impulses are controlled by the brain without any particular attention to detail [54].

Three experiments are conducted using the final version of the HSV system in an attempt to lend support to the hypothesis that no visual feedback is required when signing. The experiments involve thirty of the signers from the original database contributing a set of five signatures using a stylus that provided them with visual feedback as they wrote (that is, it reproduced the pen-tip path). Visually, the stylus-based signatures appear very similar to the regular (produced without the aid of the stylus) signatures with one signer producing identical durations in two cases. An example of a more elaborate signature can be seen in Figure 5.7 where part (a) shows the signature sample captured using a stylus and part (b) shows a sample captured without the use of a stylus.

The first experiment involved using the stylus-based signatures to train the models. The genuine signatures previously provided by the signer (without the use of the stylus) were used as test signatures along with the previously

provided forgeries (fifteen genuine signatures and five forgeries were available for testing). If the hypothesis is correct then the error rates, particularly the FRR, should be approximately the same as when there was no stylus used to produce the reference signatures. Without the use of the stylus in training the overall error rate for these thirty users is 2.4% (1.1% FAR and 1.3% FRR). When the stylus *is* used the FAR remains the same and a single extra false rejection is recorded, resulting in an overall error rate of 2.7% (1.1% FAR and 1.6% FRR).

The second experiment used the regular signatures (produced without the use of the stylus) to train the system and attempted to verify the stylus-based signatures. The FRR in this case was just 0.7% with only one of the stylus-based signatures being incorrectly classified. The FAR using this training mechanism is reported in Section 5.2.1 as 1.1%, giving an overall error rate of 1.8% (note that the set of genuine signatures is much smaller in this experiment).

The final experiment used a mixture of the two types of genuine signatures to train the system (three regular and two stylus-based). The results using this setup were identical to when no stylus-based signatures were used, at 2.4% overall error rate (1.1% FAR and 1.3% FRR).

This series of tests strongly supports the hypothesis that visual feedback is not required when reproducing practiced writings, particularly signatures.

5.3.7 Manually Adjusted Personal Thresholds

This section involves manual inspection of the class separation for comparative purposes. The "User-specific weighted sum" version of the system attempts to take into account personal weights and thresholds to maximise the verification accuracy. The approach worked quite well in most instances however it is very difficult to do this accurately with such a small training set. Manual calculations can be made to determine the ideal weight and threshold values and the resulting error rate. This error rate represents the maximum accuracy that can be achieved using the two models with this data. In other words, the optimal class separation obtained by the HSV system on the given data set. The overall error rate when manually selecting the optimal weights and thresholds is 0.47% (0.16% FAR and 0.31% FRR). The system fails to achieve complete (manual) class separation on just four signatures, generated by just two signers.

5.3.8 Signing a Password

One of the issues with HSV is that an individual's signature is generally not secret (not in the same sense that a password or personal identification number is). A potential forger may be able to obtain a copy of a genuine signature and then has the opportunity to practice (although error rates have proven to be quite good even for practiced forgeries).

In an attempt to build a variant of the HSV system that was not subject to these problems, experiments were conducted where users "signed" a password instead of their signature. Signing a password takes advantage of multiple security schemes in that a potential forger not only has to know (or guess) the user's password, but also has to be able to reproduce the structure and style of the genuine writing.

The first stage of the process involved forty-seven users being instructed to think of a password (one they could easily remember) and practice the writing over a period of one to two weeks until they felt comfortable with the writing style. The writers then provided five samples of their signed password (there was no stylus used here as to do so would have the undesirable effect of leaving a visual artifact that a potential forger can copy) on three separate occasions, totalling fifteen genuine samples per user (five of which were used in training). Three types of forgeries were gathered based on information that the forger was provided. The first type of forgery was produced by the forger where they were not given the password to be forged, so had to guess. With the second type, the forger was told the password but was not given access to the writing style. The third type of forgery involved the forger being told the password and being given a sample of the writer's natural style, but no sample of the actual written password was given. Five of each type of forgery were gathered for each user in the database.

Error rates using the three different types of forgeries are presented in Table 5.5. As can be seen, the approach works quite well with no false acceptances when the forger didn't know the password or when the forger had no access to sample handwriting. In most cases the genuine users wrote their passwords with some originality or flair (making it very difficult to forge if this style is unknown). Conversely, forgers would sometimes try to fool the verification system by drawing straight lines or squiggles when the password was unknown.

This would be a useful general purpose standalone security system in many situations, for example in granting access to a personal computer containing

Forgery Type	FAR	FRR	Overall Error Rate
No information	0.0%	0.6%	0.6%
Told password	0.0%	0.6%	0.6%
Told password, given general sample	0.9%	0.6%	1.5%

Table 5.5: *The resulting error rates using different types of forgeries in the "signing passwords" variant of the HSV system.*

sensitive information. The cost of the additional hardware is minimal and the gain in the level of security is quite high. Personal data organisers or hand-held computers with a pen interface would also be potential environments that could benefit from this type of approach.

Chapter 6

Conclusion

This thesis has presented a discussion of the development of an on-line handwritten signature verification system based on complementary pattern recognition models. The neural network-based system concentrated on global signature features whereas the hidden Markov model-based approach examined local aspects. It has been demonstrated that these models can be used in a complementary fashion and the outputs combined to achieve highly accurate classification (2.1% overall error rate) in a database of 120 writers and well over 2,000 signatures.

Different features sets have been examined for their appropriateness to various aspects of the signature verification problem and this thesis has introduced some insightful features not previously used in existing signature verification technology. A refined means of stroke extraction was also introduced that resulted in more accurate segmentation of the signature and a significant reduction in overall error rate of almost 50%. Stroke segmentation is a common module in many handwriting based systems, so this procedure may be useful in other signature verification and handwriting analysis systems.

A study of extracting forgeries required for training from a database of genuine signatures was also presented. Many of the models used in handwritten signature verification require presentation of negative examples in the form of forgeries. As it is infeasible to obtain skilled forgery attempts for every user in a realistically sized database, it is not possible to use actual forgeries as the negative examples. Similarly, it is not feasible to use genuine signatures from all other users in the database and completely re-train every time a new user is enrolled. A means of extracting the most useful forgery set from the existing user data was presented based on a signature distance metric and was shown to be successful.

Chapter 3 details the development of a neural network mechanism for learning the global aspects of a signature. Specifically, a multi-layer perceptron is used for experimentation and a study is performed into the most effective network structures, the most efficient and most accurate learning algorithms for the given features along with various other optimalities. A structure consisting of a multi-layer perceptron with one hidden layer performs most successfully over the database as a whole. The back-propagation learning algorithm displayed a greater ability to generalise than both the conjugate gradient descent and Levenberg-Marquardt learning algorithms. Despite taking longer to learn that the conjugate gradient descent algorithm, the back-propagation approach was considered superior for this application and resulted in a 3.3% overall error rate.

Various aspects of hidden Markov models are explored in Chapter 4. This experimentation includes an investigation into the number of states that is most optimal for use in handwritten signature verification (empirically found to be 0.8 times the number of strokes in the signature). A comparison of the two classical hidden Markov model learning algorithms is made, with the Segmental K-Means approach found to converge much faster (in terms of both the number of iterations and the overall time required) than the Baum-Welch approach. The Baum-Welch approach is shown to be more sensitive to a lack of training data than the Segmental K-Means approach, which resulted in an overall error rate of 2.7% using five reference signatures.

Methods of combining the output of neural networks and hidden Markov models are considered in Chapter 5. Various techniques are implemented including voting schemes, classical approaches like the sum and product rules, decision trees and neural networks. The most successful approach (in terms of error rates) is the proposed "User-specific weighted sum", which, while relatively expensive computationally, resulted in a reduced overall error rate of 2.1%.

Other experimentation includes a study into the importance of visual feedback to the signing process and confirms that visual feedback is not necessary and does not influence classification accuracy. There was also an investigation into the possibility of using a written password instead of a signature to increase the level of security (as a potential forger would need to know the password itself in addition to being able to forge the writing style). The study was very successful when tested using a database of over 1,400 writing and forgery samples, resulting in an overall error rate of 0.6% when the potential

forger does not know the password or writing style of the genuine user.

Stronger emphasis needs to be placed on the sub-problem of selecting personalised weights and thresholds because of the disparity between the current result (2.1% overall error rate) and the best possible result of 0.47% based on manually determined thresholds. This will likely be a more popular future area of research in handwritten signature verification and biometrics in general.

Another area of future work is in the analysis of the independence of the two classifiers used in this study (and indeed in classifier fusion in general). The methodologies described in previous chapters present demonstrably different classifiers, both in terms of input features and learning/classification techniques. Further work would include a detailed investigation as to the style of signatures that are incorrectly classified by both individual systems, and the tradeoff between individual classifier accuracy and independence of classifiers.

Because of the nature of handwritten signatures and the fact that some users vary between signature versions, flawless classification is the unachievable goal of handwritten signature verification systems. Performance is already far in advance of human capabilities. The process of comparison between different signature verification systems is made very difficult as a result of the different signature databases used. The presence and quantity of forgeries as well as the method in which they were obtained is a useful indicator of database quality. The forgeries captured for use in this study were performed after the forgers had viewed sample signatures being produced so that the forgeries are of high quality. Given the quality and size of the database, the verification results achieved are excellent when compared to other researchers in the area.

Appendix A

The Extremum Consistency Algorithm

This appendix details the Extremum Consistency (EC) algorithm for avoiding local maxima and minima in a specialised domain. In addition, details are presented that show the superiority of EC over thresholding, hill-climbing (or gradient-descent), simulated annealing and convolution in three large-scale practical applications.

A.1 Introduction

This appendix presents a new algorithm for avoiding local maxima and minima in a specialised domain. The performance of the algorithm is measured against a number of other classical local extremum avoidance algorithms, with quite respectable results.

Note that throughout this appendix the discussion will tend to focus on minima (also known as valleys or troughs) in the interests of brevity. Discussions can trivially be adapted to avoidance of maxima or peaks. Additionally, chiefly for convenience, this algorithm has been given a name: Extremum Consistency or EC. The meaning behind the name will be made clear later in the appendix.

The underlying problem lies in deciding whether a given extremum represents a globally optimal extremum. This of course is not a problem new to computer science, but it has never been approached in the particular way described here.

There are several existing algorithms for avoiding local extrema, however none have proven to be effective in the specific domain outlined in Section A.2.

The most appropriate of these existing algorithms were implemented in a complete system and the results and performance of the resulting system were compared.

The content is organised as follows, Section A.2 describes the problem domain and motivation for the algorithm, Section A.3 gives details of the algorithm itself and Section A.4 outlines three different software applications where the algorithm has been successfully applied. Future work is discussed in Section A.5 while Section A.6 presents the concluding remarks.

A.2 The Problem Domain - Motivation

This problem domain is based on iterative improvement strategies in that an attempt is being made to find maxima or minima while "stepping" or "traversing" along a certain stream.

Abstractly, what is required in this domain is for an algorithm to move in the direction of decreasing value until a minimum is reached. One there, the location of the minimum is recorded and, depending on the application, either proceeds in the opposite direction looking for a corresponding maximum, or "jumps" out of the minimum and begins the process again. This search continues until all of the minima (and maxima if required) are mapped and then processing of these points is performed (see Section A.4 for more information on the specific applications). Because of this pattern of searching, any local minimum encountered will adversely affect the performance of the system, not only because the recorded position is incorrect, but starting the search for the corresponding extremum too early will propagate that error.

The main problem therefore lies in detecting the *true* (or, in a sense, global) extrema in a volatile and possibly noisy environment. The algorithm presented here is a specific approach to deciding whether a particular extremum represents a true extremum.

The type of situations typically encountered in this environment appear in pictoral form in Figure A.1. The horizontal axis in these diagrams represents time and the vertical axis can represent various observations such as velocity, direction or temperature.

The initial motivation for this algorithm came when developing signature verification software in [86]. This approach was based on detecting the order of "turning points" (in other words, minima and maxima) in the pen tip direction and processing the locations of those turning points. The `EC` algorithm

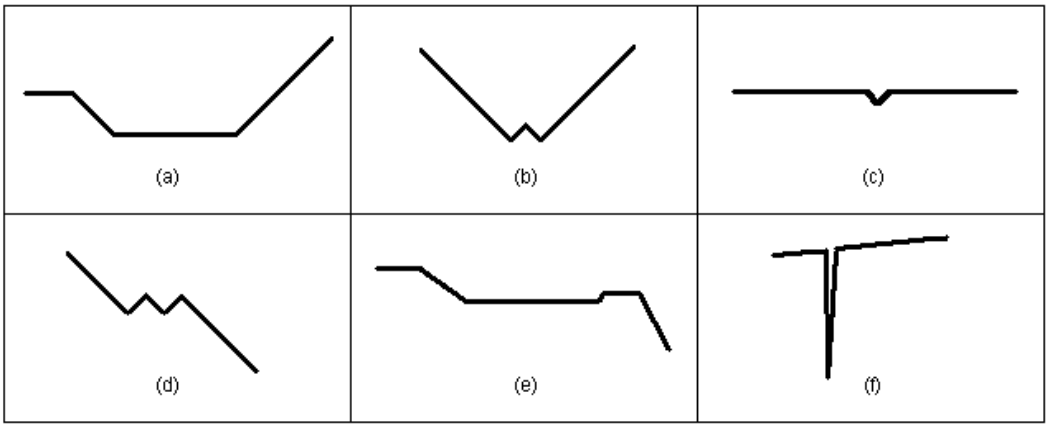

Figure A.1: *This figure represents some local minima situations which are typically encountered in processing the input stream. The horizontal axis represents increasing time and the vertical axis can represent various stream types such as velocity, direction and temperature. Specifically, (a) contains a valid minimum, (b) contains only a single valid minimum (there are actually two minima, but the second is the result of the local maximum in the centre, which should be ignored in this environment) and all others contain no "true" minima. An effective algorithm should reflect this.*

was initially implemented as part of this application (replacing hill-climbing) with significant improvements in system accuracy. See Section A.4 for more information on this and other application areas.

The challenge for this and similar algorithms is to ignore meaningless small fluctuations which appear in the stream, while recording the *meaningful* fluctuations. The problem lies in distinguishing between the two.

Local extrema can enter the data as a result of various aspects, such as a kind of rounding problem (discussed below), quantization noise or, in handwriting based applications (such as those discussed in Section A.4), something as simple as shaky hands. These local extrema can be of varying height, making them more difficult for conventional algorithms to overcome.

The rounding problem is due to the discrete resolution of hardware such as graphics tablets used to capture handwriting. The problem occurs when the actual handwriting path travels directly between two neighbouring pixels. As a result the tablet must "round" the pen tip location to the nearest pixel. Occasionally, slight variations in pen tip pressure cause the rounding to be done to a different pixel. Figure A.2 shows an example of this. The resulting handwriting path then looks (to the system) like that shown in Figure A.1(b)

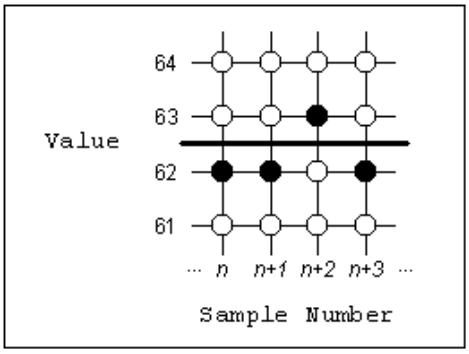

Figure A.2: *Situations like this are the result of the limited resolution of the hardware used to capture a stream. The black line represents the actual value of the stream and the black dots represent the recorded value. Time is represented on the horizontal axis. This situation typically arises when the hardware is a graphics tablet which rounds the position of the pen tip to the nearest pixel, but also comes up with (say) temperature observations when the actual temperature is rounded to the nearest tenth of a degree for recording.*

or Figure A.1(c).

A.3 The Algorithm

The algorithm itself is quite compact. It requires very little stored data (four integers), no search tree and is implemented via a series of comparisons done while traversing the surface of the feature space. Additionally, this algorithm is only executed when a potential extremum is encountered so the affect on the efficiency of the overall system is slight.

The main difference between this algorithm and others is that it examines, primarily, the *width* (or perhaps more accurately the *consistency* or *duration*) of the minima. Most other algorithms (such as convolution and thresholding) place more weight on the *depth* of the minima. The use of the term *width* here differs slightly from an intuitive understanding of the width of a valley. The width of a valley is perhaps best explained by considering the initial valley downslope and the following upslope separately. Once these two values are found, the width of the valley is simply the minimum of the individual widths. The width of a slope is defined as the number of "steps" encountered in its traversal. A step in a downslope refers to a decrease in height below the value of the current minimum. The more of these decreases there are, the larger the

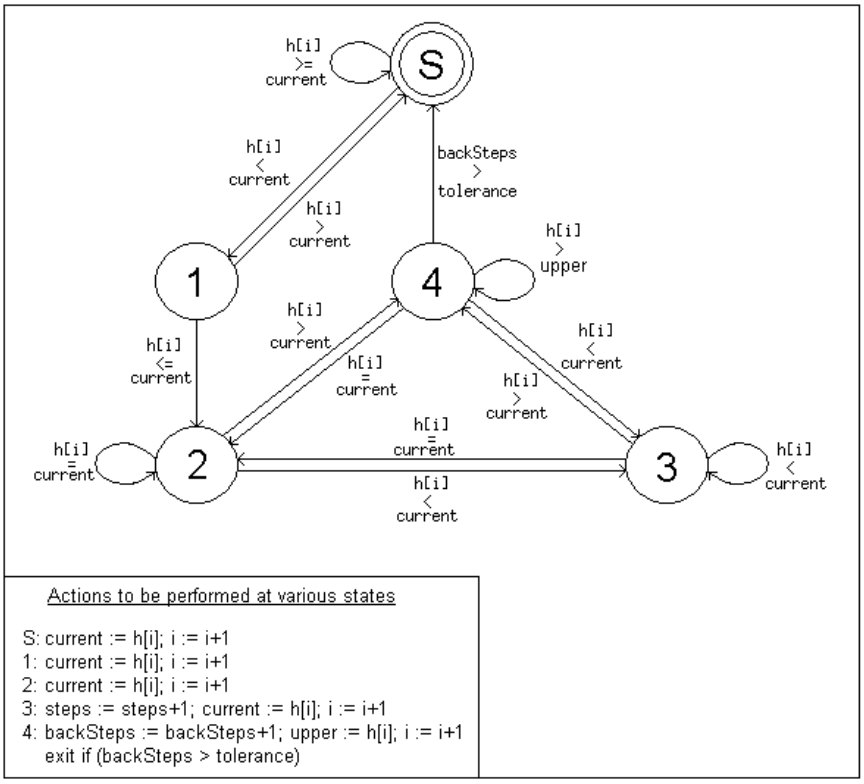

Figure A.3: *A finite state machine expressing the* `EC` *algorithm. The movements between vertices (states) are defined by the comparison between points in the input stream and the comparisons are included on the edges in the diagram. Additionally there are actions to be performed when vertices are reached - these are also included in the diagram.*

number of steps.

The operation of the `EC` algorithm is illustrated graphically in the finite state machine in Figure A.3. The remainder of this section contains textual descriptions to accompany the illustration.

Figure A.3 illustrates an example of step and width calculation. Steps have been defined as movements in the direction of a particular extremum - for example, in Figure A.3 the movement between $time = 0$ and $time = 1$, $time = 1$ and $time = 2$, $time = 3$ and $time = 4$ and $time = 5$ and $time = 6$ each constitute a single step. The term "backward steps" is now also defined as movements *away* from the extremum. For example, in Figure A.3, the movement between $time = 6$ and $time = 7$ as well as between $time = 9$ and $time = 10$ can be thought of as taking a backward step. A tolerance parameter determines how many backward steps are accepted before the algorithm

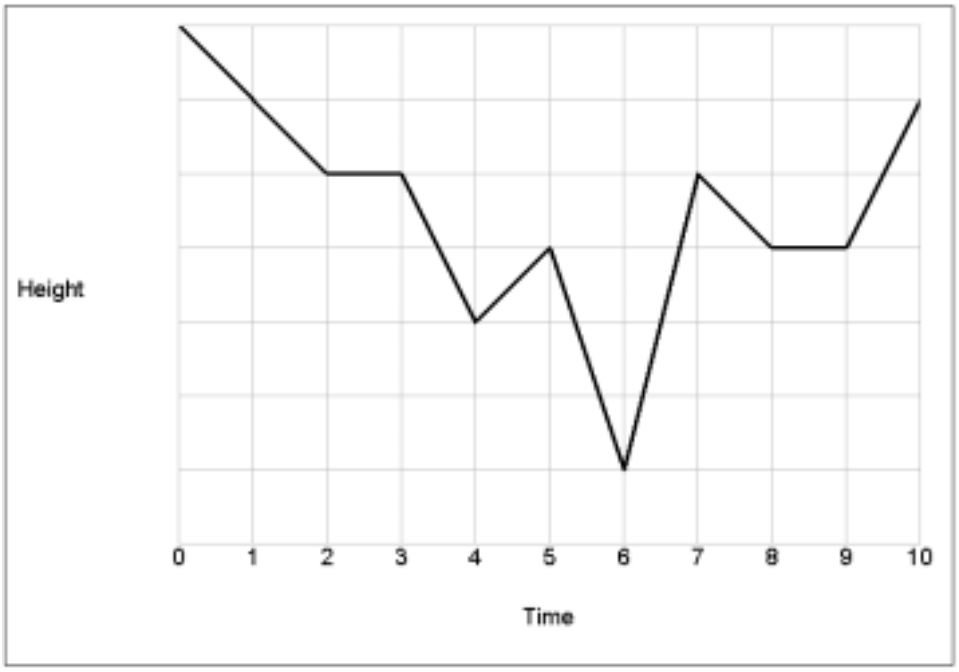

Figure A.4: *An illustration of step and width calculation. Valid steps occur between time points* 0 *and* 1, 1 *and* 2, 3 *and* 4 *and* 5 *and* 6. *Backward steps occur between time points* 6 *and* 7, *as well as between* 9 *and* 10.

terminates.

Upon termination we have the value for the width of the slope. Tolerance then becomes a significant factor as it represents the amount of time spent looking for a "better" extremum before giving up and accepting the one we have. If the location of this extremum is the only information sought (as is often the case, depending on the application), the goal has been achieved and the algorithm ends. If, however, the width of the entire valley is sought (as in the application described in Section A.4.1) then the search for the top of the post-valley upslope takes place, starting from the valley floor.

The width of the entire valley then is the *minimum* of the width values of the downslope and the upslope. It is important to take the minimum because otherwise a very large downslope with a very small upslope would erroneously appear as a very large valley. For example, see Figure A.5.

With the width of the valley obtained it is then just a matter of setting a threshold on which width sizes will be considered large enough to constitute a genuine valley (and similarly for peaks). In the experimentation performed the most successful technique for setting this threshold was trial-and-error.

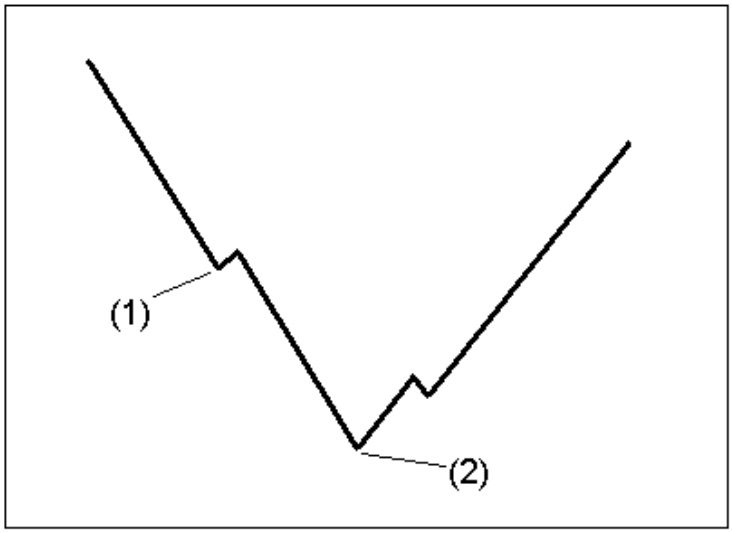

Figure A.5: *An illustration of a large downslope with a small upslope. The minimum indicated by* (1) *is more likely the result of noise than a genuine valley and should be ignored (that is, it is necessary to take the minimum of the upslope and the downslope, rather than the sum or average). The better minimum would be that indicated by* (2).

The final phase in experimentation was to try to take both the height/depth of an extremum *and* its width value into account, reflecting the kind of approach used in other areas of signal processing. It was hoped that this would give an even more accurate account of "true" extrema. The simplest and most successful method of combination was to simply take the product of the valley width and depth (or height). The result of this was that deeper valleys were now considered "better" minima than shallow valleys with similar width, which intuitively seems like a more desirable affect. In the experiments conducted, this approach consistently provided the best overall results.

Section A.4 describes three projects in which the `EC` algorithm has been successfully implemented. Subsection A.4.1 in particular presents a detailed comparison of the accuracy and execution speed of this algorithm versus other approaches such as convolution.

A.4 Successful Applications

It is worth noting at this point that the algorithm presented in Section A.3 is used as a pre-processing filter, the output of which serves as input to another

stage (for example, a signature verification system). The only real way that success of the EC algorithm is measured is by evaluating the success of the resulting application as a whole.

This section discusses three of the application areas in which the EC algorithm has been successfully applied.

A.4.1 Direction Based Handwritten Signature Verification

This was the first project to benefit from the application of the EC algorithm. It involved the design and development of a dynamic handwritten signature verification system [86]. It was an extension of the work in [46] and, at its most basic level, tracked the direction of the pen-tip when performing a signature. Peaks and valleys (maxima and minima) were detected in both the horizontal and vertical directions, ordered and converted into a character string (see [86] and [46] for more detail).

The initial attempt at the valley (minima) detection was to implement a naive gradient-descent algorithm. The gradient-descent approach simply traverses in the direction of decreasing (or equal) value until the point where a greater value is encountered. This previous (lowest) point is then deemed to be the minimum (note that as the surface is one-dimensional there is no choice as to which direction to take when traversing). The problem with this approach was that periodically there was noise introduced into the stream producing "false" extrema. Additionally there was a problem with the "rounding" of the pen-tip location to the nearest pixel in the hardware device which also produced false valleys and peaks (see Figure A.2). The result therefore was that the character string often became somewhat mis-representative of the signature, degrading the efficiency of the system as a whole.

The initial overall error rate (sum of false rejection and false acceptance rates) for the developed system was 6.9%. This was improved dramatically to 2.9% with the implementation of the EC algorithm to better detect the valleys and peaks. Specifically the false rejection rate (the proportion of genuine signatures rejected as forgeries) was improved from 4.3% to 0.9% and the false acceptance rate (forgeries accepted as genuine signatures) was improved from 2.6% to 2.0%.

In other attempts to improve the error rates, two other algorithms were implemented: simple removal of "small" valleys or peaks (called "thresholding")

and basic convolution prior to gradient-descent/hill-climbing. Thresholding simply involved determining the absolute vertical distance (in pixels) between the location of the peak and the location of the preceding valley (that is, the depth of a valley or height of a peak). If this distance was below a specified threshold then that peak was ignored and the traversal continued in the same direction.

Examples of situations in which thresholding was successful can be seen in Figure A.1(b) and Figure A.1(c) and possibly Figure A.1(d). However, the overall performance of this algorithm (in terms of error rate) was quite poor. The reason for this, it seems, is that the *depth* of a valley alone, while obviously containing some information, is not the best indicator of its validity (at least in this environment), but rather the consistency or duration is most important.

Simulated annealing is another option considered useful in avoiding local extrema. Simulated annealing involves the system "jumping ahead" some random distance when an extremum is encountered to try to find a "better" extremum [130]. While this is very effective in some domains, it was envisaged that simulated annealing would not work well in this environment. The reason for this was that there are often long periods in which the height remains the same (plateaus - see Figure A.1(e)) which can vary greatly in their duration. Small jumps ahead will work on many occasions, but not in examples such as this. Large jumps ahead will work in many situations also, but will tend to jump over smaller details. In the interest of experimentation, a modified simulated annealing algorithm from [130] was implemented which jumped forward to try to find better extrema. The number and size of jumps were limited by threshold values, which were optimized through trial-and-error. As expected, the unsuitability of this approach was reflected in the poor error rate of 24.0%.

Convolution is considered as perhaps the most useful method of "smoothing out" or "averaging" one-off "bumps" or random noise while attempting to preserve those extrema which are truly indicative of the pen-tip direction. The basic idea behind convolution is that a window of some finite length (convolution matrices are possible in environments of higher dimensionality) is scanned across the stream of values [164]. The output pixel is the weighted sum or weighted average of the input pixels within the window where the weights can be adjusted to perform various filtering tasks - when smoothing is performed the weights are generally all set to one. After the stream of input was convoluted the hill-climbing/gradient-descent approach was used to obtain the extrema. Convolution can be expressed as the following finite sum where r

Technique	Error Rate
Simple Hill Climbing	6.9%
Thresholding	17.0%
Simulated Annealing	24.0%
Convolution and Hill Climbing	5.3%
EC	2.9%
Convolution and EC	3.4%
EC (width $\times$ height)	2.3%

Table A.1: *Error rates using various methods of overcoming local extrema in a specific signature verification environment. If there are parameters involved in the operation (such as convolution window size) then the parameters which produced the lowest overall error rates were used to generate the results.*

represents the input array, s represents the output array and n is the window size:

$$s[i] = \frac{1}{n} \sum_{j=(i-(n/2))}^{(i+(n/2))} r[j]$$

Multiple attempts were made with convolution using window sizes varying from size one (the trivial case) up to fifty, with the optimal window size (that which resulted in the lowest error rate over the entire signature database) found to be five. Experiments were also conducted with the number of iterations of convolution performed, acknowledging the possibility that the first convolution run didn't smooth out all of the irrelevant extrema and further iterations were necessary. The best overall results were obtained using a single iteration of convolution with the results progressively deteriorating with further iterations, indicating that some of the true extrema were being incorrectly smoothed out.

There was also some experimentation with combining both convolution and the EC algorithm. The stream was firstly convoluted and then the EC algorithm was used to detect the extrema. This proved to be more successful than convolution with hill-climbing but less successful that EC alone. The reason for this is most probably related to convolution smoothing out small but meaningful extrema.

Table A.1 summarises the error rates of the implemented approaches when they are used in the signature verification system presented in [86]. As can be seen the EC algorithm is superior in this environment.

The other advantage of the EC algorithm over convolution is execution speed. In a real time application like signature verification, execution speed can become a serious issue. In order to perform convolution an entire extra layer of computation is required, as convolution of the raw data must be done prior to obtaining the extrema, whereas with the EC algorithm the checking is done at the same time as the search for extrema. Additionally, convolution can become expensive using a large window or with a large raw data size (for example, a typical data size in the application described in Section A.4.3 is over 12,000 entries).

Practical experimentation with the signature verification system has found that convolution causes an average slowdown of 20-25% (depending on system parameters). The number of extra calculations required in the EC algorithm compared to naive hill-climbing is quite small with the slowdown in this application experimentally found to be less than 3%.

A.4.2 Velocity Based Handwritten Word Verification

The EC algorithm was also used in a handwritten password verification system [85]. The first step in this system (as well as many other handwriting based systems like character recognition, for example [151]) was to segment the writing stream into its conceptually significant or constituent parts, commonly known as *strokes*. The strokes are continuous "pen-down" segments of writing bounded by consecutive minima in the pen-tip velocity. The approach then is to extract properties of these strokes and model these properties using, say, a hidden Markov model or neural network.

An approach along these lines was presented previously in [85] and it also made successful use of the EC algorithm. The most naive method of obtaining the velocity minima is a basic gradient-descent algorithm. This was seen as an obvious application area for the EC algorithm and it was implemented immediately. A simple gradient-descent implementation was also performed for comparative purposes.

Using basic gradient-descent as the means of segmentation produced a total error rate of of 1.7% for password verification compared with 0.79% using EC with width only, and 0.64% when using EC with the product of width and height to do the segmentation.

A.4.3 Physiology Research - Tracking Fluctuations in Infant Face Temperature

This physiological research project, initially appearing in [131], was another application which made successful use of the EC algorithm. This project involved examining fluctuating infant facial temperatures and detecting the exact location of temperature maxima. Without going into excessive detail regarding medical aspects of the project, it is theorized that a climax of increasing facial temperature closely correlates with other physiological episodes.

Initially developed software implemented a simple hill climbing approach to determine the location (in time) of the temperature maxima. These times were correlated with the nearest occurrence of a particular physiological episode and the correlation value, or p-value, was 0.072. That is, the relationship was not significant.

Subsequently the EC algorithm (using width and height) was implemented to detect the temperature maxima. The same method of correlation detection was used and the p-value was improved to 0.001 (highly statistically significant). These results would seem to indicate that the EC algorithm is providing a more accurate estimate of the true temperature maxima.

A.5 Future Work

There are two main areas of future work. Firstly, as EC is representable as a finite automaton, a focus will be to formally characterise the class of problems for which EC is most applicable. The other main area of future work is to explore the possibility of adapting the algorithm to multi-dimensional space where it would be interesting to examine its utility in (say) traversing error surfaces for more effective neural network training.

A.6 Conclusion

This appendix has presented a novel local minima and maxima avoidance algorithm for application in specialised domains. This algorithm involves examining the behaviour surrounding a local extremum and deciding whether it represents a true extremum.

The most notable algorithmic difference between this and other classical iterative improvement approaches is that the EC algorithm examines the *width*

or *consistency* of an extremum more so than the actual height or depth. Short-lived, high peaks can be encountered in many typical situations (such as noisy environments or due to hardware inaccuracies) and can cause problems with system accuracy. The `EC` algorithm is far less susceptible to these situations than hill climbing, convolution, thresholding etc. and tends to produce higher quality results.

It has been shown that this algorithm can be applied in various practical iterative improvement situations. Three specific full-scale applications have been discussed in this appendix and a comparison performed between the `EC` algorithm and hill-climbing, convolution, thresholding and simulated annealing. The `EC` algorithm has resulted in notable improvement over the other approaches in all three situations.

Appendix B

Signature Similarity Via Edit Distance

This appendix contains a brief discussion of a simple and efficient similarity measure for handwritten signatures. The primary use of this approach in the application described in this dissertation is as a means of rapidly processing a body of signatures (typically genuine signatures from a signature database) and extracting those that are most similar to a given reference signature. These extracted signatures then act as negative examples (attempted forgeries) for the purpose of system training.

Some of the elements involved in the similarity algorithm can be re-used from other processing already performed (such as signature segmentation), so the approach is quite efficient. The steps involved in the comparison process are as follows:

1. Segment the handwriting into a series of velocity-based "strokes". Most researchers consider conceptually significant segments of a signature to be delimited by consecutive minima in the pen-tip velocity [151]. The only difference in the approach described here is that the detection of velocity minima is performed using the `EC` algorithm described in Appendix A. Figure B.1 illustrates a portion of a signature that has been segmented.

2. Once the handwritten sample is segmented into a series of strokes, each stroke is examined to determine the "net direction". The net direction of a stroke is found by placing the start-point of the stroke at the origin of a cartesian plane and observing the quadrant in which the end-point lies (see Figure B.2). This observation (represented as a single character) is appended to an observation sequence that will later be used in

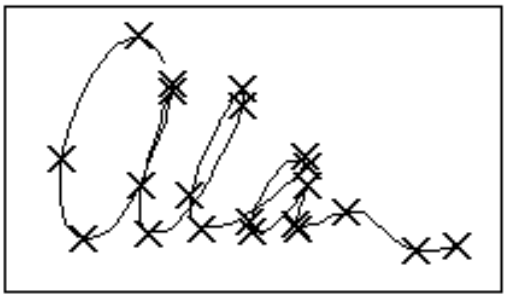

Figure B.1: *A portion of a signature that has been segmented. The crosses on the diagram represent stroke start and/or end points.*

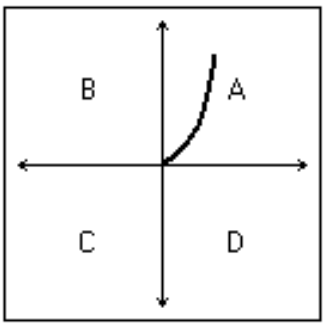

Figure B.2: *The start-point of each stroke is placed at the origin and the quadrant in which the end-point lies is recorded as the observation. The line appearing in the figure in quadrant* A *represents a stroke extracted from a handwritten word.*

the comparison. Additionally, if a pen-up event is encountered while processing the handwriting sample, a pen-up symbol (arbitrarily chosen as the character T) is appended to the sequence.

3. Finally, in an attempt to incorporate the length of time spent in a particular state, the observation symbol is repeated a number of times based on the stroke duration (it is appended to the sequence every twenty-fifth of a second or part thereof). A particularly long stroke finishing in quadrant *A*, for example, would result in a series of A's in the observation sequence.

This approach then is actually capturing the basic shape of the handwriting as well as including some sense of timing and velocity in the observation sequence.

At the end of the process, the observation sequence is a character string

representing the nature of the handwritten signature. Two signatures can therefore be compared by comparing only their character strings, and this can be done using any string edit distance algorithm. Experimentation included implementation of the *Wagner-Fischer* metric [140] as well as the *diff* dynamic programming algorithm [65]. The *diff* algorithm had the advantage in terms of memory requirements, execution speed and resulting error rate so is used as the string distance metric in this similarity measure calculation.

Bibliography

[1] Z. Bahri and B.V.K.V. Kumar. *Generalized Synthetic Discriminant Functions.* Jour. Opt. Soc. Amer. A, Vol. 5, No. 4, pp 562-571, 1998.

[2] J.K. Baker. *The DRAGON System - an Overview.* IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 23, pp 24-29, 1975.

[3] R. Bakis. *Continuous Speech Word Recognition via Centisecond Acoustic States.* Proceedings ASA Meeting, Washington, 1976.

[4] L.E. Baum and T. Petrie. *Statistical Inference for Probabilistic Functions of Finite State Markov Chains.* Ann. Math. Stat. 37, pp 1554-1563, 1966.

[5] L.E. Baum and L.A. Egon. *An Inequality with Applications to Statistical Estimation for Probabilistic Functions of a Markov Process and to a Model for Ecology.* Bulletin of the American Meteorol Soc. 73, pp 360-363, 1967.

[6] L.E. Baum and G.R. Sell. *Growth Functions for Transformations on Manifolds.* Pac. J. Math., 27(2), pp 211-227, 1968.

[7] L.E. Baum, T. Petrie, G. Soules and N. Weiss. *A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains.* Ann. Math. Stat. 41 (1), pp 164-171, 1970.

[8] L.E. Baum. *An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes.* Inequalities 3, pp 1-8, 1972.

[9] L. Bechet. *Method of Comparing a Handwriting with a Reference Writing.* US Patent No. 4, 901, 358, 1990.

[10] Y. Bengio. *Neural Networks for Speech and Sequence Recognition.* International Thomson Computer Press, 1996.

[11] Y. Bengio, Y. Le Cun, C. Nohl and C. Burges. *LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition.* Neural Computation, Vol. 7, No. 5, 1995.

[12] Y. Bengio, R. Mori, G. Flammia and R. Kompe. *Global Optimization of a Neural Network-Hidden Markov Model Hybrid.* IEEE Transactions on Neural Networks, 3(2), pp 252-259, 1992.

[13] C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[14] V. Bouletreau, N. Vincent, R. Sabourin and H. Emptoz. *Handwriting and Signature: One or two Personality Identifiers?* Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia, pp 1758-1760, 1998.

[15] R. Bolle, J. Connell, S. Pankanti, N. Ratha and A. Senior. *Guide to Biometrics.* Springer Verlag, 2003.

[16] J. Brault and R. Plamondon. *Histogram Classifier for Characterization of Handwritten Signature Dynamics.* Proceedings of the 7th International Conference on Pattern Recognition, Montreal, pp 619-622, 1984.

[17] J. Brault and R. Plamondon. *How to Detect Problematic Signers for Automatic Signature Verification.* International Carnahan Conference on Security Technology, pp 127-132, 1989.

[18] J. Brault and R. Plamondon. *A Complexity Measure of Handwritten Curves: Modeling of Dynamic Signature Forgery.* IEEE Transactions on Systems, Man and Cybernetics, Vol. 23, No. 2, pp 400-413, 1993.

[19] J. Cai and Z-Q. Liu. *Intergration of Structural and Statistical Information for Unconstrained Handwritten Numeral Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(3), pp 263-270, 1999.

[20] H. Cardot, M. Revenu, B. Victorri and M. Revillet. *A Static Signature Verification System Based on a Cooperating Neural Networks Architecture.* Series in Machine Perception and Artificial Intelligence, Vol. 13, pp 679-692, 1994.

[21] Cardweb Statistics. *How Much Fraud is There?* http://www.cardweb.com/cardlearn/faqs/2002/may/29.amp, 2002.

[22] H. Chang, J. Wang and H. Suen. *Dynamic Handwritten Chinese Signature Verification.* Proceedings of the Second International Conference on Document Analysis and Recognition, pp 258-261, 1993.

[23] J. Chappelier and A. Grumbach. *A Kohonen Map for Temporal Sequences.* Proceedings of NEURAP '95, France, pp 104-110, 1996.

[24] B. Chen and P. Varshney. *A Bayesian Sampling Approach to Decision Fusion Using Hierarchical Models.* IEEE Transactions on Signal Processing, Vol. 50, No. 8, 2002.

[25] W. Cho, S. Lee and J.H. Kim. *Modeling and Recognition of Cursive Words With Hidden Markov Models.* Pattern Recognition, Vol. 28, No. 12, pp 1941-1953, 1995.

[26] H.D. Crane and J.S. Ostrem. *Automatic Signature Verification Using a Three-axis Force-Sensitive Pen.* IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, No. 3, pp 329-337, 1983.

[27] A.M. Darwish and G.A. Auda. *A New Composite Feature Vector for Arabic Handwritten Signature Verification.* Proceedings of the IEEE International Conference on Acoustics, Vol. 2, pp 613-666, 1994.

[28] J. Daugman. *Biometric Decision Landscapes.* Technical Report No. TR482, University of Cambridge Computer Laboratory, 2000.

[29] G. Dimauro, S. Impedovo and G. Pirlo. *Component-Oriented Algorithms for Signature Verification.* International Journal of Pattern Recognition and Artificial Intelligence, Vol. 8, No. 3, pp 771-793, 1994.

[30] D. Doermann and A. Rosenfeld. *Recovery of Temporal Information from Static Images of Handwriting.* International Journal of Computer Vision, Vol. 15, pp 143-164, 1995.

[31] D. Doermann, V. Varma and A. Rosenfeld. *Instrument Grasp: A Model and its Effects on Handwritten Strokes.* Pattern Recognition, Vol. 27, No. 2, pp 233-245, 1994.

[32] J.G.A. Dolfing. *Handwriting Recognition and Verification - A Hidden Markov Approach.* Ph.D. Thesis, Technical University of Eindhoven, 1998.

[33] J.G.A. Dolfing, E.H.L. Aarts and J.J.G.M. van Oosterhout. *On-Line Signature Verification with Hidden Markov Models.* Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia, pp 1309-1312, 1998.

[34] J.G.A. Dolfing and A. Wendemuth. *Combination of Confidence Measures in Isolated Word Recognition.* Proceedings of the International Conference on Spoken Language Processing, pp 3237-3240, 1998.

[35] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley, New York, 1973.

[36] R. Dugad and U.B. Desai. *A Tutorial on Hidden Markov Models.* Technical Report No: SPANN-96-1. Indian Institute of Technology, India, 1996.

[37] M. Fairhurst and P. Brittan. *An Evaluation of Parallel Strategies for Feature Vector Construction in Automatic Signature Verification Systems.* International Journal of Pattern Recognition and Artificial Intelligence, Vol. 8, No. 3, pp 661-678, 1994.

[38] M. Fairhurst, K. Cowley and E. Sweeney. *Signature Verification Public Trials and Public Survey on Biometrics.* British Technology Group, London, 1994.

[39] R.F. Farag and Y.T. Chien. *On-line Signature Verification.* Proceedings of the International Conference on On-line Interactive Computing, Brunel University, London, p 403, 1972.

[40] L. Fausett. *Fundamentals of Neural Networks.* Prentice Hall Publishing, New York, 1994.

[41] M. Fuentes, S. Garcia-Salicetti and B. Dorizzi. *On line Signature Verification: Fusion of a Hidden Markov Model and a Neural Network via a Support Vector Machine.* Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR '02), 2002.

[42] P. Gader, M. Mohamed and J. Keller. *Fusion of Handwritten Word Classifiers.* Pattern Recognition Letters 17, pp 577-584, 1997.

[43] P. Gallinari, S. Thiria, F. Badran and F. Fogelman-Soulie. *On the Relations between Discriminant Analysis and Multilayer Perceptrons.* Neural Networks, Vol. 4, pp 349-360, 1991.

[44] G. Golub and W. Kahan. *Calculating the Singular Values and Pseudo-Inverse of a Matrix.* SIAM Numerical Analysis, B 2 (2), pp 205-224, 1965.

[45] S. Grossberg. *Adaptive Pattern Classification and Universal Recoding: Parallel Development and Coding of Neural Feature Detectors.* Biological Cybernetics, 23, pp 121-134, 1976.

[46] G. Gupta and R.C. Joyce. *A Study of Some Pen Motion Features in Dynamic Handwritten Signature Verification.* Technical Report, Computer Science Department, James Cook University, 1997.

[47] G. Gupta and A. McCabe. *A Review of Dynamic Handwritten Signature Verification.* Technical Article, Computer Science Department, James Cook University, 1997.

[48] W. Harrison. *Suspect Documents.* Praeger Publishers, New York, 1958.

[49] T. Hastie, E. Kishon, M. Clark and J. Fan. *A Model for Signature Verification.* Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Charlottesville, pp 191-196, 1991.

[50] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan Publishing, New York, 1994.

[51] N.M. Herbst and C.N. Liu. *Automatic Signature Verification Based on Accelerometry.* IBM Journal of Research and Development, pp 245-253, 1977.

[52] O. Hilton. *Scientific Examination of Documents.* Callaghan and Co, Chicago, 1956.

[53] O. Hilton. *Scientific Examination of Questioned Documents.* Revised Edition. New York: Elsevier/North-Holland, 1982.

[54] O. Hilton. *Signatures - Review and a New View.* Journal of Forensic Sciences, JFSCA, Vol. 37, No. 1, pp 125-129, 1992.

[55] J. Hochberg, K. Bowers, M. Cannon and P. Kelly. *Handwritten Document Image Analysis at Los Alamos: Script, Language and Writer Identification.* Symposium on Document Image Understanding Technology (SDIUT), Maryland, 1999.

[56] K. Hornik. *Approximation Capabilities of Multilayer Feedforward Networks.* Neural Networks, Vol. 4, pp 251-257, 1991.

[57] K. Hornik. *Some New Results on Neural Network Approximation.* Neural Networks, Vol. 6, pp 1069-1072, 1993.

[58] X.D. Huang, Y. Ariki and M.A. Jack. *Hidden Markov Models for Speech Recognition.* Edinburgh Information Technology Series, Edinburgh University Press, Edinburgh, 1990.

[59] R.A. Huber and A.M. Headrick. *Handwriting Identification: Facts and Fundamentals.* CRC Press LLC, 1999.

[60] J. Hull. *A Database for Handwritten Text Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 16, pp 550-554, 1994.

[61] S. Jaeger. *Recovering Dynamic Information from Static, Handwritten Word Images.* Ph.D. Thesis, Daimler-Benz Research and Technology, 1998.

[62] A.K. Jain, S. Prabhakar and S. Chen. *Combining Multiple Matchers for a High Security Fingerprint Verification System.* Pattern Recognition Letters, Vol. 20, No. 11-13, pp 1371-1379, 1999.

[63] A.K. Jain and A. Ross. *Learning User-specific Parameters in a Multibiometric System.* Proceedings of the International Conference on Image Processing (ICIP), Rochester, New York, September 22-25, 2002.

[64] R.S. Kashi, J. Hu, W.L. Nelson and W. Turin. *On-line Handwritten Signature Verification using Hidden Markov Model Features.* Proceedings of International Conference on Document Analysis and Recognition, 1997.

[65] B.W. Kernighan and R. Pike. *The UNIX Programming Environment.* Prentice-Hall, New Jersey, 1984.

[66] F. Kimura, S. Inoue, T. Wakabayashi, S. Tsuruoka and Y. Miyake. *Handwritten Numeral Recognition Using Autoassociative Neural Networks.* Faculty of Engineering, Mie University, Kamihama, Japan, 1993.

[67] J. Kittler, M. Hatef, R. Duin and J. Matas. *On Combining Classifiers.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 3, pp 226-239, 1998.

[68] S. Knerr and E. Augustin. *A Neural Network-Hidden Markov Model Hybrid for Cursive Word Recognition.* IEEE Pattern Analysis and Machine Intelligence, pp 1518-1520, 1998.

[69] T. Kohonen. *Self-Organized Formation of Topologically Correct Feature Maps.* Biological Cybernetics, 43, pp 59-69, 1982.

[70] Y. Le Cun. *Generalization and Network Design Strategies.* Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto, 1989.

[71] Y. Le Cun, O. Matan, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel and H. Baird. *Handwritten Zip Code Recognition with Multilayer Networks.* IEEE Proceedings of the International Conference on Pattern Recognition, 1990.

[72] F. Leclerc and R. Plamondon. *Automatic Signature Verification and Writer Identification: The State of the Art - 1989-1993.* Series in Machine Perception and Artificial Intelligence Vol. 13, pp 643-660, 1994.

[73] K. Lee and H. Hon. *Speaker-Independent Phone Recognition Using Hidden Markov Models.* Transactions on Acoustics, Speech and Signal Processing, Vol. 37, No. 11, pp 1641-1648, 1989.

[74] L.L. Lee. *On-Line Systems for Human Signature Verification.* Ph.D. Thesis, Cornell University, 1992.

[75] M. Leshno, V.Y. Lin, A. Pinkus and S. Schocken. *Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate Any Function.* Neural Networks, Vol. 6, pp 861-867.

[76] K. Levenberg. *A Method for the Solution of Certain Non-Linear Problems in Least Squares.* Quarterly Journal of Applied Mathematics II (2), pp 164-168, 1944.

[77] S.E. Levinson, L.R. Rabiner and M.M. Sondhi. *An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition.* Bell Systems Technical Journal 62, pp 1035-1074, 1983.

[78] J.S. Lew. *An Improved Regional Correlation Algorithm for Signature Verification Which Permits Small Speed Changes Between Handwriting Segments.* IBM Journal of Research and Development, Vol. 27, No. 2, 181-185, 1983.

[79] R. Lippmann. *An Introduction to Computing with Neural Nets.* IEEE Acoustics, Speech and Signal Processing Magazine, pp 4-22, 1987.

[80] C.N. Liu, N.M. Herbst and N.J. Anthony. *Automatic Signature Verification: System Description and Field Test Results.* IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-9, No. 1, pp 35-38, 1979.

[81] G. Lorette. *On-line Handwritten Signature Recognition Based on Data Analysis and Clustering.* Proceedings of 7th International Conference on Pattern Recognition, Montreal, pp 1284-1287, 1984.

[82] G. Marcialis, F. Roli and P. Loddo. *Fusion of Multiple Matchers for Fingerprint Verification.* Proceedings of the Workshop on Machine Vision and Perception, Italy, 2002.

[83] D.W. Marquardt. *An Algorithms for Least-Squares Estimation of Non-Linear Parameters.* Journal of the Society of Industrial and Applied Mathematics 11 (2), pp 431-441, 1963.

[84] A.J. Mauceri. *Feasibility Studies of Personal Identification by Signature Verification.* Report No. SID 65 24 RADC TR 65 33, Space and Information System Division, North American Aviation Co., Anaheim, California, 1965.

[85] A. McCabe. *Markov Modelling of Simple Directional Features for Effective and Efficient Handwriting Verification.* R. Mizoguchi and J. Slaney (Eds.): Pacific Rim International Conference of Artificial Intelligence (PRICAI) 2000, LNAI 1886, p 801, 2000.

[86] A. McCabe. *Implementation and Analysis of a Handwritten Signature Verification Technique.* Honours Thesis, James Cook University, 1997.

[87] W. McCulloch and W. Pitts. *A Logical Calculus of the Ideas Immanent in Nervous Activity.* Bulletin of Mathematical Biophysics, Vol. 5, pp 115-133, 1943.

[88] D.A. Mighell, T.S. Wilkinson and J.W. Goodman. *Backpropagation and Its Application to Handwritten Signature Verification.* Advances in Neural

Information Processing Systems 1, Touretzky, D. S. (ed), Morgan Kaufman Publishing, pp 340-347, 1989.

[89] B. Miller. *Vital Signs of Identity.* IEEE Spectrum, pp 22-30, 1994.

[90] M.L. Minsky and S.A. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

[91] M. Mohankrishnan, M.J. Paulik and M. Khalil. *On-Line Signature Verification Using a Nonstationary Autoregressive Model Representation.* 1993 IEEE International Symposium on Circuits and Systems, pp 2303-2306, 1993.

[92] M.E. Munich. *Applications of Hidden Markov Models to Signature Verification.* Sensory Information Processing Laboratory, California Institute of Technology, Pasadena, California, 1998.

[93] *Mutoh: Flying Colors Inside and Out.* http://www.mutoh.com/, 2003.

[94] R.N. Nagel and A. Rosenfeld. *Computer Detection of Freehand Forgeries.* IEEE Transactions on Computers, Vol. C-26, No. 9, pp 895-905, 1977.

[95] W. Nelson and E. Kishon. *Use of Dynamic Features for Signature Verification.* Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Charlottesville, pp 201-205, 1991.

[96] W. Nelson, W. Turin and T. Hastie. *Statistical Methods for On-line Signature Verification.* International Journal of Pattern Recognition and Artificial Intelligence, Vol. 8, No. 3, pp 749-770, 1994.

[97] C. Neukirchen and G. Rigoll. *Advanced Training Methods and New Network Topologies for Hybrid MMI-Connectionist/HMM Speech Recognition Systems.* Proceedings of the IEEE International Conference of Acoustics, Speech and Signal Processing, Munich, pp 3257-3260, 1997.

[98] H. Ney, U. Essen and R. Kneser. *On Structuring Probabilistic Dependencies in Stochastic Language Modelling.* Computer Speech and Language, Vol. 8, pp 1-38, 1994.

[99] A.S. Osborn. *Questioned Documents.* Boyd Printing Co., Albany, NY, 1929.

[100] M. Parizeau and R. Plamondon. *A Comparative Analysis of Regional Correlation, Dynamic Time Warping and Skeletal Tree Matching for Signature Verification.* Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 7, pp 710-717, 1990.

[101] D.B. Parker. *Learning Logic.* Technical Report TG-47, Center for Computational Research in Economics and Management Science, MIT, 1985.

[102] J.R. Parks, D.R. Carr and P.F. Fox. *Apparatus for Signature Verification.* US Patent No. 4, 485, 644, 1985.

[103] M. Pastore. *Credit Card Fraud Losses Top 700 Million.* http://www.fightidentitytheft.com/2002_MAR4_credit_fraud.html, 2002.

[104] G.S. Peake and T.N. Tan. *Script and Language Identification From Document Images.* Proceedings British Machine Vision Conference, Essex, UK, Vol. 2, pp 610-619, 1997.

[105] D.A. Pender. *Neural Networks and Handwritten Signature Verification.* Ph.D. Thesis, Department of Electrical Engineering, Stanford University, 1991.

[106] L. Pessoa. *Multilayer Perceptrons versus Hidden Markov Models: Comparisons and Applications to Image Analysis and Visual Pattern Recognition.* Qualifying Examination Report, Georgia Institute of Technology, 1995.

[107] W. Pitts and W. McCulloch. *How We Known Universals: the Perception of Auditory and Visual Forms.* Bulletin of Mathematical Biophysics, Vol. 9, pp 127-147, 1947.

[108] R. Plamondon. *The Design of an On-Line Signature Verification System: From Theory to Practice.* Series in Machine Perception and Artificial Intelligence, Vol. 13, pp 155-172, 1993.

[109] R. Plamondon. *Looking at Handwriting Generation from a Velocity Control Perspective.* Acta Psychologica, Vol. 82, pp 89-101, 1993.

[110] R. Plamondon and G. Lorette. *Automatic Signature Verification and Writer Identification: The State of the Art.* Pattern Recognition, Vol. 22, No. 2, pp 107-131, 1989.

[111] R. Plamondon and F. Maarse. *An Evaluation of Motor Models of Handwriting.* IEEE Transactions on Systems, Man and Cybernetics 19, pp 1060-1072, 1989.

[112] R. Plamondon and M. Parizeau. *Signature Verification from Position, Velocity and Acceleration Signals: A Comparative Study.* Proceedings of 9th International Conference on Pattern Recognition, Vol. 1, 260-265, 1988.

[113] S. Prabhakar and A.K. Jain. *Decision-Level Fusion in Fingerprint Verification.* Pattern Recognition, Vol. 35, No. 4, pp 861-874, 2002.

[114] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (Second ed.).* Cambridge University Press, 1992.

[115] L. Prevost and M. Milgram. *Automatic Allograph Selection and Multiple Expert Classification for Totally Unconstrained Handwritten Character Recognition.* Proceedings of the 14th International Conference of Pattern Recognition, Brisbane, Australia, Vol. I, pp 381-383, 1998.

[116] Y. Qi and B. Hunt. *Signature Verification using Global and Grid Features.* Pattern Recognition, Vol. 27, No. 12, pp 1621-1629, 1994.

[117] R. Quinlan. *Data Mining Tools See5 and C5.0.* http://www.rulequest.com/see5-info.html, 2002.

[118] L. Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE 77(2), pp 257-285, 1989.

[119] L. Rabiner and B. Juang. *The Segmental K-Means Algorithm for Estimating Parameters of Hidden Markov Models.* IEEE Transactions on Acoustic, Speech and Signal Processing, Vol. 38, No. 9, 1990.

[120] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, New Jersey, 1993.

[121] J.A. Richards. *Remote Sensing Digital Image Analysis.* Springer-Verlag, Germany, 1993.

[122] G. Rigoll. and A. Kosmala. *A Systematic Comparison between On-line and Off-line Methods for Signature Verification.* Proceedings of the International Conference on Pattern Recognition, Vol. II, pp 1755-1757, 1998.

[123] G. Rigoll, A. Kosmala, J. Rottland and C. Neukirchen. *A Comparison between Continuous and Discrete Density Hidden Markov Models for Cursive Handwriting Recognition.* Proceedings IEEE International Conference of Pattern Recognition, Vienna, Vol. II, pp 205-209, 1996.

[124] G. Rigoll, A. Kosmala and D. Willett. *A New Hybrid Approach to Large Vocabulary Cursive Handwriting Recognition.* Proceedings of the International Conference on Pattern Recognition, Vol. II, pp 1512-1514, 1998.

[125] F. Rosenblatt. *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.* Psychological Review, Vol. 65, pp 386-408, 1958.

[126] F. Rosenblatt. *On the Convergence of Reinforcement Procedures in Simple Perceptrons.* Report VG-1196-G-4, Cornell Aeronautical Laboratory, New York, 1960.

[127] A. Ross, A.K. Jain and J. Qian. *Information Fusion in Biometrics.* Proceedings Audio- and Video-Based Biometric Person Authentication '01, Sweden, pp 354-359, 2001.

[128] A. Ross and A.K. Jain. *Information Fusion in Biometrics.* Pattern Recognition Letters, Vol. 24, Issue 13, pp 2115-2125, 2003.

[129] D.E. Rumelhart, J.L. McClelland and R.J. Williams. *Learning Internal Representations by Error Propagation.* In David E. Rumelhart and James L. McClelland (Eds.): Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. 1, pp 318-362, 1986.

[130] S. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach.* Prentice Hall, New Jersey, 1995.

[131] M.J. Russell and R. Vink. *Increased Facial Temperature as an Early Warning in Sudden Infant Death Syndrome.* Accepted for publication in Medical Hypotheses.

[132] R. Sabourin, R. Plamondon and L. Beumier. *Structural Interpretation of Handwritten Signature Images.* International Journal of Pattern Recognition and Artificial Intelligence, Vol. 8, No. 3, Singapore, pp 709-748, 1994.

[133] H.E.S. Said, K.D. Baker and T.N. Tan. *Personal Identification Based on Handwriting.* Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia, pp 1761-1764, 1998.

[134] L. Schomaker. *The UNIPEN Project.* http://hwr.nici.kun.nl/unipen/, 1997.

[135] L. Schomaker, G. Abbink and S. Selen. *Writer and Writer-Style Classification in the Recognition of Online Handwriting.* Proceedings of the European Workshop on Handwriting Analysis and Recognition: A European Perspective, 1994.

[136] O.G. Selfridge. *Pandemonium: A Paradigm For Learning.* Proceedings of the Mechanisation of Thought Processes Symposium, London: HMSO, pp 513-529, 1958.

[137] A.J. Shepherd. *Second-Order Methods for Neural Networks.* Springer Publishing, New York, 1997.

[138] R.L. Sherman. *Biometric Futures.* Computers & Security, Vol. 11, pp 128-133, 1992.

[139] K. Steinbuch and U.A.W. Piske. *Learning Matrices and Their Applications.* IEEE Transactions on Electronic Computers, pp 846-862, 1963.

[140] G.A. Stephen. *String Searching Algorithms.* World Scientific Publishing, Singapore, 1994.

[141] J. Sternberg. *Automated Signature Verification Using Handwriting Pressure.* WESCON Technical Papers, No. 31/4, Los Angeles, 1975.

[142] C. Stoll. *Stalking the Wily hacker.* Communications of the ACM, pp 484-497, 1988.

[143] J. Subrahmonia. *Similarity Measures for Writer Clustering.* Proceedings of the International Workshop on Frontiers in Handwriting Recognition, 2000.

[144] K.M.C. Tan and A.B. Ruighaver. *A Practical Intrusion Detection System for UNIX.* Technical Article, The University of Melbourne, Department of Computer Science. Parkville 3052, Australia, 1994.

[145] S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press Inc., New York, 1999.

[146] S.R. Veltman and R. Prasad. *Hidden Markov Models Applied to On-line Handwritten Isolated Character Recognition.* IEEE Transactions on Image Processing, Vol. 3, No. 3, pp 314-318, 1994.

[147] A. Verikas, A. Lipnickas, K. Malmqvist, M. Bacauskiene and A. Gelzinis. *Soft Combination of Neural Classifiers: A Comparative Study.* Pattern Recognition Letters, Vol. 20, pp 429-444, 1999.

[148] B. Verma, P. Gader and W. Chen. *Fusion of Multiple Handwritten Word Recognition Techniques.* Pattern Recognition Letters, Vol. 22, pp 991-998, 2001.

[149] A.J. Viterbi. *Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm.* IEEE Transactions on Information Theory, IT-13, pp 260-269, 1967.

[150] L. Vuurpijl. *NICI Handwriting Recognition Group Home Page* http://hwr.nici.kun.nl/, 2003.

[151] L. Vuurpijl and L. Schomaker. *Coarse Writing-Style Clustering Based on Simple Stroke-Related Features.* Progress in Handwriting Recognition, Colchester, 1996.

[152] A. Webb. *Statistical Pattern Recognition.* Oxford University Press Inc., 1999.

[153] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Ph.D. Thesis, Harvard University, 1974.

[154] T. Wessels and C.W. Omlin. *Refining Hidden Markov Models with Recurrent Neural Networks.* South African Telecommunications Networks and Applications Conference (SATNAC), 1999.

[155] T. Wessels and C.W. Omlin. *A Hybrid System for Signature Verification.* South African Telecommunications Networks and Applications Conference (SATNAC), 1999.

[156] B. Widrow and M.E. Hoff. *Adaptive Switching Networks.* 1960 IRE WESCON Convention Record, pp 96-104, 1960.

[157] G. Wilensky, R. Crawford and R. Riley. *Recognition and Characterization of Handwritten Words.* Proceedings of the 1997 Symposium on Document Image Understanding Technology, pp 87-98, 1997.

[158] T.S. Wilkinson. *Novel Techniques for Handwritten Signature Verification.* Ph.D. Thesis, Department of Electrical Engineering, Stanford University, 1990.

[159] D. Willshaw, O. Buneman and H. Longuet-Higgins. *Non-Holographic Associative Memory.* Nature, 222, pp 960-962, 1969.

[160] S. Xu and M. Zhang. *An Adaptive Activation Function for Higher Order Neural Networks.* Proceedings 15th Australian Joint Conference on Artificial Intelligence, Canberra, pp 356-362, 2002.

[161] L. Yang, B. Widjaja and R. Prasad. *Applications of Hidden Markov Models for Signature Verification.* Pattern Recognition, Vol. 28, No. 2, pp 161-170, 1995.

[162] M. Yasuhara and M. Oka. *Signature Verification Experiment Based on Nonlinear Time Alignment: A Feasibility Study.* IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-7, No. 3, pp 212-216, 1977.

[163] H.S. Yoon, J.Y. Lee and H.S. Yang. *An On-line Signature Verification System Using Hidden Markov Models In Polar Space.* Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR '02), 2002.

[164] I.T. Young, J.J. Gerbrands and L.J. van Vliet. *Fundamentals of Image Processing - Convolution-based Operations.* http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Convolut-2.html, 1998.

[165] D. Yuk and J. Flanagan. *Telephone Speech Recognition using Neural Networks and Hidden Markov Models.* IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 1, pp 157-160, 1999.

[166] K.P. Zimmermann and M.J. Varady. *Handwriter Identification from One-bit Quantized Pressure Patterns.* Pattern Recognition, Vol. 18, No. 1, pp 63-72, 1985.

[167] E. Zois and V. Anastassopoulos. *Methods for Writer Identification.* Proceedings of the Third IEEE International Conference on Electronics, Circuits and Systems (ICECS), 1996.

[168] E. Zois and V. Anastassopoulos. *Decision Fusion for Writer Discrimination.* Proceedings of the Thirteenth International Conference on Digital Signal Processing, 1997.

[169] E. Zois and V. Anastassopoulos. *Fusion of Correlated Decisions for Writer Identification.* Journal of Pattern Recognition, 32, pp 1821-1823, 1999.