4

Classification of Hidden Markov Models: Obtaining Bounds on the Probability of Error and Dealing with Possibly Corrupted Observations

Eleftheria Athanasopoulou[1] and Christoforos N. Hadjicostis[2,1]
[1]*University of Illinois, Urbana-Champaign*
[2]*University of Cyprus, Nicosia*
[1,2]*USA*
[2]*Cyprus*

1. Introduction

In this chapter we consider classification of systems that can be modeled as hidden Markov models (HMMs). Given two[1] known HMMs, we discuss how to build an optimal classifier to determine, with minimal probability of error, which competing model has most likely produced a given sequence of observations. The main issue when dealing with HMMs is that the state cannot be directly observed but, instead, only the (possibly non-unique) output associated with the state and/or the transition of the model can be observed. Apart from describing how to optimally perform the classification task (in a way that minimizes the probability of error), we analyze the classification scheme by characterizing the effectiveness of the classifier in terms of a bound on the associated probability of error. We also analyze a more challenging scenario where the observations are possibly erroneous.

The likelihood that a given observed sequence has been generated from a particular HMM can be calculated as the sum of the probabilities of all possible state sequences that are consistent with the sequence of observations. This can be done using an iterative algorithm similar to the forward algorithm (Rabiner, 1989), which solves the evaluation problem in HMMs and is used frequently in speech recognition applications (Jelinek, 1998), (Rabiner, 1989), (Poritz, 1988). More specifically, given a model and an observed sequence, the evaluation problem consists of computing the probability that the observed sequence was produced by the model. When there are many competing models, these probabilities can be used to choose the model which best matches the observations, in a way that minimizes the probability of error. The forward algorithm is also used in pattern recognition applications (Fu, 1982), (Vidal et al., 2005) to solve the syntax analysis or parsing problem, i.e., to recognize a pattern by classifying it to the appropriate generating grammar, and in bioinformatics (Durbin et al., 1998), (Koski, 2001) to evaluate whether a DNA sequence or a protein sequence belongs to a particular family of sequences.

This chapter begins with an overview of optimal classification schemes for HMMs where the goal is to minimize the probability of error of the classifier. Given a particular sequence of

[1] We can easily generalize the discussion in this chapter to deal with classification of more than two models, but choose to focus on the case of two models for clarity/brevity purposes.

observations, these techniques can be used to choose the HMM that most likely generated the sequence of observations and, in the process, also characterize the associated probability of error (for the given sequence of observations). However, in order to measure the classification capability of the classifier *before* making any observations, one needs to compute the *a priori* probability that the classifier makes an incorrect decision for *any* of the possible sequences of observations. Enumerating all possible sequences of a given length (in order to evaluate their contribution to the probability of error) is prohibitively expensive for long sequences; thus, we describe ways to avoid this computational complexity and obtain an upper bound on the probability that the classifier makes an error without having to enumerate all possible output sequences. Specifically, we present a constructive approach that bounds the probability of error as a function of the observation step. We also discuss necessary and sufficient conditions for this bound on the probability of error to go to zero as the number of observations increases. After obtaining bounds on the probability of erroneous classification, we consider the additional challenge that the observed sequence is corrupted, due to noise coming from sensor malfunctions, communication limitations, or other adversarial conditions. For example, depending on the underlying application, the information that the sensors provide may be corrupted due to inaccurate measurements, limited resolution, or degraded sensor performance (due to aging or hardware failures). We consider unreliable sensors that may cause outputs to be deleted, inserted, substituted or transposed with certain known probabilities. Under such sensor malfunctions, the length of the observed sequence will generally not equal the length of the output sequence and, in fact, several output sequences may correspond to a given observed sequence. Thus, one would need to first identify all possible state sequences and the probabilities with which they agree with *both* the underlying model and the observations (after allowing, of course, for sensor failures). In particular, if symbols in the output sequence can be *deleted*, there may be an infinite number of output sequences that agree with a given observed sequence, which makes the standard forward algorithm inapplicable for classification. This inability of the standard forward algorithm can be overcome via an iterative algorithm that allows us to efficiently compute the probability that a certain model matches the observed sequence: each time a new observation is made, the algorithm simply updates the information it keeps track of and outputs on demand the probability that a given model has produced the sequence observed so far. The iterative algorithm we describe relates to (and generalizes) iterative algorithms for the evaluation problem in HMMs (Rabiner, 1989), the parsing problem in probabilistic automata (PA) (Fu, 1982), (Vidal et al., 2005), and the trellis-based decoding of variable length codes (VLC) (Bauer and Hagenauer, 2000), (Guyader et al., 2001), all of which can be modified to deal with some types of sensor failures but are not quite as general (or effective) as the iterative algorithm we describe.

We motivate the study of the above problems (bounding the probability of classification error and dealing with corrupted observations) using examples from the areas of failure diagnosis and computational biology. For instance, the problem of failure diagnosis in systems that can be modeled as finite state machines (FSMs) with known input statistics can be converted to the problem of classification of HMMs (Athanasopoulou, 2007). FSMs form a particular class of discrete event systems (DESs) that have discrete state spaces and whose evolution is event-driven, i.e., only the occurrence of discrete events forces the systems to take state transitions. Any large scale dynamic system, such as a computer system, a telecommunication network, a sensor network, a manufacturing system, a chemical process or a semiconductor manufacturing process, can be modeled as an FSM at some level of abstraction. In addition,

network protocols that describe the rules and conditions for exchanging information in distributed environments can also be modeled by FSMs. Given two known FSMs (one corresponding to the fault-free version of the underlying system and the other corresponding to a faulty version of the system) and an associated input distribution, a classifier (called diagnoser in this case) can be used to determine which of the two competing models has most likely produced a given sequence of observations. Work in this area by the authors has appeared in (Athanasopoulou et al., 2010), (Athanasopoulou and Hadjicostis, 2008). HMMs are also used in the area of computational biology to capture the behavior of DNA sequences; as we will see via examples in this chapter, these models can then be used to perform classification in order to characterize various properties of interest in DNA sequences.

2. Preliminaries and notation: FSMs, Markov chains, and hidden Markov models

A finite state machine (FSM) is a four-tuple (Q, X, δ, q_0), where $Q = \{0, 1, 2, ..., |Q| - 1\}$ is the finite set of states; X is the finite set of inputs; δ is the state transition function; and q_0 is the initial state. The FSMs we consider here are event-driven and we use n to denote the time epoch between the occurrence of the n^{th} and $(n+1)^{st}$ input. The state $Q[n+1]$ of the FSM at time epoch $n+1$ is specified by its state $Q[n]$ at time epoch n and its input $X[n+1]$ via the state transition function δ as $Q[n+1] = \delta(Q[n], X[n+1])$. A finite state machine (FSM) with outputs is described by a six-tuple $(Q, X, Y, \delta, \lambda, q_0)$, where (Q, X, δ, q_0) is an FSM; Y is the finite set of outputs; and λ is the output function. The output $Y[n+1]$ is determined by the state $Q[n]$ and the input $X[n+1]$ via the output function, i.e., $Y[n+1] = \lambda(Q[n], X[n+1])$, which maps a state and input pair to an output from the finite set of outputs Y.

We denote a time homogeneous Markov chain by $(Q, \Delta, \pi[0])$, where $Q = \{0, 1, 2, ..., |Q| - 1\}$ is the finite set of states; $\pi[0]$ is the initial state probability distribution vector; and Δ captures the state transition probabilities, i.e., $\Delta(q, q') = P(Q[n+1] = q' \mid Q[n] = q)$, for $q, q' \in Q$. If we denote the state transition probabilities by $a_{jk} = P\{(Q[n+1] = j) \mid (Q[n] = k)\}$, the state transition matrix of the Markov chain associated with the given system is $\mathcal{A} = (a_{jk})_{j,k=0,1,...,|Q|-1}$. (To keep the notation clean, the rows and columns of all matrices are indexed starting from 0 and not 1.) The state transition matrix $\mathcal{A}$ captures how state probabilities evolve in time via the evolution equation $\pi[n+1] = \mathcal{A}\pi[n]$, for $n = 0, 1, 2, \ldots$. Here, $\pi[n]$ is a $|Q|$-dimensional vector, whose j^{th} entry denotes the probability that the Markov chain is in state j at time epoch n. In our development later on, we will find it useful to define the notion of a time homogeneous Markov chain with inputs, which we denote by $(Q, X, \Delta, \pi[0])$; here, $Q = \{0, 1, 2, ..., |Q| - 1\}$ is the finite set of states; X is the finite set of inputs; $\pi[0]$ is the initial state probability distribution vector; and Δ captures the state and input transition probabilities, i.e., $\Delta(q, x_i, q') = P(Q[n+1] = q' \mid Q[n] = q, X[n+1] = x_i)$, for $q, q' \in Q$, $x_i \in X$.

An HMM is described by a five-tuple $(Q, Y, \Delta, \Lambda, \boldsymbol{\rho}[0])$, where $Q = \{0, 1, 2, ..., |Q| - 1\}$ is the finite set of states; Y is the finite set of outputs; Δ captures the state transition probabilities; Λ captures the output probabilities associated with transitions; and $\boldsymbol{\rho}[0]$ is the initial state probability distribution vector. More specifically, for $q, q' \in Q$ and $\sigma \in Y$, the state transition probabilities are given by $\Delta(q, q') = P(Q[n+1] = q' \mid Q[n] = q)$ and the output probabilities associated with transitions are given by $\Lambda(q, \sigma, q') = P(Q[n+1] = q', Y[n+1] = \sigma \mid Q[n] = q)$, where Λ denotes the output function that assigns a probability to the output σ associated with the transition from state $Q[n]$ to state $Q[n+1]$. We define the $|Q| \times |Q|$ matrix $\mathcal{A}_\sigma$, associated with output $\sigma \in Y$ of the HMM, as follows: the entry at the $(j,k)^{th}$ position of $\mathcal{A}_\sigma$ captures the probability of a transition from state k to state j that produces output σ. Note that

$\sum_{\sigma \in Y} \mathcal{A}_\sigma = \mathcal{A}$, i.e., the transition matrix whose $(j,k)^{th}$ entry denotes the probability of taking a transition from state k to state j. The joint probability of the state at step n and the observation sequence $y[1], \ldots, y[n]$ is captured by the vector $\boldsymbol{\rho}[n]$ where the entry $\boldsymbol{\rho}[n](j)$ denotes the probability that the HMM is in state j at step n and the sequence $y_1^n = y[1], \ldots, y[n]$ has been observed. More formally, $\boldsymbol{\rho}[n](j) = P(Q[n] = j, Y_1^n = y_1^n)$ (note that $\boldsymbol{\rho}$ is not necessarily a probability vector).

3. Posterior probability calculation with uncorrupted observations

In this section, we examine the simplest case where no sensor failures are present. It will become apparent later that this case does not involve any complications, such as loops in the trellis diagram, and the calculations can be easily performed iteratively by a forward-like algorithm.

Given the observation sequence $Y_1^L = y_1^L =< y[1], y[2], ..., y[L] >$ and two candidate HMMs S_1 and S_2 (together with their initial state probability distributions and their *prior* probabilities P_1 and $P_2 = 1 - P_1$), the classifier that minimizes the probability of error needs to implement the maximum *a posteriori* probability (MAP) rule by comparing $P(S_1 \mid y_1^L) \underset{<}{\overset{>}{}} P(S_2 \mid y_1^L) \Rightarrow \frac{P(y_1^L \mid S_1)}{P(y_1^L \mid S_2)} \underset{<}{\overset{>}{}} \frac{P_2}{P_1}$, and deciding in favor of S_1 (S_2) if the left (right) quantity is larger.

To calculate the probability $P(y_1^L \mid S)$ of the observed sequence given a particular model S, we first capture the evolution of S as a function of time for a given observed sequence by constructing the trellis diagram of S. The state sequences that agree with the observed sequence (consistent sequences) are those that start from any valid initial state and end at any final state while ensuring that the output at each step n matches the observed output $y[n]$. Due to the Markovian property, the probability of a specific consistent state sequence can be easily calculated as the product of the initial state probability and the state transition probabilities at each time step. Thus, to calculate the probability $P(y_1^L \mid S)$ we need to first identify all consistent sequences and their probabilities and then sum up the total probability.

The computation of $P(y_1^L \mid S)$ is not iterative in respect to the number of observation steps, hence it is not amenable for online monitoring. To make the computation iterative we can use a forward-like algorithm. For candidate HMM S, we can update $\boldsymbol{\rho}_S$ iteratively as

$$\boldsymbol{\rho}_S[n+1] = \mathcal{A}_{S,y[n+1]} \boldsymbol{\rho}_S[n], \;\; n = 0, 1, ..., L-1,$$

where $\boldsymbol{\rho}_S[0]$ is taken to be the probability distribution of the initial states for model S and $\mathcal{A}_{S,y[n+1]}$ is the matrix that describes the state transition probabilities under the output observed at time epoch $n+1$. If L is the last step, the probability that the observation sequence was produced by FSM S is equal to the sum of the entries of $\boldsymbol{\rho}_S[L]$, i.e., $P(y_1^L \mid S) = \sum_{j=0}^{|Q|-1} \boldsymbol{\rho}_S[L](j)$. This iterative algorithm is the standard forward algorithm that is used to solve the evaluation problem in HMMs.

Example 1.a: In this example we consider the finite state machine (FSM) S with known input distribution shown on the left of Figure 1: S has four states $Q = \{0, 1, 2, 3\}$, three inputs $X = \{x_1, x_2, x_3\}$, and three outputs $Y = \{a, b, c\}$. Each transition is labeled as $x_i \mid \sigma$, where $x_i \in X$ denotes the input that drives the FSM and $\sigma \in Y$ denotes the associated output produced by the FSM. If we assign equal prior probabilities to the inputs, i.e., if each input has probability $1/3$ of occurring, the resulting HMM is shown on the right of Figure 1: each transition in the HMM is labeled as $p \mid \sigma$, where p denotes the probability of the transition and $\sigma \in Y$ denotes the output produced.

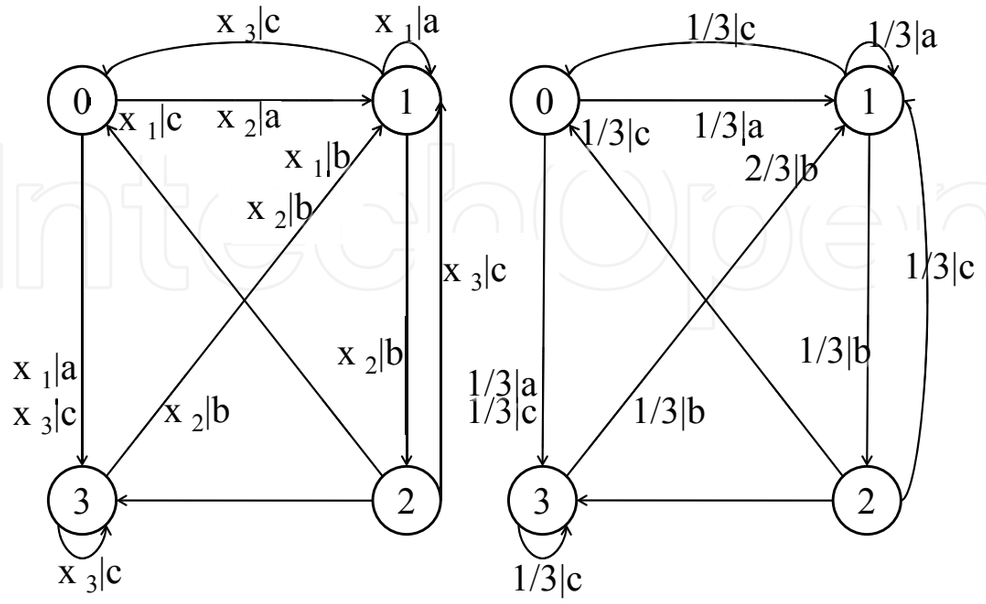


Fig. 1. State transition diagram of FSM S of Example 1 (left) and its corresponding HMM (right).

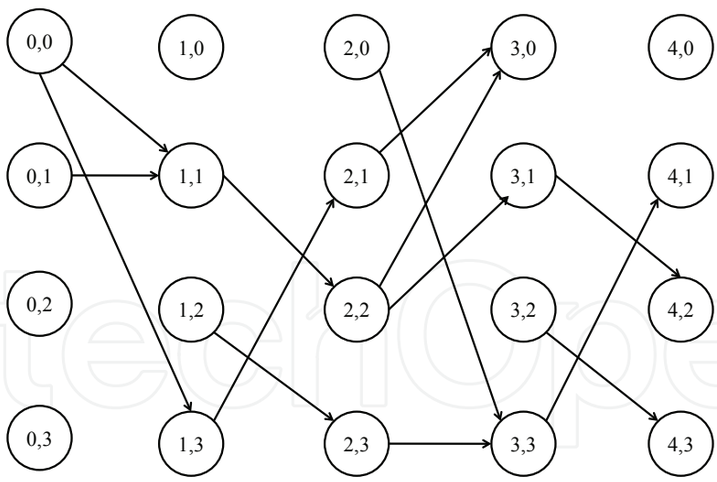


Fig. 2. Trellis diagram corresponding to S of Example 1 for observation sequence $y_1^4 =< abcb >$ (transition probabilities are not included for clarity).

Suppose that we monitor S for $L = 4$ steps and we observe the sequence $y_1^4 =< abcb >$. The corresponding trellis diagram is shown in Figure 2 (pairs of states that are associated with zero transition probabilities are not connected in the diagram and transition probabilities are

n	$\boldsymbol{\rho}_S^T[n]$				$P(y_1^n \mid S)$
0	[0.2500	0.2500	0.2500	0.2500]	1
1	[0	0.1667	0	0.0833]	0.2500
2	[0	0.0556	0.0556	0]	0.1112
3	[0.0370	0.0185	0	0]	0.0555
4	[0	0	0.0062	0]	0.0062

Table 1. Iterative calculations for vector $\boldsymbol{\rho}_S$ for the sequence of observations $y_1^4 =< abcb >$.

not included for clarity of presentation). Each state of the trellis diagram is identified by a pair (m, j), where $m = 0, 1, 2, 3, 4$ denotes the observation step and $j \in \{0, 1, 2, 3\}$ denotes the state of S. For example, the probability of a transition from state $(0, 0)$ to state $(1, 3)$ producing output a is $1/3$. Assuming uniform initial distribution, the probability that the observations were produced by S can be calculated iteratively and is given by $P(y_1^4 \mid S) = 0.0062$. Note that this probability is expected to go down as the number of observations increases. Table 1 shows the sequence of iterations in order to obtain the vector $\boldsymbol{\rho}_S[n]$ as observations are coming in. □

Example 1.b:

As another example, we consider a problem from computational biology. Specifically, we concentrate on identifying CpG islands in a DNA sequence, where the alphabet consists of the four nucleotides A, C, G, T (Durbin et al., 1998). Regions that are characterized by higher than usual concentration of CpG dinucleotides (more generally, higher concentration of C and G nucleotides) are called CpG islands.[2] It is important to be able to identify these regions because they typically appear around the promoters or start regions of many genes (Fatemi et al., 2005). Consider the following task: given a short stretch of genomic sequence, can we decide whether it comes from a CpG island or not? For illustration purposes, we assume that we are only capable of observing two output symbols, α and β, as follows: we observe the symbol α when the true output is A or C and we observe the symbol β when the true output is G or T (this could be used, for instance, to model situations where instruments are unable to distinguish between specific pairs of nucleotides). We assume that we are given two HMMs, CpG^+ and CpG^-, with known structure, which model respectively regions with and without CpG islands. As shown in Figure 3, CpG^+ and CpG^- have four states $Q = \{A, C, G, T\}$, and two outputs $Y = \{\alpha, \beta\}$. We also assume (for simplicity) that the *priors* of the two models are $p_{CpG^+} = p_{CpG^-} = 0.5$.

Suppose that we observe the sequence $y_1^4 =< \alpha\beta\alpha\beta >$ and our goal is to determine which of the two HMMs (CpG^+ or CpG^-) has most likely generated the observed sequence. According to the previously described iterative algorithm, we need to first define the transition probability matrices for each symbol for each one of the two HMMs ($\mathcal{A}_{CpG^+,\alpha}$, $\mathcal{A}_{CpG^+,\beta}$, $\mathcal{A}_{CpG^-\alpha}$, $\mathcal{A}_{CpG^-,\beta}$); for example, for HMM CpG^+, we have

$$\mathcal{A}_{CpG^+,\alpha} = \begin{bmatrix} & A & C & G & T \\ A & 0.18 & 0.17 & 0.16 & 0.08 \\ C & 0.27 & 0.37 & 0.34 & 0.36 \\ G & 0 & 0 & 0 & 0 \\ T & 0 & 0 & 0 & 0 \end{bmatrix}$$

[2] The formal definition of a CpG island is a region with at least 200 base pairs that has a GC percentage that is greater than 50% and an observed/expected CpG ratio that is greater than 60 (Gardiner-Garden and Frommer, 1987).

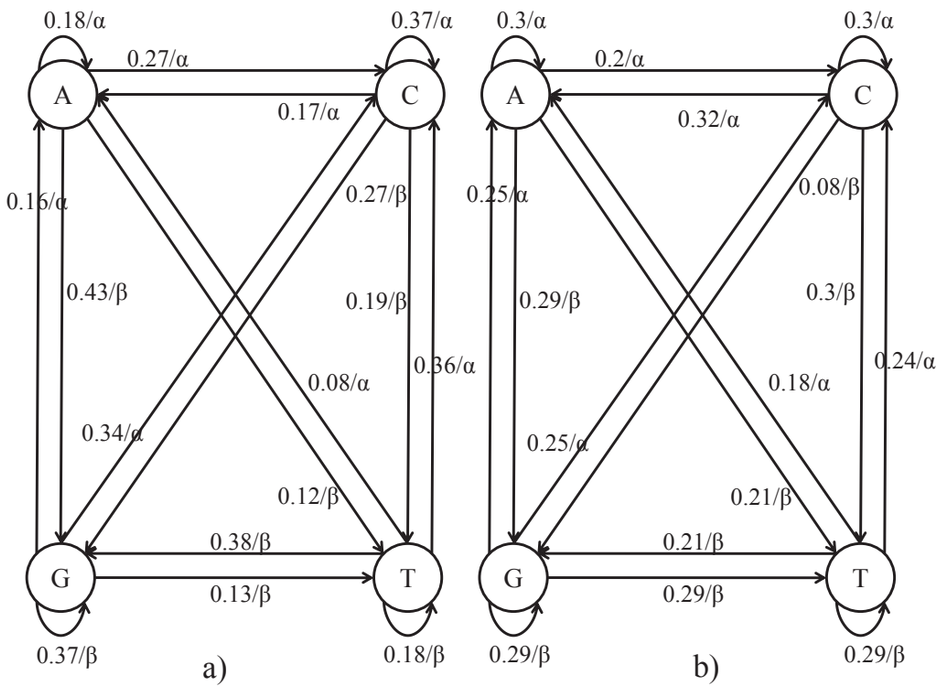


Fig. 3. State transition diagram of the two HMMs used to identify regions with or without CpG islands in Example 1.b: (a) CpG^+ (left), and (b) CpG^- (right).

n	$\boldsymbol{\rho}_+^T[n]$				$P(y_1^n \mid CpG^+)$
0	[0.2500	0.2500	0.2500	0.2500]	1
1	[0.1475	0.3350	0	0]	0.4825
2	[0	0	0.1539	0.0814]	0.2353
3	[0.0311	0.0816	0	0]	0.1127
4	[0	0	0.0354	0.0192]	0.0546

Table 2. Iterative calculations for posterior probabilities for the sequence of observations $y_1^4 =< \alpha\beta\alpha\beta >$ for the model CpG^+.

Given our previous discussion, we can calculate

$$\begin{aligned} \boldsymbol{æ}_+[4] &= \mathcal{A}_{CpG^+,\beta} \cdot \mathcal{A}_{CpG^+,\alpha} \cdot \mathcal{A}_{CpG^+,\beta} \cdot \mathcal{A}_{CpG^+,\alpha} \cdot \boldsymbol{\rho}_{CpG^+}[0] \\ \boldsymbol{æ}_-[4] &= \mathcal{A}_{CpG^-,\beta} \cdot \mathcal{A}_{CpG^-,\alpha} \cdot \mathcal{A}_{CpG^-,\beta} \cdot \mathcal{A}_{CpG^-,\alpha} \cdot \boldsymbol{\rho}_{CpG^-}[0] \end{aligned}$$

and obtain $P(y_1^4 \mid CpG^+) = 0.0546$ and $P(y_1^4 \mid CpG^-) = 0.0445$. We can now apply the MAP rule $\frac{P(y_1^4 \mid CpG^+)}{P(y_1^4 \mid CpG^-)} \underset{<}{>} \frac{p_{CpG^+}}{p_{CpG^-}}$ which reduces to $\frac{0.0546}{0.0445} \underset{<}{>} 1$, and decide in favor of CpG^+ since the left quantity is larger. Tables 2 and 3 show the various values obtained for the vectors $\boldsymbol{\rho}_+$ and $\boldsymbol{\rho}_-$ (corresponding to CpG^+ and CpG^- respectively) during the iteration.

n	$\boldsymbol{\rho}_-^T[n]$				$P(y_1^n \mid CpG^-)$
0	[0.2500	0.2500	0.2500	0.2500]	1
1	[0.2625	0.2475	0	0]	0.51
2	[0	0	0.0959	0.1294]	0.2253
3	[0.0473	0.0550	0	0]	0.1023
4	[0	0	0.0181	0.0264]	0.0445

Table 3. Iterative calculations for posterior probabilities for the sequence of observations $y_1^4 =< \alpha\beta\alpha\beta >$ for the model CpG^-.

4. Probability of error

In this section we focus on bounding the probability of classification error, i.e., the probability that the classifier makes the incorrect decision.

4.1 Preliminaries

We start by conditioning on a given observation sequence y_1^L and we compute online the conditional probability that the classifier makes an incorrect decision as follows:

$$\begin{aligned} P(\text{error at } L \mid y_1^L) &= P(\text{decide } S_2 \text{ at } L,\ S_1 \mid y_1^L) + P(\text{decide } S_1 \text{ at } L,\ S_2 \mid y_1^L) \\ &= P(\text{decide } S_2 \text{ at } L \mid S_1,\ y_1^L) \cdot P(S_1 \mid y_1^L) + \\ &\quad P(\text{decide } S_1 \text{ at } L \mid S_2,\ y_1^L) \cdot P(S_2 \mid y_1^L) \\ &= \min\{P(S_2 \mid y_1^L),\ P(S_1 \mid y_1^L)\}. \end{aligned}$$

Since both *posteriors* are already computed (for use in the MAP rule comparison), the probability of error given the observation sequence y_1^n as a function of n can be easily computed online along with the maximum likelihood decision. At each step, the classifier chooses the model with the larger *posterior* and makes an error with probability equal to the *posterior* of the other model (of course, the *posteriors* need to be normalized so that they sum up to one).

Our goal is to find a measure of the classification capability of the classifier *a priori*, i.e., before any observation is made. The probability of error at step L is given by

$$P(\text{error at } L) = \sum_{y_1^L} \left(P(y_1^L) \cdot \min\{P(S_2 \mid y_1^L),\ P(S_1 \mid y_1^L)\} \right).$$

To perform such computation, we need to find each possible observation sequence y_1^L, along with its probability of occurring, and use it to compute the *posterior* of each model conditioned on this observation sequence. To avoid the possibly prohibitively high computational complexity (especially for large L) we will focus on obtaining an easily computable upper bound and then show that, under certain conditions on the underlying HMMs, this bound on the probability of error decays exponentially to zero with the number of observation steps L.

A classifier that uses the MAP rule necessarily chooses model S_1 (S_2) if the observation sequence cannot be produced by S_2 (S_1), with no risk of making an incorrect decision. However, if the observation sequence can be produced by both models, the classifier chooses the model with the highest *posterior*, thereby risking to make an incorrect decision. The bound we obtain considers the worst case scenario where, when both models are consistent with the observation sequence y_1^L (i.e., when $P(S_i \mid y_1^L) > 0$ for $i = 1$ and 2), the classifier is assumed to always make the incorrect decision; specifically, one has

$$
\begin{aligned}
P(\text{error at } L) &= \sum_{y_1^L} \min\{P(S_1 \mid y_1^L), P(S_2 \mid y_1^L)\} \cdot P(y_1^L) \\
&= 1 - \sum_{y_1^L} \max\{P(S_1 \mid y_1^L), P(S_2 \mid y_1^L)\} \cdot P(y_1^L) \\
&= 1 - \sum_{\substack{y_1^L: P(S_i|y_1^L)=0 \\ \text{for } i=1 \text{ or } 2}} P(y_1^L) - \sum_{\substack{y_1^L: P(S_i|y_1^L)>0 \\ \text{for } i=1 \text{ and } 2}} \max\{P(S_1 \mid y_1^L), P(S_2 \mid y_1^L)\} \cdot P(y_1^L) \\
&\leq 1 - \sum_{\substack{y_1^L: P(S_i|y_1^L)=0 \\ \text{for } i=1 \text{ or } 2}} P(y_1^L) - \frac{1}{2} \sum_{\substack{y_1^L: P(S_i|y_1^L)>0 \\ \text{for } i=1 \text{ and } 2}} P(y_1^L) \\
&= 1 - \sum_{\substack{y_1^L: P(S_i|y_1^L)=0 \\ \text{for } i=1 \text{ or } 2}} P(y_1^L) - \frac{1}{2}\Big(1 - \sum_{\substack{y_1^L: P(S_i|y_1^L)=0 \\ \text{for } i=1 \text{ or } 2}} P(y_1^L)\Big) \\
&= \frac{1}{2}\Big(1 - \sum_{\substack{y_1^L: P(S_i|y_1^L)=0 \\ \text{for } i=1 \text{ or } 2}} P(y_1^L)\Big) \\
&= \frac{1}{2}\Big(1 - P_1 \sum_{\substack{y_1^L: S_2 \\ \text{incons.}}} P(y_1^L \mid S_1) - P_2 \sum_{\substack{y_1^L: S_1 \\ \text{incons.}}} P(y_1^L \mid S_2)\Big).
\end{aligned}
$$

In the previous formulas we used the fact that, when both S_1 and S_2 are consistent with the observations, then the maximum of their *posteriors* is greater than or equal to half.

4.2 Calculation of bound on probability of error

Initially, our objective is to capture the set of observation sequences that are consistent with S_1 but not with S_2 (or sequences that are consistent with S_2 but not with S_1), i.e., to capture the set of output sequences that can be produced by S_1 but not by S_2 (or the other way around). Once we have identified this set of output sequences, we need to find its probability of occurring. First, we construct the Markov chain $S_{12|1}$ (respectively MC $S_{12|2}$) to help us compute the bound on the probability that S_2 (respectively S_1) becomes inconsistent with the observations, given that the actual model is S_1 (respectively S_2). In particular, we explain how to construct MC $S_{12|1}$ starting from HMMs S_1 and S_2 in the following five steps (a similar procedure can be followed to construct MC $S_{12|2}$).

Step 1. Construct FSMs S_{1ND} and S_{2ND} from HMMs S_1 and S_2 respectively.

The set of input sequences that S_{iND} accepts is the set of output sequences that S_i is capable of producing (where $i = 1, 2$). Recall that HMM S_i is denoted by $(Q_i, Y, \Delta_i, \Lambda_i, \boldsymbol{\rho}_i[0])$ (without loss of generality[3] we assume that $Y_1 = Y_2 = Y$). Ignoring the transition probabilities of HMM S_i, we build the possibly nondeterministic FSM S_{iND} which has the same set of states as S_i and its set of inputs is equal to the set of outputs of S_i. The state transition functionality of S_{iND} is determined by the output functionality of S_i which is captured by Λ_i (although the probabilities are not important at this point). More formally, FSM S_{iND} is denoted by $S_{iND} = (Q_{iND}, X_{iND}, \delta_{iND}, q_{iND0})$, where $Q_{iND} = Q_i$; $X_{iND} = Y$; $q_{iND0} = \{j \mid \boldsymbol{\rho}_i[0](j) > 0\}$ (i.e., q_{iND0} includes all states of S_i with nonzero initial probability); and $\delta_{iND}(q_{iND}, \sigma) = \{q'_{iND} \in Q_{iND} \mid \Lambda_i(q_{iND}, \sigma, q'_{iND}) > 0\}$.

Step 2. Construct FSMs S_{1D} and S_{2D} from FSMs S_{1ND} and S_{2ND} respectively.

We can think of FSM S_{iD} as an observer for S_i because each state of S_{iD} contains the set of

[3] We can always redefine $Y = Y_1 \cup Y_2$ to be the output of both machines if Y_1 and Y_2 are different.

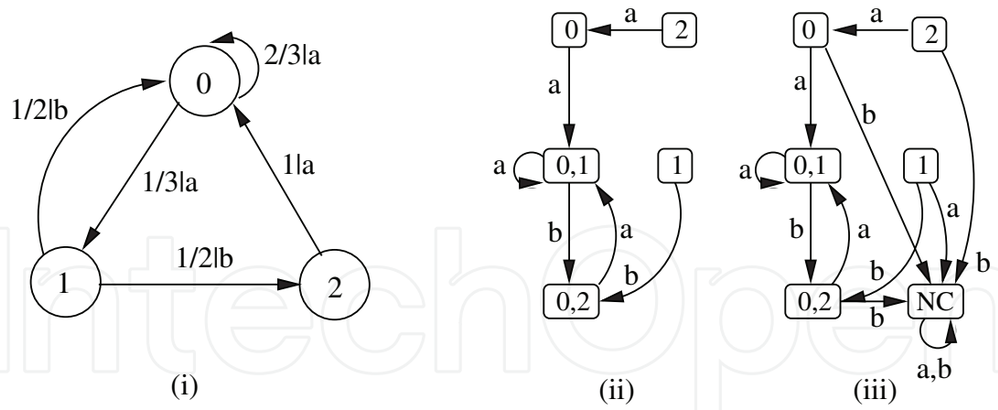


Fig. 4. State transition diagrams of (i) HMM S_1, (ii) FSM S_{1D}, and (iii) FSM S_{1DNC} of Example 2.

states that S_i may be in given the observation sequence. The number of states of S_{iD}, i.e., the deterministic version of S_{iND} could be as high as $2^{|Q_{iND}|}$. Although this may raise complexity issues, it is very common in practical scenarios for S_{iD} to have roughly the same number of states as S_{iND} (Hopcroft et al., 2001). Following the procedure of subset construction (Hopcroft et al., 2001) we use S_{iND} to build the deterministic, equivalent machine $S_{iD} = (Q_{iD}, X_{iD}, \delta_{iD}, q_{iD0})$, where Q_{iD} contains subsets of states in the set Q_i (recall that $Q_{iND} = Q_i$); the set of inputs are the same as the set of inputs of S_{iND}, i.e., $X_{iD} = Y$ (recall that $X_i = Y$); $q_{iD0} = q_{i0}$; and δ_{iD} is determined from S_{iND} by the procedure of subset construction, i.e., for $Q_S \subset Q_i$ and $\sigma \in Y$, $\delta_{iD}(Q_S, \sigma) = \{k \mid \exists j \in Q_S, k \in \delta_{iND}(j, \sigma)\}$.

Step 3. Construct FSM S_{2DNC} from FSM S_{2ND}.

Next, we append the inconsistent state NC to S_{2D} to obtain FSM S_{2DNC}. As mentioned earlier, FSM S_{2D} accepts all sequences that can be produced by S_2. FSM S_{2DNC} accepts not only the sequences that can be produced by S_2, but also all other sequences (that cannot be produced by S_2). In fact, all sequences that cannot be produced by S_2 will lead S_{2DNC} to its inconsistent state NC. More specifically, $S_{2DNC} = (Q_{2DNC}, X_{2DNC}, \delta_{2DNC}, q_{2DNC0})$, where $Q_{2DNC} = Q_{2D} \cup \{NC\}$; $X_{2DNC} = Y$; $q_{2DNC0} = q_{2D0}$ and δ_{2DNC} is given by $\delta_{2DNC}(q_{2DNC}, \sigma) =$

$$\begin{cases} \delta_{2D}(q_{2DNC}, \sigma), & \text{if } q_{2DNC} \neq NC, \delta_{2D}(q_{2DNC}, \sigma) \neq \emptyset, \\ NC, & \text{otherwise.} \end{cases}$$

Step 4. Construct FSM S_{1D2DNC} from FSMs S_{1D} and S_{2DNC}.

To capture the set of observations that can be produced by S_1 but not by S_2, we need to build the product FSM S_{1D2DNC}. FSM S_{1D2DNC} accepts all sequences that can be produced by S_1; from all of these sequences, the ones that cannot be produced by S_2 lead S_{1D2DNC} to a state of the form $\{q_{1D}, NC\}$. More specifically, $S_{1D2DNC} = S_{1D} \times S_{2DNC}$, i.e., $S_{1D2DNC} = (Q_{1D2DNC}, X_{1D2DNC}, \delta_{1D2DNC}, q_{0,1D2DNC})$, where $Q_{1D2DNC} = Q_{1D} \times Q_{2DNC}$; $X_{1D2DNC} = Y$ (recall that $X_{1D} = X_{2DNC} = Y$); $q_{0,1D2DNC} = q_{0,1D} \times q_{0,2DNC}$; and δ_{1D2DNC} is given by $\delta_{1D2DNC}(\{q_{1D}, q_{2DNC}\}, \sigma) = \{\delta_{1D}(q_{1D}, \sigma), \delta_{2DNC}(q_{2DNC}, \sigma)\}$, $\sigma \in Y$. Note that $\delta_{1D2DNC}(\{q_{1D}, q_{2DNC}\}, \sigma)$ is undefined if $\delta_{1D}(q_{1D}, \sigma)$ is undefined.

Step 5. Construct MC $S_{12|1}$ from FSM S_{1D2DNC} or from S_1, S_{1D}, and S_{2D}.

To compute the probabilities of the sequences captured by S_{1D2DNC} we construct the Markov chain with inputs $S_{12|1} = (Q_{12|1}\ X_{12|1}, \Delta_{12|1}, \boldsymbol{\rho}_{12|1}[0])$, where $Q_{12|1} = Q_1 \times Q_{1D2DNC}$; $X_{12|1} =$

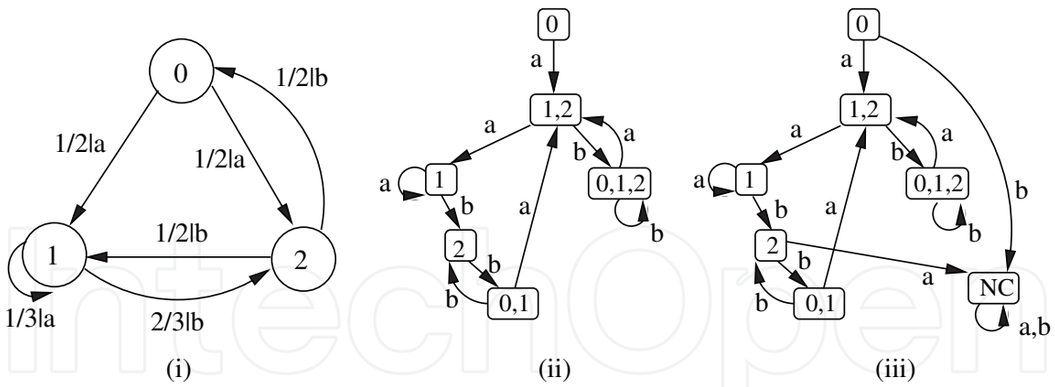


Fig. 5. State transition diagrams of (i) HMM S_2, (ii) FSM S_{2D}, and (iii) FSM S_{2DNC} of Example 2.

Y; $\boldsymbol{\rho}_{12|1}[0](\{q_1, q_{0,1D2DNC}\}) = \boldsymbol{\rho}_1[0](q_1)$, for every $q_1 \in Q_1$ and zero otherwise;[4] and $\Delta_{12|1}$ is given by $\Delta_{12|1}$ $(\{q_1, q_{1D}, q_{2DNC}\}, \sigma, \{q_1', \delta_{1D}(q_{1D}, \sigma), \delta_{2DNC}(q_{2DNC}, \sigma)\}) = \Lambda_1(q_1, \sigma, q_1')$ for all $\sigma \in Y$ such that $\delta_{1D}(q_{1D}, \sigma) \neq \emptyset$. We group all states of the form $\{q_1, q_{1D}, NC\}$ in one new state and call it NC; we also add a self-loop at state NC with probability one.

Alternatively we can build MC $S_{12|1}$ from S_1, S_{1D}, and S_{2D} as follows: $S_{12|1} = (Q_{12|1}, X_{12|1}, \Delta_{12|1}, \boldsymbol{\rho}_{12|1}[0])$, where $Q_{12|1} = Q_1 \times Q_{1D} \times Q_{2D}$; $X_{12|1} = Y$; $\boldsymbol{\rho}_{12|1}[0](\{q_1, q_{1D}, q_{2D}\}) = \boldsymbol{\rho}_1[0](q_1)$, for every $q_1 \in Q_1$, $q_{1D} \in Q_{1D}$, and $q_{2D} \in Q_{2D}$; and $\Delta_{12|1}$ is given by $\Delta_{12|1}(\{q_1, q_{1D}, q_{2DNC}\}, \sigma, \{q_1', \delta_{1D}(q_{1D}, \sigma), \delta_{2DNC}(q_{2DNC}, \sigma)\}) = \Lambda_1(q_1, \sigma, q_1')$, for all $\sigma \in Y$ such that $\delta_{1D}(q_{1D}, \sigma) \neq \emptyset$ and $\delta_{2D}(q_{2D}, \sigma) \neq \emptyset$ or $\Delta_{12|1}$ $(\{q_1, q_{1D}, q_{2DNC}\}, \sigma, \{q_1', \delta_{1D}(q_{1D}, \sigma), NC\}) = \Lambda_1(q_1, \sigma, q_1')$, for all $\sigma \in Y$ such that $\delta_{1D}(q_{1D}, \sigma) \neq \emptyset$ and $\delta_{2D}(q_{2D}, \sigma) = \emptyset$. (Note that for all q_{2D} and all σ we have $\delta_{1D}(q_{1D}, \sigma) \neq \emptyset$.) As mentioned before, we group all states of the form $\{q_1, q_{1D}, NC\}$ in one new state and call it NC; then we add a self-loop at state NC with probability one.

Notice that any path in $S_{12|1}$ that ends up in state NC represents a sequence that can be produced by S_1 but not by S_2; the probability of such path is easily computed using the Markovian property. Recall that our objective is to calculate the probability that HMM S_2 is inconsistent with the observations given that the observations are produced by S_1 (i.e., we would like to calculate $\sum_{\substack{y_1^L : S_2 \\ \text{incons.}}} P(y_1^L \mid S_1)$). Therefore, we are interested in the probability of $S_{12|1}$ being in the inconsistent state NC as a function of the observation step given by $P(S_{12|1} \text{ in state } NC \text{ at } L) = \pi_{12|1}[L](NC)$, where $\pi_{12|1}[L](NC)$ denotes the entry of $\pi_{12|1}[L]$ that captures the probability that $S_{12|1}$ is in the inconsistent state NC at L. Note that $\pi_{12|1}[L] = \mathcal{A}_{12|1}^L \pi_{12|1}[0]$, where $\mathcal{A}_{12|1}$ is the matrix that captures the transition probabilities for MC $S_{12|1}$ and $\pi_{12|1}[0] = \boldsymbol{\rho}_{12|1}[0]$ (note that at this point the particular inputs associated with a transition are not needed and are ignored — only the probabilities of these transitions matter).

[4] Abusing notation, we use $\boldsymbol{\rho}_{12|1}[0](\{q_1, q_{1D2DNC}\})$ to denote the entry of $\boldsymbol{\rho}_{12|1}[0]$ that corresponds to state $\{q_1, q_{1D2DNC}\}$; of course, $\boldsymbol{\rho}_1[0](q_1)$ denotes the entry of $\boldsymbol{\rho}_1[0]$ that corresponds to state q_1.

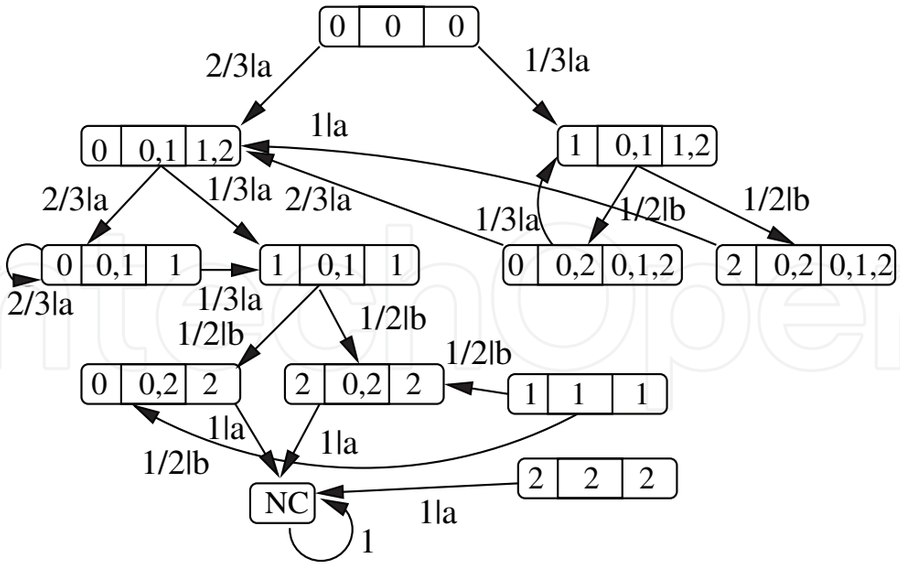


Fig. 6. State transition diagram of MC $S_{12|1}$ of Example 2.

Proposition 1: The probability of error as a function of the observation step is given by

$$\begin{aligned} P(\text{error at } L) &\leq \tfrac{1}{2}\Big(1 - P_1 \cdot \sum_{\substack{y_1^L : S_2 \\ \text{incons.}}} P(y_1^L \mid S_1) - P_2 \cdot \sum_{\substack{y_1^L : S_1 \\ \text{incons.}}} P(y_1^L \mid S_2)\Big) \\ &= \tfrac{1}{2} - \tfrac{1}{2} P_1 \cdot \pi_{12|1}[L](NC) - \tfrac{1}{2} P_2 \cdot \pi_{12|2}[L](NC), \end{aligned}$$

where $\pi_{12|1}[L](NC)$ captures the probability of $S_{12|1}$ being in state NC at step L, $\pi_{12|2}[L](NC)$ captures the probability of $S_{12|2}$ being in state NC at step L, and P_1 and P_2 denote the *priors* of S_1 and S_2. □

Example 2: We consider two candidate HMMs S_1 and S_2 with $Q_1 = Q_2 = \{0,1,2\}$, $Y_1 = Y_2 = \{a,b\}$, initial state $\{0\}$, and transition functionality, as shown in Figures 4.(i) and 5.(i), where each transition is labeled by $p_i \mid \sigma$, i.e., the probability of the transition and the output it produces. Following the procedure of subset construction we construct the deterministic FSMs S_{1D} and S_{2D} as shown in Figures 4.(ii) and 5.(ii), respectively (notice that we include states $\{1\}$ and $\{2\}$ in the state transition diagram of S_{1D} for completeness although they are not reachable from the initial state $\{0\}$). Adding the inconsistent state for each machine we get FSMs S_{1DNC} and S_{2DNC} as shown in Figures 4.(iii) and 5.(iii), respectively. Then, we construct MCs $S_{12|1}$ and $S_{12|2}$ with state transition diagrams as shown in Figures 6 and 7, respectively. For example, the sequence $< a\ a\ b\ a >$ can be produced by S_1 but not by S_2 (hence, given that this is the observation sequence, the probability that the classifier makes an incorrect decision is zero). In fact, all sequences in $S_{12|1}$ that end up in state NC can be produced by S_1 but not by S_2. □

4.3 Properties of bound on probability of error

The inconsistent state NC in MC $S_{12|1}$ is an absorbing state by construction. Therefore, the probability that $S_{12|1}$ is in state NC does not decrease as a function of the observation step; the same property holds for $S_{12|2}$. From Proposition 1 it is clear that the bound on the probability of error is a nonincreasing function of the observation step.

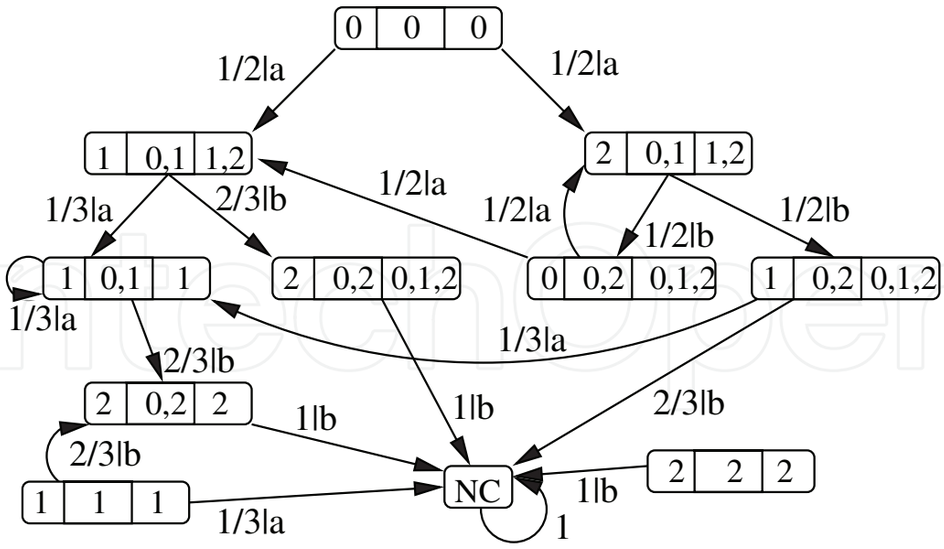


Fig. 7. State transition diagram of MC $S_{12|2}$ of Example 2.

Proposition 2: The bound on the probability of error given by Proposition 1 is a nonincreasing function of the number of observation steps. □

In fact, if MCs $S_{12|1}$ and $S_{12|2}$ have a single absorbing state each, i.e., state NC is the only absorbing state in each model, then the bound goes to zero as the number of observation steps increases. The expected number of steps to absorption, given that the initial state is the 0^{th} state of $S_{12|1}$, can be calculated using the fundamental matrix of the absorbing Markov chain $S_{12|1}$ (Kemeny et al., 1976). If $\mathcal{A}_{T12|1}$ is the substochastic transition matrix of $S_{12|1}$ that captures the transitions among all transient states (all but NC) then the fundamental matrix is given by $\sum_{i=0}^{\infty} {\mathcal{A}_{T12|1}}^i = (I - \mathcal{A}_{T12|1})^{-1}$ and its $(j,k)^{th}$ entry captures the expected number of transitions from state k to state j before absorption. The expected number of steps to absorption, given that the initial state is state $\{0\}$, is equal to the sum of the elements of the 0th column of the fundamental matrix. In fact, the rate of convergence to absorption depends on the largest eigenvalue of the substochastic matrix $\mathcal{A}_{T12|1}$ (because the rate of convergence of matrix $\mathcal{A}^m_{T12|1}$ is captured by the rate of convergence of $\lambda^m_{12|1}$, where $\lambda_{12|1}$ is the largest eigenvalue of $\mathcal{A}_{T12|1}$ and m denotes the number of steps (Kemeny et al., 1976)).

Let us now consider the scenario where neither $S_{12|1}$ nor $S_{12|2}$ includes the inconsistent state NC in their set of states. Then the bound on the probability of error will not go to zero; in fact, it will always be equal to half, thereby providing us with no useful information. This scenario corresponds to the case where all output sequences that can be produced by S_1 can also be produced by S_2 and vice versa. For this to be true, S_1 and S_2 need to be equivalent, i.e., generate the same regular language (i.e., the same set of output sequences). Of course, although the set of output sequences is the same for both models, the probabilities associated with an output sequence could be different for each model. However, the *posteriors* of the candidate models in this case would be strictly greater than zero for any observation sequence; hence, the error in the MAP decision will always be nonzero. We can check whether S_1 and S_2 are equivalent using standard approaches with complexity $O((|Q_{1D}| + |Q_{2D}|)^2)$ (Hopcroft et al., 2001). We can also easily check equivalence by using S_{1D2DNC} and S_{1DNC2D} which we have already constructed: if the inconsistent state in either S_{1D2DNC} or S_{1DNC2D} (and

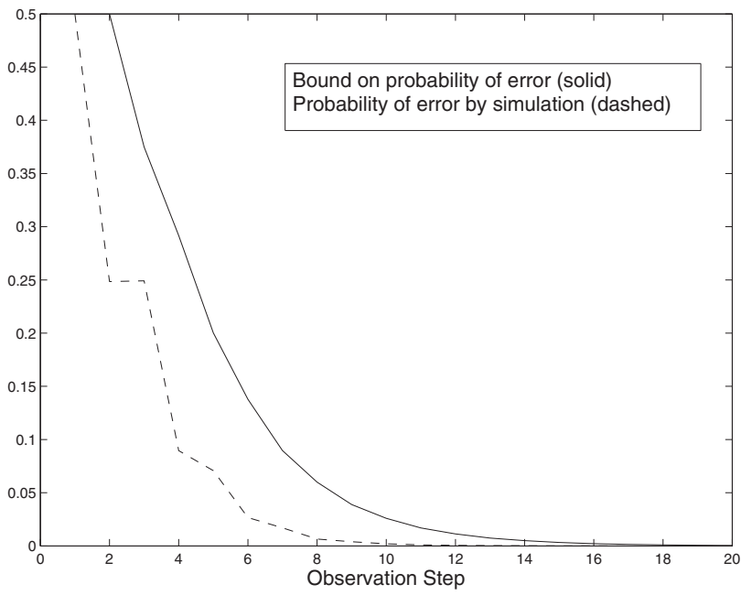


Fig. 8. Plot of the bound on the probability of error (solid) and the empirical probability of error obtained by simulation (dashed) in Example 2, both shown as functions of the observation step.

consequently $S_{12|1}$ or $S_{12|2}$) can be reached starting from the initial state, then the two models are not equivalent.

If MC $S_{12|1}$ has no absorbing state and MC $S_{12|2}$ has only the state NC as an absorbing state, then the bound on the probability of error goes to the value $\frac{P_1}{2}$. This case corresponds to the language generated by S_1 being a subset of the language generated by S_2, i.e., the set of output sequences that can be produced by S_1 can also be produced by S_2. To check for this scenario, we can check whether the inconsistent state in S_{1D2DNC} is reachable from the initial state. We formalize the above discussion in the following proposition.

Proposition 3: For two HMMs S_1 and S_2, the upper bound on the probability of error for the classification decision

- tends to zero exponentially with the number of observation steps, if (and only if) each of FSMs S_{1D2DNC} and S_{1DNC2D} has a unique absorbing state, namely the inconsistent state;
- tends to the value $P_1/2$ exponentially with the number of observation steps, if FSM S_{1D2DNC} has no inconsistent state and FSM S_{1DNC2D} has a unique absorbing state, i.e., the inconsistent state;
- tends to the value $P_2/2$ exponentially with the number of observation steps, if FSM S_{1D2DNC} has no inconsistent state and FSM S_{1DNC2D} has a unique absorbing state, i.e., the inconsistent state;
- is equal to $1/2$, if (and only if) FSMs S_{1D2DNC} and S_{1DNC2D} have no inconsistent states. □

Example 2 (continued): As shown in Figures 6 and 7, each of $S_{12|1}$ and $S_{12|2}$ have NC as the unique absorbing state. Thus, the bound on the probability of error goes to zero exponentially with the observation time; this is evident in Figure 8 where we assume equal *priors* (i.e., $P_1 =$

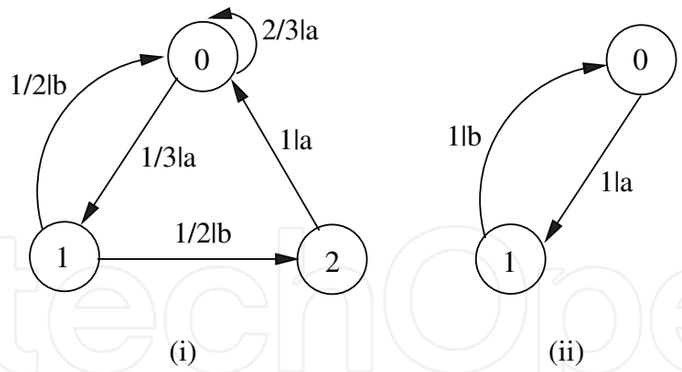


Fig. 9. State transition diagram of (i) HMM S_1 and (ii) HMM S_3 of Example 3.

$P_2 = 0.5$). After running simulations, half with the actual model being S_1 and the other half with the actual model being S_2, we obtain the empirical probability of error given S_1 (and given S_2) by recording the fraction of simulations for which the classifier incorrectly decided S_2 (and S_1, respectively). The empirical probability of error as a function of the observation step is shown in Figure 8. The expected time to absorption for $S_{12|1}$ is calculated to be 6.8 steps and the expected time to absorption for $S_{12|2}$ is 6.3 steps; hence, for equal *priors*, the expected number of steps for the bound on the probability of error to become zero is 6.55 steps. □

Example 3: Consider HMM S_1 and HMM S_3 shown in Figure 9, and assume equal *priors*. Notice that any output sequence that can be produced by S_3 can also be produced by S_1; thus, there is no inconsistent state in $S_{13|3}$ and the probability $\sum_{y_1^L : P(S_1|y_1^L)=0} P(y_1^L \mid S_3)$ is always equal to zero. On the other hand, $S_{13|3}$ has a unique absorbing inconsistent state. According to the proposition, we expect the bound on the probability of error to go to $P_1/2 = 0.25$. From Figure 10 we see that, although the bound on the probability of error indeed goes to 0.25 (as expected), the simulations show that the empirical probability of error goes to zero as the number of steps increases; for this set of candidate models, the bound is not tight, even as the number of observation steps goes to infinity. □

5. Posterior probability calculation with corrupted observations

So far, we have assumed that the output sequence of the HMM is correctly observed by the classifier. Next, we consider the scenario where sensor failures may convert the output sequence to a corrupted observed sequence.

5.1 Sensor failure model

The output sequence $y_1^L =< y[1], y[2], ..., y[L] >$ produced by the system under classification may become corrupted due to noise or sensor unreliability. When sensor failures are possible, what is actually observed by the system, denoted by $z_1^{L_z} =< z[1], z[2], ..., z[L_z] >$, may be different from the output sequence. In fact, if sensor failures are allowed to insert and/or delete outputs, the length of the observed sequence $z_1^{L_z}$ may be different from the length of the output sequence (i.e., it could be that $L_z \neq L$). We consider sensor failures that may result in the deletion, insertion, or substitution of output symbols, or the transposition of adjacent output symbols. We assume that sensor failures are transient and occur independently at each observation step with certain (known) probabilities that could *depend* on the observation step, e.g., the probability of such transient errors could vary as a function of time. We also make

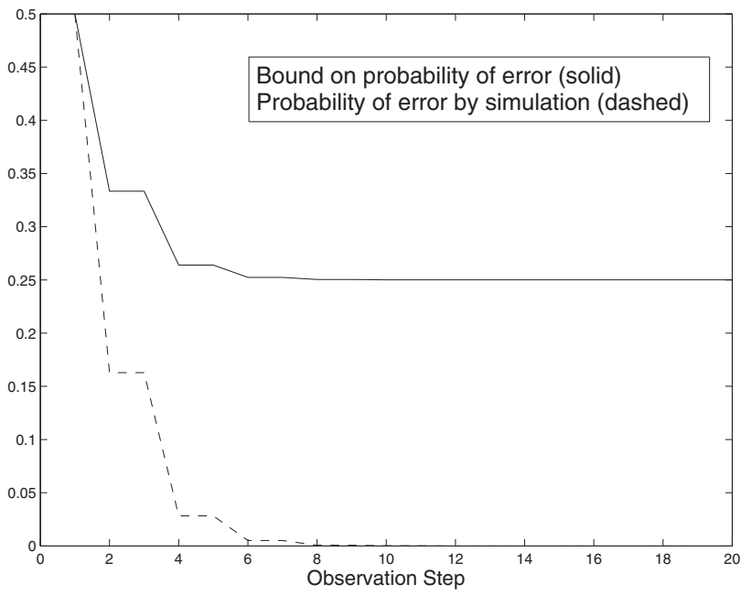


Fig. 10. Plot of the bound on the probability of error (solid) and the empirical probability of error obtained by simulation (dashed) in Example 3, both shown as functions of the observation step.

the reasonable assumption that sensor failures are conditionally independent from the HMM given the observation sequence.

If an output $\sigma \in Y$ is deleted by the sensor, then we do not observe the output, i.e., the deletion causes $\sigma \rightarrow \varepsilon$, where ε denotes the empty label. Similarly, if an output $\sigma \in Y$ is inserted by the sensor, then we observe σ instead of ε, i.e., the insertion causes $\varepsilon \rightarrow \sigma$. Also, if an output $\sigma_j \in Y$ is substituted by $\sigma_k \in Y$, then we observe σ_k, i.e., the substitution causes $\sigma_j \rightarrow \sigma_k$. Lastly, the corruption of subsequence $< \sigma_j \sigma_k >$ to $< \sigma_k \sigma_j >$ is referred to as a transposition error.

To perform the posterior probability calculation, we construct the trellis diagram of S as before but modify it to capture the sensor failures. Note that we call each column of the trellis diagram a *stage* to reflect the notion of an observation step. Deletions appear in the trellis diagram as vertical transitions within the same stage and insertions appear as one-step forward transitions. A substitution appearing at a particular observation step results in a change of the transition functionality of the HMM for that step. The transposition of two adjacent outputs appears in the trellis diagram as an erroneous transition that spans two columns.

Given the sensor failure model, we can assign probabilities to all types of errors on the observed sequence. Since the probabilities of sensor failures are known and are (conditionally) independent from the transition probabilities of the HMM, we can easily determine the probabilities associated with transitions in the modified trellis diagram that accounts for sensor failures. Due to space limitations, we focus on deletions which is the most challenging case of sensor failures because they may produce vertical cycles in the trellis diagram; the interested reader can find more details in (Athanasopoulou, 2007).

We assume that a deletion $d_{\sigma'}$ of output σ' occurs with known probability $p_{d_{\sigma'}}[m+1]$ at observation step $m+1$ when S is in a state from which a transition that outputs σ' is possible. Let $D = \{d_{\sigma_1}, d_{\sigma_2}, ..., d_{\sigma_{|D|}} \mid \sigma_1, \sigma_2, ..., \sigma_{|D|} \in Y\}$ be the set of deletions and define a function out to allow us to recover the corresponding output in the set Y given a deletion, i.e., $out(d_{\sigma'}) = \sigma'$. When constructing the trellis diagram, we assign probabilities to the transitions as follows:

1. each forward (normal) transition from state (m, j) to state $(m+1, k)$ associated with output σ is assigned probability

$$(1 - \sum_{\substack{\forall d_{\sigma'} \in D \text{ s.t.} \\ \delta(j, out(d_{\sigma'})) \neq \emptyset}} p_{d_{\sigma'}}[m]) \cdot \mathcal{A}_{\sigma}(k, j),$$

   where $(1 - \sum_{\substack{\forall d_{\sigma'} \in D \text{ s.t.} \\ \delta(j, out(d_{\sigma'})) \neq \emptyset}} p_{d_{\sigma'}}[m])$ is the probability that no deletion (possible from state j) occurs and where $\mathcal{A}_{\sigma}(k, j)$ is the probability of going from state j to state k while producing output σ;

2. each vertical transition (corresponding to deletions) from state (m, j) to state (m, k) is assigned probability

$$\sum_{\substack{\forall d_{\sigma'} \in D \text{ s. t.} \\ \delta(j, out(d_{\sigma'})) = k}} (p_{d_{\sigma'}}[m] \cdot \mathcal{A}_{\sigma'}(k, j)).$$

Note that other ways of defining these probabilities (or obtaining them from a particular sensor failure model) are possible, e.g., under different models of sensor failures. What is important (and a challenge) here are not the specific values of these probabilities but the structure of the trellis diagram (in particular the loops that are present).

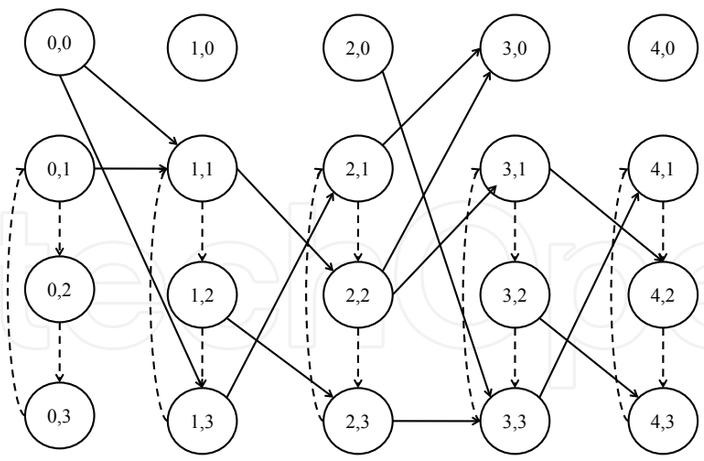


Fig. 11. Trellis diagram corresponding to S of Example 1 with deletions due to sensor failures (transition probabilities are not included for clarity).

Example 1 (continued): In S of Example 1 suppose that deletion of b may occur and that the observed sequence is $z_1^4 =< abcb >$. The output sequence y_1^L could be of any length; examples of possible output sequences are $< abbbcb >$, $< babcb >$, and $< babbcbbb >$. The

resulting trellis diagram is shown in Figure 11, where dashed arcs represent transitions due to the deletion of b. In this example, we assume that the deletion probability p_{d_b} is fixed for all observation steps. The probabilities of transitions are not included in the figure for clarity, but they can be computed as explained before the example. For instance, the (normal) transition from state $(0,0)$ to state $(1,3)$ when symbol a is observed has probability $\mathcal{A}_a(3,0)$, which is equal to the probability that S took a transition from state 0 to state 3 and produced output a. Similarly, the (erroneous) transition from state $(0,1)$ to state $(0,2)$ has probability $p_{d_b} \cdot \mathcal{A}_b(2,1)$, which is the probability that b was produced by S and then it was deleted by the sensors. □

5.2 Posterior probability calculation

The iterative algorithm described in Section 3 cannot be applied in the case of sensor unreliability (see Figure 11) because the vertical arcs within a stage can possibly form loops (as in our example) and be traversed any number of times (more loop traversals occur, of course, with lower probability). Next, we establish some notation which will help us perform the iteration.

We can view the trellis diagram for a given observed sequence $z_1^{L_z}$ as a probabilistic FSM H with $|Q_H| = (L_z+1) \cdot |Q|$ states and probabilities on transitions determined by the trellis diagram. The transition probabilities do not generally satisfy the Markovian property and the matrix $\mathcal{A}_H$ that describes the transition probabilities of FSM H is not stochastic. We can easily build H' by modifying H, so that the assigned transition probabilities produce a Markov chain and, in particular, an absorbing Markov chain. More specifically, we append $|Q|+1$ states as described in the following two steps:

1. We add an extra stage at the end of the trellis diagram, i.e., we add $|Q|$ states of the form (L_z+1, j) so that from each state of the form (L_z, j) there exists a transition with probability one to state (L_z+1, j), $j \in \{0,1,2,...,|Q|-1\}$; we also add a self-loop with probability one at each state of the form (L_z+1, j). We call each state of the form (L_z+1, j) a *consistent* state because H' being in that state implies that S is consistent with the observed sequence $z_1^{L_z}$.
2. We add state q_{in} to represent the *inconsistent* state, i.e., H is in state q_{in} when the observed sequence is not consistent with S. To achieve this, we add transitions from each state of FSM H to the inconsistent state q_{in} with probability such that the sum of the transition probabilities leaving each state is equal to one; we also add a self-loop at state q_{in} with probability one.

The resulting Markov chain H' has $|Q_{H'}| = (L_z+2) \cdot |Q| + 1$ states. The only self-loops in H' with probability one are those in the consistent states (of the form (L_z+1, j)) and in the inconsistent state (q_{in}). In fact, due to the particular structure of H' (and given that there is a nonzero probability to leave the vertical loop at each stage), the consistent and inconsistent states are the only absorbing states, while the rest of the states are transient. Therefore, when H' reaches its stationary distribution, only the absorbing states will have nonzero probabilities (summing up to one). We are interested in the stationary distribution of H' so that we can account for output sequences y_1^L of any length that correspond to the observed sequence $z_1^{L_z}$, i.e., for $L = L_z, L_z+1, \ldots, \infty$. (Recall that without sensor failures we have $L = L_z$.)

More formally, we arrange the states of H' in the order $(0,0)$, $(0,1), \ldots, (0,|Q|-1), (1,0), (1,1), \ldots, (1,|Q|-1), \ldots, (L_z+1,0), (L_z+1,1), \ldots, (L_z+1,|Q|-1), q_{in}$. Let $\pi_{H'}[0]$ be a vector with $|Q_{H'}|$ entries, each of which represents the initial probability of a

corresponding state of H'. We are interested in the stationary probability distribution of H' denoted by

$$\pi_{H'} = \lim_{n\to\infty} \pi_{H'}[n] = \lim_{n\to\infty} \mathcal{A}_{H'}^n \cdot \pi_{H'}[0],$$

where the state transition matrix $\mathcal{A}_{H'}$ of H' is in its canonical form given by

$$\mathcal{A}_{H'} = \begin{bmatrix} \mathcal{A}_H & 0 \\ R & I \end{bmatrix}. \tag{1}$$

Here $\mathcal{A}_H$ captures the behavior of the transient states of H', the $(|Q|+1)\times|Q_H|$ matrix R captures the transitions from the transient states to the absorbing states, 0 is a matrix of appropriate dimensions with all zero entries, and I is the identity matrix of appropriate dimensions. Note that since H' is an absorbing Markov chain, the limit $\lim_{n\to\infty}\mathcal{A}_{H'}^n$ exists and it is given by

$$\lim_{n\to\infty} \mathcal{A}_{H'}^n = \begin{bmatrix} 0 & 0 \\ (I-\mathcal{A}_H)^{-1}R & I \end{bmatrix}, \tag{2}$$

where $(I-\mathcal{A}_H)^{-1}$ is called the fundamental matrix (Kemeny et al., 1976).
The only nonzero entries of $\pi_{H'}$ are those that correspond to the consistent and inconsistent states. In fact, the probability that H' ends up in a consistent state is equal to the complement of the probability that H' ends up in the inconsistent state and it is equal to the probability of the observed sequence $z_1^{L_z}$ given the FSM model S, i.e.,

$$P(z_1^{L_z} \mid S) = \sum_{j=L_z\cdot|Q|}^{|Q_H|-1} \pi_{H'}(j) = 1 - \boldsymbol{\pi}_{H'}(|Q_{H'}|).$$

Note that the above approach for the posterior probability calculation is consistent with the one obtained in (Athanasopoulou et al., 2010) using the notion of the observation FSM and its composition with S.

5.3 Iterative posterior probability calculation with corrupted observations

In this section we exploit the structure of matrix $\mathcal{A}_{H'}$ which captures the transition probabilities of H' to perform the posterior probability calculations in an efficient manner. We first define the following submatrices which will be used to express $\mathcal{A}_{H'}$.

- Matrices $B_{m,m+1},\ m = 0,1,...,L_z$, capture the transitions from any state of H' at stage m to any state of H' at stage $m+1$. They can be obtained from $\mathcal{A}_{H'}$ as $B_{m,m+1}(k,j) = \mathcal{A}_{H'}((m+1)\cdot|Q|+k, m\cdot|Q|+j)$, where $k,j = 0,1,...,|Q|-1$.
- Matrices $B_m,\ m = 0,1,...,L_z$, capture the vertical transitions and account for deletions. They can be obtained from $\mathcal{A}_{H'}$ as $B_m(k,j) = \mathcal{A}_{H'}(m\cdot|Q|+k, m\cdot|Q|+j)$, where $k,j = 0,1,...,|Q|-1$. (Note that if deletions occur at each observation step with the same probability, then $B_m = B,\ m = 0,1,...,L_z$.)
- C^T is a row vector with entries $C^T(j) = 1 - \sum_{k=1}^{|Q_H|}\mathcal{A}_H(k,j)$, for $j = 1,2,...,|Q_H|$, i.e., C^T ensures that the sum of each column of $\mathcal{A}_{H'}$ is equal to 1.

We should note here that an alternative way to compute the block matrices $B_{m,m+1}$ and B_m directly, without the help of the modified trellis diagram, is by using the following equations:

$$B_m(k,j) = \sum_{\substack{\forall d_{\sigma'} \in D \text{ s.t.} \\ \delta(j,out(d_{\sigma'}))=k}} (p_{d_{\sigma'}}[m] \cdot \mathcal{A}_{\sigma'}(k,j)),$$

$$B_{m,m+1}(k,j) = (1 - \sum_{\substack{\forall d_{\sigma'} \in D \text{ s.t.} \\ \delta(j,out(d_{\sigma'}))\neq\emptyset}} p_{d_{\sigma'}}[m]) \cdot \mathcal{A}_{z[m+1]}(k,j),$$

where $k, j = 0, 1, ..., |Q| - 1$ and $p_{d_{\sigma'}}[m]$, $\mathcal{A}_\sigma$ were defined earlier.
Using the above notation, we can decompose the matrix $\mathcal{A}_{H'}$ in blocks and express it as

$$\mathcal{A}_{H'} = \left[\begin{array}{cccccc|cc} B_0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ B_{0,1} & B_1 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & B_{1,2} & B_2 & ... & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & ... & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & B_{L_z-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & ... & B_{L_z-1,L_z} & B_{L_z} & 0 & 0 \\ \hline 0 & 0 & 0 & ... & 0 & I - B_{L_z} & I & 0 \\ & & & C^T & & & 0 & 1 \end{array}\right].$$

Recall that the matrix $\mathcal{A}_{H'}$ is in its canonical form (see (1)), where the submatrices $\mathcal{A}_H$ and R are given by

$$\mathcal{A}_H = \left[\begin{array}{cccccc} B_0 & 0 & 0 & ... & 0 & 0 \\ B_{0,1} & B_1 & 0 & ... & 0 & 0 \\ 0 & B_{1,2} & B_2 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & ... & \vdots & \vdots \\ 0 & 0 & 0 & ... & B_{L_z-1} & 0 \\ 0 & 0 & 0 & ... & B_{L_z-1,L_z} & B_{L_z} \end{array}\right],$$

$$R = \left[\begin{array}{ccccc} 0 & ... & 0 & I - B_{L_z} \\ & & C^T & \end{array}\right].$$

In the initial probability distribution vector $\boldsymbol{\pi}_{H'}[0]$ the only nonzero entries are its first $|Q|$ entries, i.e., $\pi_H[0] = (\boldsymbol{\rho}[0]\ 0\ ...\ 0)^T$, where $\boldsymbol{\rho}[0]$ denotes the initial probability distribution of S (i.e., it is a $|Q|$-dimensional vector, whose j^{th} entry denotes the probability that S is initially in state j). Recall that $\pi_{H'}$ denotes the stationary probability distribution vector of H' and has nonzero entries only in the absorbing states, i.e., its last $|Q|+1$ states. Hence, given the observed sequence $z_1^{L_z}$, we can express $\boldsymbol{\pi}_{H'}$ as $\pi_{H'} = (0\ ...\ 0\ \boldsymbol{\rho}[L_z+1]\ \ p_{in}[L_z+1])^T$, where $\boldsymbol{\rho}[L_z+1]$ captures the probabilities of the consistent states and $p_{in}[L_z+1]$ denotes the probability of the inconsistent state. The following equations hold:

$$\begin{aligned} \pi_{H'} &= \lim_{n\to\infty} \mathcal{A}_{H'}^n \cdot \pi_{H'}[0] \\ (0\ ...\ 0\ \boldsymbol{\rho}[L_z+1]\ p_{in}[L_z+1])^T &= \lim_{n\to\infty} \mathcal{A}_{H'}^n \cdot (\boldsymbol{\rho}[0]\ ...\ 0)^T \\ \boldsymbol{\rho}[L_z+1] &= \lim_{n\to\infty} \mathcal{A}_{H'}^n(L_z+2,1) \cdot \boldsymbol{\rho}[0]. \end{aligned}$$

Therefore, in order to calculate the probability of the consistent states we only need the initial distribution of S and the $(L_z+2,1)^{st}$ block of the matrix $\lim_{n\to\infty} \mathcal{A}_{H'}^n$.

Next, we argue that we can compute the $\lim_{n\to\infty} \mathcal{A}_H^n(L_z+2,1)$ with much less complexity than the standard computation in (2). For simplicity, we illustrate this for the case where the observed sequence is of length 2, i.e., $L_z = 2$. The state transition matrix $\mathcal{A}_{H'}$ is given by

$$\mathcal{A}_{H'} = \left[\begin{array}{ccc|cc} B_0 & 0 & 0 & 0 & 0 \\ B_{0,1} & B_1 & 0 & 0 & 0 \\ 0 & B_{1,2} & B_2 & 0 & 0 \\ \hline 0 & 0 & I-B_2 & I & 0 \\ & C^T & & 0 & 1 \end{array}\right].$$

We can compute by induction the matrix $\mathcal{A}_{H'}$ raised to the power n and consequently find the limit of $\mathcal{A}_{H'}^n(4,1)$ as n tends to infinity as follows:

$$\begin{aligned} \lim_{n\to\infty} \mathcal{A}_H^n(4,1) &= \lim_{n\to\infty} \sum_{j_1+j_2+j_3+j_4=n-3} I^{j_4}\ I\ B_2^{j_3} B_{1,2} B_1^{j_2} B_{0,1} B_0^{j_1} \\ &= (I+B_2+B_2^2+...)B_{1,2}(I+B_1+B_1^2+...)B_{0,1}(I+B_0+B_0^2+...) \\ &= \left(\sum_{j=0}^{\infty} B_2^j\right) B_{1,2} \left(\sum_{j=0}^{\infty} B_1^j\right) B_{0,1} \left(\sum_{j=0}^{\infty} B_0^j\right) \\ &= (I-B_2)^{-1} B_{1,2} (I-B_1)^{-1} B_{0,1} (I-B_0)^{-1}\ . \end{aligned}$$

(The detailed proof is omitted due to space limitations; it can be found in (Athanasopoulou, 2007).)

As explained earlier, we are interested in the state probabilities of the *consistent* states, which will be given by the entries of the vector $\boldsymbol{\rho}[3] = \lim_{n\to\infty} \mathcal{A}_H^n(4,1)\ \boldsymbol{\rho}[0]$. From the above equation, we get

$$\boldsymbol{\rho}[3] = ((I-B_2)^{-1} B_{1,2} (I-B_1)^{-1} B_{0,1} (I-B_0)^{-1}) \boldsymbol{\rho}[0].$$

To simplify notation let us define $B'_{m,m+1} = (I-B_{m+1})^{-1} B_{m,m+1}$, $m = 0,1,2$. Hence, $\boldsymbol{\rho}[3] = B'_{1,2} B'_{0,1} (I-B_0)^{-1} \boldsymbol{\rho}[0]$.

Generalizing the above computation for any number of observations L_z, the vector that describes the probabilities of the consistent states given the observed sequence $z_1^{L_z}$ satisfies

$$\boldsymbol{\rho}[L_z+1] = \left(\prod_{i=0}^{L_z-1} B'_{m,m+1}\right) (I-B_0)^{-1}\ \boldsymbol{\rho}[0], \tag{3}$$

where $B'_{m,m+1} = (I-B_{m+1})^{-1} B_{m,m+1}$, $m = 0,1,...,L_z$.

By inspection of (3) we notice that the computation of $\boldsymbol{\rho}[L_z+1]$ can be performed iteratively as follows:

$$\begin{aligned} &\boldsymbol{\rho}[1] = B'_{0,1} (I-B_0)^{-1}\ \boldsymbol{\rho}[0], \\ &\boldsymbol{\rho}[m+1] = B'_{m,m+1}\ \boldsymbol{\rho}[m],\ m = 1,2,...,L_z, \end{aligned} \tag{4}$$

where $\boldsymbol{\rho}[m+1]$ represents the probability of consistent states given the observed sequence z_1^m. The probability that the observed sequence $z_1^{L_z}$ was produced by the particular FSM S is equal to the sum of the elements of the state probability distribution vector $\boldsymbol{\rho}[L_z+1]$, i.e., $P(z_1^{L_z} \mid S) = \sum_{j=0}^{|Q|-1} \boldsymbol{\rho}[L_z+1](j)$.

The above algorithm is described in pseudocode below.

Algorithm

Input: Matrices $\{B_m\}$, $\{B_{m,m+1}\}$, where $m = 0, 1, \ldots, L_z$; an observed (possibly corrupted) output sequence $z_1^{L_z} = \{z[1], z[2], ..., z[L_z]\}$ and the initial probability distribution $\boldsymbol{\rho}[0]$.

1. Initialization. Let $m = 0$, $z[m] = \emptyset$,
   compute $B'_{0,1} = (I - B_1)^{-1} B_{0,1}$,
   compute $\boldsymbol{\rho}[1] = B'_{0,1}(I - B_0)^{-1}\, \boldsymbol{\rho}[0]$.
2. Let $m = 1$.
3. Consider the output $z[m]$, do
   compute $B'_{m,m+1} = (I - B_{m+1})^{-1} B_{m,m+1}$,
   compute $\boldsymbol{\rho}[m+1] = B'_{m,m+1}\boldsymbol{\rho}[m]$.
4. $m = m + 1$.
5. If $m = L_z + 1$, Goto 6; else Goto 3.
6. Compute $P(z_1^{L_z} \mid S) = \sum_{j=0}^{|Q|-1} \boldsymbol{\rho}[L_z + 1](j)$. □

To gain some intuition regarding the iteration, let us consider for now the case of reliable sensors. This case corresponds to matrices B_m in $\mathcal{A}_{H'}$ being equal to zero, which means that there are no vertical transitions (transitions within the same stage) in the trellis diagram. In this case, iteration (4) becomes

$$\boldsymbol{\rho}[m+1] = B_{m,m+1}\, \boldsymbol{\rho}[m], \;\; m = 0, 1, \ldots, L_z.$$

This latter equation is the same as the iterative equation that appeared in Section 3 for the case of reliable sensors (where we denoted $B_{m,m+1}$ by $\mathcal{A}_{y[m+1]}$). Intuitively, every time we get a new observation we update the current probability vector by multiplying it with the state transition matrix of S that corresponds to the new observation. With the above intuition at hand, we now return to the case of sensor failures. Here, we also need to take into consideration the fact that any number of vertical transitions may occur. Therefore, every time we get a new observation $z[m+1]$, we multiply the current probability vector with the state transition matrix of S that corresponds to the new observation (as before) and also with $(I - B_{m+1})^{-1} = \sum_{j=0}^{\infty} B_{m+1}^j$ thereby taking into account the vertical transitions at stage $m + 1$.

The matrices $B_{m,m+1}$ have dimension $|Q| \times |Q|$, while the matrix $\mathcal{A}_H$ has dimension $|Q_H| \times |Q_H|$, where $|Q_H| = (L_z + 2) \cdot |Q|$. If we calculate $\boldsymbol{\pi}_H$ without taking advantage of the structure of $\mathcal{A}_H$, the computational complexity is proportional to $O(((L_z + 2) \cdot |Q|)^3) = O(L_z^3 \cdot |Q|^3)$. If we use the iterative approach instead, the computational complexity reduces significantly to $O((L_z + 2) \cdot (|Q|^2 + |Q|^3)) = O(L_z \cdot |Q|^3)$ (each stage requires the inversion of a new $|Q| \times |Q|$ matrix which has complexity $O(|Q|^3)$ and dominates the computational complexity associated with that particular stage). If sensor failure probabilities remain invariant at each stage, then matrix B_m at stage m only needs to be inverted once and the complexity of the iterative approach is $O((L_z + 2) \cdot |Q|^2 + |Q|^3) = O(L_z \cdot |Q|^2 + |Q|^3)$. In addition to complexity gains, the iterative nature of the calculations allows us to monitor the system under classification online and calculate the probability of the observed sequence at each observation step by first updating the state probability vector and then summing up its entries.

5.4 Discussion

We discuss here how the presented iterative algorithm relates to other known algorithms. As mentioned earlier, the techniques that we use relate to the evaluation problem in HMMs or the parsing problem in probabilistic automata with vertical loops in the resulting trellis diagram. The forward algorithm is used to evaluate the probability that a given sequence of observations is produced by a certain HMM. To do that, the standard forward algorithm uses the HMM to build a trellis diagram based on the given sequence of observations and performs the likelihood calculation online. However, the standard forward algorithm cannot handle the existence of vertical cycles in the trellis diagram. Ways around vertical cycles in the trellis diagram have been suggested in speech recognition applications, where HMMs are used to model speech patterns (Rabiner, 1989), (Ephraim and Merhav, 2002), (Jelinek, 1998), (Poritz, 1988) and may include null transitions (i.e., the HMM may move from the current state to the next state without producing any output (Jelinek, 1998), (Bahl and Jelinek, 1975)), as well as in the area of pattern recognition, where one may have to deal with null transitions when solving the parsing problem for a given probabilistic finite state automaton (Vidal et al., 2005).

While in most HMM formulations one deals with state observations, several authors have also studied the evaluation problem in HMMs with transition observations, including null transitions (i.e., transitions with no outputs). For instance, the authors of (Bahl and Jelinek, 1975), (Bahl et al., 1983), (Jelinek, 1998), develop HMMs that capture the generation of codewords in speech recognition applications via observations that are associated with transitions rather than states. These HMMs also include null transitions, i.e., transitions that change the state without producing outputs. To avoid loops in the resulting trellis diagram, the authors of (Bahl and Jelinek, 1975) eliminate them via an appropriate modification of the underlying HMM *before* constructing the trellis diagram. In (Vidal et al., 2005), an algorithm is presented to solve the parsing problem in pattern recognition applications for the case where null transitions exist in a probabilistic finite-state automaton (PFSA) model (as pointed out in (Dupont et al., 2005), HMMs are equivalent to PFSAs with no final probabilities). The authors evaluate recursively the probability that a sequence is produced by a λ-PFSA (i.e., a PFSA that includes null transitions) and their approach can be shown, after some manipulation, to be a special case of the algorithm we described here.

Also related to the described likelihood computation algorithm is the well-known Viterbi algorithm (Forney, 1973), (Viterbi, 1967), which solves the related problem of maximum-likelihood decoding of convolutional codes by choosing the most likely state sequence based on a given sequence of observations. In fact, the Viterbi algorithm is a dynamic programming algorithm amenable to online use with applications in various fields; for example, in HMMs it finds the most likely (hidden) state sequence corresponding to the observed output sequence (Rabiner, 1989). Note that, in contrast to the Viterbi algorithm, the maximum likelihood approach in this work considers the total probability of *all* paths (rather than the cost of the *most likely* path) which can be generated from the initial state(s) to the final state(s). As a consequence of this requirement, the Viterbi algorithm (or variations of it) do not obtain a solution for the problem considered here. However, it is worth pointing out that the Viterbi algorithm has been frequently suggested as a suboptimal alternative for likelihood evaluation in some applications (Rabiner, 1989). Also note that a modified Viterbi algorithm was proposed in (Bouloutas et al., 1991) to identify the correct strings of data given an FSM representation of a possibly erroneous output sequence; in (Hart and Bouloutas, 1993) the same authors proposed a channel inversion algorithm for correcting symbol sequences that

have been corrupted by errors (associated with costs) which can be described in terms of finite state automata. The work in (Amengual and Vital, 1998) proposes an efficient implementation of the Viterbi algorithm to perform error-correcting parsing using an FSM and an error model. The Viterbi algorithm can handle the case where vertical cycles exist by unwrapping cycles so that each state on the cycle is visited at most once (to avoid adding cost or decreasing the probability of the path — recall that the Viterbi algorithm only searches for the most likely path).

Before closing this discussion, it is worth pointing out that the techniques used to solve our problem also relate to maximum a posteriori (MAP) decoding of variable length codes (VLC). In MAP decoding of VLC, symbols that are generated by a source may give rise to a different number of output bits and, given an observed bit sequence, one has to recover the symbols that are transmitted according to the source codewords. The authors in (Bauer and Hagenauer, 2000), (Guyader et al., 2001) constructed a two-dimensional (symbol and bit) trellis diagram representation of the variable length coded data and then applied the BCJR algorithm (Bahl et al., 1974) to do either symbol or bit decoding. This setup resembles the setup described here when only a *finite* number of sensor failures exist in the observed sequence (in such case, one can appropriately enlarge the underlying model since no vertical cycles are present). In our formulation, however, deletions may cause vertical loops in the associated trellis diagram resulting in an infinite number of possible output sequences matching a given sequence of observations. As a consequence, the standard BCJR algorithm is insufficient for solving our problem and the techniques that we describe are crucial in obtaining the probabilities needed for our trellis-based analysis.

To summarize, the approach presented here is more general than the aforementioned approaches because it can handle different kinds of loops at different stages of the trellis diagram (loops in our setup are not introduced by null transitions in the underlying model but rather by errors in the observed sequence which can occur with time-varying probabilities). Thus, the associated probabilities in the trellis diagram can be changing with time (which cannot be handled as effectively using the techniques in (Vidal et al., 2005) or in (Bahl and Jelinek, 1975)). The problem is that the modification of the underlying model so as to match the requirements of these earlier approaches results in a quite complex HMM (in which the evaluation problem can still benefit from the techniques we describe here). Therefore, the attractive feature of the described iterative algorithm for likelihood calculation is that it can handle time-varying and infinite number of sensor failures (or, equivalently, vertical cycles in the trellis diagram) with reduced complexity.

6. Conclusions

In this chapter we considered the problem of optimal classification of HMMs in order to minimize the probability of error. Given two candidate HMMs along with their *prior* probabilities, a classifier that aims to minimize the probability of error (misclassification) needs to determine which candidate model has most likely produced the observation sequence of the system under classification. In order to find the *a priori* probability that the classifier makes an incorrect decision as a function of the observation step, one could in principle calculate all possible observation sequences (of that length), find their probabilities, and determine their contribution to the probability of misclassification. Since the complexity for calculating the exact probability of error can be prohibitively high, we described ways to obtain an upper bound on this probability, as well as necessary and sufficient conditions for the bound to go exponentially to zero as the number of observation steps increases.

Additionally, we presented an iterative methodology to calculate the probability of a given HMM under possibly erroneous observations. Our goal was to determine which of two candidate HMMs has most likely produced the observed sequence under sensor failures which can corrupt the observed sequence. Using the trellis diagram which includes all possible sequences consistent with both the observations and the given HMMs, we described an iterative algorithm that efficiently computes the total probability with which each HMM, together with a combination of sensor failures, can generate the observed sequence. The described algorithm can deal with vertical loops in the trellis diagrams which can be caused by output deletions.
As examples we considered the classification of CpG islands in DNA sequences and a failure diagnosis scenario where a system is classified as faulty or non-faulty depending on the observation sequence. The described techniques can be easily extended to classification of several hidden Markov models with applications in various fields such as document or image classification, pattern recognition, and bioinformatics.

7. Acknowledgement

The authors would like to thank Christoforos Keroglou for help with Example 1.b and for proof reading several parts of this chapter.

8. References

J. C. Amengual and E. Vidal. (1998). Efficient error-correcting Viterbi parsing, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 20, no. 10, pp. 1109–1116.

E. Athanasopoulou, L. Li, and C. N. Hadjicostis. (2010). Maximum likelihood failure diagnosis in finite state machines under unreliable observations, *IEEE Trans. Automatic Control,* vol. 55, no. 3, pp. 579–593.

E. Athanasopoulou and C. N. Hadjicostis. (2008). Probability of error bounds for failure diagnosis and classification in hidden Markov models, *IEEE Proc. of the 47th IEEE Conf. on Decision and Control,* pp. 1477-1482.

E. Athanasopoulou. (2007). *Diagnosis of finite state models under partial or unreliable observations.* Ph.D. Thesis, Department of Electrical and Computer Engineering, University of Illinois, IL.

L. R. Bahl, F. Jelinek and R. L. Mercer. (1983). A maximum likelihood approach to continuous speech recognition, *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. PAMI-5, no. 2, pp. 179–190.

L. R. Bahl and F. Jelinek. (1975). Decoding for channels with insertions, deletions and substitutions with applications to speech recognition," *IEEE Trans. Information Theory,* vol. IT-21, no. 4, pp. 404–411.

L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. (1974). Optimal decoding of linear codes for minimizing label error rate, *IEEE Trans. Information Theory,* vol. IT-20, pp. 284–287.

R. Bauer and J. Hagenauer. (2000). Symbol-by-symbol MAP decoding of variable length codes, in *Proc. 3rd ITG Conf. on Source and Channel Coding,* pp. 111–116.

A. Bouloutas, G. W. Hart, and M. Schwartz. (1991). Two extensions of the Viterbi algorithm, *IEEE Trans. Information Theory,* vol. 37, no. 2, pp. 430–436.

P. Dupont, F. Denis, and Y. Esposito. (2005). Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms, *Pattern Recognition,* vol. 38, no. 9, pp. 1349–1371.

R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, Cambridge, UK.

Y. Ephraim and N. Merhav. (2002). Hidden Markov processes, *IEEE Trans. Information Theory,* vol. 48, no. 6, pp. 1518–1569.

M. Fatemi, M. M. Pao, S. Jeong, E. N. Gal-Yam, G. Egger, D. J. Weisenberger, and P. A. Jones. (2005). Footprinting of mammalian promoters: use of a *CpG* DNA methyltransferase revealing nucleosome positions at a single molecule level, *Nucleic Acids Research*, vol. 33, no. 20, pp. e176.

G. D. Forney, Jr.. (1973). The Viterbi algorithm, *Proc. IEEE,* vol. 61, pp. 268–278.

K. S. Fu. (1982). *Syntactic Pattern Recognition and Applications.* Prentice-Hall, New York, NY.

M. Gardiner-Garden and M. Frommer (1987). *CpG* Islands in vertebrate genomes, *Journal of Molecular Biology*, vol. 196, no. 2, pp. 261-282.

A. Guyader, E. Fabre, C. Guillemot, and M. Robert. (2001). Joint source-channel turbo decoding of entropy-coded sources," *IEEE Journal on Selected Areas in Communications,* vol. 19, no. 9, pp. 1680–1696.

G. W. Hart and A. T. Bouloutas. (1993). Correcting dependent errors in sequences generated by finite-state processes, *IEEE Trans. Information Theory,* vol. 39, no. 4, pp. 1249–1260.

J. E. Hopcroft, R. Motwani, and J. D. Ullman. (2001). *Introduction to Automata Theory, Languages, and Computation,* Addison Wesley, Reading, MA.

F. Jelinek. (1998). *Statistical Methods for Speech Recognition,* The MIT Press, Cambridge, MA.

J. G. Kemeny, J. L. Snell, and A. W. Knapp. (1976). *Denumerable Markov Chains.* 2nd ed., Springer-Verlag, New York, NY.

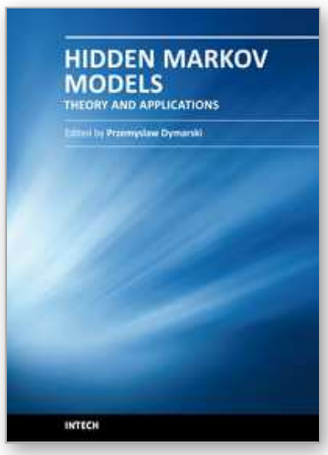
T. Koski. (2001). *Hidden Markov Models of Bioinformatics.* Kluwer Academic Publishers, Boston, MA.

A. M. Poritz. (1998). Hidden Markov models: A guided tour, *Proc. 1988 IEEE Conf. Acoustics, Speech, and Signal Processing,* vol. 1, pp. 7–13.

L. R. Rabiner. (1989). A tutorial on hidden Markov models and selected applications in speech recognition, *Proc. IEEE,* vol. 77, no. 2, pp. 257–286.

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. (2005). Probabilistic finite-state machines–part I, *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 7, pp. 1013–1025.

A. D. Viterbi. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, *IEEE Trans. Inform. Theory,* vol. IT-13, pp. 260–269.



Hidden Markov Models, Theory and Applications

Edited by Dr. Przemyslaw Dymarski

ISBN 978-953-307-208-1
Hard cover, 314 pages
Publisher InTech
Published online 19, April, 2011
Published in print edition April, 2011

Hidden Markov Models (HMMs), although known for decades, have made a big career nowadays and are still in state of development. This book presents theoretical issues and a variety of HMMs applications in speech recognition and synthesis, medicine, neurosciences, computational biology, bioinformatics, seismology, environment protection and engineering. I hope that the reader will find this book useful and helpful for their own research.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen