marcst_990304a.pdf

1D and Pseudo-2D Hidden Markov Models for Image Analysis.

Theoretical Introduction

Stéphane Marchand-Maillet - Multimedia Communications
Email: Stephane.Marchand@Eurecom.fr
Phone: +33 (0)4.93.00.26.79 - Fax: +33 (0)4.93.00.26.27

Date: March 4, 1999
Confidential: No

Eurécom's research is partially supported by its industrial members: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF,

EURECOM
Multimedia Communications

Institut EURECOM – BP 193.
06904 Sophia Antipolis – France
T.R. RR-99-49 Part A – March 4, 1999

1D and Pseudo-2D Hidden Markov Models for Image Analysis. Theoretical Introduction

Stéphane Marchand-Maillet

Stephane.Marchand@Eurecom.fr
http://www.eurecom.fr/~bmgroup/

Institut EURECOM
Department of Multimedia Communications
Technical Report RR-99-49 Part A
March 4, 1999

Abstract

This document presents a comprehensive approach to Hidden Markov Modelling. Concepts are first introduced in a one-dimensional context. Both model evaluation and training are considered and theoretical developments which lead to algorithms are presented. Major implementation issues are also addressed.

Then, results are extended to two-dimensional Hidden Markov Modelling. Again, model evaluation and training are considered and corresponding algorithms sketched. Proofs and detailed justifications of results presented can be found in the referenced bibliography.

This report is the first part of a three-fold document. Part Two [8] details the implementation of the tools presented in this document and part Three [9] specialises in applying these procedures to colour image analysis.

Résumé

Ce document présente une approche détaillée des Modèles de Markov Cachés. La théorie relative à cet outil est d'abord traitée à travers un modèle unidimensionnel. Les techniques d'évaluation du modèle par rapport aux données (reconnaissance) et le problème de l'entraînement sont détaillés. Les algorithmes sous-jacents sont aussi mis en évidence et les problèmes pratiques rencontrés classiquement traités.

Cette approche est alors étendue au cas bidimensionnel. Les formules et les algorithmes sont adaptés dans ce contexte. Pour une justification détaillée de ces résultats, le lecteur est renvoyé à la littérature citée en référence.

Ce rapport constitue la première partie d'un rapport qui en contient trois. La deuxième partie [8] détaille l'implémentation faite des outils détaillés dans ce document et la troisième partie [9] résume les résultats obtenus en appliquant ce type d'outils pour l'analyse d'images couleurs.

Keywords: Image analysis, Video analysis, Stochastic Modelling, Hidden Markov Models

Stéphane Marchand-Maillet. *1D and Pseudo-2D Hidden Markov Models for Image Analysis. Theoretical Introduction.* Tech. Rep. RR-99-49 Part A, Institut EURECOM, Multimedia Communications, March 4, 1999.

Contents

Introduction

In this report, we introduce the use of Hidden Markov Models (HMM) for 2D image segmentation. Most segmentation techniques based their performances on similarity measures. In our work, we aim to define a model-based segmentation technique whereby specific features are extracted from the image under investigation.

We mostly based this study on the example of face localisation in colour images where the part of the image containing the face is located using a pre-defined face model.

Hidden Markov Models [10, 11] are stochastic models which provide a high level of flexibility for modelling the structure of an observation sequence. They allow for recovering the (hidden) structure of a sequence of observations by pairing each observation with a (hidden) state. State duration is left free so that HMM represent a powerful technique for realising elastic matching when imposing constraints on the topology of state transitions. It is now acknowledged that the use of HMM is fundamental in automated speech analysis and recognition [7]. In this type of applications, the signal is mono-dimensional whereas pattern recognition in images requires two-dimensional operators. Nevertheless, mono-dimensional Hidden Markov Models (1DHMM) have been successfully applied to keyword spotting in binary document images [5]. The sequential aspect of written words is exploited (*i.e.*, from left to right). Each column of a word image is mapped onto a feature vector considered as a multi-dimensional observation. The sequence of such observations is then matched against different left-right models, each representing a keyword to be recognised. Recognition is based on the selection of the model of keyword that fits best the image in question. An extension of this work is found in [6]. The two-dimensional structure of the image is accounted for using a *pseudo* two-dimensional model (Pseudo 2D HMM or P2DHMM). This extension is shown to add robustness of the detection system against variations of size and slant of the fonts present in the document image.

Similarly, the potential of HMM for performing face recognition is demonstrated in [14]. The idea is again to exploit the sequential (vertical) structure of a human face. The image is divided in a sequence of overlapping horizontal stripes and the sequence of these stripes (*e.g.*, eyes-nose-mouth) is labelled using a 1DHMM. Results reported indicate that the use of HMM provides a suitable alternative to techniques classically used for this type of applications.

In this report, the aim is to test the ability of HMM to perform an image segmentation task rather than presenting a formal image segmentation system. One-dimensional Hidden Markov Models first reviewed in Chapter 1 and extended in Chapter 2 to pseudo-dimensional models to take advantage of the specific context of image segmentation.

Chapter 1

One-dimensional models

1.1 Introduction

Hidden Markov Models are mostly used for modelling the creation of an observation sequence $O = \{o_1, \ldots, o_T\}$ over time (*i.e.*, from time $t = 1$ to $t = T$), where each item o_t in the sequence is taken from a set V (*i.e.*, the vocabulary). The main idea is to design a multi-state system (finite state machine, FSM) which outputs this sequence while being in a state q_t at a given time t. At each state, the system is designed to output a certain observation from the vocabulary with a likelihood given by the *output probabilities*. At each time step the system will switch from its current state to the next (possibly stay in the same state) with a *transition probability*. The given of the transition probabilities will therefore implicitly design the (hidden) structure of the system. The state of the system at time $t = 1$ is given by its *initial state probabilities*. Figure 1.1 shows an example of a Hidden Markov Model where the system can only either loop within the same state or switch to the next state but cannot go back to a former state. This type of HMM is referred to as left-right model or Bakis model.

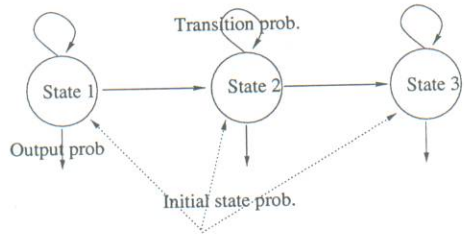

Figure 1.1: A left-right (Bakis) Hidden Markov Model.

A Hidden Markov Model can therefore conveniently be formalised via the following definition.

Definition 1 *Hidden Markov Model.*
A Hidden Markov Model can be formalised via the description $\lambda = \{S, V, A, B, \Pi\}$ *where,*

- $S = \{s_1, \ldots, s_N\}$ *is the set of* N *possible states.*
- $V = \{v_1, \ldots, v_L\}$ *is the output alphabet. In other words, for any* t*, there exist* l *such that* $o_t = v_l$.
- $A = \{a_{ij}\}_{ij=1\ldots N}$ *is the set of* transition probabilities.
- $B = \{b_i(l)\}_{i=1\ldots N, l=1\ldots L}$ *is the set of* output probabilities.
- $\Pi = \{\pi_1, \ldots, \pi_N\}$ *is the set of* initial state probabilities.

Over time, a Hidden Markov Model λ will output an observation sequence $O = \{o_1, \ldots, o_T\}$ while visiting states in the sequence $Q = \{q_1, \ldots, q_T\}$, where $q_t = s_i$[1] for some i at any time $t = 1, \ldots, T$, thus associating a hidden structure with O.

[1] Note that we will sometimes use the notation "$q_t = i$" for "$q_t = s_i$".

Remark 1 *Note that a unique observation sequence O may be generated by λ when visiting different state sequences $Q^{(1)}, \ldots, Q^{(G)}$.*

1.2 HMM parameters and model evaluation

In the context described above, details on the parameters of a Hidden Markov Model λ can be given.

Proposition 1 *Parameters of a Hidden Markov Model.*

- *A transition probability a_{ij} can be expressed as $a_{ij} = P[q_t = s_j | q_{t-1} = s_i]$[2], $1 \leq i, j \leq N$, for any $1 \leq t \leq T$.*
- *An output probability $b_i(l)$ can be expressed as $b_i(l) = P[o_t = v_l | q_t = s_i]$.*
- *Initial state probabilities are given by $\pi_i = P[q_1 = s_i]$.*

Since most of parameters above are probabilities, they impose some constraints given by,

$$\sum_{i=1}^{N} \pi_i = 1$$

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i = 1, \ldots, N$$

$$\sum_{l=1}^{L} b_i(l) = 1 \quad \forall i = 1, \ldots, N$$

This formalism now allows us to formulate the major HMM modelling problem. Given a Hidden Markov Model λ and an observation sequence O, evaluating the goodness of fit of λ with respect to O requires to calculating $P[O|\lambda]$. However, in the general case, the parameters of the model are unknown. There is therefore a need for initiating a training procedure. Following on the principle of model evaluation, training procedures will operate as follows. Starting with some given initial parameters and given a set of observation sequences $O^{(k)}$, $k = 1, \cdots, K$ new parameters will be iteratively computed so that $P[O^{(k)}|\lambda]$ is maximised. Two approaches are classically used and are presented in Section 1.2.1 and Section 1.2.2 respectively.

1.2.1 Baum-Welsh forward-backward algorithm

Following on Remark 1, a given observation sequence O may correspond to different state sequences $Q^{(1)}, \ldots, Q^{(G)}$. In other words, states sequences $\{Q^{(g)}\}$ are all the states sequences Q for which $P[O|Q, \lambda] \neq 0$. In that case,

$$P[O|\lambda] = \sum_{g=1}^{G} P[O|Q^{(g)}, \lambda] P[Q^{(g)}|\lambda]$$

However, direct computation of that equation would be much too long. The Forward-Backward approach proposes to split this calculation in two parts so that complexity is optimally reduced.

Forward procedure

This procedure uses a dynamic programming approach for computing the probability of generating an observation sequence up to a certain stage. Such a probability readily suggests the definition of forward variables.

[2] Unless necessary for comprehension, λ, the representation of the Hidden Markov Model in the probabilities will be dropped.

Definition 2 *Forward variable.*
Given a Hidden Markov Model λ, and an observation sequence O, the probability of getting the partial observation sequence $\{o_1, \ldots, o_t\}$ with being in state s_i at time t is noted $\alpha_t(i)$ and is called the forward variable. *More formally,*

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

The definition of the forward variable then allows for efficient computation using the following dynamic programming procedure. By definition, we clearly have,

$$\alpha_1(i) = \pi_i b_i(o_1)$$

Now, for any $t = 2, 3, \ldots, T$,

$$\begin{aligned}
\alpha_t(i) &= P[o_1, \ldots, o_t, q_t = s_i|\lambda] \\
&= P[o_t|q_t = s_i] \sum_{j=1}^{N} P[o_1, \ldots, o_{t-1}, q_{t-1} = s_j|\lambda] P[q_t = s_i|q_{t-1} = s_j] \\
&= b_i(o_t) \sum_{j=1}^{N} \alpha_{t-1}(j) a_{ji}
\end{aligned}$$

We finally obtain[3],

$$P[O|\lambda] = \sum_{i=1}^{N} P[o_1, \ldots, o_T, q_T = s_i|\lambda] = \sum_{i=1}^{N} \alpha_T(i)$$

Backward procedure

Although not needed for model evaluation, the Backward procedure is essential for training as will become apparent in Section 1.3. Typically for the generation of a given observation sequence O, every forward variable $\alpha_t(i)$ is paired with a backward variable $\beta_t(i)$.

Definition 3 *Backward variable.*
Given a Hidden Markov Model λ, and an observation sequence O, the probability of getting the partial observation sequence $\{o_{t+1}, \ldots, o_T\}$ with being in state s_i at time t is noted $\beta_t(i)$ and is called the backward variable. *More formally,*

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

A dynamic programming approach is again used for the computation of backward variables. We set arbitrarily,

$$\beta_T(i) = 1$$

Now, for any $t = T-1, T-2, \ldots, 1$,

$$\begin{aligned}
\beta_t(i) &= P[o_{t+1}, \ldots, o_T|q_t = s_i, \lambda] \\
&= \sum_{j=1}^{N} P[o_{t+2}, \ldots, o_T|q_{t+1} = s_j, \lambda] P[q_{t+1} = s_j|q_t = s_i] P[o_{t+1}|q_{t+1} = s_j] \\
&= \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1})
\end{aligned}$$

Following the definitions of forward and backward variables, we can define the posteriori variable $\gamma_t(i)$ as follows.

$$\gamma_t(i) = P[q_t = s_i|O, \lambda]$$

[3] Note that this formula is valid in the case of an ergodic model only. In that case, no constraints are given on the state of the HMM during its final step (see Section 1.4 for details).

$$
\begin{aligned}
&= \frac{P[O, q_t = s_i|\lambda]}{P[O|\lambda]} \\
&= \frac{P[o_1, \ldots, o_t, q_t = s_i|\lambda] P[o_{t+1}, \ldots, o_T|q_t = s_i, \lambda]}{\sum_{j=1}^{N} P[O, q_t = s_j|\lambda]} \\
&= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}
\end{aligned}
$$

Therefore $\gamma_t(i)$ is the probability of being in state s_i at time t when generating the sequence observation O with model λ. This variable is used for computing the most likely state sequence associated with an observation sequence and therefore proves fundamental when solving the recognition problem.

1.2.2 Viterbi algorithm

This procedure for model evaluation is a simplification of the above procedure. It focuses in finding the most likely state sequence $Q^* = \{q_1^*, \ldots, q_T^*\}$ by which a given observation sequence O has been generated. The corresponding dynamic programming procedure (called Viterbi algorithm) is as follows. At each time step t we calculate the value,

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

for every state s_i in the model. By induction,

$$\delta_{t+1}(i) = b_i(o_{t+1}) \max_{j=1\ldots N} (\delta_t(j) a_{ji})$$

The maximal probability will therefore be given by,

$$P[O|\lambda] = \max_{i=1\ldots N} \delta_T(i)$$

In order to be able to retrieve the sequence Q^* leading to this value, the best path generated by this maximisation is stored in the array $\phi_t(i)$.

The complete procedure is therefore given as follows.

1. $\delta_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N$
2. $\delta_t(i) = b_i(o_t) \max\{\delta_{t-1}(j) a_{ji} \ ; \ j = 1 \ldots N\} \quad 1 \leq i \leq N \quad 2 \leq t \leq T$
3. $\phi_t(i) = \arg\max\{\delta_{t-1}(j) a_{ji} \ ; \ j = 1 \ldots N\} \quad 1 \leq i \leq N \quad 2 \leq t \leq T$
4. $P[O|\lambda] = \max\{\delta_T(i) \ ; \ i = 1 \ldots N\}$
5. $q_T^* = \arg\max\{\delta_T(i) \ ; \ i = 1 \ldots N\}$

The best state sequence is then retrieved by,

$$q_t^* = \phi(q_{t+1}^*) \qquad t = T-1, \cdots, 1$$

Remark 2 *It should be noted that Viterbi algorithm is similar to Baum-Welsh algorithm where variables $\gamma_t(i)$ degenerate to 0's and 1's.*

1.3 Training

Training allows for automatically and optimally adjusting a given model to a particular type of problem. The problem of deciding of the best model structure is left open. A data set (the training set) is used for estimating models parameters. The training process of a given model λ is as follows. For each observation O sequence in the training set, we calculate $P[O|\lambda]$ and $P[O, q_t = s_i|\lambda]$ at every time step and for each state. This can be done using either of the above procedures. From these estimations, model parameters will be re-estimated by simply "counting"

event that occurred when (re-)generating the training set. For example an new estimate $\overline{a}_{ij}$ for the transition probability a_{ij} is given by:

$$\overline{a}_{ij} = \frac{\text{expected number of transitions from } s_i \text{ to } s_j}{\text{expected number of transitions from } s_i} = \frac{\sum_{t=1}^{T-1} P[q_t = s_i, q_{t+1} = s_j | O, \lambda]}{\sum_{t=1}^{T-1} P[q_t = s_i | O, \lambda]}$$

Depending on whether Baum-Welsh procedure is used or not, forward and backward variables or simple counting will be used. Next sections present re-estimation formulas when Baum-welsh procedure is used. These formulas are first presented in the case where the vocabulary V is finite and then extended to the case where V is continuous (or very large).

1.3.1 Discrete output sequence

Let us suppose first that we use the observation sequence O for training the Hidden Markov Model and that each o_t is taken from a finite vocabulary set V. We search λ such that the probability of the model to output the sequence O is maximised. We start with initial given parameters A, B and Π for λ. S and V are fixed prior to training in the design of the model.

An iterative re-estimation is then undertaken with the knowledge of O. We consider Baum-Welsh training procedure which maximises the probability of O among all possible state sequences Q (as opposed to Viterbi algorithm which finds the state sequence which maximises the probability of O). In that case, re-estimation formulas are given by,

$$\overline{\pi}_i = \frac{P[O, q_1 = s_i | \lambda]}{P[O|\lambda]} = \frac{\alpha_1(i)\beta_1(i)}{\sum_{j=1}^{N} \alpha_T(j)} = \gamma_1(i)$$

$$\overline{a}_{ij} = a_{ij} \frac{\sum_{t=1}^{T-1} \alpha_t(i) b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j'=1}^{N} a_{ij'} \sum_{t=1}^{T-1} \alpha_t(i) b_{j'}(o_{t+1}) \beta_{t+1}(j')} = a_{ij} \frac{\sum_{t=1}^{T-1} \alpha_t(i) b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T} \alpha_t(i) \beta_t(i)}$$

$$\overline{b}_i(l) = \frac{\sum_{t=1; o_t = v_l}^{T} \alpha_t(i)\beta_t(i)}{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)}$$

Remark 3 *Note that these coefficients are normalised so that they satisfy the probability constraints given earlier.*

For obtaining accurate estimates of the parameters above, the training is to be done with a set of observation sequences $\{O^{(1)}, \ldots, O^{(K)}\}$. Re-estimation formulas are generalised as,

$$\overline{\pi}_i = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \alpha_1^{(k)}(i) \beta_1^{(k)}(i)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{j=1}^{N} \alpha_T^{(k)}(j)},$$

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} a_{ij} \sum_{t=1}^{T-1} \alpha_t^{(k)}(i) b_j(o_{t+1}^{(k)}) \beta_{t+1}^{(k)}(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T} \alpha_t^{(k)}(i) \beta_t^{(k)}(i)}$$

$$\overline{b}_i(l) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1; o_t = v_l}^{T} \alpha_t^{(k)}(i) \beta_t^{(k)}(i)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T} \alpha_t^{(k)}(i) \beta_t^{(k)}(i)}$$

where, $\alpha_t^{(k)}(i)$ and $\beta_t^{(k)}(i)$ are calculated using formulas above with the training observation sequence $O^{(k)} = \{o_1^{(k)}, \ldots, o_T^{(k)}\}$ and $P_k = P[O^{(k)}|\lambda]$. It has been shown that such a iterative procedure converges to a fixed point with increasing at each iteration $P_k = P[O^{(k)}|\lambda]$, for all k.

1.3.2 Continuous output sequence

In the case where each o_t is taken from a continuous distribution (*i.e.*, the vocabulary has an infinite size), some extra parameters are to be introduced. We model the output probability distribution (which is continuous in this case) by a Gaussian mixture with a diagonal covariance matrix. More

formally,

$$b_i(o_t) = \sum_{m=1}^{M} c_{im}\mathcal{N}(o_t, \mu_{im}, \sigma_{im}) = \sum_{m=1}^{M} c_{im}\Pi_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma_{im}^{(d)}}} \exp\left(\frac{-(o_t^{(d)} - \mu_{im}^{(d)})^2}{2\sigma_{im}^{(d)}}\right),$$

where D is the dimension of the output item o_t and M is the number of mixtures in the Gaussian distribution. Corresponding constraints are imposed by

$$\int_{-\infty}^{\infty} b_i(o)do = 1 \quad \forall i = 1, \ldots, N$$

Therefore translating into,

$$\sum_{m=1}^{M} c_{im} = 1 \quad \forall i = 1, \ldots, N$$

and,

$$c_{im} \geq 0 \quad \forall i = 1, \ldots, N \quad m = 1, \ldots, M$$

In that case, the parameter $b_i(o_t)$ is re-estimated via the recalculation of c_{im}, μ_{im} and σ_{im} for $i = 1, \ldots, N$ and $m = 1, \ldots, M$. Re-estimation formulas are given by

$$\overline{c}_{im} = \frac{\sum_{t=2}^{T}\sum_{j=1}^{N} P[o_1, \ldots, o_{t-1}, q_{t-1} = s_j|\lambda]P[q_t = s_i|q_{t-1} = s_j]c_{im}\mathcal{N}(o_t, \mu_{im}, \sigma_{im})P[o_{t+1}, \ldots, o_T, q_t = s_i|\lambda]}{\Psi},$$

where Ψ is the appropriate normalisation factor for the constraints above to be respected (*i.e.*, Ψ is the sum over all mixtures m of the numerator of the above expression). Therefore, when including multiple observation sequence training,

$$\overline{c}_{im} = \frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=2}^{T} c_{im}\mathcal{N}(o_t^{(k)}, \mu_{im}, \sigma_{im})\beta_t^{(k)}(i)\sum_{j=1}^{N} a_{ji}\alpha_{t-1}^{(k)}(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{m'=1}^{M} c_{im'}\sum_{t=2}^{T}\mathcal{N}(o_t^{(k)}, \mu_{im'}, \sigma_{im'})\beta_t^{(k)}(i)\sum_{j=1}^{N} a_{ji}\alpha_{t-1}^{(k)}(j)}$$

Similarly,

$$\overline{\mu}_{im} = \frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=2}^{T} o_t^{(k)}\mathcal{N}(o_t^{(k)}, \mu_{im}, \sigma_{im})\beta_t^{(k)}(i)\sum_{j=1}^{N} a_{ji}\alpha_{t-1}^{(k)}(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=2}^{T}\mathcal{N}(o_t^{(k)}, \mu_{im}, \sigma_{im})\beta_t^{(k)}(i)\sum_{j=1}^{N} a_{ji}\alpha_{t-1}^{(k)}(j)}$$

and,

$$\overline{\sigma}_{im} = \frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=2}^{T}(o_t^{(k)} - \mu_{im})^2\mathcal{N}(o_t^{(k)}, \mu_{im}, \sigma_{im})\beta_t^{(k)}(i)\sum_{j=1}^{N} a_{ji}\alpha_{t-1}^{(k)}(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=2}^{T}\mathcal{N}(o_t^{(k)}, \mu_{im}, \sigma_{im})\beta_t^{(k)}(i)\sum_{j=1}^{N} a_{ji}\alpha_{t-1}^{(k)}(j)}$$

Re-estimation formulas for initial state and transition probabilities are clearly not influenced by the form of the output.

Formulas above can be better understood if state s_i is split into two sub-states $s_{i_{\text{init}}}$ and $s_{i_{\text{final}}}$ (Figure 1.2). Between states $s_{i_{\text{init}}}$ and $s_{i_{\text{final}}}$ the system goes through states s_{im} with probabilities c_{im} and outputs a component of the global output o_t via the Gaussian distribution $\mathcal{N}(o_t, \mu_{im}, \sigma_{im})$. Transition probabilities from a state s_{im} to state $s_{i_{\text{final}}}$ are all set to 1.

1.4 Implementation issues

Besides programming structures, the main issue in the implementation of Hidden Markov Models is the fact that the recursive formulas given in the forward and backward procedures typically multiply probability values (therefore lower than 1) between themselves. Hence, this product quickly tends to zero and goes beyond any machine storage capabilities. The idea is therefore to define a scaling procedure which will allow to obtain equivalent re-estimations while keeping values in a reasonable order of magnitude. At each step of the forward procedure, forward variables $\alpha_t(i)$

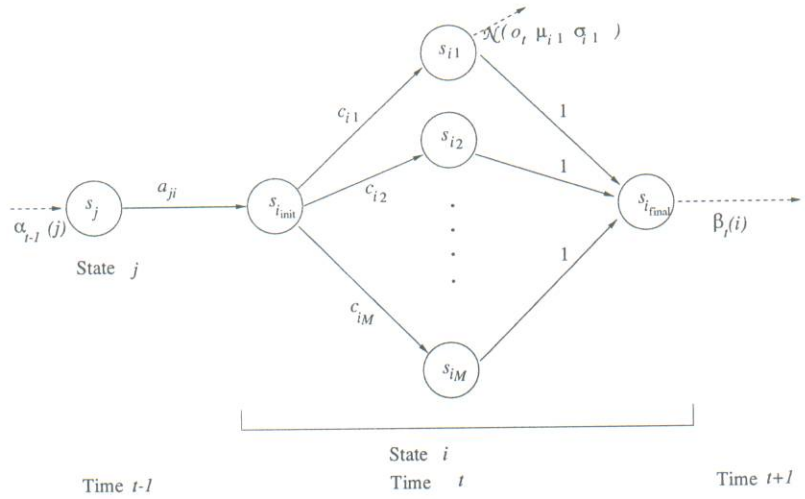

Figure 1.2: State partition.

are divided by their sum. Thus defining scaled variable $\tilde{\alpha}_t(i)$ and $\hat{\alpha}_t(i)$ as follows.

$$\begin{array}{lll}
\alpha_1(i) = \pi_i b_i(o_1) & \tilde{\alpha}_1(i) = \alpha_1(i) & \hat{\alpha}_1(i) = \frac{\tilde{\alpha}_1(i)}{\sum_{j=1}^N \tilde{\alpha}_1(j)} \\
 & \tilde{\alpha}_2(i) = b_i(o_2) \sum_{j=1}^N \hat{\alpha}_1(j) a_{ji} & \hat{\alpha}_2(i) = \frac{\tilde{\alpha}_2(i)}{\sum_{j=1}^N \tilde{\alpha}_2(j)} \\
 & \vdots & \vdots \\
 & \tilde{\alpha}_t(i) = b_i(o_t) \sum_{j=1}^N \hat{\alpha}_{t-1}(j) a_{ji} & \hat{\alpha}_t(i) = \frac{\tilde{\alpha}_t(i)}{\sum_{j=1}^N \tilde{\alpha}_t(j)}
\end{array}$$

Simple calculations show that

$$\tilde{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_{j=1}^N \alpha_{t-1}(j)} \quad i = 2 \ldots N \text{ and } \hat{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_{j=1}^N \alpha_t(j)} \quad i = 1 \ldots N$$

Defining

$$z_t = \sum_{i=1}^N \tilde{\alpha}_t(i) = \frac{\sum_{i=1}^N \alpha_t(i)}{\sum_{i=1}^N \alpha_{t-1}(i)} \quad i = 2 \ldots N \text{ and } z_1 = \sum_{i=1}^N \alpha_1(i)$$

We obtain,

$$\Pi_{t'=1}^t z_{t'} = \sum_{i=1}^N \alpha_t(i)$$

Backward variables are also scaled using the *same* scale factors z_t as in the forward procedure. Scaled forward and backward variables are therefore defined as,

$$\hat{\alpha}_t(i) = \frac{\alpha_t(i)}{\Pi_{t'=1}^t z_t} \text{ and } \hat{\beta}_t(i) = \frac{\beta_t(i)}{\Pi_{t'=t}^T z_t}.$$

In this manner, values of $\hat{\alpha}_t(i)$ and $\hat{\beta}_t(i)$ are kept close to 1. Moreover, this can be efficiently included in the recursive forward and backward procedures by dividing at each step $\alpha_t(i)$ and $\beta_t(i)$ by z_t.

Let us define

$$Z = \Pi_{t=1}^T z_t.$$

Clearly, $\alpha_T(i) = Z\hat{\alpha}_T(i)$, $\beta_1(i) = Z\hat{\beta}_1(i)$ and, for any $t = 1, \ldots, T-1$, $\alpha_t(i)\beta_{t+1}(i) = Z\hat{\alpha}_t(i)\hat{\beta}_{t+1}(i)$. Now, in the general case of an ergodic model, $P_k = P[O^{(k)}|\lambda] = \sum_{i=1}^N \alpha_T^{(k)}(i)$. Therefore, using scaled variables,

$$P_k = Z^{(k)},$$

where, $Z^{(k)} = \Pi_{t=1}^{T} z_t^{(k)}$ and $z_t^{(k)} = \sum_{i=1}^{N} \alpha_t^{(k)}(i)$. It is therefore clear that introducing scale factors in the unscaled formulas will remove factors $\frac{1}{P_k}$ since factors $Z^{(k)}$ will cancel out in all formulas.

In the case where extra constraints are given on the state of the system at specific steps, these formulas are to be adapted. For example, in the case of a left-right (Bakis) model presented in Figure 1.1, a constraint imposes

$$q_T = s_N$$

Therefore the value for the assessment of a model is

$$P_k = P[O^{(k)}, q_T = s_N|\lambda] = \alpha_T(N) = \hat{\alpha}_T^{(k)}(N) Z^{(k)}$$

In that case a factor $\frac{1}{\hat{\alpha}_T^{(k)}(N)}$ will replace the factor $\frac{1}{P_k}$ in scaled re-estimation formulas. Again, this is conveniently implemented since scaled variables $\hat{\alpha}_t^{(k)}(i)$ are easy to handle, by definition.

Following this example, different constraints can be designed for giving a specific structure to a model λ.

Chapter 2

Two-dimensional models

2.1 Introduction

The aim of this chapter is to generalise previous one-dimensional models in order to handle two-dimensional data. Typically, causality is introduced in two-dimensional data by defining a dependency between neighbouring items. For instance, consider a set of data O organised in a matrix-like manner

$$O = \begin{pmatrix} o_{11} & \cdots & o_{x1} & \cdots & o_{X1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ o_{1y} & \cdots & o_{xy} & \cdots & o_{Xy} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ o_{1Y} & \cdots & o_{xY} & \cdots & o_{XY} \end{pmatrix}.$$

Causality can be expressed as the dependency between an item o_{xy} and its neighbours in a line-scanning order. For example,

$$\begin{array}{ccccc} o_{x-1y-1} & & o_{xy-1} & & o_{x+1y-1} \\ & \searrow & \downarrow & \swarrow & \\ o_{x-1y} & \rightarrow & o_{xy} & & \end{array}$$

However, a fully connected two-dimensional structure for a HMM leads to NP-hard problems for updating parameters (training) and recovering the structure of a new 2D observation sequence (recognition).

The approach taken is therefore to introduce two-dimensional causality at the line level. In other words, a line $O_y = \{o_{1y}, \ldots, o_{Xy}\}$ is now considered as an observation and the sequence $O = \{O_1, \ldots, O_Y\}$ is also modelled by an upper-level HMM (called P2DHMM, for Pseudo-2D HMM), just as every line has been modelled by a 1DHMM in Chapter 1. The structure of a P2DHMM modelling a matrix-like two-dimensional observation sequence will therefore be composed of different (horizontal) one-dimensional models (one for each possible type of line) connected by a global (vertical) model representing relationships between lines. Figure 2.1 shows an instance of P2DHMM which is an extension of the simple 1DHMM shown in Figure 1.1.

Extending the notation proposed in Definition 1, a Pseudo 2-Dimensional HMM can be represented by the following parameters.

Definition 4 *2-Dimensional Hidden Markov Model.*

A Pseudo 2-Dimensional Hidden Markov Model can be formalised via the description $\Lambda = \{\lambda, A, B, \Pi\}$, *where,*

- $\lambda = \{\lambda^i \ ; \ i = 1, \cdots, N\}$ *is the set of* N *possible super-states in the model. Each of these super-states is a 1DHMM* λ^i *whose parameters are* $\lambda^i = \{s^i, V, A^i, \pi^i\}$, *following Definition 1. In other words,*
  - $s^i = \{s^i_1, \ldots, s^i_{N^i}\}$ *is the set of* N^i *possible states of super-state* λ^i.

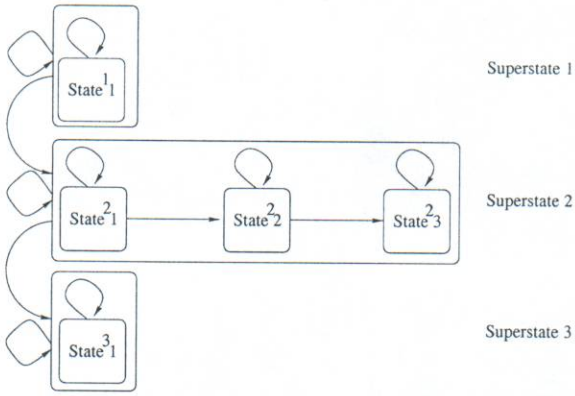

Figure 2.1: A Pseudo-2D Hidden Markov Model.

- $V = \{v_1, \ldots, v_L\}$ *is the output alphabet (common to all super-states). In other words, for any* t*, there exist* l *such that* $o_t = v_l$.
- $A^i = \{a^i_{kl}\}_{kl=1\ldots N^i}$ *is the set of* transition probabilities *within super-state* λ^i.
- $B^i = \{b^i_k(l)\}_{k=1\ldots N^i, l=1\ldots L}$ *is the set of* output probabilities *of super-state* λ^i.
- $\pi^i = \{\pi^i_1, \ldots, \pi^i_{N^i}\}$ *is the set of* initial state probabilities *of super-state* λ^i.

- $A = \{a_{ij}\}_{ij=1\ldots N}$ *is is the set of* transition probabilities *within the P2DHMM.*
- $\Pi = \{\pi_1, \ldots, \pi_N\}$ *is the set of* initial super-state probabilities *of the P2DHMM.*

Like in the one-dimensional model, the Pseudo two-dimensional Hidden Markov Model will associate a state sequence Q to an observation sequence $O = \{o_{xy}\}_{x=1\cdots X, y=1\cdots Y}$. The state sequence Q will be composed of two levels. Q is primarily a super-state sequence $Q = \{Q_1, \ldots, Q_Y\}$ indicating the super-state corresponding to the sequence of lines of observation $O = \{O_1, \ldots, O_Y\}$ Each state line Q_y is then composed of states q_{xy} ($Q_y = \{q_{1y}, \ldots, q_{Xy}\}$) indicating the state of the corresponding 1DHMM at a position (x, y).

In summary, a P2DHMM consists in two 1DHMM embedded at different levels. In this context, 2D-causality will only be modelled at line level (as opposed to pixel level).

2.2 P2DHMM parameters and model evaluation

More formally, the parameters of a P2DHMM can be expressed as follows.

- Super-state transition probability: $a_{ij} = P[q_y = \lambda^j | q_{y-1} = \lambda^i]$.
- Initial super-state probability: $\pi_i = P[Q_1 = \lambda^i | \Lambda]$.
- State transition probability of super-state λ^i: $a^i_{kl} = P[q_{xy} = s^i_l | q_{x-1y} = s^i_k]$
- State output probability of super-state λ^i: $b^i_j(l) = P[o_{xy} = v_l | q_{xy} = s^i_j]$
- Initial state probability: $\pi^i_j = P[q_{1y} = s^i_j | \lambda^i]$.

Stochastic constraints similar to that imposed on the parameters of a 1DHMM are to be fulfilled in the P2DHMM.

Following on the structure of a P2DHMM, model evaluation will be achieved using two nested one-dimensional evaluations. Both Baum-Welsh and Viterbi procedures are available and will both work on the following principle. One-dimensional (horizontal) model evaluation is first performed with respect to each line O_y of observations so that the values $P[O_y | Q_y = \lambda^i]$ can be computed. These values are then taken as output probabilities in the super-state of the upper-level HMM. Another one-dimensional (vertical) model evaluation is then initiated to calculate values $P[Q_y = \lambda^i | \Lambda]$ for every line O_y and every super-state λ^i. These values finally become weights for the calculation of $P[q_{xy} = s^i_j | \lambda^i]$.

2.2.1 Baum-Welch procedure

Forward and backward variables are defined at both levels. The notation is extended as follows.

- $\alpha^i_{xy}(k) = P[o_{1y}, \ldots, o_{xy}, q_{xy} = s^i_k | Q_y = \lambda^i, \Lambda]$.
- $\beta^i_{xy}(k) = P[o_{x+1y}, \ldots, o_{Xy}, q_{xy} = s^i_k | Q_y = \lambda^i, \Lambda]$.
- $\alpha_y(i) = P[O_1, \ldots, O_y, Q_y = \lambda^i | \Lambda]$.
- $\beta_y(i) = P[O_{y+1}, \ldots, O_Y, Q_y = \lambda^i | \Lambda]$.

Formulas and procedures for calculating $\alpha^i_{xy}(k)$ and $\beta^i_{xy}(k)$ are directly adapted from Chapter 1. In turn, they define super-state output probabilities as,

$$b_i(O_y) = P[O_y | Q_y = \lambda^i] = \sum_{j=1}^{N^i} \alpha^i_{Xy}(j).$$

Remark 4 *This formula is given in the most general case of an ergodic model. For a Bakis-like model,*

$$b_i(O_y) = P[O_y, q_{Xy} = s^i_{N^i} | Q_y = \lambda^i] = \alpha^i_{Xy}(N^i).$$

These values thus allow for the calculation of values $\alpha_y(i)$ and $\beta_y(i)$ as defined for 1DHMM while including relationships between lines of observations. At both level, probabilities of being in a certain (super-)state for a certain (line of) observation can also be calculated. More formally,

$$\gamma_y(i) = P[Q_y = \lambda^i | O, \Lambda] = \frac{\alpha_y(i)\beta_y(i)}{\sum_{j=1}^{N} \alpha_y(j)\beta_y(j)}$$

and

$$\gamma^i_{xy}(j) = P[q_{xy} = s^i_j | O_y, \lambda^i] = \frac{\alpha^i_{xy}(j)\beta^i_{xy}(j)}{\sum_{k=1}^{N^i} \alpha^i_{xy}(k)\beta^i_{xy}(k)}$$

2.2.2 Viterbi procedure

Similarly to one-dimensional model, this procedure searches for the most likely state sequence Q^* which corresponds to the given observation sequence O. Two embedded dynamic programming procedures are used. A first step calculates for each line of observations O_y and every possible super-state (*i.e.*, 1DHMM) λ^i the most likely state sequence $q_{1y}, \ldots, q_{Xy}$ through which O_y has been generated. The probability along each possible state sequence is stored using $\delta^i_{xy}(j)$ where,

$$\delta^i_{xy}(j) = \max_{q_{1y}, \cdots, q_{x-1y}} P[q_{1y}, \ldots, q_{x-1y}, q_{xy} = s^i_j, o_{1y}, \ldots, o_{xy} | \lambda^i]$$

The resulting optimal moves are stored in the array $\phi^i_{xy}(j)$ so that backtracking can be operated to retrieve this best path. This procedure is as follows.

- $\delta^i_{1y}(j) = \pi^i_j b^i_j(o_{1y})$,
  $1 \leq j \leq N^i, 1 \leq i \leq N, 1 \leq y \leq Y$
- $\delta^i_{xy}(j) = b^i_j(o_{xy}) \max\{\delta^i_{x-1y}(l) a^i_{lj} \ ; \ 1 \leq l \leq N^i\}$,
  $1 \leq j \leq N^i, 1 \leq i \leq N, 1 \leq y \leq Y, 1 < x \leq X$
- $\phi^i_{xy}(j) = \arg\max\{\delta^i_{x-1y}(l) a^i_{lj} \ ; \ 1 \leq l \leq N^i\}$,
  $1 \leq j \leq N^i, 1 \leq i \leq N, 1 \leq y \leq Y, 1 < x \leq X$

This allows for computing the value of $b_i(O_y) = P[O_y | Q_y = \lambda^i]$ as

$$b_i(O_y) = \max_{1 \leq j \leq N^i} \delta_{Xy}(j)$$

and to obtain the last state in each 1D optimal path using λ^i. This state is stored in $\chi^i(O_y)$ as

$$\chi^i(O_y) = \arg \max_{1 \leq j \leq N^i} \delta_{Xy}(j).$$

A second step finally determines the most likely sequence of super-states $Q_1, \ldots, Q_Y$ corresponding to the sequence of lines of observations $O_1, \ldots, O_Y$ (this corresponds to associating a super-state per line of observations). For each line O_y,

$$\delta_y(i) = \max_{Q_1,\ldots,Q_{y-1}} P[Q_1, \ldots, Q_{y-1}, Q_y = \lambda^i, O_1, \ldots, O_y|\Lambda].$$

This second procedure is as follows.

- $\delta_1(i) = b_i(O_1)$, $1 \leq i \leq N$.
- $\delta_y(i) = b_i(O_y) \max\{\delta_{y-1}(j)a_{ji} \ ; \ 1 \leq j \leq N\}$,
  $1 \leq i \leq N$, $2 \leq y \leq Y$.
- $\Phi_y(i) = \arg\max\{\delta_{y-1}(j)a_{ji} \ ; \ 1 \leq j \leq N\}$,
  $1 \leq i \leq N$, $2 \leq y \leq Y$.

Then,

$$P[O|\Lambda] = \max_{1 \leq i \leq N} \delta_Y(i).$$

State sequence backtracking is then done as follows.

- $Q_Y = \arg\max\{\delta_Y(i) \ ; \ 1 \leq i \leq N\}$
- For $y = Y - 1, \ldots, 1$:
  - $Q_y = \Phi_{y+1}(Q_{y+1})$
  - $q_{Xy} = \chi^{Q_y}(O_y)$
  - For $x = X - 1, \ldots, 1$:
    $q_{xy} = \phi_{xy}^{Q_y}(q_{x+1y})$

For model evaluation only the value of $P[O|\Lambda]$ is needed. Values $P[O_y|\lambda^i]$ can help is assessing further the results. States associated with each observation, be it through a probability (*i.e.*, $\gamma_{xy}^i(j)$) or a 0-1 value (Viterbi procedure) are needed for solving the recognition problem.

2.3 Training

The training of a P2DHMM will be based on essentially the same principle as the training of a 1DHMM described in Section 1.3. A set of two-dimensional observation sequences $\{O^{(1)}, \ldots, O^{(K)}\}$ is chosen. Using initial parameters, the goodness-of-fit of the model Λ is evaluated using either of the above procedures. While doing this, states are associated with each observation. Using Viterbi procedure, an observation $o_{xy}^{(k)}$ is associated with a state q_{xy} is the best state sequence. Simple statistics then allow for re-evaluating model parameters. Using Baum-Welsh procedure, each observation $o_{xy}^{(k)}$ is associated with every possible state s_j^i with a certain probability (*i.e.*, $P[q_{xy} = s_j^i|O_y, \lambda^i]$). The training procedure is then using these probabilities for re-estimating the parameters of the model at each iteration. Re-estimation formulas for state and super-state transition probabilities are as follows (we consider directly multiple sequence training).

From Section 1.3, it is clear that new estimates for state transition probabilities (and hence super-state transition probabilities) do not depend on whether training data is continuous or discrete. The same argument holds for initial (super-)state probabilities.

$$\overline{\pi}_i = \frac{\sum_{k=1}^K \frac{1}{P_k} \alpha_1^{(k)}(i)\beta_1^{(k)}(i)}{\sum_{k=1}^K \frac{1}{P_k} \sum_{j=1}^N \alpha_T^{(k)}(j)},$$

$$\overline{a}_{ij} = a_{ij} \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{y=1}^{Y-1} \alpha_y^{(k)}(i) b_j(O_{y+1}^{(k)}) \beta_{y+1}^{(k)}(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{y=1}^{Y} \alpha_y^{(k)}(i) \beta_y^{(k)}(i)},$$

where $P_k = P[O^{(k)}|\Lambda]$ and,

$$\overline{\pi}_j^i = \frac{\sum_{k=1}^{K} \sum_{y=1}^{Y} \frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})} \alpha_{1y}^{i(k)}(j) \beta_{1y}^{i(k)}(j)}{\sum_{k=1}^{K} \sum_{y=1}^{Y} \frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})} \sum_{l=1}^{N^i} \alpha_{1y}^{i(k)}(l) \beta_{1y}^{i(k)}(l)}$$

$$\overline{a}_{lm}^i = a_{lm}^i \frac{\sum_{k=1}^{K} \sum_{y=1}^{Y} \frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})} \sum_{x=1}^{X-1} \alpha_{xy}^{i(k)}(l) b_m^i(o_{xy+1}^{(k)}) \beta_{xy+1}^{i(k)}(m)}{\sum_{k=1}^{K} \sum_{y=1}^{Y} \frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})} \sum_{x=1}^{X-1} \alpha_{xy}^{i(k)}(l) \beta_{xy}^{i(k)}(l)}.$$

Extending one-dimensional re-estimation formulas to two-dimensional ones is done by adding a weight $\gamma_y^{(k)}(i)$ to each line in the re-estimation of the parameters of the super-state λ^i. This allows to bind lines together using the upper-level (vertical) model to express their inter-dependency.

Depending on whether training data is discrete (*i.e.*, V is finite) or continuous (*i.e.*, V is infinite or, in practice very large), state output probabilities $b_j^i(o_{xy})$ will take different forms, as already detailed in Sections 1.3.1 and 1.3.2. The next sections briefly suggest extensions of these formulas.

2.3.1 Discrete output sequence

A code-book stores the vocabulary $V = \{v_1, \ldots, v_L\}$. Using this look-up table, one directly accesses the value of $b_j^i(l)$. Using again the principle described above, we obtain the new estimate of the output probability using forward and backward variables as,

$$\overline{b}_j^i(l) = \frac{\sum_{k=1}^{K} \sum_{y=1}^{Y} \frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})} \sum_{x=1;o_t=v_l}^{X} \alpha_{xy}^{i(k)}(j) \beta_{xy}^{i(k)}(j)}{\sum_{k=1}^{K} \sum_{y=1}^{Y} \frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})} \sum_{x=1}^{X} \alpha_{xy}^{i(k)}(j) \beta_{xy}^{i(k)}(j)}$$

When using Viterbi procedure, simple statistics allow for re-estimating this value.

2.3.2 Continuous output sequence

When output probabilities are taken from a continuous distribution, modelling is done as in Section 1.3.2. In this case, for every state s_j^i of super-state λ^i we have,

$$b_j^i(o_{xy}) = \sum_{m=1}^{M} c_{jm}^i \mathcal{N}(o_{xy}, \mu_{jm}^i, \sigma_{jm}^i) = \sum_{m=1}^{M} c_{jm}^i \Pi_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma_{jm}^{(d,i)}}} \exp\left(\frac{-(o_{xy}^{(d)} - \mu_{jm}^{(d,i)})^2}{2\sigma_{jm}^{(d,i)}} \right),$$

where D is the dimension of the output item o_{xy} and M is the number of mixtures in the Gaussian distribution. Parameter M can be different from state to state, depending on what this distribution models. To be complete, one should replace M by M_j^i for the number of mixtures in the output distribution of state s_j^i of super-state λ^i. However, since re-calculation is independent from this value and for the sake of clarity, only M is used. In this context, new estimates for parameters of the output probability distribution are,

$$\overline{c}_{jm}^i = \frac{\sum_{k=1}^{K} \sum_{y=1}^{Y} \frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})} \sum_{x=2}^{X} c_{jm}^i \mathcal{N}(o_{xy}^{(k)}, \mu_{jm}^i, \sigma_{jm}^i) \beta_{xy}^{i(k)}(j) \sum_{l=1}^{N^i} a_{lj}^i \alpha_{x-1y}^{i(k)}(l)}{\sum_{k=1}^{K} \sum_{y=1}^{Y} \frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})} \sum_{m'=1}^{M} c_{jm'}^i \sum_{x=2}^{X} \mathcal{N}(o_{xy}^{(k)}, \mu_{jm'}^i, \sigma_{jm'}^i) \beta_{xy}^{i(k)}(j) \sum_{l=1}^{N^i} a_{lj}^i \alpha_{x-1y}^{i(k)}(l)}$$

Similarly,

$$\overline{\mu}^i_{jm} = \frac{\sum_{k=1}^{K}\sum_{y=1}^{Y}\frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})}\sum_{x=2}^{X} o_{xy}^{(k)}\mathcal{N}(o_{xy}^{(k)},\mu^i_{jm},\sigma^i_{jm})\beta_{xy}^{i(k)}(j)\sum_{l=1}^{N^i} a^i_{lj}\alpha_{x-1y}^{i(k)}(l)}{\sum_{k=1}^{K}\sum_{y=1}^{Y}\frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})}\sum_{x=2}^{X} \mathcal{N}(o_{xy}^{i(k)},\mu^i_{jm},\sigma^i_{jm})\beta_{xy}^{i(k)}(j)\sum_{l=1}^{N^i} a^i_{lj}\alpha_{x-1y}^{i(k)}(l)}$$

and,

$$\overline{\sigma}^i_{jm} = \frac{\sum_{k=1}^{K}\sum_{y=1}^{Y}\frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})}\sum_{x=2}^{X} (o_{xy}^{(k)}-\mu^i_{jm})^2\mathcal{N}(o_{xy}^{(k)},\mu^i_{jm},\sigma^i_{jm})\beta_{xy}^{i(k)}(j)\sum_{l=1}^{N^i} a^i_{lj}\alpha_{x-1y}^{i(k)}(l)}{\sum_{k=1}^{K}\sum_{y=1}^{Y}\frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})}\sum_{x=2}^{X} \mathcal{N}(o_{xy}^{(k)},\mu^i_{jm},\sigma^i_{jm})\beta_{xy}^{i(k)}(j)\sum_{l=1}^{N^i} a^i_{lj}\alpha_{x-1y}^{i(k)}(l)}$$

Remark 5 *In the above development, covariance matrices of Gaussian mixture components are considered as being diagonal. In speech recognition where HMM are widely used, components of feature vectors are de-correlated so that this approximation is valid. For the sake of generality, we will consider mixtures of Gaussian distribution will full covariance matrices Σ_{jm}. That is,*

$$b^i_j(o_{xy}) = \sum_{m=1}^{M} c^i_{jm}\frac{1}{\sqrt{(2\pi)^D|\Sigma^i_{jm}|}}\exp\left(-\frac{1}{2}(o_{xy}-\mu^i_{jm})^{\mathsf{T}}(\Sigma^i_{jm})^{-1}(o_{xy}-\mu^i_{jm})\right). \qquad (2.1)$$

Re-estimation formulas for Σ^i_{jm}, the covariance matrix associated with mixture m of state s^i_j of super-state λ^i is

$$\overline{\Sigma}^i_{jm} = \frac{\sum_{k=1}^{K}\sum_{y=1}^{Y}\frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})}\sum_{x=2}^{X} (o_{xy}^{(k)}-\mu^i_{jm})(o_{xy}^{(k)}-\mu^i_{jm})^{\mathsf{T}}\mathcal{N}(o_{xy}^{(k)},\mu^i_{jm},\sigma^i_{jm})\beta_{xy}^{i(k)}(j)\sum_{l=1}^{N^i} a^i_{lj}\alpha_{x-1y}^{i(k)}(l)}{\sum_{k=1}^{K}\sum_{y=1}^{Y}\frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})}\sum_{x=2}^{X} \mathcal{N}(o_{xy}^{(k)},\mu^i_{jm},\sigma^i_{jm})\beta_{xy}^{i(k)}(j)\sum_{l=1}^{N^i} a^i_{lj}\alpha_{x-1y}^{i(k)}(l)}$$

2.3.3 Multiple stream observation

It can be the case that an observation is composed of a combination of different information streams. In the case of an image for example, one can associate each pixel with a vector of observations which contain *e.g.*, RGB components and values of the gradients at that pixel. It is clear that RGB components and gradient values can be considered as independent and therefore to follow their own PDF. Moreover, from prior knowledge, one may allocate weights to different part of the observation vector so that the influence of each part can be modulated.

For modelling this, we will define the observation vector o_{xy} at position (x, y) as being the superimposition of different multi-component streams of observations (*i.e.*, $o_{xy} = (o_{xy}^{(1)},\ldots,o_{xy}^{(S)})^{\mathsf{T}}$). The output of each stream $o_{xy}^{(s)}$ will be modelled by a Gaussian mixture $b^i_{js}(o_{xy}^{(s)})$ as given in Equation (2.1) above. The global output probability distribution at state s^i_j of super-state λ^i can then be modelled as the product of these stream PDF (independent successive PDF). In other words

$$b^i_j(o_{xy}) = \Pi_{s=1}^{S} b^i_{js}(o_{xy}^{(s)})$$

However, in that case, no weighting scheme is readily feasible on this model.

We present here a different approach which is similar to the justification of mixture weight re-estimation. Let S be the number of streams and w_s be stream weights such that $\sum_{s=1}^{S} = 1$. In this configuration, the output PDF at a state is considered as being modelled as if each stream s was produced with a certain probability w_s. Figure 2.2 shows conceptually how the model works.

State s^i_j of super-state λ^i is first partitioned into parallel sub-states represented by output PDF $b^i_{js}(o_{xy}^{(s)})$ where each produces the part of the observation corresponding to each stream (*i.e.*, $o_{xy}^{(s)}$). The global output PDF is therefore defined by the weighted sum of these stream output PDF. Now, each of the stream output PDF is a Gaussian mixture so that the scheme proposed in Figure 1.2 is applied again. This scheme means that the global output probability $b^i_j(o_{xy})$ will be calculated

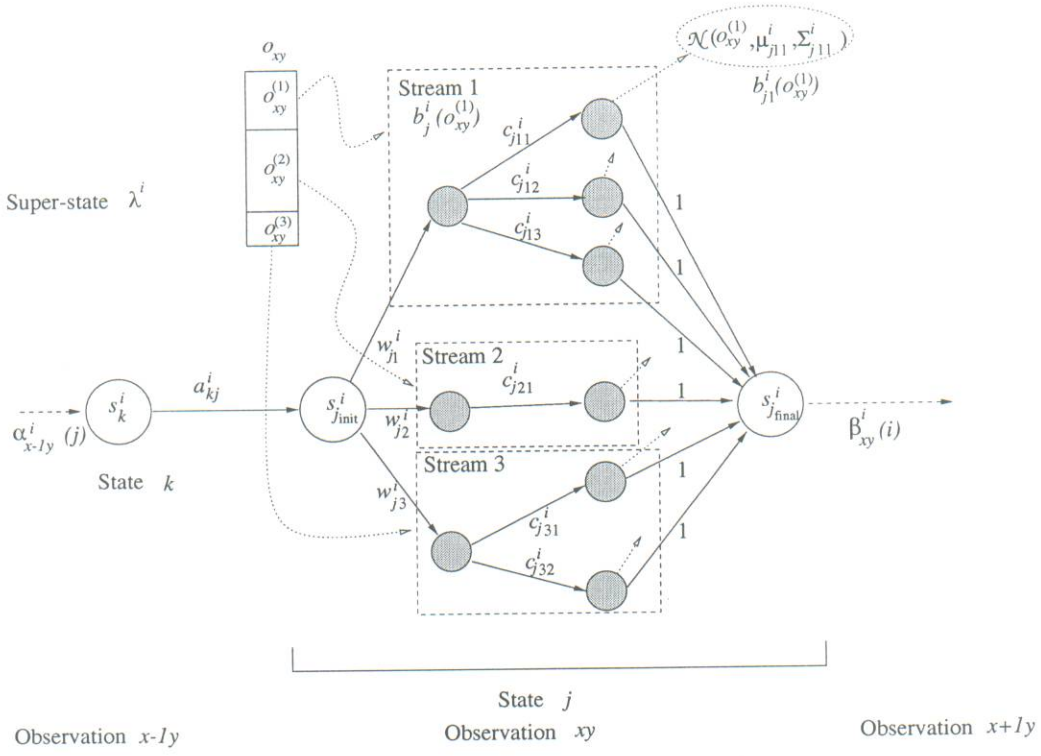

Figure 2.2: Model for multiple weighted stream output PDF.

as

$$b_j^i(o_{xy}) = \sum_{s=1}^{S} w_{js}^i b_{js}^i(o_{xy}^{(s)})$$

So that,

$$b_j^i(o_{xy}) = \sum_{s=1}^{S} w_{js}^i \sum_{m=1}^{M} c_{jsm}^i \frac{1}{\sqrt{(2\pi)^{D_s}|\Sigma_{jm}^i|}} \exp\left(-\frac{1}{2}(o_{xy}^{(s)} - \mu_{jsm}^i)^{\mathsf{T}}(\Sigma_{jsm}^i)^{-1}(o_{xy}^{(s)} - \mu_{jsm}^i)\right),$$

where D_s is the dimension of stream s. This model readily suggests re-estimation formulas for both stream weights and Gaussian mixture coefficients. Typically, a factor w_{js}^i will simply be added in front of each re-estimation formula. This is understood by the fact that w_{js}^i represents the probability of generating the complete observation vector o_{xy} by a PDF with infinite covariance between values which do not belong to stream s (*i.e.*, with the stream PDF $b_{js}^i(o_{xy}^{(s)})$ only). Equivalently, with the same token as in Section 1.3.2, one may consider that state s_j^i is split into different sub-state levels until reaching the level where a single Gaussian PDF will output a portion of the global observation corresponding to a given stream.

Re-estimation of stream weights is given by:

$$\overline{w}_{js}^i = \frac{\sum_{k=1}^{K}\sum_{y=1}^{Y}\frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})}\sum_{x=2}^{X} w_{js}^i b_{js}^i(o_{xy}^{(s)})\beta_{xy}^{i(k)}(j)\sum_{l=1}^{N^i} a_{lj}^i \alpha_{x-1y}^{i(k)}(l)}{\sum_{k=1}^{K}\sum_{y=1}^{Y}\frac{\gamma_y^{(k)}(i)}{b_i(O_y^{(k)})}\sum_{s'=1}^{S} w_{js'}^i \sum_{x=2}^{X} b_{js'}^i(o_{xy}^{(s')})\beta_{xy}^{i(k)}(j)\sum_{l=1}^{N^i} a_{lj}^i \alpha_{x-1y}^{i(k)}(l)}.$$

2.4 Implementation issues

Based on a similar argument as in Section 1.4, we emphasise the need for a scaling procedure in the actual computation of the parameters of a P2DHMM. Given a super-state λ^i and a line of observations O_y, the value $P[O_y|\lambda^i]$ is typically calculated by recursively multiplying probability

values between themselves. In Section 1.4, a scaling technique that allows for keeping forward and backward variables within an acceptable order of magnitude is presented. This technique can clearly be applied for re-estimating parameters of super-state λ^i. However, when using two-dimensional Markov modelling, another problem arises.

We introduce this shortcoming via an example. Let us set the following parameters for a given experiment:

- Observation parameters: $X = 300$, $Y = 200$, $D = 3$, $L = 100$ (L is the dimension of the vocabulary in each dimension)
- Model parameters: $N = 3$, $N^1 = 1$, $N^2 = 3$, $N^3 = 1$. This set of parameters describes the P2DHMM presented in Figure 2.1.

We obtain the following (rough) estimates:

- $b^i_j(o_{xy}) \simeq 10^{-6}$ if letters are all equally likely.
- $a_{ij} \simeq a^i_{kl} \simeq 10^{-1}$.
- $\pi_i \simeq \pi^i_j \simeq 10^{-1}$.

In this context since $P[O_y|\lambda^i] = \sum_{q_{1y}\cdots q_{Xy}} \pi^i_{q_{1y}} b^i_{q_{1y}}(o_{1y}) a_{q_{1y}q_{2y}} b^i_{q_{2y}}(o_{2y}) \ldots a_{q_{X-1y}q_{Xy}} b^i_{q_{Xy}}(o_{Xy})$, we obtain $P[O_y|\lambda^i] \simeq 10^{-2101}$. This factor will be used as output probability of the upper-level (vertical) model. The same type of estimation leads to $P[O|\Lambda] \simeq 10^{-10^5}$. It is now clear that the scaling procedure described in Section 1.4 does not allow for solving this type of problem since values of $P[O_y|\lambda^i] \simeq 10^{-10^3}$ are already beyond the storage capabilities of most of computers.

The solution therefore lies in calculating these values through their log-value. Developments are mostly based on the following (trivial and exact) formula.

Given $a_1, \ldots a_n$,

$$\log\left(\sum_{i=1}^{n} a_i\right) = \log a_{i^*} + \log\left(1 + \sum_{i=1; i\neq i^*}^{n} \exp(\log a_i - \log a_{i^*})\right) \quad \forall i^* \in \{1 \ldots n\}, \tag{2.2}$$

where $\log(.)$ is the natural logarithm (*i.e.*, $\log_e(.)$) function. In other words, the log-value of a sum can be calculated from the log-value of each of its terms. This allows for storing values through their log-value solely and operating calculations without referring to their actual values. In particular, using Equation (2.2) with a suitable choice of i^*, we will calculate parameters and variables related to the upper-level (vertical) HMM via their log-value.

2.4.1 Forward procedure

Assuming that all parameters of the super-states of the P2DHMM have been calculated using the scaling procedure described in Section 1.4, we obtain in the case of an ergodic model (the case of other structures for the P2DHMM is easily adapted from this case and from Section 1.4):

$$b_i(O^{(k)}_y) = Z^{i(k)}_y,$$

where, $Z^{i(k)}_y = \Pi^X_{x=1} z^{i(k)}_{xy}$ and $z^{i(k)}_{xy}$ are defined to scale forward variables $\alpha^{i(k)}_{xy}(j)$ (see Section 1.4).

Since the value of $b_i(O^{(k)}_y)$ is typically extremely small, we can only compute its log-value as,

$$\log(b_i(O^{(k)}_y)) = \log(Z^{i(k)}_y) = \sum_{x=1}^{X} \log(z^{i(k)}_{xy}).$$

Based on log-values of super-state output probabilities, the forward procedure is adapted to calculate log-value of $\alpha_y(i)$ using Equation (2.2). In other words, we calculate,

$$\hat{\alpha}_y(i) = \log(\alpha_y(i)) - \sum_{y'=1}^{y} \log(Z_{y'}).$$

Remark 6 *Note that the scaling procedure presented in Section 1.4 is necessary here to lower down the order of magnitude of the variables themselves.*

2.4.2 Backward procedure

Similarly to Section 1.4, backward variables will be scaled using the same scale factors Z_y as in forward variable scaling. Equation (2.2) is used again to calculate

$$\hat{\beta}_y(i) = \log(\beta_y(i)) - \sum_{y'=y+1}^{Y} \log(Z_{y'}).$$

2.4.3 Parameter re-estimation

Using the above procedures, we clearly obtain

$$\hat{\alpha}_y(i) = \log(\alpha_y(i)) - \sum_{y'=1}^{y} \log(Z_{y'}) \text{ and } \hat{\beta}_y(i) = \log(\beta_y(i)) - \sum_{y'=y}^{Y} \log(Z_{y'})$$

In this context,

$$\hat{\alpha}_y(i) + \hat{\beta}_{y+1}(i) = \log(\alpha_y(i)) - \sum_{y'=1}^{y} \log(Z_{y'}) + \log(\beta_{y+1}(i)) - \sum_{y'=y+1}^{Y} \log(Z_{y'})$$

$$= \log(\alpha_y(i)) + \log(\beta_{y+1}(i)) - \sum_{y'=1}^{Y} \log(z_{y'})$$

and (in the general case),

$$\log(P[O|\Lambda]) = \sum_{y'=1}^{Y} \log(z_{y'})$$

So that scale factors will cancel out in the re-estimation formulas of a_{ij}.

Remark 7 *Other types of P2DHMM (e.g., Bakis model) are readily adapted from Section 1.4.*

2.4.4 Calculation of output probability values

One may note that the calculation of $b_j^i(o_{xy})$ requires the calculation of both the determinant $|\Sigma_{jm}^i|$ and the inverse $(\Sigma_{jm}^i)^{-1}$ of the covariance matrix for each component of the Gaussian mixture. Since it is numerically difficult and imprecise to invert this matrix, we consider $V = (\Sigma_{jm}^i)^{-1}(o_{xy} - \mu_{jm}^i)$ and calculate the exponential part of the output probability as $(o_{xy} - \mu_{jm}^i)^{\mathsf{T}} V$. By using Gauss elimination procedure, we can solve the system $\Sigma_{jm}^i V = (o_{xy} - \mu_{jm}^i)$. Since this system must be solved for every o_{xy}, we store the covariance matrix in its LU-decomposition form, as given by the classic Gauss elimination algorithm. This in turn reduces the computation of the determinant $|\Sigma_{jm}^i|$ to the product of diagonal values of this decomposition.

Conclusion on the theoretical developments

This report introduces and details major ideas behind Hidden Markov Modelling. The aim here is to set the context and notation in order to apply these tools for colour image analysis. In parallel, the bibliography cited within this text should provide the reader with key references for a tutorial on HMM theory.

In summary, the aim here was not to introduce any new material but rather to summarise and set our context for the development of future applications.

Bibliography

[1] O. Agazzi and S.-S. Kuo. Joint normalization and recognition of degraded documents images using pseudo-2D HMM. *Proceedings of the IEEE*, pages 155–158, 1993.

[2] H. Bourlard and N. Morgan. *Connectionist Speech Recognition – A Hybrid Approach.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.

[3] A. P. Dunmur and D. M. Titterington. Computational Bayesian analysis of Hidden Markov Mesh Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-19(11):1296–1300, 1997.

[4] A. Kosmala and G. Rigoll. On-line handwritten formula recognition using statistical methods. In *Proceedings of the Int't Conf. on Pattern Recognition (ICPR'98)*, Brisbane, Australia, 1998.

[5] S.-S. Kuo and O. E. Agazzi. Automatic keyword recognition using Hidden Markov Models. *Journal of Visual Communication and Image Representation*, 5(3):265–272, 1994.

[6] S.-S. Kuo and O. E. Agazzi. Keyword spotting in poorly printed documents using Pseudo 2-D Hidden Markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-16(8):842–848, 1994.

[7] K.-F. Lee. *Automatic speech recognition.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1989.

[8] S. Marchand-Maillet. 1D and pseudo-2D Hidden Markov Models for image analysis – B: Implementation details. Technical report, EURECOM Institute – Dept of Multimedia Communications, 1999.

[9] S. Marchand-Maillet. 1D and pseudo-2D Hidden Markov Models for image analysis – C: Applications and results. Technical report, EURECOM Institute – Dept of Multimedia Communications, 1999.

[10] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[11] L. R. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[12] G. Rigoll and A. Kosmala. A systematic comparison between on-line and off-line methods for signature verification with Hidden Markov Models. In *Proceedings of the Int'l Conf. on Pattern Recognition (ICPR'98)*, Brisbane, Australia, 1998.

[13] G. Rigoll, S. Müller and C. Neukirchen. Spotting of handwritten symbols in complex environments using Pseudo-2D Hidden Markov Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, USA, 1998.

[14] F. S. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceeding of the Second IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, December 1994.

[15] G. Saon. *Modéles Markoviens uni- et bi-dimensionnels pour la reconnaissance de l'écriture manuscrite hors-ligne.* PhD thesis, Université Henri Poincaré, Nancy, France, November 1997.