Information Sciences 197 (2012) 105–130

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A survey of techniques for incremental learning of HMM parameters

Wael Khreich [a,*], Eric Granger [a], Ali Miri [b], Robert Sabourin [a]

[a] Laboratoire d'imagerie, de Vision et d'intelligence Artificielle (LIVIA), École de Technologie Supérieure, Université du Québec, 1100 Notre-Dame Ouest, Montreal, QC, Canada
[b] School of Computer Science, Ryerson University, Toronto, Canada

ARTICLE INFO

Article history:
Received 15 January 2010
Received in revised form 16 January 2012
Accepted 3 February 2012
Available online 23 February 2012

Keywords:
Incremental learning
On-line learning
Hidden Markov model
Limited training data
Expectation–maximization
Recursive estimation

ABSTRACT

The performance of Hidden Markov Models (HMMs) targeted for complex real-world applications are often degraded because they are designed a priori using limited training data and prior knowledge, and because the classification environment changes during operations. Incremental learning of new data sequences allows to adapt HMM parameters as new data becomes available, without having to retrain from the start on all accumulated training data. This paper presents a survey of techniques found in literature that are suitable for incremental learning of HMM parameters. These techniques are classified according to the objective function, optimization technique and target application, involving block-wise and symbol-wise learning of parameters. Convergence properties of these techniques are presented along with an analysis of time and memory complexity. In addition, the challenges faced when these techniques are applied to incremental learning is assessed for scenarios in which the new training data is limited and abundant. While the convergence rate and resource requirements are critical factors when incremental learning is performed through one pass over abundant stream of data, effective stopping criteria and management of validation sets are important when learning is performed through several iterations over limited data. In both cases managing the learning rate to integrate pre-existing knowledge and new data is crucial for maintaining a high level of performance. Finally, this paper underscores the need for empirical benchmarking studies among techniques presented in literature, and proposes several evaluation criteria based on non-parametric statistical testing to facilitate the selection of techniques given a particular application domain.

1. Introduction

The Hidden Markov Model (HMM) is a stochastic model for sequential data. It is a stochastic process determined by the two interrelated mechanisms – a latent Markov chain having a finite number of states, and a set of observation probability distributions, each one associated with a state. At each discrete time instant, the process is assumed to be in a state, and an observation is generated by the probability distribution corresponding to the current state. The HMM is termed *discrete* if the output alphabet is finite, and *continuous* if the output alphabet is not necessarily finite, e.g., each state is governed by a parametric density function [32,34,80].

Theoretical and empirical results have shown that, given an adequate number of states and a sufficiently rich set of data, HMMs are capable of representing probability distributions corresponding to complex real-world phenomena in terms of

* Corresponding author.
E-mail addresses: wael.khreich@livia.etsmtl.ca (W. Khreich), eric.granger@etsmtl.ca (E. Granger), ali.miri@ryerson.ca (A. Miri), robert.sabourin@etsmtl.ca (R. Sabourin).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2012.02.017

simple and compact models [8,11]. This is supported by the success of HMMs in various practical applications, where it has become a predominant methodology for design of automatic speech recognition systems (ASR) [3,48,80]. It has also been successfully applied to various other fields, such as signature verification [31,50] communication and control [32,46], bioinformatics [30,58], computer vision [13,81], and computer and network security [22,61,97]. For instance, in the area of computer and network security, a growing number of HMM applications are found in intrusion detection systems (IDSs). HMMs have been applied either to anomaly detection, to model normal patterns of behavior, or in misuse detection, to model a predefined set of attacks. HMM applications in anomaly and misuse detection have emerged in both main categories of IDS - host-based IDS [97,100,22,61,51] and network-based IDS [39,93]. Moreover, HMMs have recently begun to emerge in wireless IDS applications [19,55].

In many practical applications, the collection and analysis of training data is expensive and time consuming. As a consequence, data for training an HMM is often limited in practice, and may over time no longer be representative of the underlying data distribution. However, the performance of a generative model like the HMM depends heavily on the availability of an adequate amount of representative training data to estimate its parameters, and in some cases its topology. In static environments, where the underlying data distribution remains fixed, designing a HMM with a limited number of training observations may significantly degrade performance. This is also the case when new information emerges in dynamically-changing environments, where underlying data distribution varies or drifts in time. A HMM that is trained using data sampled from the environment will therefore incorporate some uncertainty with respect to the underlying data distribution [29].

It is common to acquire additional training data from the environment at some point in time after a pattern classification system has originally been trained and deployed for operations. Since limited training data is typically employed in practice, and underlying data distribution are susceptible to change, a system based on HMMs should allow for adaptation in response to new training data from the operational environment or other sources (see Fig. 1). The ability to efficiently adapt HMM parameters in response to newly-acquired training data, through*incremental learning*, is therefore an undisputed asset for sustaining a high level of performance. Indeed, refining a HMM to novelty encountered in the environment may reduce its uncertainty with respect to the underlying data distribution.

Following definitions by Langley [63] and Polikar et al. [77], an HMM that performs incremental learning can independently learn one new block of training data at a time. With incremental learning, HMM parameters should be efficiently updated from new data[1] without requiring access to the previously-learned training data. In addition, parameters should be updated without corrupting previously-acquired knowledge.

Standard techniques for estimating HMM parameters involve batch learning, based either on specialized Expectation–Maximization (EM) techniques [26], such as the Baum-Welch (BW) algorithm [6], or on numerical optimization techniques, such as the Gradient Descent algorithm [67]. In either case, HMM parameters are estimated over several training iterations, until some objective function, e.g., maximum likelihood over some independent validation data, is maximized. For a batch learning technique, a fixed-length sequence $O = o_1, o_2, \ldots, o_T$ of T training observations o_i is assumed to be available throughout the training process. Assuming that O is assembled into a block D of training data, each training iteration typically involves observing *all* sub-sequences in D prior to updating HMM parameters.

Given a new block D_2 of training data, a HMM that has previously been trained on D_1 through batch learning cannot accommodate D_2 without accumulating and storing all training data in memory, and training from the start using all of the cumulative data, $D_2 \bigcup D_1$. Otherwise, the previously-acquired knowledge may be corrupted, thereby compromising HMM performance. As illustrated in Fig. 2, the HMM probabilities to be optimized may become trapped in local optima of the new cost function associated with $D_2 \bigcup D_1$. In fact, probabilities estimated after training on D_1 may not constitute a good starting point for training on D_2. Updating HMM parameters on all data using some batch learning technique may therefore incur a significant cost in terms of processing time and storage requirements. The time and memory complexity of standard techniques grows linearly with the length, T, and number of training sequences R, and quadratically with the number of HMM states, N.

As an alternative, several on-line learning techniques proposed in literature may be applied for incremental learning. These include techniques based on EM, numerical optimization and recursive estimation, and assume the observation a of stream of data. Some of these techniques are designed to update HMM parameters at a symbol level (symbol-wise), while others update parameters at a sequence level (block-wise). Techniques for on-line symbol-wise learning, also referred to as recursive or sequential estimation techniques, are designed for situations in which training symbols are received one at a time, and HMM parameters are re-estimated upon observing each new symbol. Techniques for on-line block-wise learning are designed for situations in which training symbols are organized into a block of one or more sub-sequences, and HMM parameters are re-estimated upon observing each new sub-sequence of symbols. In either case, HMM parameters are updated from new training data, without requiring access to the previously-learned training data, and potentially without corrupting previously acquired knowledge.

The main advantage of applying these techniques to incremental learning is the ability to sustain a high level of performance yet bound the memory requirements, since there is no need for storing the data from previous training phases. Furthermore, since training is only performed on the new training sequences, and not on all accumulated data, on-line learning

[1] A block of training data is defined as a sequence of training observations that has been segmented into overlapping or non-overlapping sub-sequences according to a user-defined window size.

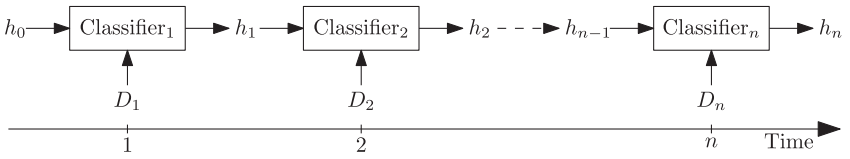

Fig. 1. A generic incremental learning scenario where blocks of data are used to update the classifier in an incremental fashion over a period of time. Let D_1, $D_2, \ldots, D_n$ be the blocks of training data available to the classifier at discrete instants in time $t_1, t_2, \ldots, t_n$. The classifier starts with initial hypothesis h_0 which constitutes the prior knowledge of the domain. Thus, h_0 gets updated to h_1 on the basis of D_1, and h_1 gets updated to h_2 on the basis of data D_2, and so forth [18].

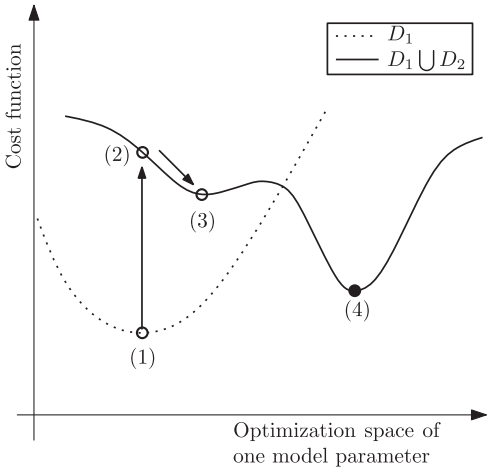

Fig. 2. An illustration of the degeneration that may occur with batch learning of a new block of data. Suppose that the dotted curve represents the cost function associated with a system trained on block D_1, and that the plain curve represents the cost function associated with a system trained on the cumulative data $D_1 \bigcup D_2$. Point (1) represents the optimum solution of batch learning performed on D_1, while point (4) is the optimum solution for batch learning performed on $D_1 \bigcup D_2$. If point (1) is used as a starting point for incremental training on D_2 (point (2)), then it will become trapped in the local optimum at point (3).

would also lower time complexity needed to learn new data. Finally, incremental learning may provide a powerful tool in a human-centric approach, where domain experts may be called upon to gradually design and update HMMs as the operational environment unfolds.

This paper contains a survey of techniques that apply to incremental learning of HMM parameters.[2] These techniques are classified according to objective function, optimization technique, and target application that involve block-wise and symbol-wise learning of parameters. An analysis of their convergence properties and of their time and memory complexity is presented. The applicability of these techniques is assessed for incremental learning scenarios in which new data is either abundant or limited. Finally, the advantages and shortcomings of these techniques are outlined, providing the key issues and guidelines for their application in different learning scenarios.

This paper is structured according to five sections. The next section briefly reviews the batch learning techniques employed to estimate HMM parameters, and introduces the formalism needed to support subsequent sections. Section 3 provides a taxonomy of on-line learning techniques form literature that apply for incremental learning of HMM parameters. An analysis of their convergence properties and resource requirements is provided in Section 4. Then, an analysis of their potential applicability in different incremental learning scenarios is presented in Section 5. This paper concludes with a discussion of the main challenges to be addressed for incremental learning of HMM parameters.

2. Batch learning of HMM parameters

A discrete-time finite-state HMM consists of N hidden states in the finite-state space $S = \{S_1, \ldots, S_N\}$ of the Markov process. Starting from an initial state S_i, determined by the initial state probability distribution π_i, at each discrete-time instant, the process transits from state S_i to state S_j according to the transition probability distribution a_{ij}. The process then emits a

[2] A predefined topology and permissible transitions between the states (e.g., ergodic or temporal) is assumed when learning HMM parameters. In many real-world applications, the best topology is determined empirically. Although adapting HMM topologies, or jointly HMM parameters and topologies, to new data may have a significant impact on performance, this issue is beyond the scope of the paper.

symbol v according to the output probability distribution $b_j(v)$ of the current state S_j (see Fig. 3). With a discrete HMM, the output $b_j(v)$ is finite, and with a continuous HMM, the output alphabet is is governed by a parametric density function.

Let $q_t \in S$ denotes the state of the process at time t, where $q_t = i$ indicates that the state is S_i at time t. An observation sequence of length T is denoted by $O = o_1, \ldots, o_T$, where o_t is the observation at time t. The sub-sequence $o_m, o_{m+1}, \ldots, o_n$, $n > m$, by the concise notation $o_{m:n}$. The HMM is then usually parametrized by $\lambda = (\pi, A, B)$, where

$\pi = \{\pi_i\}$ denotes the vector of initial state probability distribution, $\pi_i \triangleq P(q_1 = i)_{1 \leqslant i \leqslant N}$,
$A = \{a_{ij}\}$ denotes the state transition probability distribution, $a_{ij} \triangleq P(q_{t+1} = j | q_t = i)_{1 \leqslant i,\, j \leqslant N}$,
$B = \{b_j(k)\}$ denotes the state output probability distribution,

- $b_j(k) \triangleq P(o_t = v_k | q_t = j)_{1 \leqslant j \leqslant N, 1 \leqslant k \leqslant M}$ for a finite and discrete alphabet $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$, $v_k \in \mathcal{R}^L$ with M distinct observable symbols.
- $b_j(v) \triangleq f(o_t = v | q_t = j)_{1 \leqslant j \leqslant N}$ for an infinite and continuous alphabet $\mathcal{V} = \{v | v \in \mathcal{R}^L\}$. For instance, if the observation density for each state in the HMM is described by a univariate Gaussian distribution[3] $b_j(o_t) \sim \mathcal{N}(\mu_j, \sigma_j)$, for a scalar observation o_t, with a mean μ and a variance σ^2 then the state output density is given by:

$$b_j(o_t | \mu_j, \sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j}} \exp\left[-\frac{(o_t - \mu_j)^2}{2\sigma_j^2}\right] \tag{1}$$

For a finite-discrete HMM, both A, B and π' (the transpose of vector π) are row stochastic, which impose the following constraints:

$$\sum_{j=1}^{N} a_{ij} = 1 \forall i, \sum_{k=1}^{M} b_j(k) = 1 \ \forall j, \quad \text{and} \ \sum_{i=1}^{N} \pi_i = 1 \tag{2}$$

$$a_{ij}, b_j(k), \pi_i \in [0, 1], \quad \forall ijk \tag{3}$$

For a continuous HMM, other constraints arise depending on the probability density function (pdf) of the states. For example, in the Gaussian distribution case, (1), one must ensure that the standard deviation is always positive, $\sigma_j > 0$, $\forall j$.

Given a HMM initialized according to the constraints described so far, there are three tasks of interest – the evaluation, decoding, and training tasks [80]. The rest of this section focuses on batch learning techniques applied to address the third task, and in particular estimating HMM parameters, along with the definitions needed for future sections. For further details regarding the HMM, the reader is referred to the extensive literature [32,34,80].

Standard techniques for estimating HMM parameters $\lambda = (\pi, A, B)$ involve *batch learning* are based either on expectation–maximization or numerical optimization techniques. With batch learning, a finite-length sequence $O = o_1, o_2, \ldots, o_T$ of T training observations o_i is assumed to be available throughout the training process. The parameters are estimated over several training iterations, until some objective function is maximized. Each training iteration involves observing all the observation symbols prior to updating HMM parameters.

2.1. Objective functions

The estimation of HMM parameters is frequently performed according to the Maximum Likelihood Estimation (MLE) criterion. Other criteria such as the maximum mutual information (MMI), and minimum discrimination information (MDI) also be used for estimating HMM parameters [35]. However, the widespread use of the MLE for HMM is a result of its attractive statistical properties – *consistency* and *asymptotic normality* – proven under general conditions [66,10]. MLE consists in maximizing the likelihood $P(o_{1:T}|\lambda)$ or equivalently the log-likelihood of the training data with regard to the model parameters:

$$\begin{aligned} \ell_T(\lambda) \triangleq \log P(o_{1:T}|\lambda) &= \log \sum_S P(o_{1:T}, S|\lambda) = \log \sum_S P(o_{1:T}|S, \lambda) P(S|\lambda) = \log \sum_{q \in S} \prod_{t=1}^{T} P(q_t|q_{t-1}, \lambda) P(o_t|q_t, \lambda) \\ &= \log \sum_{q \in S} \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1}, q_t} b_{q_t}(o_t) = \log \sum_{q \in S} \prod_{t=1}^{T} a_{q_{t-1}, q_t} b_{q_t}(o_t); \text{by considering } a_{q_0, q_1} = \pi_{q_1} \end{aligned} \tag{4}$$

over the model parameter space Λ:

$$\lambda^* = \arg\max_{\lambda \in \Lambda} \ell_T(\lambda) \tag{5}$$

There is no known analytical solution to HMM parameters estimation since the log-likelihood function (4) depends on the unknown probability values of the latent states, S. In practice, iterative optimization procedures such as the expectation–maximization or the standard numerical optimizations techniques, which are described in the following sub-sections, are usually employed. In order to proceed iteratively any numerical optimization procedure must also evaluate the log-likelihood

[3] It is usually termed as finite-state Markov chains in white Gaussian noise in control and communication community. It is also referred to as normal HMM.

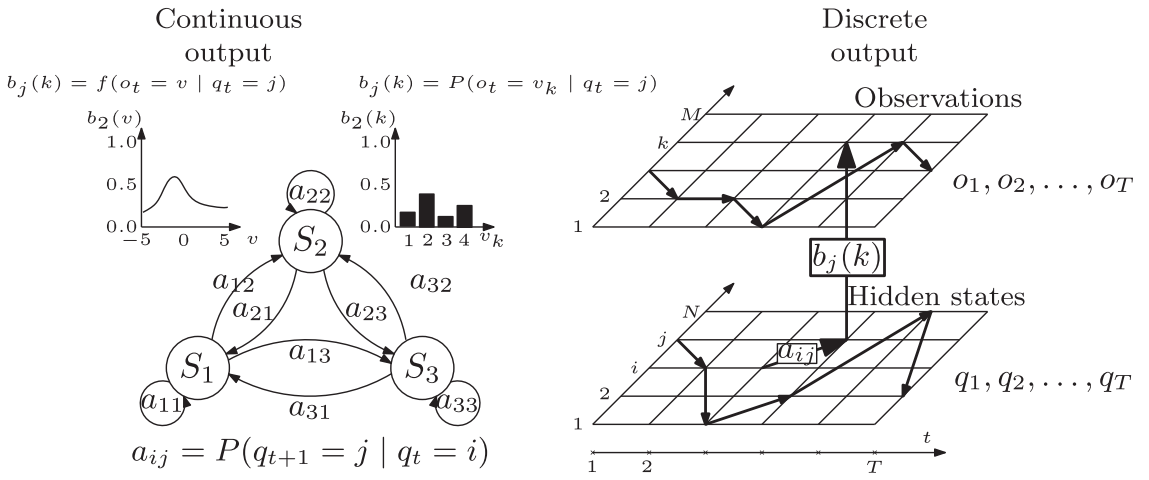

Fig. 3. An illustration of an ergodic three states HMM with either continuous or discrete output observations (left). A discrete HMM with N states and M symbols transits between the hidden states q_t, and generates the observations o_t (right).

function at any value. However, a direct evaluation of the log-likelihood function (4) requires a summation over all possible hidden state sequences $q_{1:T} \in S$, which has a prohibitively costly time complexity $\mathcal{O}(TN^T)$.

Fortunately, there exists efficient recursive procedures for evaluating the log-likelihood values as well as estimating the conditional state (6) and joint state (7) densities associated with a given observation sequence $o_{1:t}$:

$$\gamma_{\tau|t}(i) \triangleq P(q_\tau = i|o_{1:t}, \lambda) \tag{6}$$

$$\xi_{\tau|t}(i,j) \triangleq P(q_\tau = i, q_{\tau+1} = j|o_{1:t}, \lambda) \tag{7}$$

in order to provide an optimal estimate, in the minimum mean square error (MMSE) sense, for the unknown state $\hat{q}_{\tau|t}(i)$ frequency – the key problem for HMM parameters estimation:

$$\hat{q}_{\tau|t}(i) = E\{q_\tau = i|o_{1:t}\} = \sum_{\tau=1}^{t} \gamma_{\tau|t}(i) \tag{8}$$

In estimation theory, this conditional estimation problem is called *filtering* if $\tau = t$; prediction if $\tau > t$, and smoothing if $\tau < t$. The smoothing problem is termed *fixed-point* smoothing when computing the $E\{q_\tau|o_{1:t}\}$for a fixed τ and increasing $t = \tau, \tau + 1, \ldots$, *fixed-lag* smoothing when computing the $E\{q_\tau|o_{1:t+\Delta}\}$ for a fixed lag $\Delta > 0$, and *fixed-interval* smoothing when computing the $E\{q_\tau|o_{1:T}\}$ for all $\tau = 1, 2, \ldots, t, \ldots, T$.

The fixed-interval smoothing problem is therefore to find the best estimate of the states at any time conditioned on the entire observations sequence, which is typically performed in batch learning using the Forward-Backward (FB) [21,6,5] or the Forward-Filtering Backward-Smoothing (FFBS) [34,17] algorithms. Typically, fixed-interval smoothing algorithms involve an estimation of the filtered state density, for $t = 1, \ldots, T$:

$$\gamma_{t|t}(i) \triangleq P(q_t = i|o_{1:t}, \lambda) \tag{9}$$

which provides the best estimate of the conditional distribution of states given the past and present observations. It can be computed from the predictive state density (10), which provides the best estimate of the conditional distribution of states given only the past observations,

$$\gamma_{t|t-1}(i) \triangleq P(q_t = i|o_{1:t-1}, \lambda); \gamma_{1|0}(i) = \pi_i \tag{10}$$

according to the following recursion:

$$\gamma_{t|t}(i) = \frac{\gamma_{t|t-1}(i) b_i(o_t)}{\sum_{j=1}^{N} \gamma_{t|t-1}(j) b_j(o_t)} \tag{11}$$

The predictive state density (10) at time $t + 1$ can be then computed from the filtered state density (9) at time t:

$$\gamma_{t+1|t}(j) = \sum_{i=1}^{N} \gamma_{t|t}(i) a_{ij} \tag{12}$$

Given an observation sequence $o_{1:T}$, the log-likelihood is therefore evaluated with a time complexity of $\mathcal{O}(N^2T)$:

$$\ell_T(\lambda) = \sum_{t=1}^{T} \log P(o_t|o_{1:t-1}) = \sum_{t=1}^{T} \log \sum_{i=1}^{N} P(q_t = i, o_t|o_{1:t-1}) = \sum_{t=1}^{T} \log \sum_{i=1}^{N} b_i(o_t)\gamma_{t|t-1}(i) \tag{13}$$

An alternative computation of the likelihood can be written as sum of the α-variables [80]:

$$\ell_T(\lambda) = \log \sum_{i=1}^{N} \alpha_T(i) \tag{14}$$

where

$$\alpha_t(i) \triangleq P(q_t = i, o_{1:t}|\lambda), \tag{15}$$

The elements of α are not probability measures unless normalized, and hence susceptible to underflow when applied to compute the likelihood of a long sequence of observation. In practice, the α-variables are commonly scaled by their summation at each time step [67,80]:

$$\bar{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_i \alpha_t(i)} = \frac{P(q_t = i, o_{1:t}\lambda)}{P(o_{1:t}\lambda)} = P(q_t = i|o_{1:t}, \lambda)$$

making for a *filtered* state estimate (9), as detailed in [52].

The predictive state density (10) is in fact a fixed-lag smoothing with one symbol look-ahead ($\Delta = 1$). In situations where some delay or latency between receiving the observations and updating HMM parameters can be tolerated, incorporating more lag provides stability and improved performance over filtering estimation [1]. This is because the latter relies only on the information that is available at the current time. Although they provide an approximation of the conditional state distributions, both filtering and fixed-lag smoothing have been the core of several on-line learning techniques, presented in Section 3, since their recursions can go on indefinitely in time providing the state estimates without (or with a small fixed) delay. In addition, they require a linear memory complexity $\mathcal{O}(N)$ that is independent of the observation length T, while maintaining a low computational complexity $\mathcal{O}(N^2T)$.

The backward recursion of the FFBS (or FB) algorithm exploits the filtered state estimates in order to provide an exact smoothed estimate for the states, for $t = T - 1, \ldots, 1$:

$$\gamma_{t|T}(i) = \sum_{j=1}^{N} \xi_{t|T}(i,j) \tag{16}$$

$$\xi_{t|T}(i,j) = \frac{\gamma_{t|t}(i)a_{ij}}{\sum_{i=1}^{N} \gamma_{t|t}(i)a_{ij}} \gamma_{t+1|T}(j) \tag{17}$$

Although fixed-interval smoothing is the standard for batch learning, waiting until the last observation before providing the state estimates incurs long delays as well as a large memory complexity $\mathcal{O}(NT)$, to store the filtered state estimate for the entire observation sequence (see [52] for more details). These issues are prohibitive for on-line learning.

A Forward Only (FO) fixed-interval smoothing algorithm has been proposed as an alternative to reduce the memory requirement of the FB or FFBS algorithms [87,33,23]. The basic idea is to directly propagate all smoothed information in the forward pass:

$$\sigma_t(i,j,k) \triangleq \sum_{\tau=1}^{t-1} P(q_\tau = i, q_{\tau+1} = j, q_t = k|o_{1:t}, \lambda),$$

which represents the probability of having made a transition from state S_i to state S_j at some point in the past ($\tau < t$) and of ending up in state S_k at the current time t. At the next time step, $\sigma_{t+1}(i,j,k)$ can be recursively computed from $\sigma_t(i,j,k)$ using:

$$\sigma_t(i,j,k) = \sum_{n=1}^{N} \sigma_{t-1}(i,j,n)a_{nk}b_k(o_t) + \alpha_{t-1}(i)a_{ik}b_k(o_t)\delta_{jk} \tag{18}$$

from which the smoothed state densities can be obtained, at time t, by marginalizing over the states. The advantage of the FO algorithm is that it provides the exact smoothed state estimate at each time step with the linear memory complexity $\mathcal{O}(N)$. However, this is achieved at the expenses of increasing the time complexity to $\mathcal{O}(N^4T)$ as can be seen in the four-dimensional recursion (18). As described in Section 3, several authors have proposed the FO algorithm for on-line learning of HMM parameters.

2.2. Optimization techniques

Maximum-likelihood (ML) parameter estimation in HMMs can be carried out using either the expectation maximization (EM) [6,26] or standard numerical optimization techniques [76,101]. This section describes both estimation procedures for HMMs.

2.2.1. Expectation–maximization

The EM is a general iterative method to find the MLE of the parameters of an underlying distribution given a data set when the data is incomplete or has missing values. EM alternates between computing an expectation of the likelihood (E-step) – by including the latent variables as if they were observed – and a maximization of the expected likelihood (M-step) found on the E-step. The parameters found in the M-step are then used to initiate another iteration until a *monotonic* convergence to a stationary point of the likelihood [26].

In the context of HMM, the basic idea consists of optimizing an auxiliary $\mathcal{Q}$-function (also known as the intermediate quantity of EM):

$$\mathcal{Q}_T(\lambda, \lambda^{(k)}) = E_{\lambda^{(k)}}\{\log P(o_{1:T}, q_{1:T}|\lambda)|o_{1:T}, \lambda^{(k)}\} = \sum_{q\in S} P(o_{1:T}, q_{1:T}|\lambda^{(k)}) \log P(o_{1:T}, q_{1:T}|\lambda) \tag{19}$$

which is the expected value of the *complete-data* log-likelihood ($\log P(o_{1:T}, q_{1:T}|\lambda)$). By assuming the probability values of the hidden states, the $\mathcal{Q}$-function is therefore easier to optimize than the *incomplete-data* log-likelihood (4). For a particular state sequence $q \in S$, the complete data log-likelihood is given by:

$$\log P(o_{1:T}, q_{1:T}|\lambda) = \log\left[\pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}} b_{q_{t+1}}(o_{t+1})\right]$$

The $\mathcal{Q}$-function for this particular state sequence (q) can then be explicitly expressed in terms of HMM parameters:

$$\mathcal{Q}(\lambda, \lambda^{(k)}) = \sum_{q\in S} P(o_{1:T}, q_{1:T}|\lambda^{(k)}) \log \pi_{q_1} + \sum_{q\in S} P(o_{1:T}, q_{1:T}|\lambda^{(k)}) \sum_{t=1}^{T-1} \log a_{q_t q_{t+1}} + \sum_{q\in S} P(o_{1:T}, q_{1:T}|\lambda^{(k)}) \sum_{t=1}^{T-1} \log b_{q_{t+1}}(o_{t+1})$$

Finally, by marginalizing over all state sequences in the state space S the $\mathcal{Q}$-function has the following form (for a discrete output HMM):

$$\mathcal{Q}_T(\lambda, \lambda^{(k)}) = \sum_{i=1}^{N} \gamma_{1|T}^{(k)}(i) \log \pi_i + \sum_{t=1}^{T-1} \sum_{i=1}^{N} \sum_{j=1}^{N} \xi_{t|T}^{(k)}(i,j) \log a_{ij} + \sum_{t=1}^{T} \sum_{j=1}^{M} \gamma_{t|T}^{(k)}(j) \delta_{o_t v_m} \log b_j(m) \tag{20}$$

Each term on the right hand side can be maximized individually using Lagrange multipliers subject to the relevant constraints (2). The solutions provide the same update formulas as those provided with the Baum-Welch algorithm below, (21)–(23).

Starting with an initial guess $\lambda^{(0)}$, subsequent iterations, $k = 1, 2, \ldots$, of the EM consist in:

E-step: computing the auxiliary function $\mathcal{Q}(\lambda, \lambda^{(k)})$
M-step: determining the model that maximizes $\mathcal{Q}$, $\lambda^{(k+1)} = \arg\max_{\lambda} \mathcal{Q}(\lambda, \lambda^{(k)})$

If instead of maximizing $\mathcal{Q}$, we find some $\lambda^{(k+1)}$ that increases the likelihood (partial M-step) then it is called Generalized EM (GEM), which is also guaranteed to converge [99].

The Baum-Welch (BW) algorithm [7,6] is a specialized EM algorithm for estimating HMM parameters. Originally introduced to estimate the parameters when provided with a single discrete observations sequence [6], it was later extended for multiple observation sequences [67,68] and for continuous observations [69,49]. Given a sequence $o_{1:T}$ of T observations, an HMM with N states, and an initial guess $\lambda^{(0)}$, the BW algorithm proceeds as follows. During each iteration k, the FB or FF-BS computes the expected number of state transitions and state emissions based on the current model (E-step), and then re-estimated the model parameters (M-step) using:

$$\pi_i^{(k+1)} = \gamma_{1|T}^{(k)}(i) = (\text{expected frequency of state } S_i \text{ at } t = 1) \tag{21}$$

$$a_{ij}^{(k+1)} = \frac{\sum_{t=1}^{T-1} \xi_{t|T}^{(k)}(i,j)}{\sum_{t=1}^{T-1} \gamma_{t|T}^{(k)}(i)} = \frac{\text{expected \# of trans. from } S_i \to S_j}{\text{expected \# of trans. from } S_i} \tag{22}$$

$$b_j^{(k+1)}(m) = \frac{\sum_{t=1}^{T} \gamma_{t|T}^{(k)}(j) \delta_{o_t v_m}}{\sum_{t=1}^{T} \gamma_{t|T}^{(k)}(j)} = \frac{\text{expected \# of times in } S_j \text{ observing } v_m}{\text{expected \# of times in } S_j} \tag{23}$$

Each iteration of the E-step and M-step is guaranteed to increase the likelihood of the observations giving the new model until a convergence to a stationary point of the likelihood [6,5,26].

For a continuous HMM only the last term of (20) and hence (23) must be changed according to the state parametric density. For instance, for an HMM with Gaussian state densities (1), the state outputs update are given by:

$$\widehat{\mu_j} = \frac{\sum_{t=1}^{T} \gamma_t(j) o_t}{\sum_{t=1}^{T} \gamma_t(j)}, \quad \text{and } \hat{\sigma}_j^2 = \frac{\sum_{t=1}^{T} \gamma_t(j)(o_t - \mu_j)^2}{\sum_{t=1}^{T} \gamma_t(j)}$$

For multiple independent observation sequences, Levinson et al. [67] proposed averaging the smoothed conditional densities, (16) and (17), over all observation sequences in the E-step, then updating the model parameters (M-step). Li et al. [68] propose using a combinatorial method on individual observations probabilities, rather than their product, to overcome the independence assumption. For both cases, re-estimating parameters with the standard EM requires accessing all of the sequences in the training block during each iteration.

2.2.2. Standard numerical optimization

While EM-based techniques indirectly optimize the log-likelihood function (4) through the complete-data log-likelihood (19), standard numerical optimization methods work directly with the log-likelihood function and its derivatives.

For instance, starting with an initial guess of the model λ^0, first order methods such as the gradient descent update the model at each iteration k using:

$$\lambda^{(k+1)} = \lambda^{(k)} + \eta_k \nabla_\lambda \ell_T(\lambda^{(k)}) \tag{24}$$

where the learning rate η_k decrease monotonically over training iterations to ensure that the sequence $\ell_T(\lambda^{(k)})$ is non-decreasing. It is usually chosen as to maximize the objective function in the search direction:

$$\eta_k = \arg\max_{\eta \geqslant 0} \ell_T[\lambda^{(k)} + \eta \nabla_\lambda(\lambda^{(k)})] \tag{25}$$

However due to its linear convergence rate, gradient descent methods could converge slowly in large optimization spaces.

Second order methods guarantee a faster convergence. For instance, the Newton-Raphson method updates the model at each iteration k using:

$$\lambda^{(k+1)} = \lambda^{(k)} - H^{-1}(\lambda^{(k)}) \nabla_\lambda \ell_T(\lambda^{(k)}) \tag{26}$$

The convergence rate is at least quadratic at the convergence point, for which the Hessian is negative definite. However, if the initial parameters are far from those of true model parameters, the convergence is not guaranteed. A learning rate η_k similar to (25) can be introduced to control the step length in the search direction. A polynomial interpolation of $\ell_T(\lambda)$ along the line-segment between $\lambda^{(k)}$ and $\lambda^{(k+1)}$ is usually used in practice, since it is often impossible to have an exact maximum of the line search. Nonetheless, the Hessian $H = \nabla^2_\lambda \ell_T(\lambda^{(k)})$ could be non-invertible or not negative semi-definite.

To avoid these issues, quasi-Newton methods use an adaptive matrix $W^{(k)}$ which provides an approximation of the Hessian at each iteration:

$$\lambda^{(k+1)} = \lambda^{(k)} + \eta_k W^{(k)} \nabla_\lambda \ell_T(\lambda^{(k)}) \tag{27}$$

where $W^{(k)}$ is negative definite to ensure that ascent direction is chosen. The numerical issues associated with the matrix inversion are therefore avoided, while still exhibiting the convergence rate of the Newton algorithms near the convergence point.

For a discrete HMM, the gradient of the log-likelihood $\nabla_\lambda \ell_T(\lambda^{(k)})$ can be computed using the state conditional densities obtained from the fixed-interval smoothing algorithms (16) and (17) as:

$$\frac{\partial \ell_T(\lambda^{(k)})}{\partial a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi^{(k)}_{t|T}(i,j)}{a_{ij}} \tag{28}$$

$$\frac{\partial \ell_T(\lambda^{(k)})}{\partial b_j(m)} = \frac{\sum_{t=1}^{T} \gamma^{(k)}_{t|T}(j) \delta_{o_t v_m}}{b_j(m)} \tag{29}$$

An alternative computation can be achieved by using a recursive computation of the gradient itself as described in sub-Section 3.2.2. Once the gradient of the likelihood is computed, the HMM parameters are then additively updated according to, for instance, the first order (24) or second order (27) methods. For multiple independent observation sequences, the derivatives with reference to each model parameter (28) and (29) must be accumulated and averaged over all observation sequences, prior to updating the model parameters. The reader is referred to Cappe et al. [16], Qin et al. [79] for more details on quasi-Newton, and to Turner [94] on Levenberg–Marquardt optimizations for HMMs.

Standard numerical optimization methods have to ensure parameter constraints, (2) and (3), explicitly through either a constrained optimization or a re-parametrization to reduce the problem to unconstrained optimization. Levinson et al. [67] detail the early implementation of a constrained optimization for HMM using Lagrange multipliers. Among the unconstrained transformation is the soft-max parametrization proposed in [4], which maps the bounded space (a,b) to the unbounded space (u,v):

$$a_{ij} = \frac{e^{u_{ij}}}{\sum_k e^{u_{ik}}} \text{ and } b(k) = \frac{e^{v_j(k)}}{\sum_z e^{v_j(z)}} \tag{30}$$

Other unconstrained parametrization consist of a projection of the model parameters onto the constraints domain, such as a simplex or a sphere. The parametrization on a simplex [56,88]

$$\triangle = \left\{ a_{ij} \in \mathbb{R}^{N-1} : \sum_{j=1}^{N-1} a_{ij} \leqslant 1, \text{ and } a_{ij} \geqslant 0 \right\} \tag{31}$$

enforces the stochastic constraints (2) by updating all but one of the parameters in each row, $a_{l_i} = 1 - \sum_{j \neq l_i}^{N} a_{ij}, 1 \leqslant l_i \leqslant N$. However, it does not ensure the positiveness of the parameters (3). One improvement consists in using a restricted projection to the space of positive matrices for some positive ϵ [2]:

$$\triangle_\epsilon = \{ a_{ij} \in \mathbb{R}^{N-1} : \sum_{j=1}^{N-1} a_{ij} \leqslant 1 - \epsilon, \text{ and } a_{ij} \geqslant \epsilon \} \tag{32}$$

Alternatively, parametrization on a sphere adequately enforces both constraints [25]:

$$\mathbb{S}^{N-1} = \left\{ s_{ij} : \sum_{j=1}^{N} s_{ij}^2 = 1 \right\} \tag{33}$$

where $a_{ij} = s_{ij}^2$. This also has the advantage that the constraint manifold is differentiable at all points.

2.2.3. Expectation–maximization vs. gradient-based techniques

For discrete output HMMs, the EM algorithm is generally easier to apply than gradient-based technique since derivatives, Hessian inversion, line-search, etc., are not required. It is numerically more robust against poor initialization values or when the model has large number of parameters. It also has a monotonic convergence propriety which is not maintained in gradient-based techniques, and ensures parameters constraints implicitly. However the rate of convergence of the EM can be very slow, it is usually linear in the neighborhood of a maximum [26]. This is comparable to gradient descent but slower than the quadratic convergence that can be achieved with second order or conjugate gradient techniques. Nevertheless, EM steps do not always involve closed-form expressions. In such cases the maximization must be carried out numerically. Another advantage of numerical optimization techniques is that they automatically yield an estimate of parameters variances [17,101]. Since there is no clear advantage of one technique on the other, hybrid algorithms have been proposed to take advantage from both techniques [14,62].

Given a block of data with R independent sub-sequences each of length T and an N state HMM, the time an complexity per iteration for EM and gradient-based techniques are $\mathcal{O}(RN^2T)$ and their memory complexity is $\mathcal{O}(NT)$. This is because they both rely on the fixed-interval smoothing algorithms (16) and (17) to compute the state densities for the E-step or the gradient of the log-likelihood. The difference is related to the procedures employed inside gradient-based techniques such as the line search, and to the speed of convergence (see [17, Chapter 10] for more details).

3. On-line learning of HMM parameters

Several on-line learning techniques from the literature may be applied to incremental learning of HMM parameters from new training sequences. Fig. 4 presents a taxonomy of techniques for on-line learning of HMM parameters, according to objective function, optimization technique, and target application. As shown in the figure, they fall in the categories of standard numerical optimization, expectation–maximization and recursive estimation, with the objective of either maximizing the likelihood estimation (MLE) criterion, minimizing the model divergence (MMD) of parameters penalized with the MLE, or minimizing the output or state prediction error (MPE).

The target application implies a scenario for data organization and learning. Some techniques have been designed for *block-wise* estimation of HMM parameters, while others for *symbol-wise* estimation of parameters. Block-wise techniques are designed for scenarios in which training symbols are organized into a block of sub-sequences and the HMM re-estimates its parameters after observing each sub-sequence. In contrast, symbol-wise techniques, also known as recursive or sequential techniques, are designed for scenarios in which training symbols are observed one at a time, from a stream of symbols, and the HMM parameters are re-estimated upon observing each new symbol. The rest of this section provides a survey of techniques for on-line learning of HMM parameters shown in Fig. 4.

3.1. Minimum Model Divergence (MMD)

The objective function now consists of maximization of the log-likelihood as well as minimization of parameter divergence using some entropy measures. A dual cost function that maximizes the log-likelihood while minimizing the divergence of HMM parameters, using an exponentiated gradient optimization framework [54], was first proposed for the block-wise case [86], then extended to symbol-wise case [41]. Both block- and symbol-wise algorithms described below, have been evaluated on speech data.

Block-wise Based on the exponentiated gradient framework [54], Singer and Warmuth [86] proposed an objective function for discrete HMMs that minimizes the divergence between old and new HMM parameters penalized by the negative log-likelihood of each sequence multiplied by a fixed positive learning rate ($\eta > 0$):

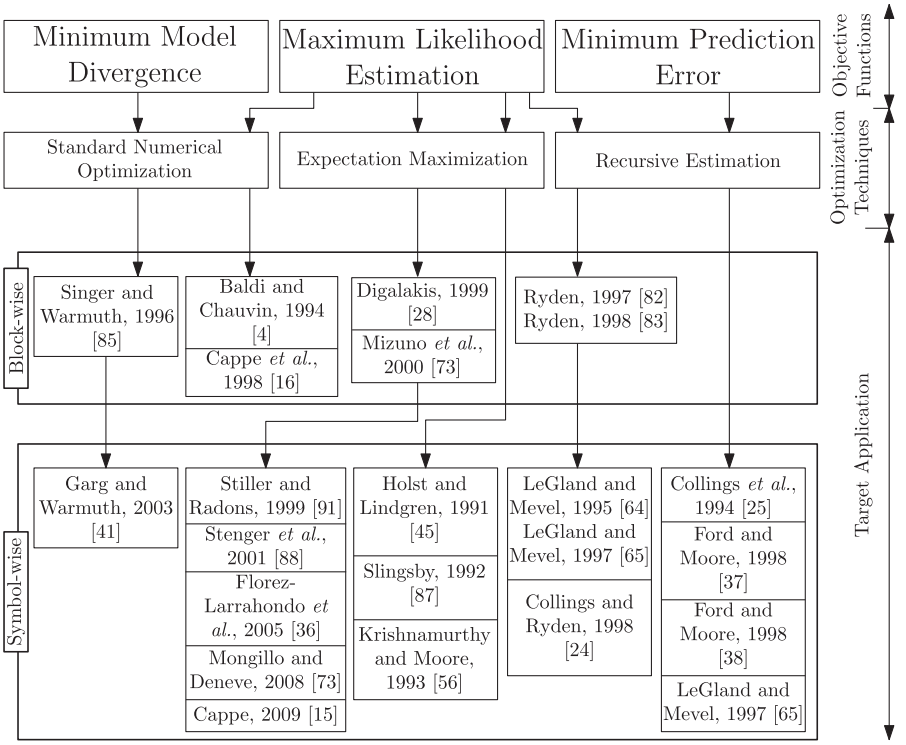

Fig. 4. Taxonomy of techniques for on-line learning of HMM parameters.

$$\hat{\lambda}^{(k+1)} = \arg\min_{\lambda}(KL(\lambda^{(k+1)}, \lambda^{(k)}) - \eta \ell_T(\lambda^{(k+1)}))$$

where $KL(\lambda^{(k+1)},\lambda^{(k)})$ is the Kullback-Leibler (KL) divergence between the distributions $\lambda^{(k+1)}$ and $\lambda^{(k)}$. KL is defined between two probability distributions $P_{\lambda^0}(o_{1:t})$ and $P_{\lambda}(o_{1:t})$ by

$$KL_t(P_{\lambda^0}\|P_{\lambda}) = E\left\{\log\frac{P_{\lambda^0}(o_{1:t})}{P_{\lambda}(o_{1:t})}\right\} = \sum_t P_{\lambda^0}(o_{1:t})\log\frac{P_{\lambda^0}(o_{1:t})}{P_{\lambda}(o_{1:t})} \tag{34}$$

which is always non-negative and attains its global minimum at zero for $P_{\lambda} \to P_{\lambda^{(0)}}$. Its limit ($t \to \infty$) is KL rate or relative entropy rate.

The model parameters are updated after processing each observation sequence:

$$a_{ij}^{(k+1)} = \frac{1}{Z_1} a_{ij}^{(k)} \exp\left(-\frac{\eta}{n_i(\lambda^{(k)})}\frac{\partial \ell_T(\lambda^{(k)})}{\partial a_{ij}}\right) \tag{35}$$

$$b_j^{(k+1)}(m) = \frac{1}{Z_2} b_j^{(k)}(m) \exp\left(-\frac{\eta}{n_j(\lambda^{(k)})}\frac{\partial \ell_T(\lambda^{(k)})}{\partial b_j(m)}\right) \tag{36}$$

where Z_1 and Z_2 are normalization factors. The expected usage of state i, $n_i(\lambda^{(k)}) = \sum_{t=1}^{T}\gamma_{t|T}(i)$ can be computed using the filtering estimation (16), and the derivatives of the log-likelihood are given by (28) and (29).

Symbol-wise Garg and Warmuth [41] extended the block-wise learning of Singer and Warmuth [86] to symbol-wise. The model divergence is however penalized by the negative of the log-likelihood increment, i.e., log-likelihood of each new symbol given all previous ones. At each time instant the following optimization is performed:

$$\hat{\lambda}_{t+1} = \arg\min_{\lambda}(KL(\lambda_{t+1}, \lambda_t) - \eta \log P(o_{t+1}|o_{1:t}, \lambda_{t+1}))$$

which can be decoupled into similar update, for both state transition and state output, as with (35) and (36), respectively. The main difference resides however in the computation of the expected usage of state i. It is computed recursively based on each symbol with $n_i(\lambda_t) = \sum_{\tau=1}^{t}\gamma_{\tau|t}(i)$, and the gradient of the log-likelihood increment. The recursive formulas proposed to $n_i(\lambda_t)$ are very similar to those for recursive maximum likelihood estimation proposed by LeGland and Mevel [64,65] (see Section 3.2).

3.2. Maximum Likelihood Estimation (MLE)

The attractive asymptotic properties of the MLE, have prompted many attempts to extend this objective function to on-line learning for different applications. In particular, indirect maximization of log-likelihood, through the complete log-likelihood (19), has been the basis for many on-line learning algorithms. Neal and Hinton [75] have proposed an "incremental" version of the EM algorithm to accelerate its convergence on a finite training data set.[4] Assuming a fixed training data set is divided into n blocks, each iteration of this algorithm performs a partial E-step (for a selected block) then updates the model parameters (M-step), until a convergence criterion is met. The improved convergence is attributed to exploiting new information more quickly, rather than waiting for the complete data to be processed before updating the parameters [42,75]. Many extensions to this algorithm have emerged for on-line block-wise [28,73,95] or symbol-wise [36,89,91] learning.

Direct maximization of the log-likelihood function is first proposed for the block-wise case using the GD algorithm [4], and then using the quasi-Newton algorithm [16] for faster convergence. A recursive block-wise estimation technique is also proposed for this optimization [83,84]. Finally, extensions to the work of Titterington [92] and Weinstein et al. [98] for independent data has lead to a symbol-wise technique to optimize the complete data likelihood (19) recursively, at each time step.

3.2.1. On-line expectation–maximization

An important property of the "incremental" version of EM resides in its flexibility [75]. There is no constraints on how data is divided into blocks. The block might range from a single observation symbol to one or multiple sequence of observations. In addition, one can choose to update the parameters by selecting data blocks cyclically, or by any other scheme. Accordingly, several algorithms have extended this approach to on-line learning of HMM parameters. As detailed next, this is achieved by initializing the smoothed densities to zero, processing the training data (symbol-wise or block-wise) sequentially, and updating the HMM after each symbol or sequence.

Block-wise Digalakis [28] derived an on-line algorithm of EM to update the parameters of a continuous HMM for automatic speech recognition (ASR), and showed faster convergence and higher recognition rate over the batch algorithm. Similarly, Mizuno et al. [73] proposed a similar algorithm for ASR using a discrete HMM however. Given the initial model λ_0, and after processing each new sequence of observation of length T, the state densities are recursively computed using:

$$\sum_{t=1}^{T-1} \hat{\xi}_{t|T}^{(k+1)}(i,j) = (1-\eta_k)\sum_{t=1}^{T-1} \xi_{t|T}^{(k)}(i,j) + \eta_k \sum_{t=1}^{T-1} \xi_{t|T}^{(k+1)}(i,j)$$

$$\sum_{t=1}^{T} \hat{\gamma}_{t|T}^{(k+1)}(j)\delta_{o_t m} = (1-\eta_k)\sum_{t=1}^{T} \gamma_{t|T}^{(k)}(j)\delta_{o_t m} + \eta_k \sum_{t=1}^{T} \gamma_{t|T}^{(k+1)}(j)\delta_{o_t m}$$

and the model parameters are then directly updated using (22) and (23). The learning rate η_k is proposed in polynomial form $\eta_k = c\left(\frac{1}{k}\right)^d$ for some positive constants c and d.

Symbol-wise Stiller and Radons [91] introduced a recursive algorithm for non-stationary discrete Mealy HMMs,[5] which is essentially based on the FO (18). The main idea is to recursively update a tensorial quantity,

$$\xi_{ijk}^t(o_t) = \sum_{\tau=1}^{t} \alpha_i^{\tau-1} a_{ij}^{\tau-1}(o_\tau)\beta_{jk}^{\tau,t}\delta_{o_t o_\tau} \tag{37}$$

which accounts for the weighted sum of transition probability from state S_i to S_j (at time τ), emitting symbol o_t and being in state k at time $t \geqslant \tau$. The model parameters (emission on arcs) are condensed in $a_{ij}^t(o_t) = P(q_t = j, o_t | q_{t-1} = i)$. The forward-vector ($\alpha^\tau$) and backward-matrix ($\beta^{\tau,t}$) are recursively computed by:

$$\alpha^\tau = \pi \prod_{r=1}^{\tau} a^{(r-1)}(o_r) \text{ and } \beta^{\tau,t} = \prod_{r=\tau+1}^{t} a^{r-1}(o_r)$$

Eq. (37) can therefore be recursively computed as function of the old $\xi_{ijk}^t(o_t)$ and the new symbol o_{t+1}:

$$\xi_{ijk}^{t+1}(o_{t+1}) = \sum_{\hat{k}} \xi_{ij\hat{k}}^t(o_t)\frac{\sum_l \xi_{\hat{k}kl}^t(o_{t+1})}{\sum_\tau \sum_{\hat{j},\hat{l}} \xi_{ij\hat{l}}^t(o_\tau)} + \eta_t \delta_{o_t o_{t+1}}\delta_{jk}\sum_{l,m}\sum_\tau \xi_{lmi}^t(o_\tau)\frac{\sum_k \xi_{ijk}^t(o_t)}{\sum_\tau \sum_{\hat{j},\hat{k}} \xi_{ij\hat{k}}^t(o_\tau)}$$

where η_t is a time-varying learning rate. Finally, the model parameters $\left(a_{ij}^\tau\right)$ and the probability of the system currently being in state $k\left(\alpha_k^\tau\right)$ can be estimated at any time using:

$$a_{ij}^t(o) = \frac{\sum_k \xi_{ijk}^t(o_t)}{\sum_t \sum_{\hat{j},\hat{k}} \xi_{ij\hat{k}}^t(o_t)} \text{ and } \alpha_k^\tau = \frac{\sum_{i,j}\sum_t \xi_{ijk}^t(o_t)}{\sum_{i,j,\hat{k}}\sum_t \xi_{ij\hat{k}}^t(o_t)}$$

[4] In contrast, the incremental scenario considered in this paper restricts accessing a block of data once it has been processed.

[5] The emission is produced on transitions (Mealy model) while for all other algorithms presented the output is produced on states (Moore model).

This algorithm have been recently proposed for Moore HMMs [74] and then generalized [15].

Stenger et al. [89] approximated the fixed-interval smoothing used in BW by the filtered state density (9) for updating the parameters of a continuous HMM in the context of adaptive background modeling from video sequences. The filtered state density is recursively computed independently of the sequence length. The model parameters are then updated, at each time step, by

$$\begin{aligned} a_{ij}^t &= \frac{\sum_{\tau=1}^{t-2}\gamma_{t|\tau}(i)}{\sum_{\tau=1}^{t-1}\gamma_{t|\tau}(i)}a_{ij}^{t-1} + \frac{\xi_{t-1|t}(i,j)}{\sum_{\tau=1}^{t-1}\gamma_{t|\tau}(i)} \\ \mu_i^t &= \frac{\sum_{\tau=1}^{t-1}\gamma_{t|\tau}(i)}{\sum_{\tau=1}^{t}\gamma_{t|\tau}(i)}\mu_i^{t-1} + \frac{\gamma_{t|t}(i)o_t}{\sum_{\tau=1}^{t}\gamma_{t|\tau}(i)} \\ \left[\sigma_i^2\right]^T = \Sigma_i^T &= \frac{\sum_{\tau=1}^{t-1}\gamma_{t|\tau}(i)}{\sum_{\tau=1}^{t}\gamma_{t|\tau}(i)}\Sigma_i^{t-1} + \frac{\gamma_{t|t}(i)(o_t-\mu_i^t)^2}{\sum_{\tau=1}^{t}\gamma_{t|\tau}(i)} \end{aligned} \tag{38}$$

An exponential forgetting factor is proposed to reduce the weight of old model estimates by tuning a fixed learning rate.

In contrast, Florez-Larrahondo et al. [36] proposed using the predictive state density[6] (10) as a better approximation of the fixed-interval smoothing. The model parameters are recursively updated using an adaptation of Stenger's recursion formulas for discrete HMMs. The transition update is the same as (38) with $\gamma_{t|\tau}$ replaced by $\gamma_{t+1|\tau}$, while the state output is given by:

$$b_j^T(k) = \frac{\sum_{\tau=1}^{t-1}\gamma_{t+1|\tau}(j)}{\sum_{\tau=1}^{t}\gamma_{t+1|\tau}(j)}b_j^{T-1}(k) + \frac{\gamma_{t+1|t}(j)\delta(o_T, v_k)}{\sum_{\tau=1}^{t}\gamma_{t+1|\tau}(j)} \tag{39}$$

In HMM-based intrusion detection application, the author argued that there is no need for exponential forgetting factors, and proposed delaying the first parameters update, until some time $t > 0$, to stabilize the re-estimation.

3.2.2. Numerical optimization methods

Similar to batch learning of HMM parameters described in Section 3.2.2, standard numerical optimization methods can be applied for on-line learning to directly optimize the log-likelihood function and its derivatives. The gradient of the log-likelihood can be computed using the state conditional densities obtained from each block of training data (instead of the entire batch of data) or by using a recursive computation of the gradient itself as described next.

Block-wise Based on the GD of the negative likelihood (24), Baldi and Chauvin [4] have introduced a block-wise algorithm for estimation of discrete HMM parameters, which related to the Generalized EM (GEM). By using the soft-max parametrization, (30), and the FF-BS algorithm, parameters are updated after each sequence as follows:

$$\begin{aligned} u_{ij}^{(k+1)} &= u_{ij}^{(k)} + \eta\sum_{t=1}^{T}(\xi_{t|T}(i,j) - a_{ij}\gamma_{t|T}(i)) \\ v_j^{(k+1)}(m) &= v_j^{(k)}(m) + \eta\sum_{t=1}^{T}(\gamma_{t|T}(j)\delta_{o_t m} - b_j(m)\gamma_{t|T}(j)) \end{aligned}$$

This is typically referred to as stochastic gradient-based techniques [12].

Based on the recursive maximum likelihood estimation of LeGland and Mevel [64,65], Cappe et al. [16] proposed a quasi-Newton (27) faster convergent technique for maximizing the likelihood of a continuous HMM with Gaussian output (1). Using the parametrization on a simplex (31) and replacing the standard deviation by its log (to ensure its positivity), the gradient of the log-likelihood (13), can be computed recursively using:

$$\nabla_\lambda \ell_T(\lambda^{(k)}) = \sum_{t=1}^{T}\frac{1}{c_t}\sum_{i=1}^{N}\left[\gamma_{t|t-1}(i)\nabla_{\lambda^{(k-1)}}b_i(o_t) + b_i(o_t)\nabla_{\lambda^{(k-1)}}\gamma_{t|t-1}(i)\right] \tag{40}$$

where $c_t = \sum_{k=1}^{N} b_k(o_t)\gamma_{t|t-1}(k)$ is a normalization factor. The gradient of the predictive density is also computed recursively, with reference to the model parameters, using:

$$\begin{aligned} \frac{\partial\gamma_{t+1|t}(j)}{\partial a_{ij}} &= \frac{1}{c_t}\sum_{i=1}^{N}\left[(a_{ij} - \gamma_{t+1|t}(j))\frac{\partial\gamma_{t|t-1}(i)}{\partial a_{ij}} + \gamma_{t|t-1}(i)\right] \\ \nabla_{\lambda_b}\gamma_{t+1|t}(j) &= \frac{1}{c_t}\sum_{i=1}^{N}(a_{ij} - \gamma_{t+1|t}(j))[\gamma_{t|t-1}(i)\nabla_{\lambda_b}b_i(o_t) + b_i(o_t)\nabla_{\lambda_b}\gamma_{t|t-1}(i)] \end{aligned}$$

and the gradient of the Gaussian output density (1), w.r.t. the mean and the standard deviation, is given by

[6] This is equivalent to setting $\beta_t(i) = \sum_{j=1}^{N} a_{ij}b_j(o_{t+1})$ in the FB algorithm.

$$\nabla_{\lambda_b} b_i(o_t) = \left[\frac{\partial b_i(o_t)}{\partial \mu_i}, \frac{\partial b_i(o_t)}{\partial \sigma_i}\right]' = \left[\frac{(o_t - \mu_i)}{\sigma_i^2} b_i(o_t), \frac{1}{\sigma_i}\left(\left(\frac{(o_t - \mu_i)}{\sigma_i}\right)^2 - 1\right) b_i(o_t)\right]' \tag{41}$$

3.2.3. Recursive Maximum Likelihood Estimation (RMLE)

The general recursive estimator for the parameters of a stochastic process is of the form [72,9]:

$$\lambda_{t+1} = \lambda_t + \eta_t H(o_{t+1}, \lambda_t)^{-1} h(o_{t+1}, \lambda_t) \tag{42}$$

where η_t is a sequence of small gains (constant or decreasing with t), $H(o_{t+1}, \lambda_t)^{-1}$ an adaptive matrix, and $h(o_{t+1}, \lambda_t)$ is a score function. Both the adaptive matrix, H, and the score function, h, determine the update of the parameter λ_t as a function of new observations. The goal of the recursive procedures is to find the roots of a regression function given the true parameters λ^0: $\lim_{t\to\infty} E\{H(o_{t+1}, \lambda_t)^{-1} h(o_{t+1}, \lambda_t)|\lambda^0\}$.

In the general case of incomplete data, the score is equal to the gradient of the log-likelihood function $h = \nabla_\lambda \log P(o_{t+1}|\lambda)$. The adaptive matrix is equal to the incomplete data observed information matrix $H = \nabla^2_\lambda \log P(o_{1:t+1}|\lambda)$, which ideally must be taken as the incomplete data Fisher information matrix $H = E_\lambda[h(o_{t+1}|\lambda)h'(o_{t+1}|\lambda)]$. However, for the incomplete data models an explicit form of the incomplete data Fisher information matrix is rarely available. Titterington [92, Eq. 9] proposed using the complete data Fisher information matrix instead $H = E[-\nabla^2_{\lambda_t} \log P(o_t, q_t|\lambda)]$. He showed that it is easy to compute and invert H because it is always block-diagonal with respect to latent data and model parameters. For several independent and identically distributed (i.i.d.) incomplete data models, this recursion is proven to be consistent and asymptotically normal [92]:

$$\lambda_{t+1} = \lambda_t + \frac{1}{t+1}(E[-\nabla^2_{\lambda_t} \log P(o_t, q_t|\lambda)])^{-1} \nabla_{\lambda_t} \log P(o_{t+1}|\lambda_t) \tag{43}$$

This recursive learning algorithm is related to on-line estimation of parameters using EM. It was also related to an EM iteration since it uses a recursive version of (19).

The efficiency and convergence is an issue with Titterington's recursion [84,96]. In fact, for HMMs, the score function must consider the previous observations:

$$h(o_{t+1}|\lambda) = \nabla_\lambda \log P(o_{t+1}|o_{1:t}, \lambda) \tag{44}$$

Another recursion relies on the relative entropy (34) as objective function, is proposed by Weinstein et al. [98, Eq. 4]:

$$\hat{\lambda}_{t+1} = \hat{\lambda}_t + \eta_t h(o_{t+1}|\hat{\lambda}_t) \tag{45}$$

where the gain η_t satisfies:

$$\lim_{t\to\infty} \eta_t = 0; \quad \sum_{t=1}^{\infty} \eta_t = \infty; \quad \sum_{t=1}^{\infty} \eta_t^2 \leqslant \infty \tag{46}$$

It was suggested that h may be calculated from the complete data using

$$h(o_{t+1}|\hat{\lambda}_t) = E_\lambda\{\nabla \log P(o_{t+1}, q_{t+1}|\lambda)|o_{t+1}\}$$

and is shown to be consistent in the strong sense and in the mean-square sense for stationary ergodic processes that satisfy some regularity conditions. However, some of these conditions do not hold for HMM [83,84].

Block-wise Ryden [83,84] considered using successive blocks of m observations taken from data stream, $\boldsymbol{o}_n = \{o_{(n-1)m+1}, \ldots, o_{nm}\}$, as independent of each other, while the Markov dependence is only maintained within the observations of each block. This assumption reduces the extraction of information from all the previous observations to a data-segment of length m. Although it gives an approximation of the MLE – a less efficient maximum pseudo-likelihood – the presented technique is easier to apply in practice. Using a projection $\mathbb{P}_G$ on the simplex (32), $\triangle_\epsilon$, for some $\epsilon > 0$, at each iteration, the recursion is given (without matrix inversion)

$$\hat{\lambda}_{n+1} = \mathbb{P}_G(\hat{\lambda}_n + \eta_n h(\boldsymbol{o}_{n+1}|\hat{\lambda}_n)) \tag{47}$$

where $h(\boldsymbol{o}_{n+1}|\hat{\lambda}_n) = \nabla_{\lambda_n} \log P(\boldsymbol{o}_{n+1}|\lambda_n)$ and $\eta_n = \eta_0 n^{-\rho}$ for some positive constant η_0 and $\rho \in (\frac{1}{2}, 1]$. It was shown to converge almost surely to the set of Kuhn-Tucker points for minimizing the relative entropy $KL_m(\lambda^0\|\lambda)$ defined in (34), which attains its global minimum at $\hat{\lambda}_n \to \lambda^0$ provided that the HMM is identifiable [66,82], hence the requirement $m \geqslant 2$.

Symbol-wise Holst and Lindgren [45, Eq. 16] proposed a similar recursion to (43) for estimating the parameter of an HMM however. They used the conditional expectation over (q_{t-1}, q_t) given $o_{1:t}$, which can be efficiently calculated using a forward recursion. It is guided by

$$H_t^{-1} = \frac{1}{t}\sum_{\tau=1}^{t} h(o_t|\hat{\lambda}_{t-1}) h'(o_t|\hat{\lambda}_{t-1})$$

which is different from the score function (44). The adaptive matrix is suggested by an empirical estimate to the incomplete data information matrix given by

$$H_t^{-1} = \frac{1}{t}\sum_{\tau=1}^{t} h(o_t|\hat{\lambda}_{t-1})h'(o_t|\hat{\lambda}_{t-1})$$

which can be computed recursively without matrix inversion using:

$$H_t = \frac{t}{t-1}\left(H_{t-1} - \frac{H_{t-1}h_t h_t' H_{t-1}}{t-1+h_t' H_{t-1} h_t}\right)$$

Ryden [83] argued that the recursion of Holst and Lindgren aims at local minimization of the relative entropy rate (34) between λ^0 and $\hat{\lambda}_t$. Moreover, he showed that if $\hat{\lambda}_{t+1} \to \lambda^0$, then $\sqrt{t}(\hat{\lambda}_{t+1} - \lambda^0)$ is asymptotically normal with zero mean and covariance matrix given by the inverse of this expectation $\lim_{t\to\infty} E\{h(o_{t+1}, \lambda^0)h'(o_{t+1}, \lambda^0)\}$.

Slingsby [88] and Krishnamurthy and Moore [56] derived an on-line algorithm for HMMs based on the recursive version of the EM (43) proposed in [92,98], for digital signal processing and telecommunication applications. The algorithm follows a two-step procedure, similar to EM, as each observation symbol arrives:

E-step: Recursively computes the complete data likelihood

$$\mathcal{Q}_{t+1}(\lambda, \lambda_t) = E_\lambda[\log P(o_{t+1}, q_{t+1}|\lambda)|o_{1:t+1}, \lambda_t] = \sum_{\tau=1}^{t+1} \mathcal{Q}_\tau(\lambda, \lambda_t) \tag{48}$$

M-step: Estimates $\lambda_{t+1} = \arg\max_\lambda \mathcal{Q}_{t+1}(\Lambda_t, \lambda)$

where $\mathcal{Q}_\tau(\lambda, \lambda_t)$ is defined in (19). The model is updated using:

$$\lambda_{t+1} = \lambda_t + [\nabla^2_{\lambda_t} \mathcal{Q}_{t+1}(\lambda, \lambda_t)]^{-1} \nabla_{\lambda_t} \mathcal{Q}_{t+1}(\lambda, \lambda_t)$$

Since each term of (48) depends on only one of the model parameters, $\nabla^2_{\lambda_t} \mathcal{Q}_{t+1}(\lambda, \lambda_t)$ is a block diagonal matrix for each of the model parameters. A projection into the constraint domain along with the application of smoothing (fixed-lag or more conveniently sawtooth-lag) and forgetting show, through empirical experiments, a convergence to the correct model.

Slingsby [88] presented the parameters update for a discrete HMM using a projection on a simplex (31):

$$a_{ij}^{(t+1)} = a_{ij}^{(t)} + \frac{1}{d_j^{(i)}}\left(g_j^{(i)} - \frac{\sum_{h=1}^{N} g_h^{(i)}/d_h^{(i)}}{\sum_{h=1}^{N} 1/d_h^{(i)}}\right) \tag{49}$$

where

$$d_j^{(i)} = \frac{\sum_{\tau=1}^{t+1}\xi_\tau(i,j)}{\tilde{a}_{ij}^2}; \quad g_j^{(i)} = \frac{\xi_{t+1}(i,j)}{\tilde{a}_{ij}} \tag{50}$$

$$b_j^{(t+1)}(k) = b_j^{(t)}(k) - b_j(o_{t+1})\left(\frac{\gamma_{t+1}(j)}{\sum_{\tau=1}^{t+1}\gamma_\tau(j)\delta(o_{t+1},k)}\right)\left(\frac{\frac{b_j(k)^2}{\sum_{\tau=1}^{t+1}\gamma_\tau(j)\delta(o_{t+1},k)}}{\sum_{p=1}^{M}\left(\frac{b_j(p)^2}{\sum_{\tau=1}^{t+1}\gamma_\tau(j)\delta(o_{t+1},p)}\right)}\right); \quad k \neq o_{t+1}$$

$$b_j^{(t+1)}(k) = 1 - \sum_{p\neq k}^{M} b_j^{(t)}(p); \quad k = o_{t+1}$$

Krishnamurthy and Moore [56] focused on continuous HMM with Gaussian output (1). The state transitions update is the same as (50) and (49), and the state output update is given by:

$$\mu_i^{(t+1)} = \mu_i^{(t)} + \frac{\gamma_{t+1}(i)(o_{t+1} - \mu_i^{(t)})}{\sum_{\tau=1}^{t+1}\gamma_{\tau|t+1}(i)}$$

$$\left[\sigma_w^2\right]^{(t+1)} = \Sigma_w^{(t+1)} = \Sigma_w^{(t)} + \frac{\sum_{i=1}^{N}\gamma_{t+1}(i)\left(o_{t+1} - \mu_i^{(t)}\right)^2 - \Sigma_w^{(t)}}{t+1}$$

Since the positiveness of parameters is not maintained with the simplex parametrization, the projection on a sphere (33) may provide a better alternative [25]:

$$d_j^{(i)} = \sum_{\tau=1}^{t+1}\left[\frac{2\xi_\tau(i,j)}{s_{ij}^2} + 2\gamma_{t+1}(i)\right]; \quad g_j^{(i)} = \frac{2\xi_{t+1}(i,j)}{s_{ij}} - 2\gamma_{t+1}(i)s_{ij},$$

from which the transition parameters update is given by

$$s_{ij}^{(t+1)} = s_{ij}^{(t)} + \frac{g_j^{(i)}}{d_j^{(i)}}, \quad a_{ij} = s_{ij}^2$$

LeGland and Mevel [64,65] suggested and proved the convergence and the asymptotic normality of the RMLE (and the RCLSE described in the next section) without any stationary assumption, using the geometric ergodicity and the exponential forgetting of their prediction filter and its gradient. This approach is based on the observation that the log likelihood can be expressed as an additive function of an extended Markov chain, i.e., as sum of terms depending on the observations and the state predictive filter (10):

$$\ell_\tau(\lambda) = \sum_{t=1}^{\tau} \log \sum_{i=1}^{N} b_i(o_t)\gamma_{t|t-1}(i)$$

Taking the gradient of the log-likelihood increment gives the score: (as a function of the extended Markov chain $Z_t = F(o_t, \gamma_{t|t-1}, w_t)$)

$$h(Z_t|\lambda) = \nabla_\lambda \ell_\tau(\lambda) = \frac{1}{c_t} \sum_{i=1}^{N} \left[\gamma_{t|t-1}(i)\nabla_\lambda b_i(o_t) + b_i(o_t) w_t^i\right]$$

where $c_t = \sum_{k=1}^{N} b_k(o_t)\gamma_{t|t-1}(k)$ is a normalization factor, and $w_t^i = \nabla_\lambda \gamma_{t|t-1}(i)$ is the gradient of the state prediction filter, which can be also computed recursively using:

$$w_{t+1}^i = R_1(o_t, \gamma_{t|t-1}, \lambda) w_t^i + R_2^i(o_t, \gamma_{t|t-1}, \lambda) \tag{51}$$

where

$$R_1(o_t, \gamma_{t|t-1}, \lambda) = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}\left[1 - \frac{\gamma_{t|t-1}(i)\sum_{k=1}^{N} b_k(o_t)}{c_t}\right]\frac{b_i(o_t)}{c_t}$$

$$R_2^i(o_t, \gamma_{t|t-1}, \lambda) = \sum_{j=1}^{N} a_{ij}\left[1 - \frac{b_i(o_t)\sum_{k=1}^{N}\gamma_{t|t-1}(k)}{c_t}\right]\frac{\gamma_{t|t-1}(i)}{c_t}\nabla_\lambda b_i(o_t) + \frac{\gamma_{t|t-1}(i) b_i(o_t)}{c_t}\sum_{j=1}^{N}\nabla_\lambda a_{ij}$$

The parameters update is then done for each row i using:

$$\hat{\lambda}_{t+1}^i = \mathbb{P}_\epsilon\left(\hat{\lambda}_t^i + \eta_{t+1} h(Z_t^i|\lambda)\right)$$

where, $\eta_t = \frac{1}{t}$, and $\mathbb{P}_\epsilon$ is the projection on the simplex (32), Δ_ϵ, for some $\epsilon > 0$.

The update can be decoupled into transition and state output probabilities where R_1 does not change. For transition probabilities, $\nabla_\lambda b_i(o_t)$ are zeros, accordingly h and R_2 reduces to:

$$h(Z_t|\lambda) = \frac{1}{c_t}\sum_{i=1}^{N} b_i(o_t) w_t(i)$$

$$R_2^i(o_t, \gamma_{t|t-1}, \lambda) = \frac{\gamma_{t|t-1}(i) b_i(o_t)}{c_t}\sum_{j=1}^{N}\nabla_\lambda a_{ij}$$

where, $\nabla_\lambda a_{ij} = 1$ at position i, j and zero otherwise. For discrete state outputs,

$$h(Z_t|\lambda) = \begin{cases} \frac{1}{c_t}\sum_{i=1}^{N}\left[\gamma_{t|t-1}(i)\nabla_\lambda b_i(o_t) + b_i(o_t) w_t^i\right]; & \text{if } o_t = v_k \\ \frac{1}{c_t}\sum_{i=1}^{N}\left[b_i(o_t) w_t^i\right]; & \text{if } o_t \neq v_k \end{cases}$$

$$R_2^i(o_t, \gamma_{t|t-1}, \lambda) = \begin{cases} \sum_{j=1}^{N} a_{ij}\left[1 - \frac{b_i(o_t)\sum_{k=1}^{N}\gamma_{t|t-1}(k)}{c_t}\right]\frac{\gamma_{t|t-1}(i)}{c_t}\nabla_\lambda b_i(o_t), & \text{if } o_t = v_k \\ 0 & \text{if } o_t \neq v_k \end{cases}$$

For the Gaussian state outputs (1):

$$h(Z_t|\lambda) = \frac{1}{c_t}\sum_{i=1}^{N}\left[\gamma_{t|t-1}(i)\nabla_{\lambda_b} b_i(o_t) + b_i(o_t) w_t^i\right]$$

$$R_2^i(o_t, \gamma_{t|t-1}, \lambda) = \sum_{j=1}^{N} a_{ij}\left[1 - \frac{b_i(o_t)\sum_{k=1}^{N}\gamma_{t|t-1}(k)}{c_t}\right]\frac{\gamma_{t|t-1}(i)}{c_t}\nabla_{\lambda_b} b_i(o_t)$$

where $\nabla_{\lambda_b} b_i(o_t)$ is given by (41).

Collings and Ryden [24] proposed a similar RMLE approach, for continuous HMM with Gaussian output (1). The difference consists of using the sphere parametrization (33), instead of the simplex used in [64,65]. The gradient of the transition probabilities and the recursive computation of the gradient of the state predictive density should be now taken w.r.t. the projected parameters $s_{ij}\left(a_{ij} = s_{ij}^2\right)$. Although for different purpose, these derivations are very similar to (52) and (53) presented next.

3.3. Minimum Prediction Error (MPE)

Minimizing the output or state prediction errors provides alternate objective functions, which have been proposed in application of HMMs to signal processing. It consists of measuring the error of HMM output prediction [25,65] or of HMM filtered state [38,37], and then providing an updated estimate of HMM parameters with each new observation symbol.

Recursive Prediction Error (RPE) is first proposed for continuous range Gauss-Markov process [71], and for a general recursive stochastic gradient algorithm [2]. RPE extends the concept of least squares from linear to non-linear functions. It locates the locale minimum of the prediction error cost function $J(\lambda)$ and provides an updated estimate of the model with each new observation. It is formulated by considering the minimum variance of the prediction error based on the best model estimate at the time

$$\hat{\lambda} = \arg\min_{\lambda} \left\{ J(\lambda) = \frac{1}{2} E[o_t - \hat{o}_t]^2 \right\}$$

where $\hat{o}_t$ is the output prediction. The expectation of the off-line minimum variance, for an observation sequence of length T, is approximated by its time average:

$$J_T(\lambda) = \frac{1}{2T} \sum_{t=1}^{T} [o_t - \hat{o}_t]^2$$

which is commonly minimized with the Gauss-Newton method – a simplified Newton method specialized to the non-linear least square problem – where each pass k through the data the model is updated using:

$$\hat{\lambda}^{(k+1)} = \hat{\lambda}^{(k)} - \eta_t \left[\frac{1}{T} \sum_{t=1}^{T} \psi_{\lambda^{(k)}} \psi'_{\lambda^{(k)}} \right]^{-1} \left[\frac{1}{T} \sum_{t=1}^{T} \psi_{\lambda^{(k)}} \varepsilon_{\lambda^{(k)}} \right]$$

where $\psi_{\lambda^{(k)}} = \nabla_{\lambda^{(k)}} J_T(\lambda)$ and $\varepsilon_{\lambda^{(k)}} = o_t - \hat{o}_t$. An on-line version of the Gauss-Newton method is also possible if the gradient, and the inverse of the empirical information matrix, can be built up recursively, at each time instant, as new observation arrive. A general practical implementation is given by Ljung and Soderstrom [72].

Collings et al. [25] extended the RPE for continuous HMM where model variances are assumed. The model is updated at each time step using:

$$\hat{\lambda}_{t+1} = \mathbb{P}_s \left(\hat{\lambda}_t + \eta_{t+1} R_{t+1}^{-1} \psi'_{t+1} \mathcal{E}_{t+1} \right)$$

where η_t is a gain sequence satisfying (46) and R_t is the adaptive matrix which is computed by:

$$R_{t+1} = R_t + \eta_{t+1} (\psi_{t+1} \psi'_{t+1} - R_t)$$

or alternatively, to avoid the matrix inversion, R_t^{-1} could be computed as

$$R_{t+1}^{-1} = \frac{1}{1 - \eta_{t+1}} \left(R_t^{-1} - \frac{\eta_{t+1} R_t^{-1} \psi_{t+1} \psi'_{t+1} R_t^{-1}}{(1 - \eta_{t+1}) + \eta_{t+1} \psi'_{t+1} R_t^{-1} \psi_{t+1}} \right)$$

However, now for the HMM case, the output error is given by

$$\mathcal{E}_t = o_t - \hat{o}_{t|t-1}$$

where, the output prediction, $\hat{o}_{t|t-1} = E[o_t | o_{1:t-1}, \lambda_t]$, is conditioned on previous values and given by (using the unnormalized state variable $\alpha_t(i)$ (15), of the Forward-Backward algorithm)

$$\hat{o}_{t|t-1} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} \alpha_{t-1}(i) b_j(o_t)}{\sum_{i=1}^{N} \alpha_{t-1}(i)}$$

In addition, a projection on the sphere, $\mathbb{P}_s$, is made at each time step to ensure model constrains (33). Accordingly, the gradient of the output error is given by:

$$\psi_t = [-\nabla_{\lambda} \hat{\varepsilon}_t]' = [\nabla_{\mu_j} \hat{o}_{t|t-1}, \nabla_{s_{ij}} \hat{o}_{t|t-1}]'$$

which can be also computed recursively:

$$\begin{aligned}\nabla_{\mu_j}\hat{o}_{t+1|t} &= \frac{\sum_{i=1}^{N} a_{ij}\alpha_t(i) + \sum_{i=1}^{N} a_{ij}\varsigma_t(i)b_j(o_t)}{\sum_{i=1}^{N}\alpha_{t-1}(i)} - \frac{\sum_{i=1}^{N} a_{ij}\alpha_t(i)\varsigma_t(i)b_j(o_t)}{\sum_{i=1}^{N}\alpha_{t-1}^2(i)} \\ \nabla_{s_{ij}}\hat{o}_{t+1|t} &= \frac{2\sum_{i=1}^{N} s_{ij}\alpha_t(i)\left(\mu_j - \sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij}^2 b_j(o_t)\right) + \sum_{i=1}^{N} a_{ij}\zeta_t(i,j)b_j(o_t)}{\sum_{i=1}^{N}\alpha_{t-1}^2(i)} \\ &\quad - \frac{\sum_{i=1}^{N} a_{ij}\alpha_t(i)\zeta_t(i,j)b_j(o_t)}{\sum_{i=1}^{N}\alpha_{t-1}^2(i)}\end{aligned} \tag{52}$$

where $\varsigma_t(j) = \nabla_{\mu_j}\alpha_t(j)$ and $\zeta_t(j,m,n) = \nabla_{s_{ij}}\alpha_t(j)$ are given by the following recursions:

$$\begin{aligned}\varsigma_{t+1}(j) &= \sum_{i=1}^{N} a_{ij}\varsigma_t(j)b_j(o_{t+1}) + \frac{o_{t+1} - b_j(o_t)}{\sigma_w^2}\sum_{i=1}^{N} a_{ij}\alpha_t(i)b_j(o_{t+1}) \\ \zeta_{t+1}(i,j) &= \sum_{i=1}^{N} a_{ij}\zeta_t(i,j)b_j(o_{t+1}) - 2s_{ij}\alpha_t(i)\sum_{j=1}^{N} b_j(o_{t+1})s_{ij}^2 + 2\alpha_t(i)s_{ij}b_j(o_{t+1})\end{aligned} \tag{53}$$

The authors studied the convergence properties of the algorithm empirically. A convergence problem for this algorithm has been observed for small error variances (low noise condition), cf. [24, Section 5] and [37, Section 3-C].

Ford and Moore [38] proposed an ELS and RPE schemes that exploit the filtered state estimates, however for cases where the transition probabilities are assumed known. These schemes have been later extended to the recursive state prediction error (RSPE) algorithm [37], where the state transition probabilities are now estimated from the data. Local convergence analysis for RSPE is presented using the ordinary differential equation (ODE) approach developed for RPE methods. The objective function is given (as a function of the filtered state error) by:

$$\hat{\lambda} = \arg\min_{\lambda}\left\{J_t(\lambda) = \frac{1}{2}\sum_{t=1}^{\tau}\left[\gamma_{t|t}(i) - \sum_{i=1}^{N}\gamma_{t-1|t-1}(i)a_{ij}\right]^2\right\}$$

and the model is updated, for each row i, using:

$$\begin{aligned}\hat{\lambda}_t^i &= \hat{\lambda}_{t-1}^i + \left[H_t^i\right]^{-1} w_t^i \\ \left[H_t^i\right]^{-1} &= \left[H_{t-1}^i\right]^{-1} + \gamma_{t-1|t-1}(i)\end{aligned}$$

where H_t^{-1} is an approximation to the second derivative of w_t, and

$$w_t^i = \frac{\partial J_t(\lambda)}{\partial a_{ij}}$$

can be computed recursively, in a way similar to (51).

In addition to the RMLE described in the previous section, LeGland and Mevel [65] proved the convergence of the recursive conditioned least squares estimator (RCLSE), which is a generalization of the RPE approach [25]. A similar objective function is used:

$$J_\tau(\lambda) = \frac{1}{2\tau}\sum_{t=1}^{\tau}[o_t - \hat{o}_{t|t-1,\lambda}]^2,$$

where the output prediction given by

$$\hat{o}_{t|t-1} = \sum_{j=1}^{N} b_j(o_t)\gamma_{t|t-1}(j)$$

is based on the state predictive density. Accordingly, its gradients (w_t^i) can be computed recursively using (51). The model update for transition probabilities is given, for each row i, by

$$\hat{\lambda}_{t+1}^i = \mathbb{P}_\epsilon\left(\hat{\lambda}_t^i + \eta_{t+1}\left[\mu_i\left(o_t - \sum_{j=1}^{N}\mu_j\gamma_{t|t-1}(j)\right)\right]w_t^i\right),$$

and for the continuous state outputs by

$$\hat{\lambda}_{t+1}^i = \mathbb{P}_\epsilon\left(\hat{\lambda}_t^i + \eta_{t+1}\left(\sum_{i=1}^{N}\left[\mu_i\left(o_t - \sum_{j=1}^{N}\mu_j\gamma_{t|t-1}(j)\right)\right]w_t^i + \sum_{i=1}^{N}\left[\nabla_i\mu_i\left(o_t - \sum_{j=1}^{N}\mu_j\gamma_{t|t-1}(j)\right)\right]\gamma_{t|t-1}(i)\right)\right)$$

As for the discrete state outputs, the algorithm is only applicable for finite alphabet. In this case, the same formulas apply, however with $\mu_i = \sum_{t=1}^{T} o_t b_i(o_t)$ and of course using the gradient recursion (51) for the discrete case.

4. An analysis of the on-line learning algorithms

Algorithms for on-line learning of HMM parameters are mostly designed to learn observations from a source generating an infinite amount of data. For instance, these algorithms can be employed to re-estimate HMM parameters from a large number of sub-sequences (e.g., speech sentences), or a stream of symbols (e.g., signal in a noisy communication channel). Block-wise techniques re-estimate HMM parameters after observing each new sub-sequence, while symbol-wise techniques re-estimate HMM parameters upon observing each new symbol (Fig. 5a). The objective is to optimize HMM parameters to fit the generating source of data through one observation of the data. In contrast, batch learning algorithms re-estimate HMM parameters upon observing all accumulated sequences or blocks of sub-sequences of observations (Fig. 5b). Batch learning algorithms optimize HMM parameters to fit the accumulated data over several iterations.

On-line learning is challenging since algorithms must converge rapidly to the true model with minimum resource requirements. It is assumed that the true generative model resides within the HMM parameters space. Finding recursive formulas to update the HMM parameter is necessary for on-line learning but not sufficient. Statistical properties such as consistency and asymptotic normality must be proven to ensure the convergence of an on-line learning algorithm. This section provides an analysis of the convergence properties and of the computational complexity of the on-line algorithms presented in Section 3.

4.1. Convergence properties

The extension of the batch EM to an incremental EM version, using a partial E-step followed by a direct update of HMM parameters in the M-step, is a well known technique that overcomes the resource requirements when processing a large fixed-size data set [44]. Neal and Hinton [75] showed that this EM variant provides non-increasing divergence, and that local minima in divergence are local maxima in the likelihood function. However, as argued by Gunawardana and Byrne [43], this is insufficient to conclude that it converges to local maxima in the likelihood function. They showed that this incremental EM variant is not in fact an EM procedure and hence the standard GEM convergence properties [99] do not apply. Indeed their E-step is modified to use summary statistics from the posterior distributions of previous estimates. By using the Generalized Alternating Minimization (GAM) procedure in an information geometric framework, Gunawardana and Byrne [43] proved that the incremental EM converges to stationary points in likelihood, although *not monotonically*.

The authors could find no proof of convergence for on-line EM-based algorithms when HMM parameters estimation is performed from infinite amount of data, for both block-wise [28,73] and symbol-wise [91,89,36,74,**?**] techniques. Furthermore, statistical analysis such as consistency and asymptotic normality are not provided. Although the rate of convergence is not studied analytically, applying stochastic techniques in literature such as averaging [78] has been shown to help improve convergence properties [15]. Among the block-wise algorithms, the gradient-based algorithm proposed by Cappe et al. [16] should achieve the fastest convergence rate since it is based on a quasi-Newton method (second order approximation), while others should have a slower convergence rate. Similarly, for the recursive EM symbol-wise algorithms [56,88] since the complete likelihood is optimized by a second order method, although no proof of convergence is provided.

Among the methods based on RMLE, Ryden [83,84] provided convergence analysis for his block-wise RMLE. He proved the consistency by using the classical result of Kushner and Clark [60]. Independently, LeGland and Mevel [64,65] proved the convergence and asymptotic normality of their symbol-wise RMLE, under the assumption that the transition probability matrix is primitive, yet without any stationary assumption. This is accomplished by using the geometric ergodicity and the exponential forgetting of the predictive filter and its gradient. Krishnamurthy and Yin [57] extended the RMLE results of LeGland and Mevel [65] to auto-regressive models with Markov regime, and added results on convergence, rate of

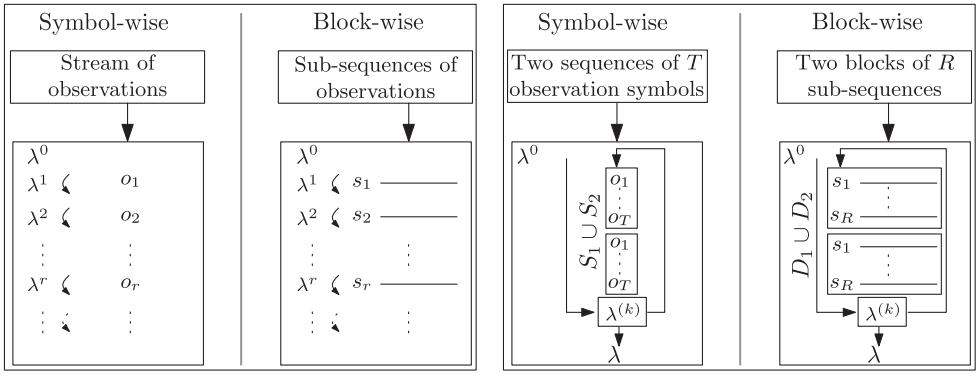

Fig. 5. An illustration of on-line learning (a) from an infinite stream of observation symbols (o_i) or sub-sequences (s_i) versus batch learning (b) from two accumulated sequences of observations symbols ($S_1 \cup S_2$) or blocks of sub-sequences ($D_1 \cup D2$).

convergence, model averaging, and parameter tracking. In particular, they showed that the above assumption can be relaxed to the condition in which the transition probability matrix is aperiodic and irreducible. The authors also suggested using the iterate averaging [78], as well as observation averaging for faster convergence and lower variance.

RPE based techniques were first extended and applied for recursive estimation of HMM parameters due to their quadratic convergence rate (although it is asymptotically sub-optimal). This convergence speed comes at the expenses of an increased complexity, and as discovered later, the RPE algorithm proposed by Collings et al. [25] suffers from numerical issues. Local convergence analysis is only shown for the RSPE [38] using the ordinary differential equation. Finally, similar to the RMLE, LeGland and Mevel [65] also proved the convergence of the RCLSE.

4.2. Time and memory complexity

The time complexity of an algorithm, can be analytically defined as the sum of the worst-case running time $\mathcal{T}_p$ for each operation p required to re-estimate HMM parameters based on new training sequence (block-wise) or new observation (symbol-wise):

$$\mathcal{T} = \sum_p \mathcal{T}_p = \sum_p t_p n_p$$

where t_p is the constant time needed to execute an operation p (e.g., multiplication, division, etc.), and n_p is the number of times this operation is executed. The *growth rate* is then obtained by making the key parameters (T and N) of the worst-case time complexity tend to ∞. For simplicity, only most expensive operations are considered – multiplication, division, and exponent. Time complexity is independent from the convergence time, which varies among different algorithmsas discussed in Section 4.1. Memory complexity is estimated as the number of *32* bit registers needed during learning process to store temporary variables. The worst-case memory required for estimation of sufficient statistics such as conditional distributions of states and gradients of the log-likelihood is considered.

Tables 1 and 2 present a breakdown of time and memory complexity for some representative algorithms. Table 1 compares the complexity of block-wise algorithms applied to on-line learning of an observation sequence $o_{1:T}$ of length T and then re-estimating HMM parameters. Block-wise techniques typically employ a fixed-lag smoothing approach for estimating HMM states upon receiving each sub-sequence. In general, all these algorithms have the same time complexity, $\mathcal{O}(N^2T)$. However, EM-based block-wise techniques [28,73] are easier to implement than gradient-based [4,86,16] and RMLE [83,84] techniques. EM-based techniques maintain the parameters constraints (2) and (3) implicitly, and do not employ derivatives and projections of gradient filters. Accordingly, they require fewer computations to update the HMM parameters upon receiving an observation symbol (see Table 1). Although, block-wise algorithms have the same memory complexity of $\mathcal{O}(NT)$, the memory requirements of the algorithm proposed by Cappe et al. [16] is independent of the sequence length T. This is due to the recursion on the gradient itself and could be useful when T is large. However, it requires additional time complexity due to its line search technique.

Table 2 compares the complexity of symbol-wise algorithms for on-line learning from a stream of observation symbols o_t. The time complexity is based on the learning of one observation. Therefore, it must be multiplied by the length, T, of a finite observation sequence for comparison with block-wise techniques. Algorithms that requires more than N^2 computational complexity per time step may become less attractive when the number of HMM states N is large. As for block-wise techniques, EM-based symbol-wise filtering and prediction techniques [89,36] are generally easier to implement than gradient-based [64,65,24,41] and minimum prediction error [25,38,37] techniques. On the other hand, although EM-based smoothing techniques [91,74,15] have been shown to provide more accurate states estimate than filtering and prediction [1], their time complexity per observation is $\mathcal{O}(N^4)$, which makes them less attractive when the number of HMM states N is large. In general the memory requirement of symbol-wise techniques is independent of T, since the HMM re-estimation is either based the current symbol (filtering) or on small finite predictions – one symbol look-ahead or a larger fixed-lag smoothing.

Block-wise algorithms typically require fewer computations than symbol-wise algorithms when learning from a large observation sequence, at the expenses of an increased memory requirements. This is mainly due to fewer update of HMM parameters that is required with block-wise learning algorithms. Fig. 6 illustrates the time and memory complexity required by the EM-based symbol-wise prediction algorithm proposed by Florez-Larrahondo et al. [36] versus its block-wise fixed-lag smoothing counterpart proposed by Mizuno et al. [73], when learning an observation sequence of length $T = 1000{,}000$ symbols, with an output alphabet of size $M = 50$ symbols. While symbol-wise algorithms can be directly employed to learn such sequence of observations, block-wise algorithms require segmenting the sequence into non-overlapping sub-sequences using a sliding window of size W, which is also the fixed-lag value.

As shown in Fig. 6a–c, block-wise algorithms are more time efficient with smaller number of states N and smaller window sizes W. While symbol-wise algorithms require T updates of HMM parameters, block-wise algorithms require about $\frac{T}{W}$ updates when learning the same observation sequence. When the N value increases the computational time of block-wise algorithms may approach that of symbol-wise for large window sizes. In such cases, the computational time of parameters update is dominated by that of state densities which scales quadratically with N and linearly with W. Therefore, for a given N value, smaller window sizes should be favored since they reduce the time complexity for block-wise learning from a large observation sequence. Memory complexity also increases linearly with the window size as shown in Fig. 6d. Increasing the

Table 1
Time and memory complexity for some representative block-wise algorithms used for on-line learning of a new sub-sequence $o_{1:T}$ of length T. N is the number of HMM states and M is the size of alphabet symbols.

Algorithms	Step	# Multiplications	# Divisions	# Exp	Memory
Baldi and Chauvin [4]	γ and ξ	$6N^2T + 3NT - 6N^2 - N$	$N^2T + 3NT - N^2 - N$		$NT + N^2 + 2N$
	A	$2N^2$	N^2	N^2	$2N^2$
	B	$2NM$	MN	NM	$2NM$
	Total	$6N^2T + 3NT - 4N^2 + 2NM - N$	$N^2T + 3NT + MN - N$	$N^2 + NM$	$NT + 3N^2 + 2NM + 2N$
Cappe et al. [16], quasi-Newton based	$\nabla\ell$ and γ_t	$5N^2T + 2NT$	$3N^2T + NT$		$N(N^2 + NM)$
	A	$2N^2$	N^2		$2N^2$
	B	$2N$	N		$2N$
	Total	$5N^2T + 2NT + 2N^2 + 2N$	$4N^2T + NT + 2N + N$		$2N^3 + 2N^2$
Mizuno et al. [73]	γ and ξ	$6N^2T + 3\ NT - 6N^2 - N$	$N^2T + 3NT - N^2 - N$		$NT + N^2 + 2N$
	A	N^2	N^2		N^2
	B	NM	NM		NM
	Total	$6N^2T + 3NT - 5N^2 + NM - N$	$N^2T + 3NT + NM - N$		$NT + 2N^2 + NM + 2N$

Table 2
Time and memory complexity of some representative symbol-wise algorithms used for on-line learning of new symbol o_i. N is the number of HMM states and M is the size of alphabet symbols.

Algorithms	Step	# Multiplications	# Divisions	Memory
Stiller and Radons [91], smoothing-based approach	$\xi^t_{ijk}(o_t)$	$N^4 + N^2$	$2N^3$	$2N^3M$
	$A = \{a_{ij}(o_t)\}$	–	N^2M	N^2M
	Total	$N^4 + N^2$	$2N^3 + N^2M$	$2N^3M + N^2M$
Florez-Larrahondo et al. [36], prediction-based approach	γ and ξ	$5N^2 + N$	$N^2 + N$	$N^2 + 3N$
	A	N^2	$2N^2$	N^2
	B	NM	$2NM$	NM
	Total	$6N^2 + NM + N$	$3N^2 + 2NM + N$	$2N^2 + NM + 3N$
Slingsby [87]	d, g, γ	$6N^2 + N$	$2N^2 + N$	$2N^2 + N$
	A	N^2	$N^2 + 2N$	N^2
	B	$NM + 3N$	$NM + 2N$	NM
	Total	$7N^2 + NM + 4N$	$3N^2 + NM + 5N$	$3N^2 + NM + N$
Legland and Mevel [64]	R_1 and R_2	$N^3 + N^2M$	$N^2 + NM$	$N^2 + NM + 4N$
	A	N^2	$2N^2$	N^2
	B	NM	$2NM$	NM
	Total	$N^3 + N^2(M + 1) + NM$	$3N^2 + 3NM$	$2N^2 + 2NM + 4N$
Collings et al. [25]	$R_t^{-1}, \psi_t, \mathcal{E}_t$	$N^4 + 4N^3 + N^3 + 5N^2 + 5M^2 + 3NM$	$3N^2 + M^2 + M$	$N^2 + NM + 2N$
	A	– (only N^2 additions) –		N^2
	B	– (only NM additions) –		NM
	Total	$N^4 + 4N^3 + N^3 + 5N^2 + 5M^2 + 3NM$	$3N^2 + M^2 + M$	$2N^2 + 2NM + 4N$

fixed-lag value W may yield a more accurate local estimate of HMM states, and hence improves algorithms stability. If memory constraints are an issue, block-wise techniques that employ recursions on the gradient itself [16] should be favored since their memory complexity are independent of W. When the application imposes stricter constraints on the memory space, symbol-wise algorithms should be employed. The strategy for selecting a value for W is thereby an additional issue to address with block-wise techniques.

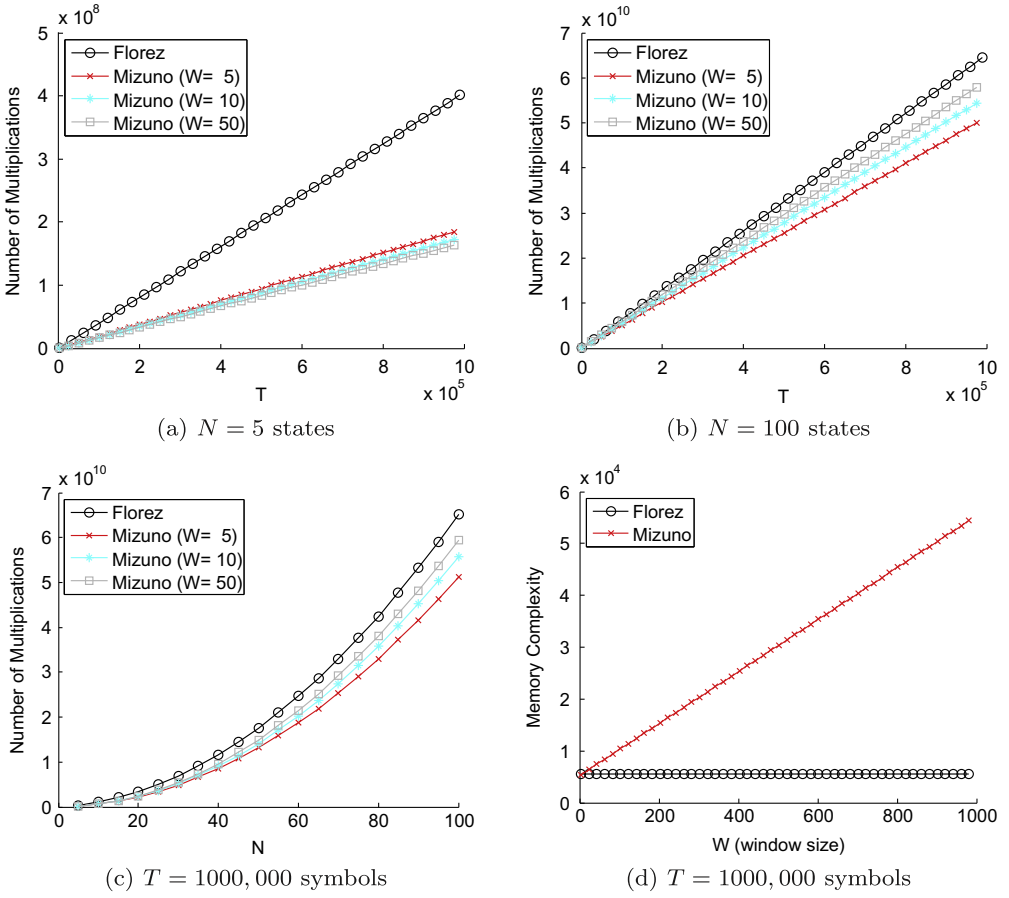

Fig. 6. An example of the time and memory complexity of a block-wise [73] vs. symbol-wise [36] algorithm for learning an observation sequence of length $T = 1000{,}000$ with an output alphabet of size $M = 50$ symbols.

5. Guidelines for incremental learning of HMM parameters

The target application implies a scenario for data organization and learning. Depending on the application, incremental learning may be performed on an abundant or limited amount of new data. In addition, this data may be organized into a block of observation sub-sequences or into one long observation sequence. Within the incremental learning scenario (Fig. 1), HMM parameters should be re-estimated from newly-acquired training data. However, the corruption of previously acquired knowledge remains a key issue. In fact, none of the algorithms described in Section 4 can fully overcome this issue during incremental learning. Several factors such as the choice of the learning rate and sub-sequence length may help alleviating this issue, by optimizing HMM parameters such that contributions of new data and pre-existing knowledge is balanced. In contrast, ensemble of classifiers trained independently on new training data and combined with previously-trained classifiers may provide an alternate solution [47,53]. The rest of this section provides guidelines and underscores challenges faced when applying on-line techniques to supervised incremental learning of new training data, when the data is either abundant or limited.

5.1. Abundant data scenario

Under the abundant data scenario, it is assumed that a long sequence of training observations becomes available to update HMM parameters through incremental learning. A more complete view of phenomena is therefore presented to the learning algorithm. As stated previously, symbol-wise algorithms can be directly employed to learn such sequences, one observation at a time, while block-wise algorithms require buffering some amount of data using for instance a sliding window according to a user-defined buffer size W.

When learning is performed in a static environment from a large sequence of observations, one view of the data is typically sufficient to capture the underlying structure by exploiting patterns redundancy. This is because the abundant data provide a more complete view of phenomena. Resource constraints would be another reason behind restricting the iterative

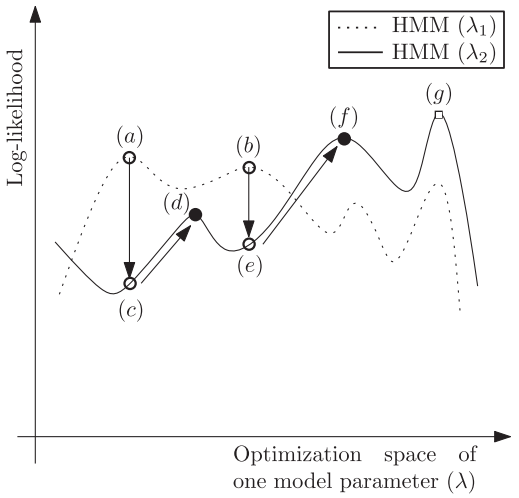

Fig. 7. The dotted curve represents the log-likelihood function associated with an HMM (λ_1) trained on block D_1 over the space of one model parameter λ. After training on D_1, the on-line algorithm estimates $\lambda = \lambda_1$ (point (a)) and provides HMM (λ_1). The plain curve represents the log-likelihood function associated with HMM (λ_2) trained on block D_2 over the same optimization space.

learning procedure to one pass. Therefore, the convergence properties of an on-line algorithm must be known to determine the speed and limits of convergence behavior. As described in Section 4.1, when the true model is contained in the set of solutions, some algorithms are shown to be consistent and asymptotically normal by properly choosing a monotonically decreasing learning rate η_t and by applying Polyak-Ruppert averaging [78]. For on-line learning in static environment, decreasing step-sizes are essential conditions of convergence. Fixed step-sizes may yield the convergence of algorithms to oscillate around their limiting values with variances proportional to the step-size. However, some novelty criteria on the new data should be employed to reset the monotonically decreased learning rate when provided with a new sequence of observations.

In contrast, when learning is performed in dynamically-changing environments, the notion of optimality is no longer valid because the on-line algorithm should forget past knowledge and adapt to the newly acquired information. Detecting slow, systematic or abrupt changes would require specialized novelty detection strategies [59], and this remains an open issue that is outside the scope of this paper. Tracking non-stationary environments and handling slow drift is typically achieved by choosing a fixed learning rate η. For abrupt drift however, a data driven learning rate is required to detect changes and adapt the step-sizes according to the incoming data [85,90].

In both static and dynamically-changing environments, algorithms with low time and memory complexity are favored to learn the long sequence of observations. As discussed in Section 4.2, symbol-wise algorithms that requires more than N^2 time complexity upon receiving each symbol are less attractive, especially when the number of HMM states N is large. For block-wise algorithms, smaller window sizes W are favored for reducing both time and memory complexity. However, the stability of algorithms must also be considered when selecting the window size.

Some issues that require further investigation when adapting the HMM parameters from abundant data. Empirical benchmarking studies of these techniques could offer further insight on trade-offs for selecting algorithms and user-defined parameters for an application domain. For instance, the performance of techniques based on MLE (such as EM and gradient based) should be compared to those based on MMD, and recursive techniques for MPE. An interesting comparison would also involve symbol-wise versus block-wise and filtering versus fixed-lag smoothing. Significant improvement, accuracy, speed and stability of convergence, computing resources, and amount of training data required to reach or maintain a given level of performance should be among the evaluation criteria. To this end, it is recommended to conduct non-parametric tests for statistical comparisons of multiple classifiers over multiple data sets [27,40]. In addition to assessing the strength and weakness of each algorithm, such comparison may lead to efficient hybrid on-line algorithms that require fewer resources (see Section 2.2.3).

5.2. Limited data scenario

Under the limited data scenario, it is assumed that a short sequence of observations becomes available to update the HMM parameters, providing therefore a limited view of phenomena. For block-wise algorithms the sequence is typically segmented into shorter sub-sequences using a sliding window with a user-defined window size W. In contrast to the abundant data case, when provided with limited data, several iterations over the new sequence or block of observations are required to re-estimate HMM parameters. This local optimization raises additional challenges. Specialized strategies are needed for managing the learning rate which, at each iteration must balance integration of pre-existing knowledge of the HMM and

the newly-acquired training data. In addition, the learning technique may become trapped in local maxima on the cost function on the parameter space.

To illustrate the difficulty, assume that the training block D_2, which contains one or multiple sub-sequences, becomes available after a HMM has been previously trained on a block D_1 and deployed for operations. After training on D_1, the HMM has a parameter settings of λ_1. An on-line algorithm that optimizes, for instance, the log-likelihood function requires re-estimating the HMM parameters over several iterations until this function is maximized for D_2. The plain curve in Fig. 7 represents the log-likelihood function of an HMM(λ_1) that was previously learned D_1, and then has incrementally learned D_2 over the space of on HMM parameter (λ). Since λ_1 is the only information at hand from previous data D_1, the training process starts from HMM(λ_1). Optimization of HMM parameters depends on the shape and position of the log-likelihood function of HMM(λ_2) with respect to that of HMM(λ_1). For instance, if λ_1 was selected according to point (a) in Fig. 7, then the optimization will probably lead to point (d), which degrades HMM performance. If λ_1 was selected as point (b), the optimization would probably lead to a better solution (f). In contrast, training a new HMM(λ_2) on D_2 by starting from the start on different initializations may lead to point (g) which, depending on the new data, may yield higher log-likelihood than all previous models.

Boolean combination of responses from HMM(λ_1) and HMM(λ_2) in the receiver operating characteristics (ROC) space may significantly improve system performance [53]. An incremental Boolean combination of ensembles of HMMs (EoHMMs) have been proposed to overcome knowledge corruption with a single HMM system. When a new block of data becomes available, a pool of HMMs is generated and combined to previously-generated HMMs. The HMMs are trained from each new data block with different number of states and initializations, which allows to capture different underlying structures of the data and hence increases diversity among pool members. The responses from these newly-trained HMMs are then combined with those of the previously-trained HMMs in ROC space using a novel incremental Boolean combination technique [53].

Although incremental learning techniques of single HMM parameters are unable to overcome knowledge corruption, typical procedures for unbiased performance estimation may still help alleviating the corruption. This involves stopping the training iterations when the log-likelihood of an HMM on independent validation data no longer improves. Using hold-out or cross-validation procedures reduces the effects of overfitting and improves generalization performance during operations. When the situation allows, some proportion of a new training data should be dedicated to validation, providing a good stopping criterion and improving the generalization capabilities of the classification system. In order to perform such validation, a set of observations must be stored and updated over time in a fixed-sized buffer. Managing this validation set over time depends on the application environment and should be selected and maintained according to some relevant selection criteria. For instance, in dynamically-changing environments, older validation data should be discarded and replaced with new observations data that is more representative of the underlying data distribution. In static or cyclically-stationary environments, older data should be preserved or rotated in a first-in-first-out manner. An information theoretic measure called surprise has been recently proposed to capture uncertainty of a new observation with respect to the current knowledge of the learning system [70]. The concept of surprise could be applied for maintaining the validation set, since it can classify new data into abnormal, learnable or redundant for the current system [70].

A potential advantage of applying the on-line learning techniques over batch learning techniques to limited data scenario resides in their added stochasticity, which stems from the rapid re-estimation of HMM parameters. This may aid escaping local maxima during the early adaptation to newly provided data. Managing the internal learning rate, associated with each sub-sequence or observation symbol, is therefore different for limited data than for abundant data. In general, employing a fixed learning rate along with the validation strategies described above would maintaining the level of performance in both static and dynamically-changing environments. However, more insights are required in this regard to determine if applying a monotonically decreasing or auto-adaptive learning rates at the iteration, sequence, or symbol levels are beneficial for escaping local maxima and hence improving the performance.

As presented in Section 4.2, the resource requirements for the on-line algorithms when learning from limited data are comparable to that of when learning from abundant data. However, the time complexity presented for both block- and symbol-wise techniques must be multiplied by the number of training iterations, which varies according to the algorithm employed, validation strategy and stopping criteria, and training data. Learning from limited data is less restrictive to memory and time complexity in the abundant data. Therefore, even the most compute-intensive symbol-wise algorithms – requiring $\mathcal{O}(N^4)$ time complexity upon receiving each symbol – or block-wise algorithms – requiring $\mathcal{O}(N^4T)$ time and $\mathcal{O}(NT)$ memory complexity upon receiving each sub-sequence – can be afforded as long as they improve or maintain the HMM performance. In limited data scenario, escaping local maxima, managing learning rates and stopping criteria may be more critical factors for selecting an on-line algorithm than minimizing the resource requirements.

Finally, as for the abundant data scenario, applying the on-line learning techniques from this survey to adapt HMM parameters to limited new data requires comparative benchmarking and statistical testing at the objective functions, optimization techniques and algorithms (symbol-wise versus block-wise) levels, and according to various user-defined parameters (learning rate and window size). Empirical evaluations of these techniques across various applications, such as those presented in [20], would provide useful insights.

6. Conclusion

The performance of Hidden Markov Models (HMMs) in real-world applications are often degraded because they face complex environments that change during operations, and because they are designed a *priori* using limited training data and prior knowledge. To sustain a high level of performance, a HMM should be capable of efficiently adapting its parameters, in response to new data observations from the environment, through incremental learning. Incremental learning allows to update HMM parameters from new data without accessing the previously-learned data, however corrupting previously-acquired knowledge remains an issue. Standard techniques for estimating HMM parameters involve batch learning, which assume a finite amount of training data available throughout the training process. The HMM parameters are estimated over several training iterations, where each iteration requires processing the entire training data, until some objective function is maximized. To avoid knowledge corruption, learning new data with these techniques would require storing cumulative data in memory and training from the start using all this data. This may represent a very costly solution.

This paper present a survey of techniques found in literature that are suitable for incremental learning of HMM parameters. These include on-line learning algorithms that have been initially proposed for learning from a long training sequence, or block of sub-sequences. In this paper, these techniques are categorized according to the objective functions, optimization techniques and target application. Some techniques are designed to update HMM parameters upon receiving a new symbol (symbol-wise), while others update the parameters upon receiving a sequence (block-wise). Convergence properties of these techniques are presented, along with an analysis of their time and memory complexity. In addition, the challenges faced when these techniques are applied to incremental learning is assessed for scenarios in which the new training data is limited and abundant.

When the new training data is abundant (scenario 1), the HMM parameters should be re-estimated to fit the source generating the data through one pass over a sequence of observations. When new data corresponds to a long sequence or block of sub-sequences generated from a stationary source (static environment), these techniques must employ a specialized strategy to manage the learning rate such that new data and existing knowledge are integrated without compromising the HMM performance. This includes resetting a monotonically decreasing learning rate when provided with new data or employing an auto-adaptive learning rate. Rapid convergence with limited resource requirements are other major factors in such cases. However few learning algorithms have been provided with a proof of convergence or other relevant statistical properties. Another important issue is the ability to operate in dynamically-changing environments. In such cases, some novelty criteria on the new data should be employed to detect changes and trigger adaptation by resting a monotonically decreasing learning rate or fine-tuning an auto-adaptive learning rate.

When the new training data is limited (scenario 2), HMM parameters should be re-estimated over several iterations due to the limited view of phenomena. Therefore, HMM parameters should be able to escape local maxima of the cost function associated with the new data. Given the limited data, early stopping criteria through hold-out or cross-validation must be considered when learning the new block of data to reduce the effects of overfitting. Accumulating and updating representative validation data set over time must be considered. However, this requires investigating some selection criteria for maintaining the most informative sequences of observations and discarding less relevant ones. Another major challenge resides in adapting the current operational HMM to the newly-acquired data without corrupting existing knowledge. One possible solution involves using an adaptive learning rate which, at each iteration, controls the weight given to the current HMM with reference to the new information. This learning rate should preferably be inferred from the data.

Finally, this paper underscores the need for comparative benchmarking studies of these on-line algorithms for incremental learning in both abundant and limited data scenarios. Some interesting comparative studies, evaluation criteria and statistical testing methods are suggested for selecting an algorithm for a particular situation. Knowledge corruption and performance degradation may arise because incremental learning of new data is initiated from a pre-existing HMM. Learn-and-combine approaches [53] are among the promising solutions to limit the impact of knowledge corruption.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada, and le Fonds Québécois de la Recherche sur la Nature et les Technologies.

References

[1] B. Anderson, From Wiener to hidden Markov models, IEEE Control Systems Magazine 19 (3) (1999) 41–51.
[2] A. Arapostathis, S.I. Marcus, Analysis of an identication algorithm arising in the adaptive estimation of Markov chains, Mathematics of Control, Signals and Systems 3 (1) (1990) 1–29.
[3] L.R. Bahl, F. Jelinek, R.L. Mercer, A maximum likelihood approach to continuous speech recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-5 2 (1982) 179–190.
[4] P. Baldi, Y. Chauvin, Smooth on-line learning algorithms for hidden Markov models, Neural Computation 6 (2) (1994) 307–318.
[5] L.E. Baum, An inequality and associated maximization technique in statistical estimation for probalistic functions of Markov processes, Inequalities III, 1972, pp. 1–8 (Proc. 3rd Symp., Univ. Calif., Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin).
[6] L.E. Baum, G.S. Petrie, N. Weiss, A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains, Annals of Mathematical Statistics 41 (1) (1970) 164–171.

[7] L.E. Baum, T. Petrie, Statistical inference for probabilistic functions of finite state Markov chains, Annals of Mathematical Statistics 37 (1966) 1554–1563.
[8] Y. Bengio, Markovian models for sequential data, Neural Computing Surveys 2 (1999) 129–162.
[9] A. Benveniste, P. Priouret, M. Métivier, Adaptive Algorithms and Stochastic Approximations, Springer-Verlag New York, Inc., New York, NY, USA, 1990.
[10] P.J. Bickel, Y. Ritov, T. Ryden, Asymptotic normality of the maximum-likelihood estimator for general hidden Markov models, Annal of Statistics 26 (4) (1998) 1614–1635.
[11] J.A. Bilmes, What HMMs can do. Tech. Rep. UWEETR-2002-0003, Dept of EE, University of Washington Seattle WA, 2002.
[12] L. Bottou, Stochastic learning, in: O. Bousquet, U. von Luxburg (Eds.), Advanced Lectures on Machine Learning. No. LNAI 3176 in Lecture Notes in Artificial Intelligence, Springer Verlag, Berlin, 2004, pp. 146–168.
[13] M. Brand, V. Kettnaker, Discovery and segmentation of activities in video, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8) (2000) 844–851.
[14] J. Bulla, A. Berzel, Computational issues in parameter estimation for stationary hidden Markov models, Computational Statistics 23 (1) (2008) 1–18.
[15] O. Cappe, Online EM algorithm for hidden Markov models, 2009 (preprint).
[16] O. Cappe, V. Buchoux, E. Moulines, Quasi-Newton method for maximum likelihood estimation of hidden Markov models, in: Proceedings of the 1998 IEEE International Conference onAcoustics, Speech, and Signal Processing, ICASSP98, vol. 4, 1998, pp. 2265–2268.
[17] O. Cappe, E. Moulines, Recursive computation of the score and observed information matrix in hidden Markov models, in: IEEE/SP 13th Workshop on Statistical Signal Processing, 2005, pp. 703–708.
[18] D. Caragea, A. Silvescu, V. Honavar, Towards a theoretical framework for analysis and synthesis of agents that learn from distributed dynamic data sources, in: Emerging Neural Architectures Based on Neuroscience, Springer-Verlag, 2001, pp. 547–559.
[19] A.A. Cardenas, V. Ramezani, J.S. Baras, HMM sequential hypothesis tests for intrusion detection in MANETs, Tech. Rep. ISR TR 2003-47, Department of Electrical and Computer Engineering and Institute for Systems Research University of Maryland College Park, MD 20740, USA, 2003.
[20] P. Cavalin, R. Sabouri, C. Suen, A. Britto Jr., Evaluation of incremental learning algorithms for HMM in the recognition of alphanumeric characters, Pattern Recognition 42 (2009) 3241–3253.
[21] R. Chang, J. Hancock, On receiver structures for channels having memory, IEEE Transactions on Information Theory 12 (4) (1966) 463–468.
[22] S.-B. Cho, S.-J. Han, Two sophisticated techniques to improve HMM-Based intrusion detection systems, in: RAID, 2003, pp. 207–219.
[23] A. Churbanov, S. Winters-Hilt, Implementing EM and viterbi algorithms for hidden Markov model in linear memory, BMC Bioinformatics 9 (224) (2008).
[24] I. Collings, T. Ryden, A new maximum likelihood gradient algorithm for on-line hidden Markov model identification, in: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP98, vol. 4, 1998, pp. 2261–2264.
[25] I.B. Collings, V. Krishnamurthy, J.B. Moore, On-line identication of hidden Markov models via recursive prediction error techniques, IEEE Transactions on Signal Processing 42 (12) (1994) 3535–3539.
[26] A. Dempster, N. Laird, D. Rubin, Maximum likelihood estimation from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B 39 (1) (1977) 1–38.
[27] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[28] V.V. Digalakis, Online adaptation of hidden Markov models using incremental estimation algorithms, IEEE Transactions on Speech and Audio Processing 7 (3) (1999) 253–261.
[29] P. Domingos, A unified bias-variance decomposition and its applications, in: 17th International Conference on Machine Learning. Morgan Kaufmann, 2000, pp. 231–238.
[30] S.R. Eddy, Profile hidden Markov models, Bioinformatics 14 (9) (1998) 755–763.
[31] A. El-Yacoubi, M. Gilloux, R. Sabourin, C. Suen, Unconstrained handwritten word recognition using hidden Markov models, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (8) (1999) 752–760.
[32] R.J. Elliott, Exact adaptive filters for Markov chains observed in gaussian noise, Automatica 30 (9) (1994) 1399–1408.
[33] R.J. Elliott, L. Aggoun, J.B. Moore, Hidden Markov Models: Estimation and Control, Springer-Verlag, 1995.
[34] Y. Ephraim, N. Merhav, Hidden Markov processes, IEEE Transactions on Information Theory 48 (6) (2002) 1518–1569.
[35] Y. Ephraim, L. Rabiner, On the relations between modeling approaches for information sources [speech recognition], IEEE Transactions on Information Theory 36 (2) (1990) 372–380.
[36] G. Florez-Larrahondo, S. Bridges, E.A. Hansen, Incremental estimation of discrete hidden Markov models based on a new backward procedure, Proceedings of the National Conference on Artificial Intelligence 2 (2005) 758–763.
[37] J. Ford, J. Moore, Adaptive estimation of HMM transition probabilities, IEEE Transactions on Signal Processing (USA) 46 (5) (1998) 1374–1385.
[38] J. Ford, J. Moore, On adaptive HMM state estimation, IEEE Transactions on Signal Processing (USA) 46 (2) (1998) 475–486.
[39] F. Gao, J. Sun, Z. Wei, The prediction role of hidden Markov model in intrusion detection, in: Canadian Conference on Electrical and Computer Engineering, vol. 2, Montreal, Canada, 2003, pp. 893–896.
[40] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.
[41] A. Garg, M.K. Warmuth, Inline updates for HMMs, in: EUROSPEECH-2003, September 1–4, 2003, pp. 1005–1008.
[42] Y. Gotoh, M.M. Hochberg, H.F. Silverman, Efficient training algorithms for HMM's using incremental estimation, IEEE Transactions on Speech and Audio Processing 6 (6) (1998) 539–548.
[43] A. Gunawardana, W. Byrne, Convergence theorems for generalized alternating minimization procedures, Journal of Machine Learning Research 6 (2005) 2049–2073.
[44] R.J. Hathaway, Another interpretation of the EM algorithm for mixture distributions, Statistics and Probability Letters 4 (1986) 53–56.
[45] U. Holst, G. Lindgren, Recursive estimation in mixture models with Markov regime, IEEE Transactions on Information Theory 37 (6) (1991) 1683–1690.
[46] G.E. Hovland, B.J. McCarragher, Hidden Markov models as a process monitor in robotic assembly, International Journal of Robotics Research 17 (2) (1998) 153–168.
[47] J. Hu, X. Yu, D. Qiu, H.-H. Chen, A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection, IEEE Network 23 (1) (2009) 42–47.
[48] X. Huang, H.-W. Hon, Spoken Language Processing: A Guide to Theory, Algorithm, and System Development. Prentice Hall PTR, Upper Saddle River, NJ, USA, foreword By-Raj Reddy, 2001.
[49] B.-H. Juang, S. Levinson, M. Sondhi, Maximum likelihood estimation for multivariate mixture observations of Markov chains (corresp.), IEEE Transactions on Information Theory 32 (2) (1986) 307–309.
[50] E. Justino, F. Bortolozzi, R. Sabourin, A comparison of SVM and HMM classifiers in the off-line signature verification, Pattern Recognition Letters 26 (9) (2005) 1377–1385.
[51] W. Khreich, E. Granger, A. Miri, R. Sabourin, Iterative Boolean combination of classifiers in the ROC space: an application to anomaly detection with HMMs, Pattern Recognition 43 (8) (2010) 2732–2752.
[52] W. Khreich, E. Granger, A. Miri, R. Sabourin, On the memory complexity of the forward-backward algorithm, Pattern Recognition Letters 31 (2) (2010) 91–99.
[53] W. Khreich, E. Granger, A. Miri, R. Sabourin, Adaptive ROC-based ensembles of HMMs applied to anomaly detection, Pattern Recognition 45 (1) (2012) 208–230.
[54] J. Kivinen, M.K. Warmuth, Exponentiated gradient versus gradient descent for linear predictors, Information and Computation 132 (1) (1997) 1–63.

[55] J. Konorski, Solvability of a Markovian model of an IEEE802.11 LAN under a backoff attack, in: 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Atlanta, GA, USA, 2005, pp. 491–498.
[56] V. Krishnamurthy, J. Moore, On-line estimation of hidden Markov model parameters based on the Kullback–Leibler information measure, IEEE Transactions on Signal Processing 41 (8) (1993) 2557–2573 (see also IEEE Transactions on Acoustics, Speech, and Signal Processing).
[57] V. Krishnamurthy, G.G. Yin, Recursive algorithms for estimation of hidden Markov models and autoregressive models with Markov regime, IEEE Transactions on Information Theory 48 (2) (2002) 458–476.
[58] A. Krogh, B. Larsson, G. von Heijne, E.L.L. Sonnhammer, Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes, Journal of Molecular Biology 305 (3) (2001) 567–580.
[59] L.I. Kuncheva, Classifier ensembles for changing environments, in: Multiple Classifier Systems, LNCS, vol. 3077, Springer-Verlag, Cagliari, Italy, 2004, pp. 1–15.
[60] H. Kushner, D. Clark, Stochastic Approximation for Constrained and Unconstrained Systems, vol. 26, Springer-Verlag, New York, 1978.
[61] T. Lane, C.E. Brodley, An empirical study of two approaches to sequence learning for anomaly detection, Machine Learning 51 (1) (2003) 73–107.
[62] K. Lange, A quasi-Newton acceleration of the EM algorithm, Statistica Sinica 5 (1) (1995) 1–18.
[63] P. Langley, Order effects in incremental learning, in: P. Reimann, H. Spada (Eds.), Learning in Humans and Machines: Towards an Interdisciplinary Learning Science, Elsevier, Amsterdam, 1995.
[64] F. LeGland, L. Mevel, Recursive identication of HMM's with observations in a finite set, in: Proceedings of the 34th IEEE Conference on Decision and Control, New Orleans, December 1995, pp. 216–221.
[65] F. LeGland, L. Mevel, Recursive estimation in hidden Markov models, in: Proceedings of the 36th IEEE Conference on Decision and Control, vol. 4, San Diego, CA, December 1997, pp. 3468–3473.
[66] B. Leroux, Maximum-likelihood estimation for hidden Markov models, Stochastic Processes and its Applications 40 (1992) 127–143.
[67] S.E. Levinson, L.R. Rabiner, M.M. Sondhi, An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition, Bell System Technical Journal 62 (1983) 1035–1074.
[68] X. Li, M. Parizeau, R. Plamondon, Training hidden Markov models with multiple observations – A combinatorial method, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (4) (2000) 371–377.
[69] L. Liporace, Maximum likelihood estimation for multivariate observations of Markov sources, IEEE Transactions on Information Theory 28 (5) (1982) 729–734.
[70] W. Liu, J. Príncipe, S. Haykin, Kernel Adaptive Filtering: A Comprehensive Introduction, John Wiley & Sons, Inc., 2010.
[71] L. Ljung, Analysis of recursive stochastic algorithms, IEEE Transactions on Automatic Control 22 (4) (1977) 551–575.
[72] L. Ljung, T. Soderstrom, Theory and Practice of Recursive Identification, MIT Press, Boston, 1983.
[73] J. Mizuno, T. Watanabe, K. Ueki, K. Amano, E. Takimoto, A. Maruoka, On-line estimation of hidden Markov model parameters, in: Proceedings of Third International Conference on Discovery Science, DS 2000 1967, 2000, pp. 155–169.
[74] G. Mongillo, S. Deneve, Online learning with hidden Markov models, Neural Computation 20 (7) (2008) 1706–1716.
[75] R.M. Neal, G.E. Hinton, A new view of the EM algorithm that justifies incremental, sparse and other variants, in: M.I. Jordan (Ed.), Learning in Graphical Models, Kluwer Academic Publishers., 1998, pp. 355–368.
[76] J. Nocedal, S.J. Wright, Numerical Optimization, second ed., Springer Series in Operations Research and Financial Engineering, Springer, 2006.
[77] R. Polikar, L. Upda, S. Upda, V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, IEEE Transactions on Systems, Man and Cybernetics, Part C 31 (4) (2001) 497–508.
[78] B.T. Polyak, New method of stochastic approximation type, Automation and Remote Control 7 (1991) 937–946.
[79] F. Qin, A. Auerbach, F. Sachs, A direct optimization approach to hidden Markov modeling for single channel kinetics, Biophysical Journal 79 (4) (2000) 1915–1927.
[80] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.
[81] J. Rittscher, J. Kato, S. Joga, A. Blake, A probabilistic background model for tracking, in: Proceedings of the 6th European Conference on Computer Vision-Part II, ECCV00, Springer-Verlag, London, UK, 2000, pp. 336–350.
[82] T. Ryden, Consistent and asymptotically normal parameter estimates for hidden Markov models, Annals of Statistics 22 (4) (1994) 1884–1895.
[83] T. Ryden, On recursive estimation for hidden Markov models, Stochastic Processes and their Applications 66 (1) (1997) 79–96.
[84] T. Ryden, Asymptotic efficient recursive estimation for incomplete data models using the observed information, Metrika 44 (1998) 119–145.
[85] N. Schraudolph, Local gain adaptation in stochastic gradient descent, in: Ninth International Conference on Artificial Neural Networks, ICANN99, vol. 2, 1999, pp. 569–574.
[86] Y. Singer, M.K. Warmuth, Training algorithms for hidden Markov models using entropy based distance functions, in: Neural Information Processing Systems, 1996, pp. 641–647.
[87] S. Sivaprakasam, S.K. Shanmugan, A forward-only recursion based HMM for modeling burst errors in digital channels, in: IEEE Global Telecommunications Conference, GLOBECOM95, vol. 2, 1995, pp. 1054–1058.
[88] P. Slingsby, Recursive parameter estimation for arbitrary hidden Markov models, in: IFAC Adavptive Systems in Control and Signal Processing, Grenoble, France, 1992.
[89] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, J. Buhmann, Topology free hidden Markov models: application to background modeling, Proceedings of the IEEE International Conference on Computer Vision 1 (2001) 294–301.
[90] J. Stiller, Adaptive online learning of generative stochastic models, Complexity (USA) 8 (4) (2003) 95–101.
[91] J. Stiller, G. Radons, Online estimation of hidden Markov models, IEEE Signal Processing Letters (USA) 6 (8) (1999) 213–215.
[92] D.M. Titterington, Recursive parameter estimation using incomplete data, Journal of the Royal Statistical Society, Series B (Methodological) 46 (2) (1984) 257–267.
[93] U. Tosun, Hidden Markov models to analyze user behaviour in network traffic, Tech. rep., Bilkent University 06800 Bilkent, Ankara, Turkey, 2005.
[94] R. Turner, Direct maximization of the likelihood of a hidden Markov model, Computational Statistics and Data Analysis 52 (9) (2008) 4147–4160.
[95] D. Vasquez, T. Fraichard, C. Laugier, Incremental learning of statistical motion patterns with growing hidden Markov models, IEEE Transactions on Intelligent Transportation System 10 (2009) 403–416.
[96] S. Wang, Y. Zhao, Almost sure convergence of Titterington's recursive estimator for mixture models, Statistics and Probability Letters 76 (18) (2006) 2001–2006.
[97] C. Warrender, S. Forrest, B. Pearlmutter, Detecting intrusions using system calls: alternative data models, in: Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA, 1999, pp. 133–145.
[98] E. Weinstein, M. Feder, A.V. Oppenheim, Sequential algorithms for parameter estimation based on the Kullback-Leibler information measure, IEEE Transactions on Acoustics Speech and Signal Processing 38 (9) (1990) 1652–1654.
[99] C.F.J. Wu, On the convergence properties of the EM algorithm, Annals of Statistics 11 (1) (1983) 95–103.
[100] D.-Y. Yeung, Y. Ding, Host-based intrusion detection using dynamic and static behavioral models, Pattern Recognition 36 (1) (2003) 229–243.
[101] W. Zucchini, I. MacDonald, Hidden Markov and Other Models for Discrete-valued Time Series: An Introduction Using R, Monographs on Statistics and Applied Probability, vol. 110, Chapman & Hall, 2009.