Dynamic Character Recognition Using Hidden Markov Models.[1]

M. S. Ryan* and G. R. Nudd.

Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England.

*e-mail address - (msr@dcs.warwick.ac.uk)

19th May 1993.

Abstract.

Hidden Markov Model theory is an extension of the Markov Model process. It has found uses in such areas as speech recognition, target tracking and word recognition. One area which has received little in the way of research interest, is the use of Hidden Markov Models in character recognition. In this paper the application of Hidden Markov Model theory to dynamic character recognition is investigated. The basic Hidden Markov Model theory is reviewed, and so are the algorithms associated with it. A quick overview of the dynamic character recognition process is considered. Then three types of describing characters are considered, position, inclination angle of small vectors and stroke directional encoding using a Freeman code. A system using each of these descriptions, using Hidden Markov Models in the comparison stage, is described. It is recognised that experiments using the different encoding systems have to be carried out to check the validity of this chosen method.

1. Introduction.

It is the purpose of this paper to investigate a form of pattern recognition known as character recognition, more specifically that of dynamic character recognition using Hidden Markov Models (HMMs) during the comparison stage of recognition. Dynamic character recognition uses information about the characters written by a person to identify which characters have been written. The information generated by the handwriting process is captured by the computer at real-time, via the use of a digitising tablet, and is usually presented to the recognition system in the form of a time ordered sequence of x and y co-ordinates. This method of character recognition is also referred to as on-line character recognition, and has the advantage over static character recognition of not only containing static information such as position, but also dynamic information such as the order in which the character was written and dynamic features such as velocity and time taken to write the character.

HMMs can be viewed of being made up of two statistical processes, one which is not seen i.e. the underlying process, which can only be seen through another statistical process that produces the signals seen emitted by the underlying model. The HMM theory can be used in estimation problems like target tracking, or recognition problems such

1. Research Report No.244.

as character recognition, which is the subject of this paper. Unlike the area of speech recognition, where HMMs have found extensive uses [1,2], there has been little work carried out on the use of HMMs in character recognition. Some work has been carried out in the area of static character recognition such as that carried out by Jeng *et al* [3], Tanaka *et al* [4], and Vlontzos and Kung [5], with very good results e.g. 91.5% to 97% recognition accuracy for multi-font Chinese characters in the case of Jeng *et al*, and 97% to 99% accuracy for the recognition of multi-font English characters in the case of Vlontzos and Kung for their two level character recognition system.

The area of dynamic character recognition using HMMs does not seem to have been considered in the literature. Nag *et al* [6] have carried out work in the area of dynamic cursive word recognition, using quantised inclination angles of small vectors to describe the script making up the word. Cameillerapp *et al* [7] work concerns the recognition of on-line cursive handwriting using fully connected models of a full word, and stroke directions as features. The reason for choosing to model a word instead of individual characters is so that segmentation does not have to be carried out on the word, which is particularly difficult when the characters run into each other. But this leaves the problem of needing a separate model for each word to be recognised.

The motivation behind the use of HMMs in character recognition is the great variation of handwriting between different writers, and even in the same writer over a long time. Other reasons for the choosing of HMMs in the recognition of dynamic characters is that only the number of states, and the constraints to be imposed on the model have to be decided upon. Any rules about the nature of the characters can be established by example during the training phase of each HMM. HMMs also remove the need of explicit time alignment associated with the use of other techniques such as dynamic programming, or even Markov Models such as those used by Farag to model hand-written cursive words, [8].

In this paper, the concept of HMMs and various algorithms and implementation issues surrounding HMMs is examined in Section 2. Then a short description of the various processes involved in dynamic character recognition is considered in Section 3. In Section 4 the application of HMM theory to dynamic character recognition is considered, using three different types of character description.

2. Hidden Markov Models.

A HMM can be represented by a Finite State Machine, which in turn can be represented by either a connected graph or a special form of connected graph called a trellis. Each node in this graph represents a state, where the signal being modelled has a distinct set of properties, and each edge a possible transition between two states at consecutive discrete time intervals. An example of a trellis and graph of a 4 state fully connected HMM is shown in Figure 1.

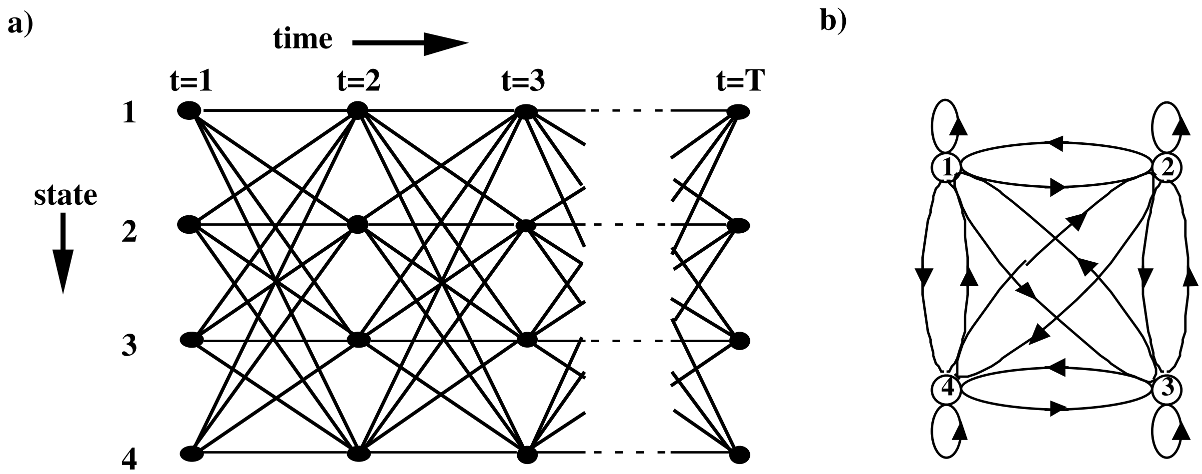


Figure 1. Showing **a)** Trellis Diagram and **b)** corresponding graph of 4 state HMM.

The actual structure of the Finite State Machine to be modelled by the HMM is hidden from view, therefore the structure of the HMM is usually a general one to take into account all the possible occurrences of the states, with the HMM's states and transitions being estimates of the true underlying Finite State Machine. Associated with each transition and observation symbol for a particular HMM are a set of metrics, which are used to obtain a measure of likelihood that the observations seen were produced by the model represented by the HMM. Another type of Finite State Machine that is similar in nature to a HMM is a Markov Model. Markov Models have found extensive use in communications, [9], where the underlying process is known, i.e. all the possible states and transitions are known for the signal to be modelled. Only the observation process is statistical in nature, due to the noise introduced into the observation, though what observation is produced by which state is also known. In this case only the Viterbi Algorithm, see section 2.2, is needed to decode the output.

Before looking at the various aspects of HMMs and the algorithms associated with them some notation associated with HMMs has to be defined as follows :-

t - The discrete time index.

N - Total number of states in the FSM.

M - Total number of possible observation symbols.

- The ith state of the FSM.

x_i

- The observation sequence generated by the process to be

$\boldsymbol{O} = O_1, O_2, \ldots, O_T$

modelled. Each is a symbol from the discrete set V of

O_t

possible observation symbols.

- The set of the M different possible observation symbols,

$$V = \{v_1, v_2, \ldots, v_M\}$$

usually an optimal set, where each symbol is discrete in nature. A single member of this set is represented by the symbol .

$$v_k$$

- The state sequence for the corresponding observation sequence.

$$I = i_1, i_2, \ldots, i_T$$

- The survivor path which terminates at time t, in the ith state of the FSM.

$$sp_t(i)$$

It consists of an ordered list of 's visited by this path from time t = 1 to

$$x_i$$

time t.

T - overall length of the observation sequence.

- Initial state metric for the ith state at t = 0. Defined as the probability that

$$\pi_i$$

the ith state is the most likely starting start, i.e. .

$$Prob(x_i \; at \; t = 0)$$

- The transition metric for the transition from state at time t - 1 to the

$$a_{ij}$$

$$x_i$$

state at time t. Defined as the probability that given that state occurs at

$$x_j$$

$$x_i$$

time t - 1, the state will occur at time t, i.e. ,

$$x_j$$

$$Prob(x_j \; at \; t \,|\, x_i \; at \; t - 1)$$

known as the Markovian property. These metrics can also change with time.

- The observation metric at time t, for state . Defined as the probability

$$b_j(k)$$

$$x_j$$

that the observation symbol would occur at time t, given that we are in

v_k

the state at time t, i.e. . This metric is fixed for the

x_j

$$Prob(v_k \ at\ t | x_j \ at\ t)$$

state regardless of where the state was entered.

- The survivor path metric of . This is defined as the Product of the

$\Gamma_\tau(\iota)$

$sp_t(i)$

metrics (, and) for each transition in the ith survivor path,

π_ι

a_{ij}

$b_j(k)$

from time t = 0 to time t.

- compact notation used to represent the model itself, where A is

$\lambda = (A, B, \Pi)$

the 2 dimensional matrix of the 's, B is the 2 dimensional

a_{ij}

matrix of the 's, and Π is the 1 dimensional matrix of the

$b_j(k)$

π_ι

for the given model.

Before looking at the problems and algorithms associated with HMMs, it should be pointed out that the observation symbols considered throughout this paper are discrete. Though HMMs can be adapted to handle continuous symbols for the observations quite easily, by handling the probability densities as continuous. This adaptation is beyond

$b_j(k)$

the scope of this paper, and has been investigated by other authors such as Rabiner in [1].

To allow a HMM to be used in real world applications, 3 problems have to be solved associated with the model process. These are :-

1. How to compute the likelihood of the observation sequence ,

$$\Pr ob(O|\lambda)$$

given the model and observation sequence.
2. How to find the optimal underlying state sequence given the observations i.e. finding the structure of the model.

3. How to train the model i.e. estimate A, B, Π to maximise .

$$\Pr ob(O|\lambda)$$

Having established these as the problems that have to be dealt with in some manner, which look at some solutions to these three problems.

2.1 The Forward-Backward Procedure.

The solution to problem 1 above is based upon the forward-backward procedure. With the forward procedure the forward probabilities for the model are calculated. This probability can be defined as the joint probability that the partial observation sequence and

$$(O_1, K\ , O_t)$$

a state at t of occurs given the model λ, i.e. . This can be computed recursively by the

$$x_i$$

$$\alpha_\tau(\iota) = \Pi\rho o\beta(O_1 L\ O_t, i_t = \xi_\iota|\lambda)$$

following algorithm :-

Initialisation :-

(1a)

$$\alpha_1(\iota) = \pi_\iota \beta_\iota(O_1) \qquad \forall\iota\ \omega\eta\varepsilon\rho\varepsilon\ 1 \le \iota \le N.$$

Recursion :-

(1b)

For $t = 1,2,K\ , T - 1$ and $\forall\varphi\omega\eta\varepsilon\rho\varepsilon 1 \le \varphi \le N.$

$$\alpha_{\tau+1}(\varphi) = \left[\sum_{\iota=1}^{N} \alpha_\tau(\iota)\alpha_{\iota\varphi}\right]\beta_\varphi(O_{\tau+1})$$

Termination :-

(1c)

$$\Pr ob(O|\lambda) = \sum_{\iota=1}^{N} \alpha_T(\iota)$$

This can be used to score the model, but is also used during the re-estimation stage to obtain partial observation probabilities for states at each time interval.

There is also a need to define a backward procedure to calculate the backward probabilities for the model, but these are only used during the re-estimation stage, and is not part of the solution to problem 1. A backward probability can be defined as, which is

$$\beta_t(i) = Prob(O_{t+1}, O_{t+2}, \ldots, O_T | i_t = x_i, \lambda)$$

the probability of the partial observation sequence from time t+1 to T, given the present state is x_i at t, and given the model, λ. This can be calculated recursively by the following algorithm :-

Initialisation :-

$$\beta_T(i) = 1 \qquad \forall i \text{ where } 1 \le i \le N. \tag{2a}$$

Recursion :-

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad \forall i \text{ where } 1 \le i \le N. \tag{2b}$$

So it can be seen that both the forward and backward probabilities are defined for all t, from 1 to T. The $\alpha_t(i)$'s and $\beta_t(i)$'s produced by the above algorithms rapidly approach zero numerically and this can cause arithmetic under flow, so a re-scaling technique is required. A re-scaling technique is derived for $\alpha_t(i)$'s in [1], and a similar re-scaling algorithm can be defined for the $\beta_t(i)$'s. This technique requires all $\alpha_t(i)$'s at a particular t to be re-scaled if one of the $\alpha_t(i)$'s falls below a minimum threshold, so that under flow is prevented. The re-scaling algorithm for the $\beta_t(i)$'s at a particular t is given by equation 3 :-

$$\hat{\beta}_t(i) = \frac{\beta_t(i)}{\sum_{i=1}^{N} \beta_t(i)} \tag{3}$$

This re-scaling algorithm can be used at each iteration of the Backward procedure or only when one of the $\beta_t(i)$'s at a particular time is in danger of causing arithmetic under flow, a minimum threshold measure is used to judge when this is going to occur.

2.2 The Viterbi Algorithm.

The solution to problem 2 is based on the Viterbi Algorithm [10,11]. The Viterbi Algorithm finds the joint probability of the observation sequence and the specific state sequence, I, occurring given the model, λ. most likely state sequence for the observation sequence given, , i.e. it finds the most likely state sequence. It is also usual to take the

$$\Pr ob(O, I|\lambda)$$

natural logarithms of the A, B, and Π parameters, so that arithmetic under flow is prevented in the Viterbi Algorithm during calculations. It should be noted that the Viterbi Algorithm is similar to the Forward Procedure in implementation, though the Forward Procedure sums over t and the Viterbi Algorithm uses a maximisation, and it also provides the most likely state sequence. The algorithm can be calculated as follows :-

Initialisation :-

(4a)

$$\forall \iota\ \omega\eta\epsilon\rho\epsilon 1 \le \iota \le \mathrm{N}.$$
$$\Gamma_1(\iota) = \lambda\nu\pi_\iota + \lambda\nu\beta_\varphi(\mathrm{O}_1)$$

(4b)

$$sp_1(i) = [\xi_\iota]$$

Recursion :-

(4c)

$$\forall \tau\omega\eta\epsilon\rho\epsilon 2 \le \tau \le \mathrm{T},\ \forall \varphi\omega\eta\epsilon\rho\epsilon 1 \le \varphi \le \mathrm{N}.$$
$$\Gamma_\tau(\varphi) = \mu\alpha\xi_{1\le\iota\le\mathrm{N}}[\Gamma_{\tau-1}(\iota) + \lambda\nu\alpha_{\iota\varphi}] + \lambda\nu\beta_\varphi(\mathrm{O}_\tau)$$

(4d)

$$sp_t(j) = \mathrm{A}\pi\pi\epsilon\nu\delta[\xi_\varphi, \sigma\pi_{\tau-1}(\iota)]$$
$$\sigma\upsilon\chi\eta\ \tau\eta\alpha\tau\ \Gamma_{\tau-1}(\iota) + \lambda\nu\alpha_{\iota\varphi} + \lambda\nu\beta_\varphi(\mathrm{O}_\tau) = \Gamma_\tau(\varphi)$$

where Append in equation (4c), takes the state and adds it to the head of the list of

x_j

the survivor path , whose transition had the maximum metric at t for the transition to

$sp_t(i)$

state .

$$x_j$$

Decision :-

(4e)

$$\Gamma_T = \mu\alpha\xi_{1\le\iota\le N}[\Gamma_T(\iota)]$$

(4f)

$$sp_T = \sigma\pi_T(\iota)\ \sigma\upsilon\chi\eta\ \tau\eta\alpha\tau\ \Gamma_T(\iota) = \Gamma_T$$

In English the Viterbi Algorithm looks at each state at time t, and for all the transitions that lead into that state, it decides which of them was the most likely to occur, i.e. the transition with the greatest metric. If two or more transitions are found to be maximum, i.e. their metrics are the same, then one of the transitions is chosen randomly as the most likely transition. This greatest metric is then assigned to the state's survivor path metric, . The Viterbi Algorithm then discards the other transitions into that state, and appends

$$\Gamma_\tau(\iota)$$

this state to the survivor path of the state at t - 1, from where the transition originated. This then becomes the survivor path of the state being examined at time t. The same operation is carried out on all the states at time t, at which point the Viterbi Algorithm moves onto the states at t + 1 and carries out the same operations on the states there. When we reach time t = T (the truncation length), the Viterbi Algorithm determines the survivor paths as before and it also has to make a decision on which of these survivor paths is the most likely one. This is carried out by determining the survivor with the greatest metric, again if more than one survivor is the greatest, then the most likely path followed is chosen randomly. The Viterbi Algorithm then outputs this survivor path, ,

$$sp_T$$

along with it's survivor metric, .

$$\Gamma_T$$

2.3 The Baum-Welch Re-estimation Algorithms.

The solution to problem 3 is based on a training method known as the Baum-Welch Re-estimation Algorithms, [12]. This algorithm finds a local optimal probability for ,

$$Prob(O|\lambda)$$

in this case a maximum. The re-estimation algorithms for the 3 model parameters, Π, A and B can be defined in English as follows, [1] :-

(5a)

$$\bar{\pi}_\iota = \varepsilon\xi\pi\varepsilon\chi\tau\varepsilon\delta\ \nu\upsilon\mu\beta\varepsilon\rho\ o\phi\ \tau\iota\mu\varepsilon\sigma\ \iota\nu\ \alpha\tau\ \tau = 1$$

(5b)

$$\bar{a}_{ij} = \frac{\text{εξπεχτεδ νυμβερ οφ τρανσιτιονσ φρομ } ξ_ι \text{ το } ξ_φ}{\text{εξπεχτεδ νυμβερ οφ τρανσιτιονσ φρομ } ξ_ι}$$

(5c)

$$\bar{b}_j(k) = \frac{\text{εξπεχτεδ νυμβερ οφ τιμεσ ιν } ξ_φ \text{ ανδ οβσερϖινγ } ϖ_κ}{\text{εξπεχτεδ νυμβερ οφ τιμεσ ιν } ξ_φ}$$

where , and are the re-estimated parameters of the previous model parameters , and .

$\bar{\pi}_\iota$

$\bar{a}_{ij}$

$\bar{b}_j(k)$

π_ι

a_{ij}

$b_j(k)$

From the above definitions the following equations can be derived (see Appendix for the derivation of equations (6a), (6b) and (6c)) :-

(6a)

$$\bar{\pi}_\iota = \sum_{\varphi=1}^{N} \left(\frac{\alpha_1(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_2)\beta_2(\varphi)}{\sum_{\nu=1}^{N}\sum_{\mu=1}^{N}\alpha_1(\nu)\alpha_{\nu\mu}\beta_\mu(O_2)\beta_2(\mu)} \right)$$

(6b)

$$\bar{a}_{ij} = \sum_{\tau=1}^{T-1} \left(\frac{\alpha_\tau(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_{\tau+1})\beta_{\tau+1}(\varphi)}{\sum_{\varphi=1}^{N}\alpha_\tau(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_{\tau+1})\beta_{\tau+1}(\varphi)} \right)$$

(6c)

$$\bar{b}_j(k) = \frac{\sum_{\substack{\tau=1 \\ \sigma\tau.\ o_\tau = \varpi_k}}^{T-1} \left(\frac{\alpha_\tau(\varphi)\beta_\tau(\varphi)}{\sum_{\iota=1}^{N} \alpha_\tau(\iota)\beta_\tau(\iota)} \right)}{\sum_{\tau=1}^{T} \left(\frac{\alpha_\tau(\varphi)\beta_\tau(\varphi)}{\sum_{\iota=1}^{N} \alpha_\tau(\iota)\beta_\tau(\iota)} \right)}$$

The notation represents the re-estimated model. The training of the model is carried

$$\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\Pi})$$

out using a training set, consisting of observation sequences appropriate for that model. The training sequence can be summarised by the following diagram, Figure 2.

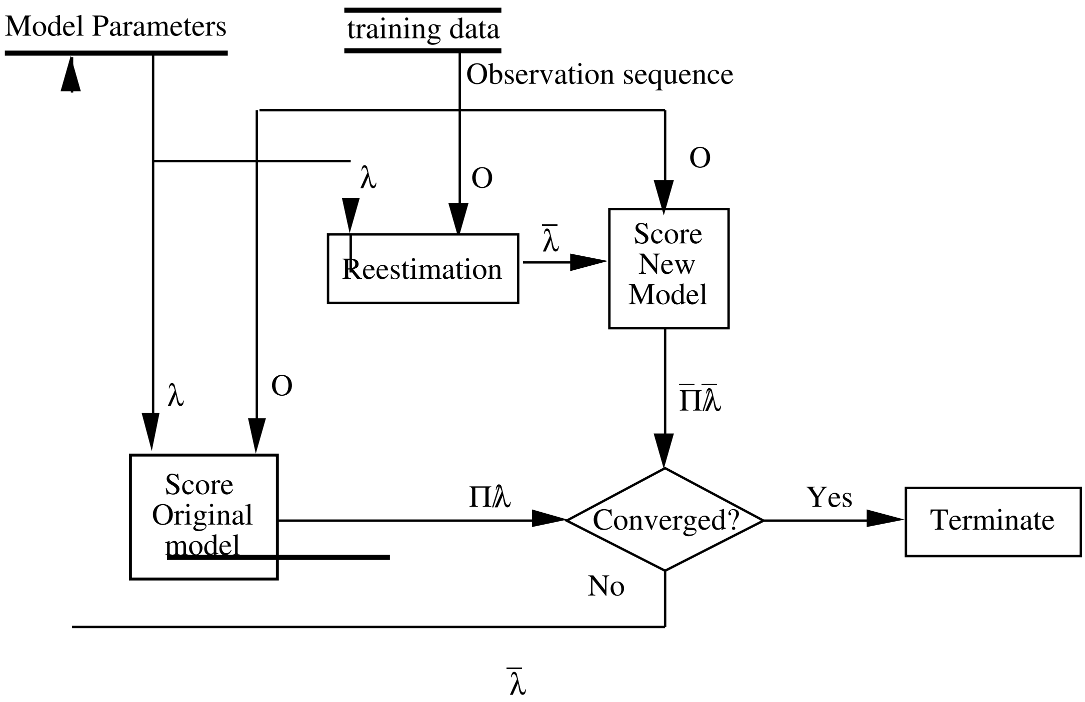


Figure 2. Showing training procedure for a HMM.

The scoring of the model depends on whether the Viterbi Algorithm or the Forward

Procedure is to be used during recognition. If the Viterbi Algorithm is used then ,

$$P = \Pi\rho\omega\beta(O, I|\lambda)$$

whereas if the Forward procedure is used then . has the same meaning as P but de-

$$P = \Pi\rho\omega\beta(O|\lambda)$$

$$\bar{P}$$

scribes the probability for the re-estimated model, . For each observation sequence in

$$\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\Pi})$$

the training set, the likelihood of the present model λ, has to be compared with the new re-estimated model . If then the present model λ is replaced by , meaning the new mod-

$$\bar{\lambda}$$

$$\bar{P} > \Pi$$

$$\bar{\lambda}$$

el is more likely to have produced the observation sequence than the previous model. The training procedure is then repeated on the same observation sequence, with , to see

$$\lambda = \bar{\lambda}$$

if an even better model can be found. Convergence is reached when , or after a set num-

$$P = \bar{\Pi}$$

ber of re-estimation steps. Once convergence has been achieved for the present observation sequence, then the new model is trained on the next observation sequence in the training set, until there are no more observation sequences left in the training set. This training method is carried out on all the HMMs required for a system, using their own specific training data.

During each re-estimation stage, the Baum-Welch algorithms keep the statistical consistency needed when dealing with probability distributions i.e.

(7a)

$$\sum_{\iota=1}^{N} \bar{\pi}_{\iota} = 1$$

(7b)

$$\sum_{\varphi=1}^{N} \bar{a}_{ij} = 1, \quad \phi o\rho\ \epsilon\alpha\chi\eta\omega\eta\epsilon\rho\epsilon 1 \leq \iota \leq N.$$

(7c)

$$\sum_{\kappa=1}^{M} \bar{b}_{j}(k) = 1, \quad \phi o\rho\ \epsilon\alpha\chi\varphi\omega\eta\epsilon\rho\epsilon 1 \leq \varphi \leq N.$$

The initial estimates of the probability densities for the model have to be established in some manner. According to Rabiner, [1], this can be achieved in a number of ways.

A finite training set has to be used to train HMMs, usually due to the infinite variations that are encountered for a particular models. This means that special consideration has to be given to transitions or observation symbols that do not occur in the training set for a particular model. This affects the a_{ij}'s and $b_j(k)$'s parameters, but the affect on the π_ι's occurs anyway due to the fact that only one π_ι will occur for each of the observation sequences considered in the training set. If for example a particular observation does not occur in the training set then the $b_j(k)$ value for that particular observation occurring will be set to zero. It may be that this particular observation will never occur for that particular model, but due to the finite nature of the training set there can be no assurance that it will not occur in the recognition stage. There are a few ways to deal with this problem according to Rabiner [1]. One is increasing the size of the training set, increasing the chance that all possible transitions and observation symbol occurrences associated with the model are covered. However this methods usually impractical both computationally and in obtaining the observation sequences for the training set. The model itself can be changed by reducing the number of states or number of observation symbols used, but the main disadvantage with this approach is that the model used has been chosen due to practical reasons.

Another method, and possibly the most appealing, is to add thresholds to the transition and observation metrics, so that an individual value cannot fall below a minimum value. This method is defined by Levinson *et al* in [13] for the $b_j(k)$'s, but it is redefined here for clarity for the a_{ij}'s. If, for example, we set up a threshold for the a_{ij}'s as

$$a_{ij} \geq \varepsilon \tag{8}$$

then any a_{ij} for a particular i falls below this threshold then it is reset to the value ε. this is done for all a_{ij} for a particular i. But, there is a need to re-scale the other a_{ij}'s which have not fallen below ε so that the statistical consistency laid down by the condition given in equation (7b), can be kept. To do this re-scaling the following equation has to be used :-

(9)

$$\tilde{a}_{ij} = (1-\lambda\varepsilon)\frac{\alpha_{t\varphi}}{\sum \alpha_{t\varphi}, \forall\varphi \notin \{\forall\varphi | \alpha_{t\varphi} < \varepsilon\}}, \quad \forall\varphi \notin \{\forall\varphi | \alpha_{t\varphi} < \varepsilon\}$$

where is the minimum threshold set, is the re-scaled , l is the number of 's in the set ,

ε

$\tilde{a}_{ij}$

a_{ij}

a_{ij}

$\{\forall\varphi | \alpha_{t\varphi} < \varepsilon\}$

which in turn is the set of all the 's which fell below for a particular i. This process is

a_{ij}

ε

continued until all the 's for a particular i met the constraint. The re-scaling algorithm

a_{ij}

for the 's is similar. This algorithm can be used after each step of the re-estimation pro-

$b_j(k)$

cess, or after the whole re-estimation process for a particular observation sequence is completed.

It can now be seen that in most cases all three algorithms, i.e. the Forward-Backward Procedure, the Viterbi Algorithm and the Baum-Welch re-estimation Algorithms, are used during the training stage. Once training is complete, then the models can be used to recognise the objects that it represents. During recognition, the Viterbi Algorithm or the Forward Procedure are used to score the unknown observation sequence against the model/s to be considered. The forward Procedure can be dropped in favour of the Viterbi Algorithm, since in a majority of applications either the most likely state sequence has to be known, such as in target tracking [14] or word recognition [15], or the maximum of is also the most significant term in , [16].

$\Pr ob(O, I|\lambda)$

$\Pr ob(O|\lambda)$

2.4 Types of HMM.

An important aspect of implementing HMM theory in real world applications, along with such matters as the number of states to use and the observation symbols to use, is the type of HMM to use. The two most commonly used types of HMMs, are the

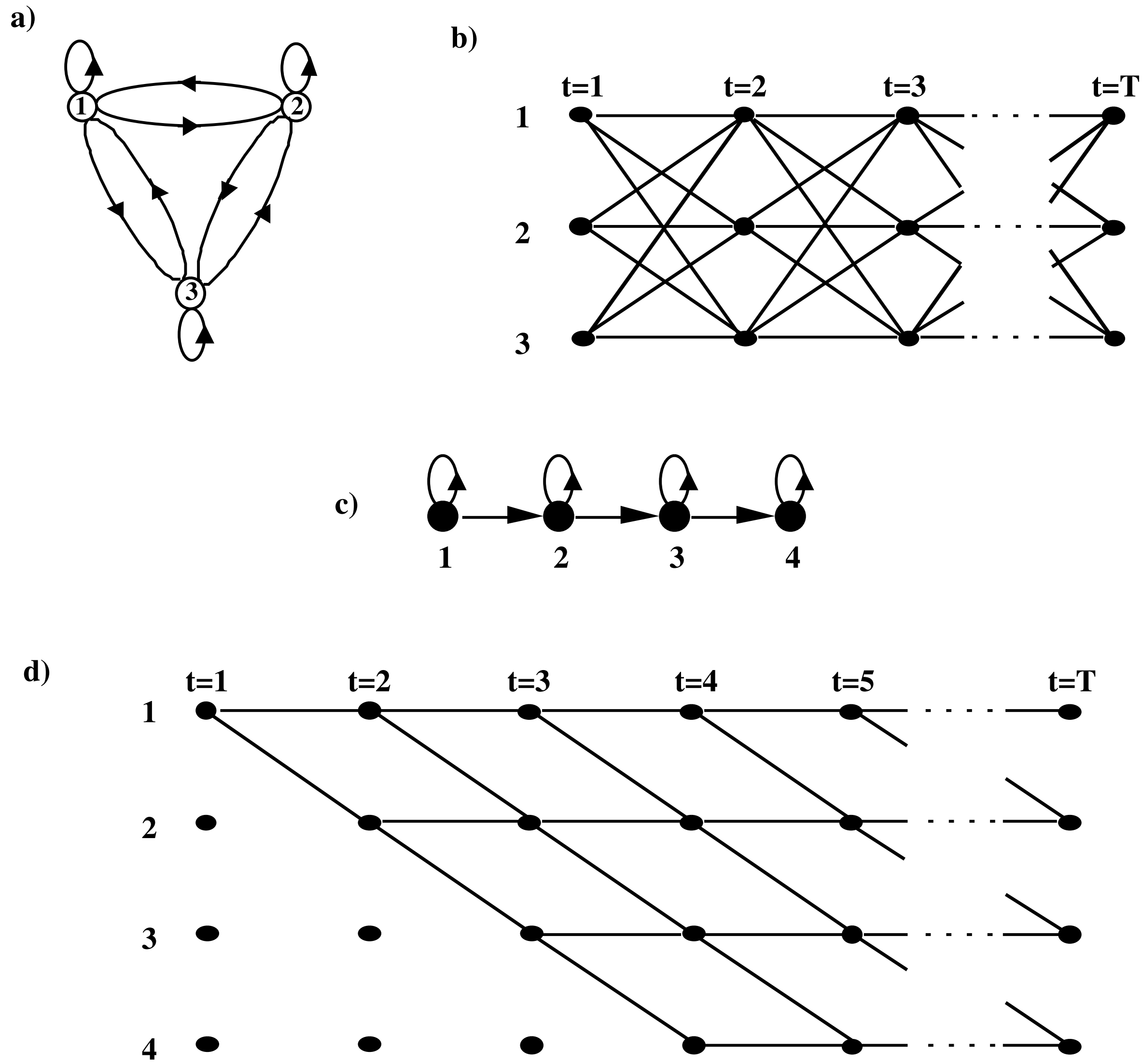


Figure 3. Showing 3 state fully connected HMM as **a)** a fully connected graph, **b)** a fully connected trellis, and **c)** a left-to-right model for a 2-transition 4 state model and **d)** it's corresponding trellis diagram.

ergodic type models and the left-to-right models. The ergodic models are the type of model already considered in this paper, which are fully connected models, i.e. each state can be accessed at the next time interval from every other state in the model. An example of a 3 state ergodic model is shown in Figure 3.a in the form of a connected graph, and with the corresponding trellis diagram in Figure 3.b. The most important thing to note about the ergodic models are that the 's are all positive.

$$a_{ij}$$

The other main type of HMM, the left-to-right model is used to model signals that change properties over time, e.g. speech. This type of HMM has the unique property that a transition can only occur to a state with the same or higher state index i, as time increases, i.e. the state sequence proceeds from left to right. This means that the 's have

$$a_{ij}$$

the following property :-

$$a_{ij} = 0, \quad \iota\phi\varphi < \iota. \tag{10}$$

and that the initial state is only entered if it begins in state 1, i.e.:-

$$\pi_{\iota} = \begin{cases} 0, & \iota\phi\iota \neq 1. \\ 1, & \iota\phi\iota = 1. \end{cases} \tag{11}$$

it also means that the last state N has the following property

$$\begin{aligned} &a_{NN} = 1, \\ &\alpha_{N\iota} = 0, \quad \iota\phi\iota \neq \mathrm{N}. \end{aligned} \tag{12}$$

since the state sequence cannot go back along the model. An example of a 2 transition, 4 state left-to-right model is shown in Figure 3.c.

Even with these constraints placed on this type of model, the same algorithms are used without any modifications. So if a state becomes zero ant time during the training then it stays at zero from then on. The above two types of HMMs are the two most used HMMs, there are many variations on the theme, and the decision to use which model should be based on the application and the signals to be modelled.

For a more detailed analysis of the issues connected with HMMs discussed in this section and other issues, the reader is pointed to the work of Rabiner, [16] and [1].

3. Dynamic Character Recognition.

Before describing the system using HMMs in the next section, a brief look at the various stages associated with dynamic character recognition is taken. Since character recognition is a form of pattern recognition, it shares many of the same processes as other forms of recognition within this broad base of problems. These are show in Figure 4 below.

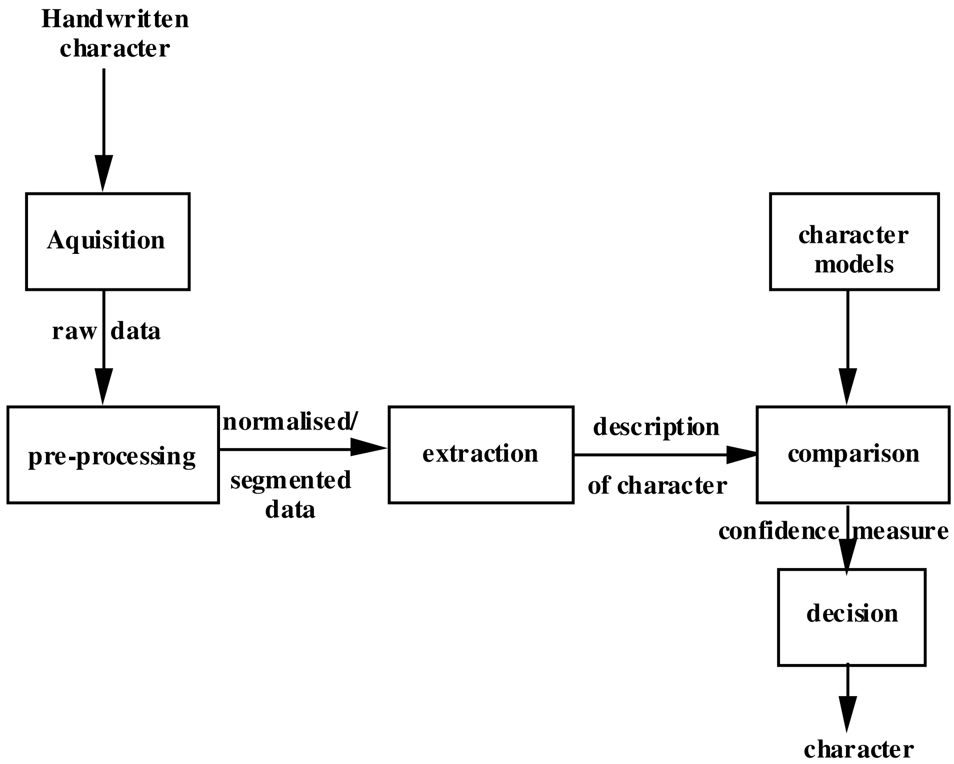


Figure 4. Showing processing stages of Character Recognition.

Acquisition in dynamic character recognition is usually carried out by a digitising tablet. This supplies raw digital data in the form of and co-ordinate data of the pen posi-

$$\boldsymbol{x}(\boldsymbol{t})$$

$$y(t)$$

tion on the paper. This usually comes with pen-down/pen-up information, a single bit, to indicate whether the pen is in contact with the paper or not. Some special pens have been developed to extract specific dynamic features such as force, [17], but have found better use in such areas as signature verification [18].

Pre-processing is the process where raw data is taken and various processing steps are carried out to make data acceptable to use in the extraction stage. This stage reduces the amount of information needed, removes some of the noisy data, and normalises it to certain criteria. Segmentation of the characters is the first process carried out, so that characters are presented to the comparison stage isolated from each other. The individual character is then smoothed and filtered to remove noise from such sources as the digitiser, and human induced noise such as pauses during the writing process. It also removes redundant and useless information such as isolated points, which could cause misclassification. The final stage of pre-processing to be carried out on the character is that of normalisation. The character is normalised according to size, rotation and location depending upon the description of characters that is used for comparison. Some or all of these stages are carried out depending upon features used in the comparison stage.

Once pre-processing has been carried out on the data, the next stage carried out on the data is a extraction stage. In this stage a meaningful description of the character to be recognised has to be extracted from the data provided by the pre-processing stage. It should be said that this part of the character recognition process is the most important, though what makes a good description of a character is not really known. Some typical descriptions are based on static features such as the presence of a dot, or how many semi-circles are contained within the character, e.t.c. Others include transform co-efficients such as those generated by Fourier transforming characters. other descriptions can be based on dynamic characteristic such as positional order, direction of the writing to the horizontal or force exerted by the writer on the paper.

The last two stages are the comparison and decision stages, where the description of the unknown character is compared to models of the different known characters. The comparison stage can sometimes be split into two stages. The first stage, if implemented, reduces the number of possible character models that the unknown character has to be compared against. This is achieved by using such methods as dynamic programming or using such criteria as the presence of a dot or no dot. The next stage is the comparison of the unknown character description against some or all the models of known characters. This stage can produce a confidence measure, based on a minimal, statistical or Euclidean distance measure, of how close the fit is between the unknown character and each of the character models in the systems. The decision stage then decides which model more closely resembles the unknown character and outputs the character with the best measure as the final decision. Decision trees can also be used during the comparison-decision stage, particularly if shapes make up the description of the unknown character.

A more detailed analysis of the different stages and techniques used in dynamic character recognition can be found in [19] and [20].

4. Use of HMMs in Character Recognition.

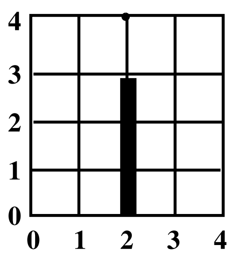
In this section a dynamic character recognition system using HMMs in the comparison stage is investigated. For purposes of this first paper only the areas of extraction, comparison and decision will be considered initially, it will be assumed that a character has been acquired and pre-processed to the requirements of the system, though some normalisation and quantisation techniques are presented which are specific to a particular descriptive method.

In this system, the character just written by a person is represented by a sequence of co-ordinates in the x and y directions, an example of a character 'i' that has been captured dynamic by using a digitising tablet is shown in Figure 5.

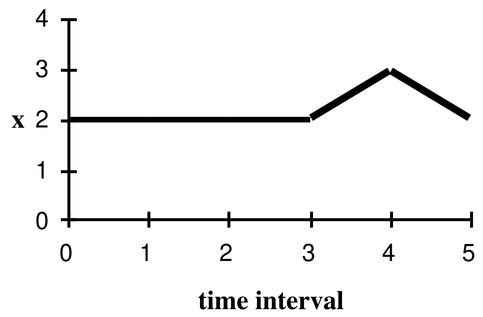
a)

$$\begin{aligned}(x,y,penup)_t = \ & (2.0,3.0,0)_0,\\ & (2.0,2.0,0)_1,\\ & (2.0,1.0,0)_2,\\ & (2.0,0.0,0)_3,\\ & (3.0,2.0,1)_4,\\ & (2.0,4.0,0)_5.\end{aligned}$$

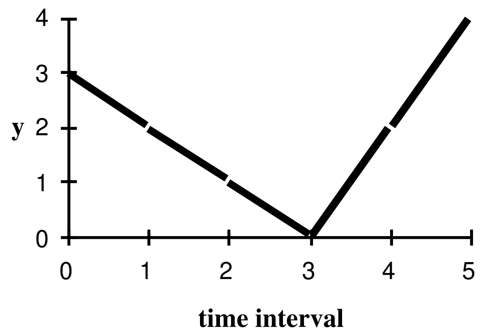
b)



c)



d)



e)

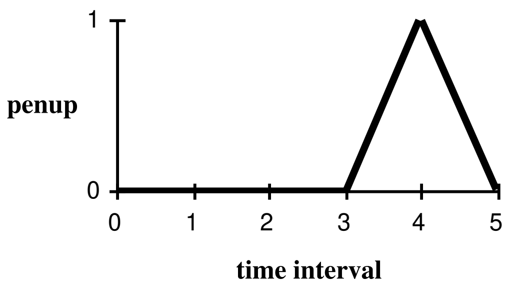


Figure 5. Showing **a)** data sequence obtained from digitiser for character 'i',
b) x-y plot of character 'i' for pendown positions only,
c) x-t plot, **d)** y-t plot and **e)** penup-t plot for the data sequence of the 'i'.

As can be seen in Figure 5, the character 'i' can be represented by four distinct variables, notably the x, y, penup and the time interval data. The majority of English characters are written with one continuous stroke, as opposed to the two that are usually associated with characters such as 'i' and 'x'. It is initially proposed to look at the recognition of the 26 lower case letters of the English alphabet, but the system should be easily adapted so that capital letters and digits can be recognised as well.

4.1 And Position Description Method.

$$\mathbf{x}(\mathbf{t})$$
$$\mathbf{y}(\mathbf{t})$$

In this method the co-ordinates obtained from the digitiser are used as a description of each character. For each character to be used either in training or recognition, two observation sequences are generated for the HMM comparison stage. These two observation sequences are which represents the normalised x co-ordinates observed up to

$$O^x = (O_1^{\xi}, K , O_T^x)$$

time , and which represents the corresponding normalised y co-ordinates for the same

$$t = \mathrm{T}$$

$$O^y = (O_1^{\psi}, K , O_T^y)$$

time interval as that for , i.e. T in both observation sequences is the same. Each obser-

$$O^x$$

vation sequence contains points representing the position of the pen at that time, and also contains the character p to represent the penup code if the character contains more than one stroke, as in the case of a 't' or 'x'. Let the points that occur between penups be referred to as a segment. Note that the penup symbol will appear at the same time interval in both observation sequences. An example of the coding of the letter 'i' as shown in Figure 5 above, using this method is:-

$$O^x = (0.0, 0.0, 0.0, 0.0, \pi, 0.0).$$
$$\mathrm{O}^{\psi} = (0.8, 0.6, 0.2, 0.0, \pi, 1.0).$$

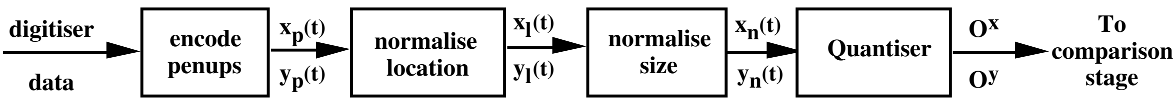


Figure 6. Showing the stages in the positional data method.

Figure 6 shows the various stages involved in constructing the observation sequences for the comparison stage, for this method. The pre-processing stage for this method consists of three stages, the first being to find penups in the digitiser data, the second being the normalisation of the character's location, and the last stage is normalising the character for size. The first stage of encoding penups extracts the penup data at each point in the sequence obtained from the digitiser, and sets the and to the symbol 'p' at

$$\boldsymbol{x}(\boldsymbol{t})$$
$$y(t)$$

a particular time interval, if the penup bit is set to 1, i.e. a penup has occurred at this time interval, equation 13 where is the length in time of the digitiser data sequences

$$T_d$$

$$x(t)$$

and .

$$y(t)$$

(13)

$$\forall\tau \quad \omega\eta\epsilon\rho\epsilon 1 \le \tau \le \mathrm{T}_\delta,$$
$$\iota\phi(\ \pi\epsilon\nu\upsilon\pi_\tau \ne 0)\ \tau\eta\epsilon\nu$$
$$\xi_\pi(\tau) = \pi$$
$$\psi_\pi(\tau) = \pi$$
$$\epsilon\nu\delta\iota\phi$$

This stage produces the and sequences to be used in the normalisation stage. The

$$x_p(t)$$

$$y_p(t)$$

character 'i' that has been considered as an example would have the following sequences as output of the encode penups stage :-

$$x_p(t) = (2.0, 2.0, 2.0, 2.0,\ \pi, 2.0).$$
$$\psi_\pi(\tau) = (3.0, 2.0, 1.0, 0.0,\ \pi, 4.0).$$

The first stage of the normalisation process, normalising for location, finds the minimum and values over whole of the encoded penup sequences, and , this includes extra

$$x(t)$$

$$y(t)$$

$$x_p(t)$$

$$y_p(t)$$

segments for that character, e.g. a character x is made up of two segments and some penup values. It should be noted that penup values are ignored when finding the minimum value in either direction. To find the minimum point in the x direction the following algorithm can be used, equation (14a).

(14a)

$$\min\_x = \infty;$$

$$\forall \tau\ \omega\eta\varepsilon\rho\varepsilon 1 \le \tau \le \mathrm{T}_{\delta},$$
$$\mathrm{I}\phi \xi_{\pi}(\tau) \ne \pi\ \tau\eta\varepsilon\nu$$
$$\quad \mathrm{I}\phi \xi_{\pi}(\tau) < \mu\iota\nu\_\xi\ \tau\eta\varepsilon\nu$$
$$\qquad \mu\iota\nu\_\xi = \xi_{\pi}(\tau)$$

(14b)

$$\forall \tau \omega\eta\varepsilon\rho\varepsilon 1 \le \tau \le \mathrm{T}_{\delta},$$
$$\mathrm{I}\phi \xi_{\pi}(\tau) \ne \pi\ \tau\eta\varepsilon\nu$$
$$\quad \xi_{\lambda}(\tau) = \xi_{\pi}(\tau) - \mu\iota\nu\_\xi$$
$$\varepsilon\lambda\sigma\varepsilon$$
$$\quad \xi_{\lambda}(\tau) = \xi_{\pi}(\tau)$$

Once the minimum point has been found then the minimum value in the x direction, , is subtracted from all the values in the observation sequence, except for penup points,

$$\min\_x$$
$$x_p(t)$$

using equation (14b). This process is also carried out on the values using found by us-

$$y_p(t)$$
$$\min\_y$$

ing equation (14a) replacing x by y, and then carrying out the subtraction stage using equation (14b) as for the x values previous. This process produces two sequences of data and which are now passed onto the next normalisation stage. An example of the

$$x_l(t)$$
$$y_l(t)$$

encoded sequence produced by the normalise location process for the character 'i' in Figure 5 is :-

$$x_l(t) = (0.0, 0.0, 0.0, 0.0, \pi, 0.0).$$
$$\psi_{\lambda}(\tau) = (3.0, 2.0, 1.0, 0.0, \pi, 4.0).$$

After normalising for location, each character is normalised for size by normalising each point in both location normalised sequences, and , to a value between 0 and 1.

$x_l(t)$

$y_l(t)$

This is done by finding the maximum value in each of the input sequences and then dividing each point in by , and the same for the sequence. This size normalisation pro-

$x_l(t)$

$\max\_x$

$y_l(t)$

cess can be carried out for the data sequence by using the following equations, equation

$x_l(t)$

(15a) to find and equation (15b) to calculate the normalised component .

$\max\_x$

$x_n(t)$

(15a)

$$
\begin{array}{l}
\max\_x = 0; \\
\\
\forall\tau \ \omega\eta\epsilon\rho\epsilon 1 \le \tau \le \mathrm{T}_\delta \\
\mathrm{I}\phi \xi_\lambda(\tau) \ne \pi \ \tau\eta\epsilon\nu \\
\quad \mathrm{I}\phi \xi_\lambda(\tau) > \mu\alpha\xi\_\ \xi \ \tau\eta\epsilon\nu \\
\quad\quad \mu\alpha\xi\_\ \xi = \xi_\lambda(\tau)
\end{array}
$$

(15b)

$$
\begin{array}{l}
\forall\tau\omega\eta\epsilon\rho\epsilon 1 \le \tau \le \mathrm{T}_\delta, \\
\mathrm{I}\phi(\xi_\lambda(\tau) \ne 0) \ \omicron\rho(\xi_\lambda(\tau) \ne \pi) \ \tau\eta\epsilon\nu \\
\quad \xi_\nu(\tau) = \dfrac{\xi_\lambda(\tau)}{\mu\alpha\xi\_\ \xi} \\
\epsilon\lambda\sigma\epsilon \\
\quad \xi_\nu(\tau) = \xi_\lambda(\tau)
\end{array}
$$

This final stage of pre-processing produces two normalised data sequences, and , and

$x_n(t)$

$y_n(t)$

an example of these sequences for the character 'i' shown in Figure 5 is:-

$$x_n(t) = (0.0, 0.0, 0.0, 0.0, \pi, 0.0).$$
$$\psi_v(\tau) = (0.75, 0.5, 0.25, 0.0, \pi, 1.0).$$

After this stage a quantisation stage is carried out which quantises the normalised position data, and , to a set of discrete evenly spaced points. This process also determines

$$x_n(t)$$
$$y_n(t)$$

the number of observation symbols to be considered by the HMM for each letter. To carry out the quantisation process on the data from the normalisation process, a gate size has to be determined for both the x and y data sequences. This value can be the same or different for both gate sizes, depending upon how many observation symbols the HMMs for the x and y data sequences of a character have.

Let the gate sizes of the x and y data sequence quantisers be and respectively. It will

$$g_x$$
$$g_y$$

be initially assumed that , and so the gate size for both the x and y data sequences can

$$g_x = \gamma_\psi$$

be represented by . Only this gate size , or and , has to be specified for the quantiser,

$$g$$
$$g$$
$$g_x$$
$$g_y$$

all other factors associated with the quantiser can be calculated as follows from this gate size. From , the number of different possible quantisation levels, R, can be calculated

$$g$$

using equation (16).

(16)

$$R = \frac{1}{\gamma} + 1$$

where the addition of the 1 is needed to allow the inclusion of the 0.0 quantisation level. The set can be defined as the set of possible quantisation level values where is a spe-

$$Q$$
$$Q = \{\theta_1, K, q_r, K, q_R\}$$
$$q_r$$

cific quantisation level value and can be calculated by the following equation, equation (17).

(17)

$$q_r = \gamma \times (\rho - 1)$$

The and observation sequences can be constructed from the corresponding and data

O^x

O^y

$x_n(t)$

$y_n(t)$

sequences by the following quantisation equation, equation (18).

(18)

$$\forall\tau \;\; \omega\eta\epsilon\rho\epsilon 1 \le \tau \le \mathrm{T}_\delta,$$
$$\iota\phi(\xi_\nu(\tau) \ne \pi)\;\tau\eta\epsilon\nu$$
$$O_\tau^\xi = \theta_\rho \;\iota\phi\left(\theta_\rho - \frac{\gamma}{2}\right) \le \xi_\nu(\tau) \le \left(\theta_\rho + \frac{\gamma}{2}\right)\;\alpha\nu\delta\,\iota\phi\theta_\rho \in \Theta$$
$$\epsilon\lambda\sigma\epsilon$$
$$O_\tau^\xi = \xi_\nu(\tau)$$

where and can be substituted with their y counterparts to obtain the quantisation equa-

O_t^x

$x_n(t)$

tion for the y data sequence. These quantised values of the pendown and values, along

$x_n(t)$

$y_n(t)$

with the penup points make up the observation sequences, and , which will be used by

O^x

O^y

the HMMs in the comparison and decision stage to determine which character was written, see section 4.4. The number of observation symbols, M, generated by this method can be calculated from the following equation, equation (19), where the extra symbol p is generated by the encode penup stage.

(19)

$$M = \mathrm{P} + 1$$

If is considered for the gate size of the quantiser for the example character 'i', 6 quan-

$g = 0.2$

tisation levels are obtained for both the x and y quantisers. The resulting observation sequences obtained from this quantising stage will be :-

$$O^x = (0.0, 0.0, 0.0, 0.0, \pi, 0.0).$$

$$O^{\psi} = (0.8, 0.6, 0.2, 0.0, \pi, 1.0).$$

This will also mean that there are 7 observation symbols for each model to be considered in the comparison stage, i.e. the 6 quantisation levels of 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0, and the symbol p for penup.

4.2 Inclination Angle of Vectors Description Method.

This form of describing the character read in from the digitiser, is based on the ideas suggested by Nag et al, [6]. They describe a system for cursive word recognition, i.e. each HMM represents the whole word but each word is described as a sequence of quantised inclination angles, with special codes to represent sharp turning points. In our system using this descriptive method, the quantised inclination angles will make up the majority of the observation sequence for the comparison stage, but special codes for penups and single dots will also make up some of the set of possible observations. The special code used by Nag in his work, i.e. the sharp turning point code, will be implemented in our system later to see if better recognition results are obtained.

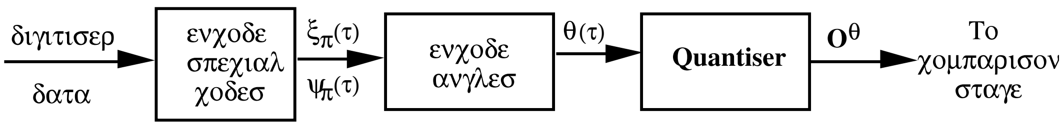


Figure 7. Showing the stages in the inclination angle method.

Figure 7 above, shows the various stages involved in constructing the observation sequence for this method. It will become obvious to the reader that there are no normalisation stages required in this method since the inclination angle of a small vector, that is used as the main observation feature, is size and location independent for any handwritten character.

The first stage of encoding the digitiser data for a character involves encoding penups into the and data sequences, as was done for the positional method, see section 4.2.

$x(t)$

$y(t)$

The other special code introduced at this encode special codes stage, is that for a dot, as is found in the characters 'i' and 'j'. This is needed particularly in the case of a character i to distinguish it from an l. A dot can be defined as a point where a penup occurs at the previous and the next time interval in the digitiser data sequence. If the present point is the last in the digitiser data sequence and the previous one was a penup, then the present point can be classified as a dot. This will be the most commonly written form of a dot since in the majority of cases the dot in a character i or j is written last. To represent this special case a d is inserted at the point in the and data where the dot oc-

$x(t)$

$y(t)$

curred, by using the following algorithm, equation (20).

$$(20)$$

$$
\begin{aligned}
&\forall\tau\ \ \omega\eta\varepsilon\rho\varepsilon 1 \le \tau \le \mathrm{T}_{\delta},\\
&\iota\phi(\pi\varepsilon\nu\upsilon\pi_{\tau-1} \ne 0)\ \alpha\nu\delta((\tau = \mathrm{T}_{\delta})\ o\rho(\pi\varepsilon\nu\upsilon\pi_{\tau+1} \ne 0))\ \tau\eta\varepsilon\nu\\
&\quad \xi_{\pi}(\tau) = \delta\\
&\quad \psi_{\pi}(\tau) = \delta\\
&\varepsilon\nu\delta\iota\phi
\end{aligned}
$$

For the example character 'i' considered in the last section, the output from this encode special characters stage would be :-

$$x_p(t) = (2.0, 2.0, 2.0, 2.0, \pi, \delta).$$

$$\psi_{\pi}(\tau) = (3.0, 2.0, 1.0, 0.0, \pi, \delta).$$

The resulting sequences, and , are then feed into the next stage which encodes the

$x_p(t)$

$y_p(t)$

inclination angles.

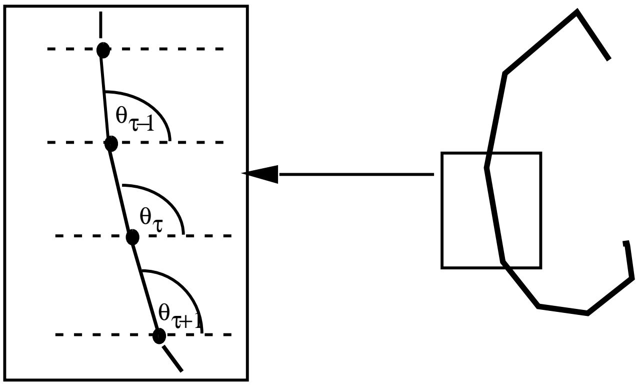


Figure 8. Showing Inclination angle for a sequence of small vectors of a letter c.

In the next stage, encode angles, the and data sequences are encoded into one data

$x_p(t)$

$y_p(t)$

sequence - the sequence of inclination angles of consecutive points within each seg-

$\theta(\tau)$

ment that makes up the character. Let represent an inclination angle of the vector be-

θ_τ

tween the points at t and at t+1, Figure 8. The inclination angle for each set of consecutive points can be found by the following equation.

(21)

$$\theta_\tau = \tau\alpha\nu^{-1}\left(\frac{\Delta\psi_\pi}{\Delta\xi_\pi}\right) = \tau\alpha\nu^{-1}\left(\frac{\psi_\pi(\tau+1) - \psi_\pi(\tau)}{\xi_\pi(\tau+1) - \xi_\pi(\tau)}\right)$$

where and are the changes in the x and y directions between consecutive points respec-

$\Delta\xi_\pi$

$\Delta\psi_\pi$

tively. So the output of this stage for the example character 'i' will be :-

$$\theta(\tau) = (270, 270, 270, \pi, \delta).$$

where the angles are measured in degrees.

Once the angles have been worked out then they are passed onto the quantiser, as the data sequence. The quantiser stage quantises the angles in this data sequence to a set

$\theta(\tau)$

number, , of possible angles. As for the quantiser in the previous method, section 4.1,

R_θ

only the gate size of the quantiser in degrees has to be specified. only one quantiser is needed in this method and it's gate size can be represented by . The number of possible

g_θ

angles, , can be calculated by adapting equation (16) as in equation (22).

R_θ

(22)

$$R_\theta = \frac{360}{\gamma_\theta}$$

Note that no additional 1 has to be added to equation (22) as in equation (16), this is because for this purpose, therefore no additional quantisation level has to be included

$360^\circ = 0^\circ$

for . As for the quantiser in section 4.1, the set represents the set of possible quantisa-

$$0^{\circ}$$

$$Q = \{\theta_1, K\ , q_r, K\ , q_R\}$$

tion level values and can be calculated as in equation (17) in section 4.1. The observa-

$$q_r$$

$$O^{\theta}$$

tion sequence which is passed onto the comparison stage can be constructed from the corresponding data sequences by using the same quantisation equation, equation (21),

$$\theta(\tau)$$

as was used for the position method by substituting the values defined here into the equation. The number of observation symbols, M, generated by this encoding method can be found using equation (23).

$$M = \mathrm{P}_{\theta} + 2 \tag{23}$$

Where the additional 2 symbols generated are the special codes for dots and penups. An example of the final observation sequence generated by this method for the example character 'i', where , thus giving the number of different possible observation symbols

$$g_{\theta} = 5$$

of 74, is:-

$$O^{\theta} = (270, 270, 270, \pi, \delta).$$

4.3 Directional Encoding Method.

This method is based upon the work by Farag on cursive word recognition, [8]. Farag did not use HMM but Markov Models to represent each word that had to be recognised by the system, using time dependent transition matrices. This caused problems when there was not one-to-one correspondence between the word to be recognised and the model of the word, which would need to be solved by the use of dynamic programming to allow non-linear time warping of each word to fit it to the same time length as the model. In Farag's method each word was described by a direction code of each vector connecting consecutive points that make up a word. The directional encoding scheme only allowed 8 possible directions each vector could have, as is shown in Figure 9.

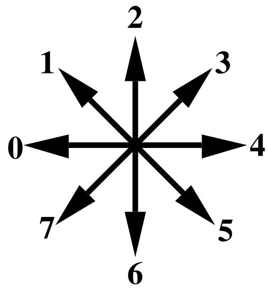


Figure 9. Showing directional coding scheme.

The directional encoding scheme is sometimes referred to as a Freeman chain code in the literature, and it can be argued that this method is the same as Nag's inclination angle method. Figure 10 shows the various stages involved in constructing the observation sequence, , for this method when adapted to dynamic character recognition.

O^d

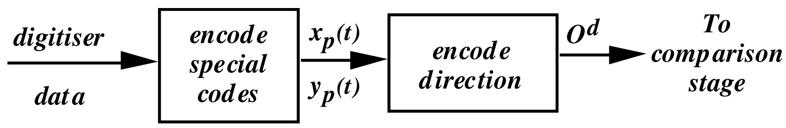


Figure 10. Showing the stages in the directional encoding method.

The first stage of encoding the digitiser data is the same as the previous method, i.e. encoding the special codes for penups and dots, and the data sequences and will be the

$x_p(t)$

$y_p(t)$

same for both methods. The encode direction stage encodes the direction of writing for consecutive points at t and t+1 to one of eight possible directions. This can be done by using the equation given in [8], as reproduced here, equation (24).

(24)

$$\frac{\Delta\psi}{\Delta\xi} = \iota \quad \iota\phi\left(\frac{\pi\iota}{4} - \frac{\pi}{8}\right) \le \tau\alpha\nu^{-1}\left(\frac{\Delta\psi}{\Delta\xi}\right) < \left(\frac{\pi\iota}{4} + \frac{\pi}{8}\right) \alpha\nu\delta\iota\phi\iota \in \phi$$

where and are defined as for the previous method in section 4.2, and is the set of pos-

$\Delta\xi$

$\Delta\psi$

f

sible directions, in this case . Note that instead of the separate stages of calculating the

$f = \{0,1,2,3,4,5,6,7\}$

inclination angle and then quantising these angles, in this method these are carried out in the same process. This process is carried out on all the pairs of consecutive points within a segment, for all segments in the and data sequences which are separated by

$$x_p(t)$$

$$y_p(t)$$

penups and do not include dots. The output of this process is the observation sequence which like the previous methods is passed onto the comparison stage. Using the char-

$$O^d = \{O_1^\delta, O_2^\delta, \ldots, O_T^d\}$$

acter 'i' as the example, the observation sequence obtained by this coding method will be :-

$$O^d = (6, 6, 6, \pi, \delta).$$

This method generates 10 possible observation symbols, the 8 direction codes, and the penup and dot special codes. The major difference between this encoding method and the inclination angle encoding method presented in the previous section, is that the observation sequences passed onto the next stage in the inclination angle are in terms of degrees, whereas in the method considered in this section the output consists mainly of integers.

4.4 Comparison and Decision Processes.

Now that the encoding scheme to be investigated in our research, have been described in some detail, our attention is turned to the comparison and decision stages. Before describing the actual comparison and decision processes, the HMMs to be used within the comparison stage have to be considered. For whatever encoding scheme used to construct the observation sequence/s the same general model can be used. The left-to-right HMM described in section 2.4 will be used to model each character to be recognised by the system. The first type of left-to-right model to be considered will be a 3 state model with skip states, Figure 11. This type of model will allow variations not just in the observation data in a specific state, but it will variations in writing speed to occur, with the self-looping back into the same state taking into account slower than normal writing, and the skip state from state 1 to state 3 taking into account faster than normal writing.

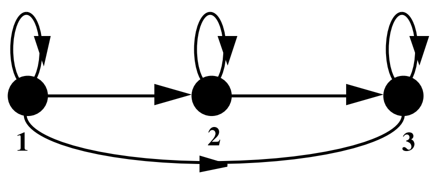


Figure 11. Showing a 3 state left-to-right HMM with skip states.

It should be noted that only one left-to-right model is required to represent each different character in the inclination angle and directional encoding schemes, i.e. only 26 models are required to represent the 26 letters of the alphabet. In the poisitonal encoding method two left-to-right models are required for each character to be modelled in the system, one for the x and one for the y directions, thus 52 model are required.

Though this number can be halved if the O^x and O^y are combined into one observation sequence as was done by Jeng *et al* [3].

Of course this type of model already constraints the Π and A matrices, which will initially be set as follows for all models in whatever description method used :-

$$\Pi = [1 \quad 0 \quad 0]$$

$$A = \begin{bmatrix} 0.34 & 0.33 & 0.33 \\ 0.0 & 0.5 & 0.5 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

where Π and A can be partially derived from the conditions laid down for left-to-right models in section 2.4. What about the B matrix for each character? This will be different for each method, since the symbols are dependent on the way the observation sequence is constructed. According to Rabiner, [1], only the B estimates have to be good estimates, but how are these estimates obtained. Rabiner suggest as number of methods including manual segmentation of observation sequences into states and then averaging observations within states, or maximum likelihood segmentation of observation sequences with averaging within states or by the use of segmental k-means segmentation with clustering as considered by Rabiner in [1] and [21]. It is proposed for our system to use manual segmentation of a small number of observation sequences for each character to be recognised, in the training set and using averaging within each state of the observations. For example, if the character i considered throughout this section as an example is used to build the observations, then using the observation sequence O^x for the example i as training data for segmenting and establishing the initial B matrix, the B matrix for the x HMM will be :-

$$B = \begin{bmatrix} 0.9994 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 \\ 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.9994 \\ 0.9994 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 \end{bmatrix}$$

where the rows represent the present state j, and the columns represent the possible observation symbols, the 6 quantisation levels 0.0 to 1.0, and the last column represents the symbol p for penup. Note that the penup that occurs in the character i allows for easy segmentation of the observations between the states in the model.

Once the initial estimates for the B matrix for a particular character model have been established, then the training procedure begins. In this training period a number of observation sequences of the character the model is being trained to represent are used. The training procedure using the Baum-Welch re-estimation algorithms will be used to train each model, see section 2.3, but only the re-estimation algorithms for the and

$$a_{ij}$$

$$b_j(k)$$

have to be used, equations (6b) and (6c), since the constraint that the left-to-right model always starts in state 1 has already been set. The forward procedure described in section 2.1, will be used to score the models during training and recognition. The use of the Viterbi Algorithm, described in section 2.2, in scoring the models will also be investigated.

For this system the character description just supplied to the comparison stage is compared to all the models formed during the training stage. The forward procedure is then used to compare the observation received from the extraction stage, to determine how likely the character read in matches each model in the system. This will produce a confidence measure for each model for that particular observation sequence, , where is the

$$\Pr ob(O|\lambda_\chi)$$

$$\lambda_\chi$$

cth character model. The more closely the observation sequence matches a particular model, the higher the confidence measure and the more likely that the character the model represents the unknown character just written. Once a confidence measure has been obtained for each model a decision can be made on which character model best represents the unknown character just written. The simplest criteria to use for this decision is to say that the character model which has produced the best confidence measure is the unknown character. This decision stage can be represented by equation (25).

(25)

$$
\begin{aligned}
& max\_prob = -\infty \\
& \beta\varepsilon\sigma\tau\_\mu o\delta\,\varepsilon\lambda = 1 \\
& \\
& \forall\chi \quad \omega\eta\varepsilon\rho\varepsilon 1 \le \chi \le \mathrm{X}, \\
& \iota\phi(\Pi\rho o\beta(\mathrm{O}|\lambda_\chi) > \mu\alpha\xi\_\pi\rho o\beta)\ \tau\eta\varepsilon\nu \\
& \quad \mu\alpha\xi\_\pi\rho o\beta = \Pi\rho o\beta(\mathrm{O}|\lambda_\chi) \\
& \quad \beta\varepsilon\sigma\tau\_\mu o\delta\,\varepsilon\lambda = \chi \\
& \varepsilon\nu\delta\iota\phi
\end{aligned}
$$

where is the total number of separate characters that are modelled in the system, in our

$$C$$

system and is the probability measure obtained by the forward procedure, for the ob-

$$C = 26$$

$$\Pr ob(O|\lambda_\chi)$$

servation sequence given the model, , for the cth character. The output of this stage is

$$\lambda_\chi$$

the reference number for the best fitting model, c, and it's confidence measure, . In the

$$Prob(O|\lambda_\chi)$$

case of the position encoding method is given by combining the confidence measure

$$\Pr ob(O|\lambda_\chi)$$

for the x and y HMMs, as is shown in equation (26).

$$\Pr ob(O|\lambda_\chi) = \Pi\rho o\beta(O^\xi|\lambda_\chi^\xi) + \Pi\rho o\beta(O^\psi|\lambda_\chi^\psi) \quad (26)$$

Both equations (25) and (26) can be easily adapted if the Viterbi Algorithm is used to score the models instead of the forward procedure.

This would be the simplest form of character recognition, but no method of character recognition is perfect, so this is where higher level models can be used to correct wrongly recognised characters. Here we can see the inherent advantage of using HMMs in character recognition, since a confidence measure is produced for each character model, not only can the most likely character be passed to higher level models, but the next n most likely characters along with their confidence measures can be passed, thus supplying higher level models with more information about the characters it has recognised. This will be of more use when two character models have similar confidence measures. Such higher level models take into account the whole word, and then the sentence that each word is contained in. Such a system has already been developed for static character recognition by Vlontzos and Kung, [5], using Hidden Markov Models at the character and word levels. All that would have to be done to their system is substitute the static character system with one of the systems described in this section.

5. Summary and Conclusions.

The application of Hidden Markov Models to dynamic character recognition has been considered. A review of the theory and some of the implementation issues has been presented. It should be noted that experiments need to be conducted to see how well the various methods suggested in this report are in reality. It should be obvious to the reader the advantage of the inclination angle and directional encoding schemes in that they require no normalisation of the incoming data to derive a description.

An investigation into what type of model, left-to-right, ergodic or a mixture, best represents the underlying Markov process in dynamic character recognition has to be carried, similar to the one suggested by Nag *et al* [6] for dynamic cursive word recognition. Also an investigation into what other types of descriptions can be used and how well they perform against the other method already described in this paper, especially what combinations of descriptions work best and which descriptions have little or no relevant information to the character.

Two obvious applications where the work presented in this paper can be used, are dynamic signature verification and single writer adaptation. The dynamic signature verification work would require adaptation of the methods presented in this paper, including the need of only on HMM to represent the signature, and the finding of more writer dependent features. The application of HMMs to dynamic signature verification has been considered by Hanan *et al* in [22], representing the and position sequences

$$x(t)$$

$$y(t)$$

as a set of up and down slopes and the time interval for each of these slopes.

Another problem that is not considered in the literature is that of punctuation recognition, i.e. recognising such characters as ?, or ", though the recognition of mathematical equations and line drawings have been considered, [23,24]. Whether it is static or dynamic recognition, this problem has to be considered just as important as the character recognition problem, and should really be considered as part of the same process. The reasoning behind this is that the English language is not just made up of letters which make up words, but digits and punctuation which give the language extra meaning, and allow us the distinguish between the end of one sentence and the beginning of another for example. If character recognition systems, both static and dynamic, are to become viable and accepted commercially for use in large scale document reading, then the ability to recognise punctuation has to be included.

Acknowledgements.

The authors would like to thank the members of the Parallel Systems Group in the Computer Science Department of the University of Warwick for their encouragement, help and useful discussions about character recognition and Hidden Markov Models.

Appendix. Derivation of Re-estimation Formulas for Model Parameters.

In this appendix the re-estimation formulas for , and are derived using the formulas

$$\pi_\iota$$

$$a_{ij}$$

$$b_j(k)$$

given in [1]. Before the derivation of the re-estimation formulas can be carried out, two more variables have to be introduced. The first is the state probability, , which can be

$$\gamma_\tau(\iota)$$

defined as the probability of being in state at time t, given the observation sequence

$$x_i$$

and the model, equation (A1).

$$\gamma_\tau(\iota) = \Pi\rho o\beta(\iota_\tau = \xi_\iota | O, \lambda) \qquad \text{(A1)}$$

Equation (A1) can be expressed in terms of the forward-backward variables as in equation (A2).

(A2)

$$\gamma_\tau(\iota) = \frac{\alpha_\tau(\iota)\beta_\tau(\iota)}{\Pi\rho\omega\beta(O|\lambda)} = \frac{\alpha_\tau(\iota)\beta_\tau(\iota)}{\sum_{\iota=1}^{N}\alpha_\tau(\iota)\beta_\tau(\iota)}$$

To be able to derive the equations for and the probability of a particular transition

π_ι

a_{ij}

occurring has to defined. The branch probability, , can be defined

$\xi_\tau(\iota,\varphi)$

according to [1] as the joint probability that given the observation sequence and the model, that at time t the model is in state and at time t+1 the model is in state , equation

x_i

x_j

(A3). This can be expressed in terms of the model parameters, and the forward-backward variables as in equation (A4).

(A3)

$$\xi_\tau(\iota,\varphi) = \Pi\rho\omega\beta(\iota_\tau = \xi_\iota, \iota_{\tau+1} = \xi_\varphi | O,\lambda)$$

(A4)

$$\xi_\tau(\iota,\varphi) = \frac{\alpha_\tau(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_{\tau+1})\beta_{\tau+1}(\varphi)}{\sum_{\nu=1}^{N}\sum_{\mu=1}^{N}\alpha_\tau(\nu)\alpha_{\nu\mu}\beta_\mu(O_{\tau+1})\beta_{\tau+1}(\mu)}$$

So the state probability , can be expressed in terms of the branch probability , as

$\gamma_\tau(\iota)$

$\xi_\tau(\iota,\varphi)$

shown in equation (A5).

(A5)

$$\gamma_\tau(\iota) = \sum_{\varphi=1}^{N}\xi_\tau(\iota,\varphi)$$

The derivation of the re-estimation formula for can now be looked at. From section

π_ι

2.3, it can be seen that the definition of the re-estimation formula, equation (5a), for a particular initial state is the expected number of times in that state at . This can be ex-

x_i

$t = 1$

pressed in terms of the state probability as in equation (A6).

$\gamma_\tau(\iota)$

(A6)

$$\bar{\pi}_\iota = \gamma_1(\iota)$$

Looking at the right hand side of equation (A6) it can be seen that can be expressed in

$\gamma_\tau(\iota)$

terms of as in equation (A7). on the right hand side of this equation can be substituted

$\xi_\tau(\iota, \varphi)$

$\xi_1(\iota, \varphi)$

by the right hand side of equation (A4), as in equation (A8).

(A7)

$$\gamma_1(\iota) = \sum_{\varphi=1}^{N} \xi_1(\iota, \varphi)$$

(A8)

$$\sum_{\varphi=1}^{N} \xi_1(\iota, \varphi) = \sum_{\varphi=1}^{N} \left(\frac{\alpha_1(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_2)\beta_2(\varphi)}{\sum_{\nu=1}^{N}\sum_{\mu=1}^{N} \alpha_1(\nu)\alpha_{\nu\mu}\beta_\mu(O_2)\beta_2(\mu)} \right)$$

Substituting equation (A8) into (A6) and we obtain the following equation in terms of the model parameters and the forward-backward variables, equation (A9), for the re-estimation formula for .

π_ι

(A9)

$$\bar{\pi}_\iota = \sum_{\varphi=1}^{N} \left(\frac{\alpha_1(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_2)\beta_2(\varphi)}{\sum_{\nu=1}^{N}\sum_{\mu=1}^{N} \alpha_1(\nu)\alpha_{\nu\mu}\beta_\mu(O_2)\beta_2(\mu)} \right)$$

Now that the re-estimation formula for π_ι has been derived, it is time to look at the derivation of the a_{ij} re-estimation formula. From section 2.3, equation (5b) it can be seen that the definition of the transition re-estimation formula, can be expressed in terms of $\gamma_\tau(\iota)$ and $\xi_\tau(\iota, \varphi)$ as in equation (A10).

$$\bar{a}_{ij} = \frac{\sum_{\tau=1}^{T-1} \xi_\tau(\iota, \varphi)}{\sum_{\tau=1}^{T-1} \gamma_\tau(\iota)} \qquad \text{(A10)}$$

The numerator of equation (A10) can be simply expressed in terms of the forward-backward variables and the model parameters as shown in equation (A11), by substituting equation (A4) in place of $\xi_\tau(\iota, \varphi)$.

$$\sum_{\tau=1}^{T-1} \xi_\tau(\iota, \varphi) = \sum_{\tau=1}^{T-1} \left(\frac{\alpha_\tau(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_{\tau+1})\beta_{\tau+1}(\varphi)}{\sum_{\nu=1}^{N}\sum_{\mu=1}^{N} \alpha_\tau(\nu)\alpha_{\nu\mu}\beta_\mu(O_{\tau+1})\beta_{\tau+1}(\mu)} \right) \qquad \text{(A11)}$$

The denominator of equation (A10) can be expressed in terms of $\xi_\tau(\iota, \varphi)$ as in equation (A12), according to equation (A5) above. The right hand side of equation (A12) can then be expressed in terms of the forward-backward variables and the model parameters, by substituting the definition for $\xi_\tau(\iota, \varphi)$ into this equation to give equation (A13).

$$\sum_{\tau=1}^{T-1} \gamma_\tau(\iota) = \sum_{\tau=1}^{T-1} \sum_{\varphi=1}^{N} \xi_\tau(\iota, \varphi) \qquad \text{(A12)}$$

(A13)

$$\sum_{\tau=1}^{T-1}\gamma_\tau(\iota)=\sum_{\tau=1}^{T-1}\sum_{\varphi=1}^{N}\left(\frac{\alpha_\tau(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_{\tau+1})\beta_{\tau+1}(\varphi)}{\sum_{\nu=1}^{N}\sum_{\mu=1}^{N}\alpha_\tau(\nu)\alpha_{\nu\mu}\beta_\mu(O_{\tau+1})\beta_{\tau+1}(\mu)}\right)$$

The expressions for the denominator and numerator can now be substituted back into equation (A10), to obtain the expression in equation (A14) below. It can be seen that the bottom expression of the denominator and the numerator are the same, i.e. . It can

$$\sum_{\nu=1}^{N}\sum_{\mu=1}^{N}\alpha_\tau(\nu)\alpha_{\nu\mu}\beta_\mu(O_{\tau+1})\beta_{\tau+1}(\mu)$$

also be seen that the bottom of the denominator can be factored out as in equation (A15), since the sum to j does not affect this part of the denominator in any way. Another reason for this factoring out is that the denominator is a sum of ratios, and returns the same value as the sum of the top of the expression being calculated first followed by a division by , as shown in equation (A15). This also means that is a common term

$$\sum_{\nu=1}^{N}\sum_{\mu=1}^{N}\alpha_\tau(\nu)\alpha_{\nu\mu}\beta_\mu(O_{\tau+1})\beta_{\tau+1}(\mu)$$

$$\sum_{\nu=1}^{N}\sum_{\mu=1}^{N}\alpha_\tau(\nu)\alpha_{\nu\mu}\beta_\mu(O_{\tau+1})\beta_{\tau+1}(\mu)$$

in both the denominator and numerator, and can therefore be cancelled out of the equation, leaving the expression for the re-estimation formula as given in equation (A16)

$$a_{ij}$$

below.

(A15)

$$\bar{a}_{ij}=\sum_{\tau=1}^{T-1}\left(\frac{\dfrac{\alpha_\tau(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_{\tau+1})\beta_{\tau+1}(\varphi)}{\sum_{\nu=1}^{N}\sum_{\mu=1}^{N}\alpha_\tau(\nu)\alpha_{\nu\mu}\beta_\mu(O_{\tau+1})\beta_{\tau+1}(\mu)}}{\dfrac{1}{\sum_{\nu=1}^{N}\sum_{\mu=1}^{N}\alpha_\tau(\nu)\alpha_{\nu\mu}\beta_\mu(O_{\tau+1})\beta_{\tau+1}(\mu)}\sum_{\varphi=1}^{N}\alpha_\tau(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_{\tau+1})\beta_{\tau+1}(\varphi)}\right)$$

(A16)

$$\bar{a}_{ij} = \sum_{\tau=1}^{T-1} \left(\frac{\alpha_\tau(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_{\tau+1})\beta_{\tau+1}(\varphi)}{\sum_{\varphi=1}^{N} \alpha_\tau(\iota)\alpha_{\iota\varphi}\beta_\varphi(O_{\tau+1})\beta_{\tau+1}(\varphi)} \right)$$

The last formula to derive is the re-estimation formula for the model parameters. The

$b_j(k)$

definition for this formula, given in section 2.3, equation (5c), can be expressed solely in terms of as in equation (A17).

$\gamma_\tau(\iota)$

(A17)

$$\bar{b}_j(k) = \frac{\sum_{\substack{\tau=1 \\ \sigma\tau.\ o_\tau = \varpi_\kappa}}^{T} \gamma_\tau(\varphi)}{\sum_{\tau=1}^{T} \gamma_\tau(\varphi)}$$

Since the re-estimation formula is a sum to T in comparison to the sum to T-1 of re-

$b_j(k)$

a_{ij}

estimation formula, equation (A4) for cannot be used since no observation exists for

$\gamma_\tau(\iota)$

T+1. So the original expression for in terms of the forward-backward variables, equa-

$\gamma_\tau(\iota)$

tion (A2), has to be substituted into both expressions for the denominator and numerator of equation (A17), since both and are defined for all T. So this leaves the following

$\alpha_\tau(\varphi)$

$\beta_\tau(\varphi)$

expression, equation (A18), for the numerator of equation (A17).

(A18)

$$\sum_{\tau=1}^{T} \gamma_\tau(\varphi) = \sum_{\tau=1}^{T}\left(\frac{\alpha_\tau(\varphi)\beta_\tau(\varphi)}{\sum_{\iota=1}^{N}\alpha_\tau(\iota)\beta_\tau(\iota)}\right)$$

The expression for the denominator of equation (A17) is similar to equation (A18) except for the condition that . Substituting these expressions back into equation (A17)

$$O_t = \varpi_k$$

leaves the following expression for the re-estimated formula, equation (A19).

$$b_j(k)$$

(A19)

$$\bar{b}_j(k) = \frac{\sum\limits_{\substack{\tau=1 \\ \alpha\tau.\ o_t=\varpi_k}}^{T-1}\left(\frac{\alpha_\tau(\varphi)\beta_\tau(\varphi)}{\sum_{\iota=1}^{N}\alpha_\tau(\iota)\beta_\tau(\iota)}\right)}{\sum\limits_{\tau=1}^{T}\left(\frac{\alpha_\tau(\varphi)\beta_\tau(\varphi)}{\sum_{\iota=1}^{N}\alpha_\tau(\iota)\beta_\tau(\iota)}\right)}$$

References.

1. L. R. Rabiner., A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition., Proceedings of the IEEE., February 1989.; 77. (2) : pp. 257 - 286..

2. R. Nag, *Speech and Speaker Recognition Using Hidden Markov Models and Vector Quantisation*, Ph.D. dissertation, Cambridge University Engineering Department, 1988.

3. B. S. Jeng, M. W. Chang, S. W. Sun, C. H. Shih and T. M. Wu, Optical Chinese Character Recognition with a Hidden Markov Model Classifier - A Novel Approach., Electronics Letters., August 1990.; 26 (18) : pp. 1530 - 1531.

4. Hatsukazu Tanaka, Y. Hirakawa and S. Kaneku, Recognition of Distorted Patterns Using the Viterbi Algorithm., IEEE Transactions on Pattern Analysis and Machine Intelligence., January 1982.; PAMI - 4 (1) : pp. 18 - 25.

5. J. A. Vlontzos, and S. Y. Kung, Hidden Markov Models for Character Recognition, UNK, 1992 : pp. 1719 - 1722.

6. R. Nag, K. H. Wong and F. Fallside, Script Recognition Using Hidden Markov Models, ICASSP86, 1986 : pp. 2071 - 2074.

7. J. Camillerapp, G. Lorette, G. Menier, H. Oulhadj and J. C. Pettier, Off-line and on-line methods for cursive handwriting recognition. In *From Pixels to Features III : Frontiers in Handwriting Recognition*. Elsevier Science Publishers B. V., Impedovo, S. and Simon, J.C., pp. 273 - 287, 1992.

8. R. F. H. Farag, Word-Level Recognition of Cursive Script, IEEE Transactions on Computers, February 1979; C - 28 (2) : pp. 172 - 175.

9. G. David Forney, Jr, The Viterbi Algorithm., Proceedings of the IEEE., March 1973; 61 (3) : pp. 268 - 278.

10. A. J. Viterbi, Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm., IEEE Transactions on Information Theory., April. 1967; IT - 13 (2) : pp. 260 - 269.

11. A. J. Viterbi, Convolutional Codes and Their Performance in Communication Systems., IEEE Transactions on Communications Technology., October 1971; COM - 19 (5) : pp. 751 - 772.

12. L. E. Baum, A Maximization Technique Occuring in the Statistical Analysis of Probabilistic Functions of Markov Chains, Ann Mathematical Statistics, 1970; 41 : pp. 164 - 171.

13. 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, The Bell System Technical Journal, April 1983; 62 (4) : pp. 1035 - 1074.

14. R. L. Streit, and R. F. Barrett, Frequency Line Tracking Using Hidden Markov Models., IEEE Transactions on Acoustics, Speech, and Signal Processing., April 1990; ASSP - 38 (4) : pp. 586 - 598.

15. Amlan Kunda, Y. He and P. Bahl, Recogintion of Handwritten Word: First and Second Order Hidden Markov Model Based Approach., Pattern Recognition, 1989.; 22. (3.) : pp. 283-297..

16. L. R. Rabiner and B. H. Juang, An Introduction to Hidden Markov Models., IEEE ASSP Magazine., January 1986 : pp. 4 - 16.

17. H. D. Crane and R. E. Savoie, An on-line data entry system for handprinted characters, Computer, 1977; 10 : pp. 43 - 50.

18. H. D. Crane and J. S. Ostrem, Automatic Signature Verification Using a Three-Axis Force-Sensitive pen, IEEE Transactions on Systems, Man, and Cybernetics, May / June 1983; SMC - 13 (3) : pp. 329 - 337.

19. Charles C. Tappert, Ching Y. Suen, and T. Wakahara, The State of the Art in On-Line Handwritting Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, August 1990; PAMI - 12 (8) : pp. 787 - 808.

20. F. Nouboud. and R. Plamondon, On-Line Recognition of Handprinted Characters : Survey and Beta Tests, Pattern Recognition, 1990; 23 (9) : pp. 1031 - 1044.

21. B. H. Juang, and L. R. Rabiner, The Segmental K-Means Algorithm for estimating Parameters of Hidden Markov Models., IEEE Transactions on Acoustic, Speech, and Signal Processing., September 1990; ASSP - 38 (9) : pp. 1639 - 1641.

22. M. Hanan, N. M. Herbst and J. M. Kutzberg, Finite State Machine Decision Procedure Model For Signature Verification, IBM Technical Disclosure Bulletin, January 1978; 20 (8) : pp. 3355 - 3360.

23. H. Murase and T. Wakahara, Online hand-sketched figure recognition, Pattern Recognition, 1986; 19 : pp. 37 - 46.

24. A. Belaid, and J. P. Haton, A syntatic approach for handwritten mathematical formula recognition, IEEE Transactions of Pattern Analysis and Machine Intelligence, 1984; PAMI - 6 : pp. 105 - 111.