Classification using Hidden Markov Models

Report #03

Luiz S. Oliveira and Marisa Morita

Federal University of Parana (UFPR)
Department of Informatics (DInf)
R. Rua Cel. Francisco H. dos Santos, 100, Curitiba, PR, Brazil
lesoliveira@inf.ufpr.br

1 Introduction

This report describes the activities performed in Aug/Sep 2013, which consist in using Hidden Markov Models (HMM) to classify handwritten words using the global features introduced in the last report. In order to make this report self-contained, in Section 2 we present the theory of HMMs. Then, in Section 3 we describe how we have built the HMM models to recognise handwritten words while in Section 4 we report some experimental results. Finally, Section 5 concludes this report pointing out our next steps. The source codes of all experiments are available in the SVN repository.

2 Introduction to HMM

A Hidden Markov Model (HMM) is a doubly stochastic process, with an underlying stochastic process that is not observable (hence the word hidden), but can be observed through another stochastic process that produces the sequence of observations [1]. The hidden process consists of a set of states connected to each other by transitions with probabilities, while the observed process consists of a set of outputs or observations, each of which may be emitted by each state according to some output probability density function (pdf). Depending on the nature of this pdf, several kinds of HMMs can be distinguished. If the observations are naturally discrete or quantized using vector quantization [2], and drawn from an alphabet or a codebook, the HMM is called discrete. If these observations are continuous we are dealing with a continuous HMM, with a continuous pdf usually approximated by a mixture of normal distributions. In the context of this thesis we consider discrete HMMs.

In some applications, it is more convenient to produce observations by transitions rather than by states. Furthermore, it is sometimes useful to allow transitions with no output in order to

model, for instance, the absence of an event in a given stochastic process. If we add the possibility of using more than one feature set to describe the observations, we must modify the classic formal definition of HMMs [1]. These modifications can be found in [3] and they are also described in this appendix. In this case, the following parameters are required:

- T: length of the observation sequence $O = \{o_0, o_1, \ldots, o_{T-1}\}$, where $o_t = (o_t^0, o_t^1, \ldots, o_t^{P-1})$, the observation o_t^p at time t being drawn from the p^{th} finite feature set, and $p = 0, 1, \ldots, P-1$.
- N: number of states in the model.
- M_p: number of possible observation symbols for the p^{th} feature set.
- $S = \{s_0, s_1, \ldots, s_{N-1}\}$: set of possible states of the model.
- $Q = \{q_t\}$: q_t denotes the state at time t.
- $V_p = \{v_0^p, v_1^p, \ldots, v_{M-1}^p\}$: codebook or discrete set of possible observation symbols corresponding to the p^{th} feature set.
- $A = \{a_{ij}\}$, $a_{ij} = P[q_{t+1} = s_j | q_t = s_i]$: probability of going from state s_i at time t to state s_j at time $(t+1)$, and at the same time producing a real observation o_t at time t.
- $A' = \{a'_{ij}\}$, $a'_{ij} = P[q_t = s_j | q_t = s_i]$: probability of null transition from state s_i at time t to state s_j at time t, producing null observation symbol Φ. Note here that there is no increase over time since no real observation is produced.
- $B_p = \{b_{ij}^p(k)\}$, $b_{ij}^p(k) = P[o_t^p = v_k^p | q_t = s_i, q_{t+1} = s_j]$: output pdf of observing the k^{th} symbol in the p^{th} feature set when a transition from state s_i at time t to state s_j at time $(t+1)$ is taken. If we assume the P output pdfs are independent (multiple codebooks), we can compute the output probability $b_{ij}(k)$ as the product of P output probabilities:

$$b_{ij}(k) = \prod_{p=0}^{P-1} b_{ij}^p(k) \tag{1}$$

- $\pi = \{\pi_i\}$, $\pi_i = P[q_0 = s_i]$: initial state distribution. In general, it is more convenient to have predefined initial and final states s_0 and s_{N-1} that do not change over time. In this case, $\pi_0 = 1$ and $\pi_i = 1, 2, \ldots, N-1$.

A, A', B_p, and π obey the stochastic constraints:

$$\sum_{j=0}^{N-1} [a_{ij} + a'_{ij}] = 1 \qquad \sum_{k=0}^{M_p-1} b_{ij}^p(k) = 1 \qquad \sum_{i=0}^{N-1} \pi_i = 1 \tag{2}$$

$$p = 0, 1, \ldots, P-1$$

It can be seen that a complete specification of an HMM requires specification of two model parameters, N and M, specification of observation symbols, and the specification of the four sets of probability measures A, A', B_p and π where $p = 0, 1, \ldots, P-1$. For convenience, we use the compact notation $\lambda = (A, A', B_p, \pi)$ to indicate the complete parameter set of the model. This parameter set, of course, defines a probability measure for O, i.e., $P(O|\lambda)$, which we discuss along this Appendix.

2.1 Types of HMMs

There are two important types of HMMs: ergodic and left-right or Bakis model [1]. The ergodic model is a specific case of a fully-connected model when all a_{ij} are positive. In this type of model, the states are interconnected in such a way that any state can be reached from any other state. Figure 1(a) shows a 4-state ergodic HMM model. The left-right model presents an important kind of state interconnection for text recognition modeling, which has the property:

$$a_{ij} = 0, \quad j < i \tag{3}$$

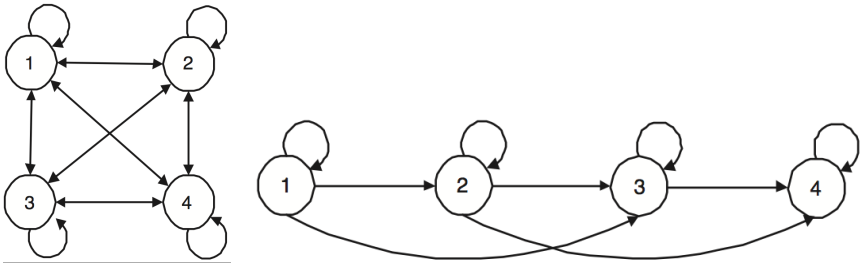
This property means that no transitions are allowed to states whose indices are lower than that of the current state. Since the state sequence must begin in state 0 (and end in state $N-1$), the initial state probabilities have the following property:

$$\pi_i = \begin{cases} 0, & i \neq 0 \\ 1, & i = 0 \end{cases} \tag{4}$$

Often, with left-right models, additional constraints are used in order to avoid great changes in state indices, such as:

$$a_{ij} = 0, \quad j > i + \Delta \tag{5}$$

The value of Δ is used as a limit for jumps. For instance, in 1(b), Δ is two, that is, no jumps of more than two states are allowed.



(a) (b)

Figure 1: Types of HMMs: (a) Ergodic model, and (b) Left-right model

2.2 The Three Basic Problems for HMMs

Given a model, three basic problems of interest must be solved for the model to be useful in real-word applications. These problems are the following:

- The evaluation problem: given an observation sequence $O = (o_0, o_1, \ldots, o_{T-1})$, and a model $\lambda = (A, A', B, \pi)$, how do we compute $P(O|\lambda)$, the probability of O given λ ?
- The decoding problem: given the observation sequence $O = (o_0, o_1, \ldots, o_{T-1})$, and the model λ, how do we find the optimal state sequence in λ that has generated O ?
- The training problem: given a set of observation sequences and an initial model λ, how can we re-estimate the model parameters so as to increase the likelihood of generating this set of sequences ?

2.2.1 The Evaluation Problem

To compute $P(O|\lambda)$, we modify the well-known Forward-Backward procedure [1] to take into account the assumption that symbols are emitted along transitions, the possibility of null transitions, and the use of multiple codebooks. Hence, we define the forward probability $\alpha_t(i)$ as:

$$\alpha_t(i) = P(o_0, o_1, \ldots, o_{t-1}, q_t = s_i|\lambda) \tag{6}$$

where $\alpha_t(i)$ is the probability of the partial observation sequence $(o_0, o_1 \ldots, o_{t-1})$ (until time $t-1$) and the state s_i reached at time t given the model λ. $\alpha_t(i)$ can be inductively computed as follows.

1. *Initialization*

$$\alpha_0(0) = 1.0 \tag{7}$$

$$\alpha_0(j) = \sum_{i=0}^{N-1} a'_{ij}\alpha_0(i) \quad j = 0, 1, \ldots, N-1$$

given that s_0 is the only possible initial state.

2. *Induction*

$$\alpha_t(j) = \sum_{i=0}^{N-1} \left[a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_{t-1}) \right) \alpha_{t-1}(i) + a'_{ij}\alpha_t(i) \right] \tag{8}$$

$$j = 0, 1, \ldots, N-1 \quad \textit{and} \quad t = 1, 2, \ldots, T$$

3. *Termination*

$$P(O|\lambda) = \alpha_T(N-1) \tag{9}$$

given that s_{N-1} is the only possible terminal state. Similarly, we define the backward probability $\beta_t(i)$ by:

$$\beta_t(i) = P(o_t, o_{t+1}, \ldots, o_{T-1}|q_t = s_i, \lambda) \tag{10}$$

where $\beta_t(i)$ is the probability of the partial observation sequence from the time t to the end, given state s_i reached at time t and the model λ. $\beta_t(i)$ can also be inductively computed as follows.

1. *Initialization*

$$\beta_T(N-1) = 1.0 \tag{11}$$

$$\beta_T(i) = \sum_{j=0}^{N-1} a'_{ij}\beta_T(j) \quad i = 0, 1, \ldots, N-1$$

given that s_{N-1} is the only possible terminal state.

2. *Induction*

$$\beta_t(i) = \sum_{j=0}^{N-1} \left[a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t) \right) \beta_{t+1}(j) + a'_{ij}\beta_t(j) \right] \tag{12}$$

$$t = T-1, T-2, \ldots, 0 \quad and \quad i = 0, 1, \ldots, N-1$$

3. *Termination*

$$P(O|\lambda) = \beta_0(0) \tag{13}$$

given that q_0 is the only possible initial state.

2.2.2 The Decoding Problem

The decoding problem is solved using a near-optimal procedure, the Viterbi algorithm, by looking for the best state sequence $Q = (q_0, q_1, \ldots, q_T)$ for the given observation sequence $O = (o_0, o_1, \ldots, o_{T-1})$. Again, we modify the classic algorithm [1] in the following way.

$$\delta_t(i) = \max_{q_0, q_1, \ldots, q_{t-1}} P[q_0, q_1, \ldots, q_t, q_t = s_i, o_0, o_1, \ldots, o_{t-1}|\lambda] \tag{14}$$

where $\delta_t(i)$ is the probability of the best path that accounts for the first t observations and ends at state s_i at time t. The function $\Psi_t(i)$ is defined to recover the best state sequence by a procedure called Backtracking. $\Psi_t(i)$ e $\delta_t(i)$ can be recursively computed as follows.

1. *Initialization*

$$\delta_0(0) = 1.0 \tag{15}$$

$$\Psi_0(0) = 0$$

$$\delta_0(j) = \max_{0 \leq i \leq N-1} [\delta_0(i) a'_{ij}] \quad j = 0, 1, \ldots, N-1$$

$$\Psi_0(j) = \arg \max_{0 \leq i \leq N-1} [\delta_0(i) a'_{ij}] \quad j = 0, 1, \ldots, N-1$$

given that s_0 is the only possible initial state.

2. *Recursion*

$$\delta_t(j) = \max_{0 \le i \le N-1} \left[a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_{t-1}) \right) \delta_{t-1}(i); a'_{ij} \delta_t(i) \right] \tag{16}$$

$$t = 1, 2, \ldots, T \quad and \quad j = 0, 1, \ldots, N-1$$

$$\Psi_t(j) = \arg \max_{0 \le i \le N-1} \left[a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_{t-1}) \right) \delta_{t-1}(i); a'_{ij} \delta_t(i) \right] \tag{17}$$

$$t = 1, 2, \ldots, T \quad and \quad j = 0, 1, \ldots, N-1$$

3. *Termination*

$$P^* = \delta_T(N-1) \tag{18}$$

$$q_T^* = (N-1) \tag{19}$$

given that s_{N-1} is the only possible terminal state.

4. *Backtracking procedure*

$$q_t^* = \Psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 0 \tag{20}$$

As shown above, except for the Backtracking procedure, Viterbi and Forward procedures are similar. The only difference is that the summation is replaced by maximization.

2.2.3 The Training Problem

The main strength of HMMs is the existence of a procedure called the Baum-Welch algorithm [4, 1] that iteratively and automatically adjusts HMM parameters given a training set of observation sequences. This algorithm, which is an implementation of the Expectation-Maximization algorithm [5], guarantees the model to converge to a local maximum of the probability of observation of the training set according to the maximum likelihood estimation criterion. This maximum depends strongly on the initial parameters.

To re-estimate HMM parameters, we first define $\xi_t^1(i,j)$, the probability of being in state s_i at time t and in state s_j at time $(t+1)$, producing a real observation O_t, given the model and the

observation O, and $\xi_t^2(i,j)$, the probability of being in state i at time t and in state j at time t, producing the null observation Φ, given the model and the observation O.

$$\xi_t^1(i,j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda) \tag{21}$$

$$\xi_t^2(i,j) = P(q_t = s_i, q_t = s_j | O, \lambda) \tag{22}$$

The development of these quantities leads to:

$$\xi_t^1(i,j) = \frac{\alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t) \right) \beta_{t+1}(j)}{P(O|\lambda)} \tag{23}$$

$$\xi_t^2(i,j) = \frac{\alpha_t(i) a'_{ij} \beta_t(j)}{P(O|\lambda)} \tag{24}$$

We also define $\gamma_t(i)$ as the probability of being in state s_i at time t, given the model and the observation sequence.

$$\gamma_t = P(q_t = s_i | O, \lambda) \tag{25}$$

$\gamma_t(i)$ is related to $\xi_t^1(i,j)$ and $\xi_t^2(i,j)$ by the following equation:

$$\gamma_t = \sum_{j=0}^{N-1} [\xi_t^1(i,j) + \xi_t^2(i,j)] = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \tag{26}$$

The re-estimations of HMM parameters $\{a_{ij}\}$, $\{a'_{ij}\}$ e $\{b_{ij}^p(k)\}$ are:

$$\overline{a_{ij}} = \frac{\text{expected number of transitions from } s_i \text{ at time } t \text{ to } s_j \text{ at time } (t+1)}{\text{expected number of being in } s_i} \tag{27}$$

$$\overline{a'_{ij}} = \frac{\text{expected number of transitions from } s_i \text{ at time } t \text{ to } s_j \text{ at time } t \text{ and observing } \Phi}{\text{expected number of being in } s_i} \tag{28}$$

$$\overline{b_{ij}^p(k)} = \frac{\text{expected number of symbols } v_k^p \text{ in transition from } s_i \text{ at time } t \text{ to } s_j \text{ at time } (t+1)}{\text{expected number of transitions in } s_i \text{ at time } t \text{ to } s_j \text{ at time } (t+1)} \tag{29}$$

Given the definitions of $\xi_t^1(i,j)$, $\xi_t^2(i,j)$, and $\gamma_t(i)$, it is easy to see, when we are using one observation sequence O:

$$\overline{a_{ij}} = \frac{\sum_{t=0}^{T-1} \xi_t^1(i,j)}{\sum_{t=0}^{T} \gamma_t(i)} = \frac{\sum_{t=0}^{T-1} \alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t)\right) \beta_{t+1}(j)}{\sum_{t=0}^{T} \alpha_t(i)\beta_t(i)} \tag{30}$$

$$\overline{a'_{ij}} = \frac{\sum_{t=0}^{T-1} \xi_t^2(i,j)}{\sum_{t=0}^{T} \gamma_t(i)} = \frac{\sum_{t=0}^{T-1} \alpha_t(i) a'_{ij} \beta_t(j)}{\sum_{t=0}^{T} \alpha_t(i)\beta_t(i)} \tag{31}$$

$$\overline{b_{ij}^p(k)} = \frac{\sum_{t=0}^{T-1} \delta(o_t^p, v_k^p)\xi_t^1(i,j)}{\sum_{t=0}^{T-1} \xi_t^1(i,j)} = \frac{\sum_{t=0}^{T-1} \delta(o_t^p, v_k^p)\alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t)\right) \beta_{t+1}(j)}{\sum_{t=0}^{T-1} \alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t)\right) \beta_{t+1}(j)} \tag{32}$$

where $\delta(x,y) = \begin{pmatrix} 1 & if x = y \\ 0 & if x \neq y \end{pmatrix}$

For a set of training sequences $O(0), O(1), \ldots, O(U-1)$ (size U), as is usually the case in real world applications, the above formulas become:

$$\overline{a_{ij}} = \frac{\sum_{u=0}^{U-1} \sum_{t=0}^{T-1} \xi_t^1(i,j,u)}{\sum_{u=0}^{U-1} \sum_{t=0}^{T} \gamma_t(i,u)} = \frac{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T-1} \alpha_t(i,u) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t^p(u))\right) \beta_{t+1}(j,u)}{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \alpha_t(i,u)\beta_t(i,u)} \tag{33}$$

$$\overline{a'_{ij}} = \frac{\sum_{u=0}^{U-1} \sum_{t=0}^{T-1} \xi_t^2(i,j,u)}{\sum_{u=0}^{U-1} \sum_{t=0}^{T} \gamma_t(i,u)} = \frac{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T-1} \alpha_t(i,u) a'_{ij} \beta_t(j,u)}{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \alpha_t(i,u)\beta_t(i,u)} \tag{34}$$

$$\overline{b_{ij}^p(k)} = \frac{\sum_{u=0}^{U-1} \sum_{t=0}^{T-1} \delta(o_t^p(u), v_k^p)\xi_t^1(i,j,u)}{\sum_{u=0}^{U-1} \sum_{t=0}^{T-1} \xi_t^1(i,j,u)} \tag{35}$$

$$\overline{b_{ij}^p(k)} = \frac{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T-1} \delta(o_t^p(u), v_k^p)\alpha_t(i,u) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t^p(u))\right) \beta_{t+1}(j,u)}{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T-1} \alpha_t(i,u) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t^p(u))\right) \beta_{t+1}(j,u)}$$

In the above equations, the index u is introduced into α, β, ξ^1, ξ^2, and γ to indicate the observation sequence $O(u)$ currently used. Note that a new quantity $P(u) = P(O(u)|\lambda)$ appears, since this term is now included in the summation and cannot be eliminated as before.

If we define the current model as $\lambda = (A, A', B_p, \pi)$ and the re-estimated model as $\overline{\lambda} = (\overline{A}, \overline{A'}, \overline{B_p}, \overline{\pi})$, and we iteratively use $\overline{\lambda}$ in place of λ and repeat the re-estimation calculation, we can then improve the probability of O being observed from the model until some limiting point is reached. The final result of this re-estimation procedure is a maximum likelihood estimate of the HMM. It should be pointed out that the Forward-Backward algorithm leads to a local maxima only, and that in most problems of interest, the likelihood function is very complex and has many local maxima.

3 Modelling Handwritten Words using HMMs

Based on our previous experience, we decided to represent the characters using the maximum of three graphemes. In light of this, four states are necessary for each character model as depicted in Figure 2. This model is based on the architecture proposed by El Yacoubi et al [6].

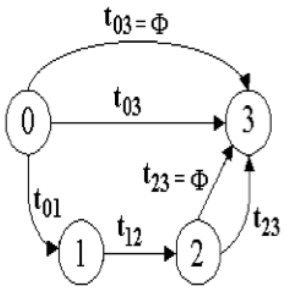


Figure 2: Character model architecture

In this model, the observation sequences are emitted from the model transitions in order to take advantage of the explicit segmentation adopted during feature extraction. All possible segmentation outcomings are reported in Table 1. Considering the database we are using for the experiments so far, the month alphabet is composed of 20-character classes, therefore,we have 40 HMMs considering uppercase and lowercase characters.

Table 1: Transitions allowed by the character model

Transition	Description
$t_{03} = \Phi$	character under-segmentation
t_{03}	no segmentation detected
$t_{01} - t_{12} - (t_{23} = \Phi)$	character over-segmented into 2 graphemes
$t_{01} - t_{12} - t_{23}$	character over-segmented into 3 graphemes

3.1 Training the Models

Since the writing style (uppercase/lowercase) of each training word image is available, the word model is generated from the concatenation of appropriate character models. The last state of a character model becomes the initial state of the next model, as depicted in Figure 3.

In order to train the models, we are using the Baum-Welch algorithm (described in Section 2.2.3) with cross-validation procedure. The program used to train the HMM models is available in the following directory:

`~/tools/orand/batch/crossv`

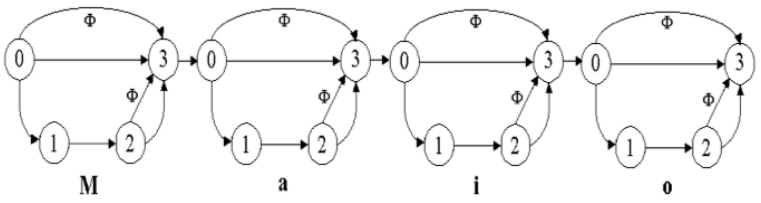


Figure 3: Training model of class “Maio”

To execute the training, use:

```
./orand_crossanmt feat1 ex01 symbol_1.ex01 m_month.ex01
```

3.2 Recognition

Since we are dealing with unconstrained handwriting, one can write the month using uppercase, lowercase, or a mix of both. With this in mind, we have used three different HMM configurations for recognition. The programs used for recognition are available in the following directory:

```
~/tools/orand/batch/rec/an
```

First we have built 24 models (similar to the one presented in Figure 3), 12 for lowercase and 12 for uppercase. To recognise using 24 models, use:

```
./orand_recanmt24cb 1 feat1 ex01 symbol_1.ex01 m_month.ex01
```

In the second experiment, we have added another 12 models, summing up 36 models, where the first letter is uppercase and all the others lowercase. To recognise using 36 models, use:

```
./orand_recanmt36cb 1 feat1 ex01 symbol_1.ex01 m_month.ex01
```

However, none of these strategies are able to handle the problem of mixed handwriting styles. To deal with that, the third model, inspired in the work of El Yacoubi [6], is composed of two character models in parallel (uppercase and lowercase), as depicted in Figure 4.

The word model consists of an initial state (I) and a final state (F), and two consecutive character models linked by four transitions: two uppercase characters (UU), two lowercase characters (LL), one uppercase character followed by one lowercase character (UL) and one lowercase character followed by one uppercase character (LU). The probabilities of these transitions are estimated by their occurrence frequency in the training set. In the same manner, the probabilities of beginning a word by an uppercase character (0U) or lowercase character (0L) are also estimated. This architecture handles the problem related to the mixed handwritten words detecting implicitly the writing style during recognition using the Backtracking procedure of the Viterbi algorithm. To recognise using 12 models (in parallel), use:

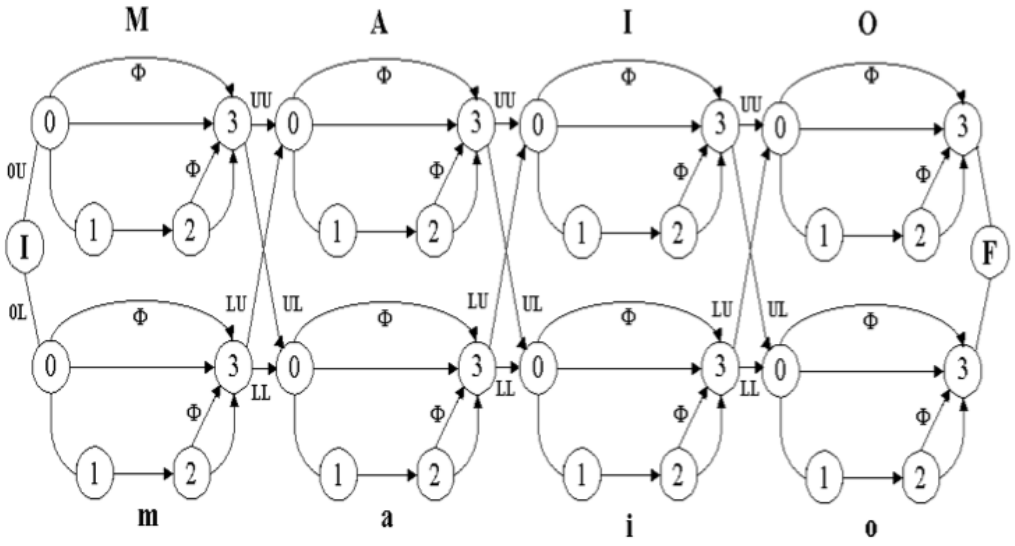


Figure 4: Training model of class “Maio”

```
- /orand_recanmt12 1 feat1 ex01 symbol_1.ex01 m_month.ex01
```

4 Experiments

In these experiments we have used the Brazilian month word database, which is composed of 2,000 images. It was divided into 1,190, 408 and 400 images for training, validation, and testing, respectively. The results reported in this section were carried out on the test using the Forward procedure. Table 2 summarises the results of the three experiments using the Global features (20 components).

Table 2: Summary of the recognition rates obtained in the experiments

Experiment	Number of HMM Models	Rec. Rate (%)
I	24	80.9
II	36	82.2
III	12	82.4

As one can notice, the third model achieve a slightly better result with a considerable smaller number of HMM models. These results show that the parallel model can handle better the mixture of handwriting styles. In spite of the similar results, by analysing the confusion matrices (Tables 3, 4, and 5), one can notice that in some cases the three models make different mistakes. This suggests that this kind of complementarity may be explored by fusing the three different strategies in order to increase the recognition rates. This will be subject of further investigation during the project.

Table 3: Confusion Matrix (%) - Experiment I

Class	1	2	3	4	5	6	7	8	9	10	11	12
1	**73.7**	10.5	5.3	-	-	5.3	2.6	-	2.6	-	-	-
2	3.1	**84.4**	6.3	-	-	-	-	-	-	3.1	3.1	-
3	-	2.8	**69.4**	2.8	25.0	-	-	-	-	-	-	-
4	-	-	-	**82.1**	12.8	-	-	2.6	2.6	-	-	-
5	-	2.6	-	5.3	**86.8**	-	-	5.3	-	-	-	-
6	3.4	-	-	-	3.4	**89.7**	-	-	3.4	-	-	-
7	3.1	-	3.1	-	6.3	12.5	**75.0**	-	-	-	-	-
8	-	-	3.6	-	7.1	-	-	**82.1**	-	-	-	7.1
9	3.2	-	-	-	-	-	-	-	**83.9**	9.7	-	3.2
10	-	-	-	-	-	-	-	-	13.3	**86.7**	-	-
11	2.9	-	-	-	-	-	-	-	5.9	2.9	**88.2**	-
12	-	-	-	-	-	-	-	6.1	15.2	9.1	-	**69.7**

Table 4: Confusion Matrix (%) - Experiment II

Class	1	2	3	4	5	6	7	8	9	10	11	12
1	**76.3**	7.9	5.3	-	-	5.3	2.6	-	2.6	-	-	-
2	3.1	**90.6**	3.1	-	-	-	-	-	-	3.1	-	-
3	-	2.8	**66.7**	2.8	27.8	-	-	-	-	-	-	-
4	2.6	-	-	**84.6**	12.8	-	-	-	-	-	-	-
5	-	-	-	10.5	**86.8**	-	-	2.6	-	-	-	-
6	3.4	-	-	-	3.4	**93.1**	-	-	-	-	-	-
7	3.1	-	6.3	-	3.1	12.5	**71.9**	3.1	-	-	-	-
8	-	-	3.6	-	7.1	-	-	**85.7**	-	-	-	3.6
9	3.2	-	-	-	-	-	-	-	**83.9**	9.7	-	3.2
10	-	-	-	-	-	-	-	-	16.7	**83.3**	-	-
11	-	2.9	-	-	-	-	-	-	2.9	2.9	**91.2**	-
12	-	-	-	-	-	-	-	3.0	12.1	9.1	3.0	**72.7**

Table 5: Confusion Matrix (%) - Experiment III

Class	1	2	3	4	5	6	7	8	9	10	11	12
1	**73.6**	7.8	5.2	-	-	2.6	2.6	-	5.2	2.6	-	-
2	3.1	**84.3**	3.1	-	-	-	-	-	3.1	3.1	3.1	-
3	-	2.7	**66.6**	2.78	27.7	-	-	-	-	-	-	-
4	-	-	-	**89.7**	10.2	-	-	-	-	-	-	-
5	-	-	-	7.8	**86.8**	-	-	2.6	2.6	-	-	-
6	3.4	-	-	-	3.4	**93.1**	-	-	-	-	-	-
7	3.1	-	6.2	-	3.1	12.5	**71.8**	3.1	-	-	-	-
8	-	-	3.5	-	7.1	-	-	**85.7**	-	-	-	3.5
9	3.2	-	-	-	-	-	-	-	**83.8**	6.4	-	6.4
10	-	-	-	-	-	-	-	-	16.6	**83.3**	-	-
11	-	-	-	-	-	-	-	-	2.9	2.9	**94.1**	-
12	-	-	-	-	-	-	-	3.0	12.1	9.0	-	**75.7**

Another aspect worth of noticing is the TOP3 result for the model used in the third experiment. In this case, considering TOP3 (Table 6), the classifier would produce a recognition rate of 96.4%.

This result indicates that a verification strategy would be a possible alternative to recover some of these confusions. This subject also will be investigated in this project.

Table 6: Confusion Matrix (%) - TOP3 for Experiment III

Class	1	2	3	4	5	6	7	8	9	10	11	12
1	**92.1**	-	2.6	-	-	2.6	-	-	-	2.6	-	-
2	-	**96.8**	-	-	-	-	-	-	-	3.1	-	-
3	-	-	**100.0**	-	-	-	-	-	-	-	-	-
4	-	-	-	**92.3**	7.6	-	-	-	-	-	-	-
5	-	-	-	2.6	**92.1**	-	-	2.6	2.6	-	-	-
6	-	-	-	-	3.4	**96.5**	-	-	-	-	-	-
7	3.1	-	-	-	-	-	**96.8**	-	-	-	-	-
8	-	-	-	-	-	-	-	**100.0**	-	-	-	-
9	3.2	-	-	-	-	-	-	-	**96.7**	-	-	-
10	-	-	-	-	-	-	-	-	3.3	**96.6**	-	-
11	-	-	-	-	-	-	-	-	-	-	**100.0**	-
12	-	-	-	-	-	-	-	-	3.0	-	-	**96.9**

5 Conclusions

In this report we have summarised the results of the experiments using the global features to train the HMM models to recognise unconstrained handwritten month words. These results show that the parallel model seems to be a good alternative for our application providing a recognition rate of 82.5% and 96.4% for TOP1 and TOP3, respectively. Our next steps will be towards the combination of different architectures of HMMs and the implementation of a different feature set, which is based on concavities.

References

[1] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. of IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[2] X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh Univ. Press, 1990.

[3] A. El Yacoubi, R. Sabourin, M. Gilloux, and C. Y. Suen, "Off-line handwritten word recognition using hidden markov models," in *Knowledge Techniques in Character Recognition*, L.C. Jain and B. Lazzerini, Eds. CRC Press LLC, April 1999.

[4] F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, 1997.

[5] T. K. Moom, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, pp. 42–52, 1996.

[6] A. El Yacoubi, M. Gilloux, R. Sabourin, and C. Y. Suen, "An HMM-based approach for off-line unconstrained handwritten word modeling and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 752–760, 1999.