Learning observable operator models via the ES algorithm[1]

Herbert Jaeger, Mingjie Zhao, Klaus Kretzschmar, Tobias Oberstein,
Dan Popovici, Andreas Kolling

June 16, 2005

[1]To appear in: Haykin, S., Principe, J., Sejnowski, T., McWhirter, J. (eds.): New directions in statistical signal processing: from systems to brains. MIT Press.

Abstract

Hidden Markov Models (HMMs) today are the method of choice for blackbox modelling of symbolic, stochastic time series with memory. HMMs are usually trained using the expectation-maximization (EM) algorithm. This learning algorithm is note entirely satisfactory due to slow convergence and the presence of many globally suboptimal solutions. Observable operator models (OOMs) present an alternative. At the surface OOMs appear almost like HMMs: both can be expressed in structurally identical matrix formalisms. However, the matrices and state vectors of OOMs may contain negative components, whereas the corresponding components in the world of HMMs are non-negative probabilities. This freedom in sign gives OOMs algebraic properties that radically differ from HMMs, and leads to novel learning algorithms that are fast and yield asymptotically correct model estimates. Unfortunately, the basic versions of these algorithms are statistically inefficient, which has so far precluded a widespread use of OOMs. This chapter gives, first, a tutorial introduction to OOMs, and second, introduces a novel approach to OOM estimation called *efficiency sharpening* (ES). The ES method is iterative. In each iteration, the model estimated in the previous round is used to construct an estimator with a better statistical efficiency than the previous one. The computational load per iteration is comparable to one EM iteration, but only 2 to 5 iterations are typically needed. The chapter gives an analytical derivation of the ES principle and describes two learing algorithms that build on this principle, a simple "poor man's" version and a more complicated but superior version which is based on a suffix-tree representation of the training string. The quality of the latter algorithm is demonstrated on a task of learning a model of a long belletristic text, where OOM models markedly outperform HMM models in quality, requiring only a fraction of learning time.

1 Introduction

Observable operator models (OOMs) are mathematical models of stochastic processes. In their basic version, they describe stationary, finite-valued, discrete-time processes — in other words, symbol sequences. We will restrict ourselves to this basic type of processes in this chapter.

A number of models for stochastic symbol sequences are widely used. Listed in order of increasing expressiveness, the most common are elementary Markov chains, higher-order Markov chains and hidden Markov models (HMMs) [38; 2]. Well-understood learning algorithms to estimate such models from data exist. Specifically, HMMs are usually trained by versions of the expectation-minimization (EM) algorithm [6]. HMMs currently mark the practical limit of analytical and algorithmic tractability, which has earned them a leading role in application areas such as speech recognition [32], biosequence analysis [11] and control engineering [13].

In this chapter we wish to establish OOMs as a viable alternative to HMMs — albeit as yet only for the case of modeling stationary symbol processes. We see three main advantages of OOMs over HMMs:

- The mathematical theory of OOMs is expressed purely in terms of linear algebra and admits a rigorous, transparent semantic interpretation.
- OOMs properly generalize HMMs, that is, the class of processes that have finite-dimensional OOM properly includes the processes characterized by finite-dimensional HMMs.
- New learning algorithms for OOMs, derived from a novel principle which we would like to call *efficiency sharpening* (ES), yields model estimates in a fraction of the computation time that EM-based algorithms require for HMM estimation. Furthermore, on most datasets that have been investigated so far, the OOM models obtained via ES are markedly more accurate than HMM models.

However, at the current early state of research there remain also painful shortcomings of OOMs. Firstly, the OOMs learnt from data are prone to predict negative "probabilities" for some (rare) sequences, instead of small non-negative values. Currently only heuristic methods to master this problem are available. Secondly, our OOM learning algorithms tend to become instable for large model dimensions. Again, heuristic coping strategies exist, which are detailed out in this chapter.

This chapter has two main parts. The first part (Sections 2 through 9) contains a tutorial introduction to the basic theory of OOMs, including the basic version of the learning algorithm. This material has been published before [30] but has been almost completely rewritten with a more transparent notation and a new didactic approach. We hope that this tutorial part becomes the standard introductory text on OOMs. The second part (Sections 10 through 15), as an original contribution, establishes the ES principle and two learning algorithms are derived from it. Two case studies round off the presentation.

2 The Basic Ideas Behind OOMs

In this section we first describe the essence of OOMs in informal terms and then condense these intuitions into a mathematical formalism.

Envision a soccer-playing robot[1] engaged in a soccer game. In order to play well, the robot should make predictions about possible consequences of its actions. These consequences are highly uncertain, so in one way or the other they must be internally represented to the robot as a distribution of future trajectories (Figure 1a).

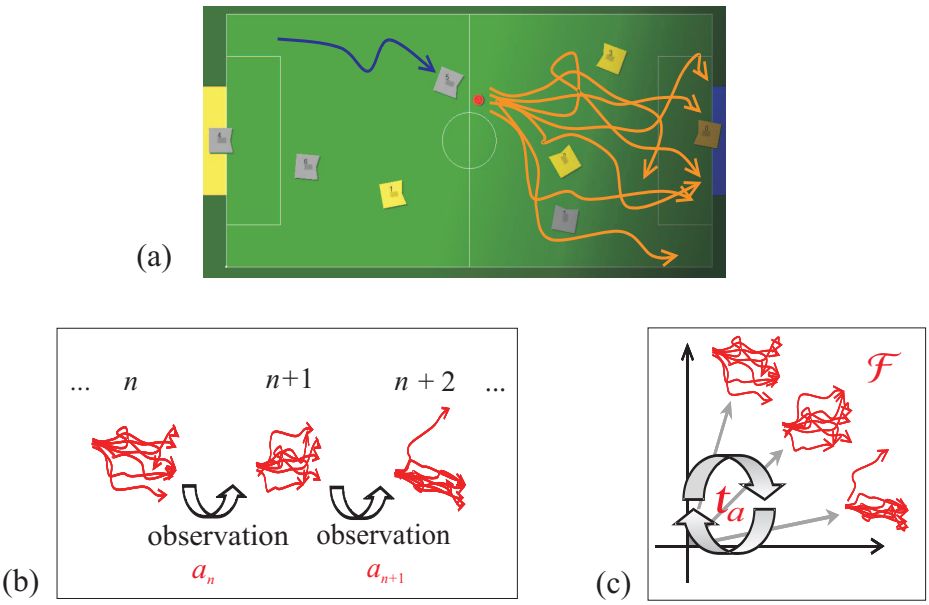

Figure 1: (a) A robot's future depicted as a "spaghetti bundle" of expected possible future trajectories. (b) The robot's expected futures change due to incoming observations a_n of information. (c) An operator τ_a associated with an observation a yields an update operation on the vector space of future distributions.

Soccer is a dynamic game, and the robot has to update its expectations about the future in an update cycle from time n to $n+1$, assuming a unit cycle time. OOMs are a mathematical model of this kind of update operation. Clearly the update is steered by the information that the robot collects during an update interval. This comprises incoming sensory information, communications from other robots, but also the robot's own issued motor commands or even results from some planning algorithms that run on it — in short, *everything* that is

[1] The first author originally devised OOMs while thinking about action selection algorithms for mobile robots.

of some informational value for the expected future. We comprise all of these bits of information under the term of an *observation*. At the root of OOMs lies the assumption that *nothing but* the observation a_n between n and $n+1$ controls the update of future expectations, and that such update operations can be *identified with* observations (Figure 1b). Thus, in OOMs we have for every possible observation one operator that can be used to update expected futures. This identification of observations with update operators has given OOMs their name, *observable operator* models.

Mathematically, a future of a stochastic process is a probability distribution on the set of potential future trajectories after the current time n. Such distributions can be specified by a real-valued function f in various ways. For instance, f may be a probability density funciton, or one may use a function f which assigns probabilities to finite-length future sequences, that is, a function on words over the observation alphabet. At this point we do not care about the particular format of f, we only assume that some real-valued function can describe a future's distribution (for general abstract treatment see [29]).

The real-valued functions f over some set can be added and multiplied with scalars and hence span a vector space F. Identifying observations with update operators on futures, and identifying futures with functions f which are vectors in F, we find that observations can be seen as operators on F. In the OOM perspective, each possible observation a is identified with an operator t_a on F (Figure 1c).

The key to OOMs is the observation that these *observable operators* are linear. We now give a formal treatment of the case where the stochastic process is of a particular simple kind, namely, discrete-time, finite-valued, and stationary. Let $(X_n)_{n\in\mathbb{N}}$, or for short, (X_n) be a stationary, discrete-time stochastic process with values in a finite alphabet $O = \{a^1, \ldots, a^\alpha\}$ of possible observations.

We shall use the following shorthand. For $P(X_n = a_0, \ldots, X_{n+r} = a_r)$ we write $P(a_0 \ldots a_r)$ or even shorter $P(\bar{a})$. For conditional probabilities $P(X_n = b_0, \ldots, X_{n+r} = b_r \,|\, X_{n-s} = a_0, \ldots, X_{n-1} = a_{-1})$ we write $P(b_0 \ldots b_r \,|\, a_0 \ldots a_{s-1})$ or $P(\bar{b} \,|\, \bar{a})$. Unconditional probabilities $P(\bar{a})$ can be seen as conditional probabilities conditioned by the empty sequence ε, that is $P(\bar{b}) = P(\bar{b} \,|\, \varepsilon)$.

The distribution of (X_n) is uniquely characterized by the probabilities of finite substrings, i.e. by all probabilities of the kind $P(\bar{b})$, where $\bar{b} \in O^*$ (O^* denotes the set of all finite strings over O including the empty string).

For every $\bar{a} \in O^*$, we define a real-valued function

$$\begin{array}{rcl} f_{\bar{a}} : O^* & \to & \mathbb{R}, \\ \bar{b} & \mapsto & \begin{cases} P(\bar{b} \,|\, \bar{a}), & \text{if } P(\bar{a}) \neq 0, \\ 0, & \text{if } P(\bar{a}) = 0, \end{cases} \end{array} \tag{1}$$

with the understanding that $f_{\bar{a}}(\varepsilon) = 1$ if $P(\bar{a}) > 0$, else it is 0.

A function $f_{\bar{a}}$ describes the future distribution of the process after an initial realization $\bar{a}$. In our robot illustration in Figure 1, $\bar{a}$ would correspond to the past that the robot has in its short-term memory (symbolized by the blue trajectory), and $f_{\bar{a}}$ would correspond to the "spaghetti bundle" of future trajectores,

as anticipated at that moment. We call these $f_{\bar{a}}$ the *prediction functions* of the process.

Let F be the functional vector space spanned by the prediction functions. Thus F can be seen as the (linear closure of the) space of future distributions of the process (X_t).

We now define the observable operators. In order to specify a linear operator on a vector space, it suffices to specify the values the operator takes on a basis of the vector space. Choose a set $(f_{\bar{a}_i})_{i \in I}$ of prediction functions that is a basis of F. Define, for every $a \in O$, a linear *observable operator* $t_a : F \to F$ by putting

$$t_a(f_{\bar{a}_i}) = P(a \,|\, \bar{a}) f_{\bar{a}_i a} \tag{2}$$

for all $i \in I$ ($\bar{a}a$ denotes the concatenation of the sequence $\bar{a}$ with a). It is easy to verify [30] that (2) carries over from basis elements $f_{\bar{a}_i}$ to all $\bar{a} \in O^*$:

Proposition 1 *For all $\bar{a} \in O^*$, $a \in O$, the linear operator t_a satisfies the condition*

$$t_a(f_{\bar{a}}) = P(a \,|\, \bar{a}) f_{\bar{a}a}. \tag{3}$$

Furthermore, the definition of observable operators does not depend on the choice of basis of F.

Intuitively, the observable operator t_a describes the change of knowledge about a process' future due to an incoming observation of a – which is just the idea of our update operators. A new ingredient that we find here is that the updated future distribution $f_{\bar{a}a}$ becomes weighted by $P(a \,|\, \bar{a})$. This circumstance can be used to express the probability of a sequence $P(a_0 \dots a_r)$ in terms of the operators $t_{a_0}, \dots, t_{a_r}$. Let $\sigma : F \to \mathbb{R}$ be the linear function that returns 1 on all basis vectors $f_{\bar{a}_i}$. Then the following proposition holds (proof in [30]):

Proposition 2 *For all $a_0 \dots a_r \in O^*$,*

$$P(a_0 \dots a_r) = \sigma \; t_{a_r} \cdots t_{a_0} \; f_\varepsilon. \tag{4}$$

Note that Eqns. (3) and (4) are valid for any choice of basis vectors $f_{\bar{a}_i}$. Equation (4) is the fundamental equation of OOM theory. It reveals how the distribution of any stationary symbol process can be expressed purely by means of linear algebra. Furthermore, the observable operators and f_ε are uniquely determined by the the distribution of (X_t). This leads to the following definition:

Definition 1 *Let $(X_n)_{n \in \mathbb{N}}$ be a stationary stochastic process with values in a finite set O. The structure $(F, (t_a)_{a \in O}, f_\varepsilon)$ is called the* observable operator model *of the process. The vectors $f_{\bar{a}}$ are called* states *of the process; the state f_ε is called the* initial state*. The vector space dimension of F is called the* dimension of the process*.*

We will soon introduce matrix representations of OOMs. If we wish to distinguish the abstract OOMs introduced above from matrix representations, we will speak of "functional" vs. "matrix" OOMs, respectively.

We have only treated the discrete time, discrete value, stationary case here. However, OOMs can be defined in a similar way also for non-stationary, continuous-time, arbitrary-valued processes [29]. It turns out that in those cases the resulting observable operators are linear too. In the sense of updating prediction functions, the change of knowledge about a process due to incoming observations is a linear phenomenon.

3 From HMMs to OOMs: Matrix Representations of OOMs

If one wishes to carry out concrete computations, one has to work with finite-dimensional matrix representations of OOMs. Instead of deriving them from the abstract Definition 1, we will introduce matrix representations of OOMs in a very different way, by showing how they can be obtained as a generalization of HMMs.

A basic HMM specifies the distribution of a discrete-time, discrete-value stochastic process $(Y_n)_{n\in\mathbb{N}}$, where the random variables Y_n have outcomes in an alphabet $O = \{a^1, \ldots, a^\alpha\}$. To specify $(Y_n)_{n\in\mathbb{N}}$, first a Markov chain $(X_n)_{n\in\mathbb{N}}$ is considered that produces sequences of *hidden* states from a state set $\{s_1, \ldots, s_m\}$. Second, when the Markov chain is in state s_j at time n, it "emits" an observable outcome a_i with a time-invariant probability $P(Y_n = a_i \,|\, X_n = s_j)$.

We now represent a HMM in a matrix formalism that is a bit different from the one customarily found in the literature. The Markov chain state transition probabilities are collected in an $m \times m$ stochastic matrix M which at position (i, j) contains the transition probability from state s_i to s_j. For every $a \in O$, we collect the emission probabilities $P(Y = a \,|\, X = s_j)$ in the diagonal of an $m \times m$ matrix O_a that is otherwise zero.

In order to fully characterize a HMM, one must supply an initial distribution $w_0 = (P(X_0 = s_1), \ldots, P(X_0 = s_m))^\top$ (superscript $^\top$ denotes transpose of vectors and matrices). The process described by the HMM is stationary if w_0 is an invariant distribution of the Markov chain [10], namely, if it satisfies

$$M^\top w_0 = w_0. \tag{5}$$

We consider only stationary processes here. The matrices M, O_a and w_0 can be used to compute the probability of finite observation sequences. Let $\mathbf{1} = (1, \ldots, 1)$ denote the m-dimensional row vector of units, and let $T_a := M^\top O_a$. Then the probability to observe the sequence $a_0 \ldots a_r$ among all possible sequences of length $r + 1$ is obtained by

$$P(a_0 \ldots a_r) = \mathbf{1} T_{a_r} \cdots T_{a_0} w_0. \tag{6}$$

Equation (6) is a matrix notation of the well-known forward algorithm for determining probabilities of observation sequences in HMMs. Proofs may be found in [24] and [23].

Matrix M can be recovered from the operators T_a by observing that

$$M^\top = M^\top \cdot \mathbf{id} = M^\top (O_{a^1} + \cdots + O_{a^\alpha}) = T_{a^1} + \cdots + T_{a^\alpha}, \tag{7}$$

where **id** denotes the identity matrix. Equation (6) shows that the distribution of the process (Y_t) is specified by the operators T_a and the vector w_0. Thus, the matrices T_a and w_0 contain the same information as the original HMM specification in terms of M, O_a and w_0. Namely, one can rewrite a HMM as a structure $(\mathbb{R}^m, (T_a)_{a\in O}, w_0)$, where $\mathbb{R}^m$ is the domain of the operators T_a.

From here one arrives at the definition of a finite-dimensional OOM in matrix representation by (i) relaxing the requirement that $M^\top$ be the transpose of a stochastic matrix, to the weaker requirement that the columns of M^T each sum to 1, and by (ii) requiring from w_0 merely that it has a component sum of 1. That is, negative entries are now allowed in matrices and vectors, which are forbidden in the stochastic matrices and probability vectors of HMMs. Using the symbol τ in OOMs in places where T appears in HMMs, and introducing $\mu = \sum_{a\in O} \tau_a$ in analogy to (7) we get:

Definition 2 *An m-dimensional (matrix) OOM is a triple $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$, where $w_0 \in \mathbb{R}^m$ and $\tau_a : \mathbb{R}^m \to \mathbb{R}^m$ are linear maps represented by matrices, satisfying three conditions:*

1. $\mathbf{1}w_0 = 1$,
2. $\mu = \sum_{a\in\mathcal{O}} \tau_a$ *has column sums equal to 1,*
3. *for all sequences $a_0 \dots a_r$ it holds that $\mathbf{1}\tau_{a_r} \cdots \tau_{a_0} w_0 \geq 0$.*

Conditions 1 and 2 reflect the relaxations (i) and (ii) mentioned previously, while condition 3 ensures that one obtains non-negative values when the OOM is used to calculate probabilities. While the non-negativity of matrix entries in HMMs guarantees non-negativity of values obtained from the right-hand-side (rhs) of Eq. (6), non-negativity must be expressedly assured for OOMs. Unfortunately, for given operators $(\tau_a)_{a\in O}$ there exists no known way to decide whether condition 3 holds. This is our first encounter with the central unresolved issue in OOM theory, and we will soon hear more about (and suffer from) it.

Since concatenations of operators like $\tau_{a_r} \cdots \tau_{a_0}$ will be much used in the sequel, we introduce a shorthand notation: for $\tau_{a_r} \cdots \tau_{a_0}$ we also write $\tau_{a_0 \cdots a_r}$ (be aware of the reversal of indices) or even $\tau_{\bar{a}}$.

A matrix-based OOM specifies a stochastic process as in (4):

Proposition 3 *Let $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ be an OOM according to the previous definition. Let $\Omega = O^\infty$ be the set of all right-infinite sequences over O, and $\mathfrak{A}$ be the σ-algebra generated by all finite-length initial sequences on Ω. Then, if one computes the probabilities of finite-length sequences by*

$$P_0(\bar{a}) = \mathbf{1}\tau_{\bar{a}} w_0, \tag{8}$$

where the numerical function P_0 can be uniquely extended to a probability measure P on $(\Omega, \mathfrak{A})$, giving rise to a stochastic process $(\Omega, \mathfrak{A}, P, (X_n)_{n\in\mathbb{N}})$, where $X_n(a_1a_2\ldots) = a_n$. If w_0 is an invariant vector of μ, i.e., if $\mu w_0 = w_0$, the process is stationary.

A proof can be found in [30]. Since we introduce matrix OOMs here by generalizing away from HMMs, it is clear that every process that can be characterized by a finite-dimensional HMM can also be described by a matrix OOM of dimension at most the number of hidden HMM states.

Conversely, there exist processes that can be described by a matrix OOM, but that cannot be characterized by a finite-dimensional HMM. One way to construct examples of such processes is to design one of the operators τ_a to be a rotation of $\mathbb{R}^m$ by a non-rational angle ϕ. Such a rotation gives rise to a "probability oscillation", that is, the sequence $P(a \mid a^n)_{n\geq 0}$ converges to an oscillation with angular velocity ϕ (radian per unit time step). Intuitively, the reason why such a process cannot be modelled by an HMM is that a matrix describing a rotation needs to contain some negative entries. If a HMM for such a process would exist, reinterpreting it as an OOM according to the construction $T_a = M^\top O_a$ would yield a purely non-negative matrix for the rotating operator, which is impossible. A concrete example of such a process (dubbed the "probability clock") and a proof that it is not a hidden Markov process was given in [30].

In Section 2 we introduced abstract OOMs in a top down fashion, by starting from a stochastic process and transforming it into its OOM. In this section we introduced matrix OOMs in a bottom-up fashion by abstracting away from HMMs. These two are related as follows (for proofs, see [30; 25]):

- A matrix OOM of matrix dimension m specifies a stochastic process of process dimension $m' \leq m$.
- A process of finite dimension m has matrix OOMs of matrix dimension m.
- A process of finite dimension m has no matrix OOMs of smaller matrix dimension.

When we refer to OOMs in the remainder of this chapter we mean matrix OOMs.

4 OOMs as Generators and Predictors

In this section we describe how an OOM can be used to generate a random sequence, and to compute the probabilities of possible continuations of a given initial sequence.

Concretely, assume that an OOM $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ describes a process $(X_n)_{n\geq 0}$, where $O = \{a^1, \ldots, a^\alpha\}$. Then, the task is to use $\mathcal{A}$ to produce at times $n = 0, 1, 2, \ldots$ observations $a_0, a_1, a_2, \ldots$, such that (i) at time $n = 0$, the probability of producing a is equal to $P(X_0 = a)$, and (ii) at every time

step $n > 0$, the probability of producing a (after $a_0, \ldots, a_{n-1}$ have already been produced) is equal to $P(X_n = a \,|\, X_0 = a_0, \ldots, X_{n-1} = a_{n-1})$. We address (i) and (ii) in turn.

(i). For generating the first symbol we need the probability vector $\mathbf{p}_0 = (P(X_0 = a^1) \cdots P(X_0 = a^\alpha))^\top$. This could be done by calculating $P(X_0 = a) = \mathbf{1}\tau_a w_0$ for all $a \in O$. A faster way is to precalculate the row vectors $\mathbf{1}\tau_a$ for all a, and assemble them in a matrix

$$\Sigma = \begin{bmatrix} \mathbf{1}\tau_{a^1} \\ \vdots \\ \mathbf{1}\tau_{a^\alpha} \end{bmatrix}, \tag{9}$$

and directly obtain

$$\mathbf{p}_0 = \Sigma\, w_0. \tag{10}$$

This probability vector is then used to randomly generate the symbol a_0 with the correct distribution.

(ii). In order to obtain $P(X_n = a \,|\, X_0 = a_0, \ldots, X_{n-1} = a_{n-1})$ we make use of Eq. (8):

$$\begin{aligned} & P(X_n = a \,|\, X_0 = a_0, \ldots, X_{n-1} = a_{n-1}) \\ &= \mathbf{1}\tau_a\tau_{a_{n-1}} \cdots \tau_{a_0} w_0 \;/\; \mathbf{1}\tau_{a_{n-1}} \cdots \tau_{a_0} w_0 \\ &= \mathbf{1}\tau_a \left(\frac{\tau_{a_{n-1}} \cdots \tau_{a_0} w_0}{\mathbf{1}\tau_{a_{n-1}} \cdots \tau_{a_0} w_0}\right). \end{aligned} \tag{11}$$

Introducing the notation

$$w_{a_0 \ldots a_{n-1}} = \frac{\tau_{a_{n-1}} \cdots \tau_{a_0} w_0}{\mathbf{1}\tau_{a_{n-1}} \cdots \tau_{a_0} w_0}, \tag{12}$$

Equation (11) can be more concisely written as $P(X_n = a \,|\, X_0 = a_0, \ldots, X_{n-1} = a_{n-1}) = \mathbf{1}\tau_a w_{a_0 \ldots a_{n-1}}$. A vector $w_{\bar{a}}$ of the kind (12) that arises after a sequence $\bar{a}$ has been observed is called a *state vector* of an OOM. Note that state vectors have unit component sum. Again we can use Σ to obtain all of the probabilities $P(a^i \,|\, \bar{a})$ in a single operation:

$$\mathbf{p}_n = (P(a^1 \,|\, \bar{a}) \cdots P(a^\alpha \,|\, \bar{a}))^\mathsf{T} = \Sigma\, w_{\bar{a}}. \tag{13}$$

Observing that the next state vector can be obtained from the previous one by

$$w_{\bar{a}a} = \tau_a w_{\bar{a}} \;/\; \mathbf{1}\tau_a w_{\bar{a}}, \tag{14}$$

the entire generation procedure can be neatly executed as follows:

1. State vector initialization: put $w = w_0$.
2. Assume that at time n a state vector w_n has been computed, then determine the probability vector $\mathbf{p}$ of the $(n+1)$-st symbol as $\Sigma\, w_n$, and choose a_n according to that vector.

3. Update the state vector by $w_{n+1} = \tau_{a_n} w_n \;/\; \mathbf{1}\tau_{a_n} w_n$ and resume at step *2.*

Now we consider the task of predicting the probability $P(\bar{b}\,|\,\bar{a})$ of a continuation $\bar{b}$ of an initial sequence $\bar{a}$ that has already been observed. It is easy to see that an iterated application of eq. (11) yields

$$\begin{aligned} P(X_{n+1} = b_{n+1}, \ldots, X_{n+r} = b_{n+r} \,|\, X_0 = a_0, \ldots, X_{n-1} = a_{n-1}) \\ = \mathbf{1}\tau_{b_{n+r}} \cdots \tau_{b_{n+1}}\, w_{a_0 \cdots a_n}, \end{aligned} \tag{15}$$

which in our shorthand notation becomes $P(\bar{b}\,|\,\bar{a}) = \mathbf{1}\tau_{\bar{b}}\, w_{\bar{a}}$. If one is interested in repeated predictions of the probability of a particular continuation $\bar{b}$ (for instance, an English word), then it pays off to precalculate the row vector $\sigma_{\bar{b}} = \mathbf{1}\tau_{\bar{b}}$ and obtain $P(\bar{b}\,|\,\bar{a}) = \sigma_{\bar{b}}\, w_{\bar{a}}$ by a single inner product computation.

5 Understanding Matrix OOMs by Mapping Them to Functional OOMs

OOM states are conceptually quite different from HMM states. This conceptual issue is complicated by the circumstance that the term "state" is used in two different ways for HMMs. First, it may denote the finite set of *physical states* that the target system is assumed to take. Second, it is used for the *current* probability distribution over these physical states that can be inferred from a previous observation sequence. In both cases, the notion is connected to the assumed physical states of the target system. By contrast, OOM states represent the expectation about the system's *future* and *outwardly observable* development given an observed past. In no way do OOM states refer to any assumed physical state structure of the target system — they are purely epistemic, one might say. Incidentally, this agrees with the perspective of modern physics and abstract systems theory: "[...] a state of a system at any given time is the information needed to determine the behaviour of the system from that time on" [42]. This perspective was constitutional for the construction of functional OOMs in Section 2. We will now add further substance to this view by showing how matrix OOMs map to functional OOMs, and thereby how the finite state vectors of matrix OOMs represent the process' future. As by-products our investigation will yield a construction for minimizing the dimension of a matrix OOM, and an algebraic characterization of matrix OOM equivalence.

Definition 3 *Let $\mathcal{A} = (\mathbb{R}^l, (\tau_a)_{a\in O}, w_0)$ be a matrix OOM of the process $(X_n)_{n\geq 0}$. Let $\mathcal{F} = (F, (t_a)_{a\in O}, f_\varepsilon)$ be the functional OOM of the same process. Let W be the linear subspace of $\mathbb{R}^l$ spanned by the state vectors $\{w_{\bar{a}} \mid \bar{a} \in O^*\}$. Let $\{w_{\bar{a}_1}, \ldots, w_{\bar{a}_d}\}$ be a basis of W. Define a linear mapping $\pi : \mathbb{R}^l \to F$ through $\pi(w_{\bar{a}_i}) = f_{\bar{a}_i} \quad (i = 1, \ldots, d)$. This mapping is called the* canonical projection *of $\mathcal{A}$.*

This definition is independent of the choice of basis, and the canonical projection has the following properties:

Proposition 4 *1.* $\forall\, \bar{a} \in O^*\ \ \pi(w_{\bar{a}}) = f_{\bar{a}}$.

2. π *is surjective.*

3. $\forall\, w \in W\ \ \sigma\pi(w) = \mathbf{1}w$.

4. $\forall\, \bar{a} \in O^*,\ w \in W\ \ \pi(\tau_{\bar{a}} w) = t_{\bar{a}}\, \pi(w)$.

The proof of *1.* – *3.* is given in [25], the proof of *4.* is in the Appendix A. Note that *3.* implies that the matrix dimension l of $\mathcal{A}$ is at least as great as the process dimension m.

Our goal is now to distil from the l-dimensional state vectors of the matrix OOM those parts which are relevant for representing the process' future. Intuitively, if the process dimension is m, only projections of the matrix OOM states on some m-dimensional subspace of $\mathbb{R}^l$ contain relevant information.

First observe that a basis $\{w_{\bar{a}_1}, \ldots, w_{\bar{a}_d}\}$ of the linear subspace W can be effectively constructed from $\mathcal{A}$, as follows. Construct a sequence of sets $(S_j)_{j=0,1,\ldots,r}$ of states as follows:

1. Let $S_0 = \{w_0\}$.
2. Obtain S_{j+1} from S_j by first adding to S_j all states from the set $\{\tau_a w \,/\, \mathbf{1}\tau_a w \,|\, a \in O, w \in S_j\}$, and then deleting from the obtained set as many states as necessary to get a maximal set of linearly independent states.
3. When the size of S_{j+1} is equal to the size of S_j, stop and put $r = j$; else resume at 2.

It is clear that the size of the sets $(S_j)_{j=0,1,\ldots,r}$ properly grows throughout the sequence, and that the vectors contained in S_r yield the desired basis for W.

To determine the "prediction relevant" portions in the states w, we investigate the kernel, denoted as *ker* π, of the canonical projection.

Proposition 5

$$\forall x \in W\ \ x \in \mathit{ker}\ \pi \ \ \Leftrightarrow\ \ \forall \bar{a} \in O^*\ \ \mathbf{1}\tau_{\bar{a}} x = 0. \tag{16}$$

The proof is in the Appendix B. As a special case we get $\mathbf{1}x = 0$ for all $x \in \mathit{ker}\ \pi$. Using this insight, a basis for *ker* π can be constructed from $\mathcal{A}$ as follows. Again build a sequence $(S_j)_{j=0,1,\ldots,s}$ of sets of (row) vectors:

1. Let $S_0 = \{\mathbf{1}\}$.
2. Obtain S_{j+1} from S_j by first adding to S_j all vectors from the set $\{u\tau_a \,|\, a \in O, u \in S_j\}$, and then delete from the obtained set as many vectors as necessary to get a maximal set of linearly independent vectors.
3. When the size of S_{j+1} is equal to the size of S_j, stop and put $s = j$; else resume at Step 2.

It follows from Proposition 5 that

$$ker\ \pi = \{x \in W \,|\, \forall u \in S_s \;\; x \perp u^\top\}, \tag{17}$$

from which some orthonormal basis for $ker\ \pi$ is readily constructed. Since π is surjective we have $dim\ ker\ \pi = d - m$. Let $\{x_1, \ldots, x_{d-m}\}$ be such a basis. Consider the orthogonal complement of the kernel:

$$V = \{v \in W \,|\, v \perp ker\ \pi\}. \tag{18}$$

where V is a linear subspace of W and has a dimensionality of m. It is an easy exercise to obtain a concrete representation of V through creating an orthonormal basis for V.

For $w \in W$, let $\tilde{w}$ denote the orthogonal projection of w on V. From linearity of orthogonal projections and Proposition 5 we obtain that

$$\mathbf{1}\tilde{w} = \mathbf{1}w \tag{19}$$

for all $w \in W$. Let π_0 be the restriction of π on V. In light of (19) and Proposition 4(*3*), π_0 preserves our probability measuring functionals $\mathbf{1}$ (in $\mathcal{A}$) and σ (in $\mathcal{F}$) in the sense that $\mathbf{1}v = \sigma\pi_0(v)$ for all $v \in V$.

Furthermore, define restrictions $\tilde{\tau}_a$ of the observable operators τ_a by

$$\tilde{\tau}_a v = \widetilde{\tau_a v} \tag{20}$$

for all $v \in V$. It is easy to see that $\tilde{\tau}_a$ is linear, and a matrix representation for $\tilde{\tau}_a$ is readily obtained from the bases of V and $ker\ \pi$. The projection π_0 maps $\tilde{\tau}_a$ on t_a by virtue of

$$\forall\ v \in V \;\; \pi_0(\tilde{\tau}_a v) = \pi_0(\widetilde{\tau_a v}) = \pi(\tau_a v) = t_a \pi(v), \tag{21}$$

where the last equality follows from Proposition 4(*4*). Assembling our findings we see that

$$\pi_0 : (V, (\tilde{\tau}_a)_{a \in O}, \tilde{w}_0) \cong (F, (t_a)_{a \in O}, f_\varepsilon) \tag{22}$$

induces an isomorphism of vector spaces and operators which maps $\mathbf{1}$ on σ. This is just another way of saying that $(V, (\tilde{\tau}_a)_{a \in O}, \tilde{w}_0)$ is an OOM for our process. Note that V is represented here as a linear subspace of $\mathbb{R}^l$ and the matrices $\tilde{\tau}_a$ have a size of $l \times l$. A little more elementary linear algebra would finally transform $(V, (\tilde{\tau}_a)_{a \in O}, \tilde{w}_0)$ into an m-dimensional (and thus minimal-dimensional) matrix OOM.

We are now prepared to provide a simple answer to the question when two matrix OOMs $\mathcal{A}$ and $\mathcal{A}'$ are equivalent in the sense of yielding identical probabilities for finite sequences. To decide the equivalence between $\mathcal{A}$ and $\mathcal{A}'$, we first transform them into minimal-dimensional OOMs of dimension m (if their minimal dimensions turn out not to be identical, they are not equivalent), we then obtain the following proposition:

Proposition 6 *Two minimal-dimensional OOMs* $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ *and* $\mathcal{A}' = (\mathbb{R}^m, (\tau'_a)_{a\in O}, w'_0)$ *are equivalent if and only if there exists a bijective linear map* $\varrho : \mathbb{R}^m \to \mathbb{R}^m$*, satisfying the following conditions:*

1. $\varrho(w_0) = w'_0$,
2. $\tau'_a = \varrho\tau_a\varrho^{-1}$ *for all* $a \in \mathcal{O}$,
3. $\mathbf{1}w = \mathbf{1}\varrho w$ *for all* $w \in \mathbb{R}^m$.

Sketch Proof. (for detailed proof see [25]). The "if" direction is a mechanical verification. The interesting direction is to show that if $\mathcal{A}$ and $\mathcal{A}'$ are equivalent then a map ϱ exists. First observe that for minimal-dimensional OOMs, the canonical projection π coincides with π_0 and is an isomorphism of the matrix OOM with the functional OOM. Let π, π' be the canonical projections $\mathcal{A}$ and $\mathcal{A}'$, respectively, then $\varrho = \pi'^{-1}\,\pi$ satisfies the conditions of the proposition.

A matrix ϱ satisfies condition *3* of Proposition 6 from the proposition if and only if each column of ϱ sums to unity. Thus, if we have one minimal-dimensional OOM $\mathcal{A}$, we get all the other equivalent ones by applying any transformation matrix ϱ with unit column sum.

6 Characterizing OOMs via Convex Cones

The problematic non-negativity condition *3* from Definition 2 can be equivalently stated in terms of convex cones. This sheds much light on the relationship between OOMs and HMMs, and also allows one to appreciate the difficulty of the issue. I first introduce some cone-theoretic concepts, following the notation of a standard textbook [3].

With a set $S \subseteq \mathbb{R}^n$ we associate the set S^G, the *set generated by* S, which consists of all finite nonnegative linear combinations of elements of S. A set $K \subseteq \mathbb{R}^n$ is defined to be a *convex cone* if $K = K^G$. A convex cone K^G is called *n-polyhedral* if K has n elements. A cone K is *pointed* if for every nonzero $w \in K$, the vector $-w$ is not in K.

Using these concepts, the following proposition gives a condition which is equivalent to condition *3* from Definition 2, and clarifies the relationship between OOMs and HMMs.

Proposition 7 (i) *Let* $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ *be a structure satisfying the first two conditions from Definition 2, i.e.* $\mathbf{1}w_0 = 1$ *and* $\mu = \sum_{a\in O}\tau_a$ *has unit column sums. Then* $\mathcal{A}$ *is an OOM if and only if there exists a pointed convex cone* $K \subset \mathbb{R}^m$ *satisfying the following conditions:*

1. $\mathbf{1}w \geq 0$ *for all* $w \in K$,
2. $w_0 \in K$,
3. $\forall a \in O :\ \tau_a K \subseteq K$.

(ii) *Assume that $\mathcal{A}$ is an OOM, then there exists a HMM equivalent to $\mathcal{A}$ if and only if a pointed convex cone K according to* **(i)** *exists which is n-polyhedral for some n, where n can be selected such that it is not greater than the minimal state number for HMMs equivalent to $\mathcal{A}$.*

Part **(i)** can be proven by reformulating a similar claim [30] that goes back to [21] and has been renewed in [23][2]. Part **(ii)** was shown in [23]. These authors considered a class of stochastic processes called "linearly dependent processes" that is identical to what we introduced as processes with finite dimension m; they did not use observable operators to characterize the processes.

Part **(ii)** has the following interesting implications:

- Every two-dimensional OOM is equivalent to some HMM, because all cones in $\mathbb{R}^2$ are 2-polyhedral. A nice exercise left to the reader is to construct a 2-dimensional OOM whose smallest equivalent HMM has 4 states (hint: derive a 2-dimensional OOM from a HMM defined not by emitting observations from states but from state transitions).

- If an OOM contains an operator τ_a that rotates $\mathbb{R}^m$ by a non-rational multiple of π, then this OOM has no equivalent HMM because τ_a leaves no polyhedral cone invariant.

- Three-dimensional OOMs can be constructed whose equivalent minimal-size HMMs have at least p states (for any prime $p \geq 3$), by equipping the OOM with an operator that rotates $\mathbb{R}^3$ by $2\pi/p$. This is so because any polyhedral cone left invariant by such an operator is at least p-polyhedral.

Proposition 7 is useful to *design* interesting OOMs, starting with a cone K and constructing observable operators satisfying $\tau_a K \subseteq K$. Unfortunately it provides no means to *decide*, for a given structure $\mathcal{A}$, whether $\mathcal{A}$ is a valid OOM, since the proposition is non-constructive w.r.t. K.

If one would have effective algebraic methods to decide, for a set of k linear operators on $\mathbb{R}^m$, whether they leave a common cone invariant, then one could decide whether a candidate structure $(\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ is a valid OOM. However, this is a difficult and unsolved problem of linear algebra. For a long time, only the case of a single operator ($k = 1$) was understood [3]. Recently however there was substantial progress in this matter. In [12] interesting subcases of $k = 2$ were solved, namely, the subcases of of $m = 2$ and of polyhedral cones.

7 Interpretable OOMs

OOM states represent future distributions, but the previous section might have left the impression that this representation is somewhat abstract. We will now see that within the equivalence class of a given minimal-dimensional OOM,

[2]Heller and Ito used a different definition for HMMs, which yields a different version of the minimality statement in part **(ii)**

there are some members whose states can be interpreted immediately as future distributions – *interpretable* OOMs. Interpretable OOMs are pivotal for OOM learning algorithms.

Because this concept is so important for OOM theory we will first illustrate it with an informal example. Assume we have a 26-dimensional OOM $\mathcal{A}$ over the English alphabet $O = \{a, \ldots, z\}$ — the OOM dimension and the alphabet size accidentally coincide. Assume that $\mathcal{A}$ models the distribution of letter sequences in English texts. Utilizing the generation procedure from Section 4, $\mathcal{A}$ can be run to generate strings of pseudo-English. Remember that at time n, the state w_n is used to compute a 26-dimensional probability vector $\mathbf{p}_{n+1}$ of the nth occurring letter via $\mathbf{p}_n = \Sigma\, w_n$, where Σ's rows are made from the column sums of the 26 observable operators (Eqn. (13)).

Wouldn't it be convenient if we had $\mathbf{p}_{n+1} = w_n$ and $\Sigma = \mathbf{id}$ (where $\mathbf{id}$ denotes the identity matrix)? Then we could immediately take the next letter probabilities from the current state vector, spare us the computation of $\Sigma\, w_n$, and directly "see" the development of very interesting probabilities in the state evolution.

We will now see that such an *interpretable* OOM can be constructed from $\mathcal{A}$. The definition of interpretable OOMs is more general than this example suggests in that it admits a more comprehensive notion of the future events whose probabilities become the state vector's entries. In our example, these events that we will call *characteristic events* — were just the singletons $a, \ldots, z$. Here is the general definition of such events:

Definition 4 *Let $(X_n)_{n\geq 0}$ be an m-dimensional stationary process with observables from O. Let, for some sufficiently large l, $O^l = B_1 \cup \cdots \cup B_m$ be a partition of the set of strings of length l into m disjoint, non-empty sets B_i. Then this partition is called a set of* characteristic events *B_i $(i = 1, \ldots, m)$, if some sequences $\bar{a}_1, \ldots, \bar{a}_m$ exist such that the matrix $(P(B_i \,|\, \bar{a}_j))_{1\leq i,j\leq m}$ is nonsingular.*

Here by $P(B_i \,|\, \bar{a}_j)$ we mean $\sum_{\bar{b}\in B_i} P(\bar{b} \,|\, \bar{a}_j)$. We introduce some further notational commodities. For a state vector w of an OOM $\mathcal{A}$ of $(X_n)_{n\geq 0}$ and a sequence $\bar{b}$ let $P(\bar{b} \,|\, w) = \mathbf{1}\tau_{\bar{b}} w$ denote the probability that the OOM will produce $\bar{b}$ when started in state w. Furthermore, let $P(B_i \,|\, w) = \sum_{\bar{b}\in B_i} P(\bar{b} \,|\, w)$. Now we are equipped to define interpretable OOMs:

Definition 5 *Let $B_1, \ldots, B_m$ be characteristic events for an m-dimensional process with observables O, and let $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ be an OOM for that process. Then $\mathcal{A}$ is* interpretable *w.r.t. $B_1, \ldots, B_m$ if the states w of $\mathcal{A}$ have the property*

$$w = (P(B_1 \,|\, w) \cdots P(B_m \,|\, w))^\top. \tag{23}$$

Here is a method to transform a given OOM $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ for $(X_n)_{n\geq 0}$ into an OOM that is interpretable w.r.t. characteristic events $B_1, \ldots, B_m$. Define $\tau_{B_i} := \sum_{\bar{b}\in B_i} \tau_{\bar{b}}$. Define a mapping $\varrho : \mathbb{R}^m \to \mathbb{R}^m$ by

$$\varrho(x) := (\mathbf{1}\tau_{B_1} x \cdots \mathbf{1}\tau_{B_m} x)^\top. \tag{24}$$

The mapping ϱ is obviously linear. It is also bijective, since according to the definition of characteristic events, sequences $\bar{a}_j$ exist such that the matrix $(P(B_i \,|\, \bar{a}_j)) = (\mathbf{1}\tau_{B_i} x_j)$, where $x_j = \tau_{\bar{a}_j} w_0 / \mathbf{1}\tau_{\bar{a}_j} w_0$, is nonsingular. Furthermore, ϱ preserves component sums of vectors, since for $j = 1, \ldots, m$ it holds that $\mathbf{1}x_j = 1 = \mathbf{1}(P(B_1 \,|\, x_j) \cdots P(B_m \,|\, x_j))^\top = \mathbf{1}(\mathbf{1}\tau_{B_1} x_j \cdots \mathbf{1}\tau_{B_m} x_j)^\top = \mathbf{1}\varrho(x_j)$ (a linear map preserves component sums if it preserves component sums of basis vectors). Hence ϱ satisfies the conditions of Proposition 6. We therefore obtain an OOM equivalent to $\mathcal{A}$ by

$$\mathcal{A}' = (\mathbb{R}^m, (\varrho\tau_a\varrho^{-1})_{a\in O}, \varrho w_0) = (\mathbb{R}^m, (\tau'_a)_{a\in O}, w'_0). \tag{25}$$

Equation (23) holds in $\mathcal{A}'$. To see this, let w'_n be a state vector obtained in a generation run of $\mathcal{A}'$ at time n, and w_n the state obtained in $\mathcal{A}$ after the same sequence has been generated. Then it concludes that

$$\begin{aligned} w'_n &= \varrho\varrho^{-1} w'_n \\ &= (\mathbf{1}\tau_{B_1}(\varrho^{-1} w'_n) \cdots \mathbf{1}\tau_{B_m}(\varrho^{-1} w'_n))^\top \\ &= (\mathbf{1}\tau_{B_1} w_n \cdots \mathbf{1}\tau_{B_m} w_n)^\top \\ &= (P(B_1 \,|\, w_n) \cdots P(B_m \,|\, w_n))^\top \\ &= (P(B_1 \,|\, w'_n) \cdots P(B_m \,|\, w'_n))^\top, \end{aligned}$$

where the last equality follows from the equivalence of $\mathcal{A}$ and $\mathcal{A}'$.

We will sometimes denote $\mathcal{A}'$ by $\varrho\mathcal{A}$. The $m \times m$ matrix corresponding to ϱ can be obtained from the original OOM $\mathcal{A}$ by observing that

$$\varrho = (\mathbf{1}\tau_{B_i} e_j), \tag{26}$$

where e_i is the i-th unit vector.

The following fact lies at the heart of the learning algorithm presented in the next section:

Proposition 8 *In an OOM that is interpretable w.r.t. $B_1, \ldots, B_m$ it holds that*

1. $w_0 = (P(B_1) \cdots P(B_m))^\top$,
2. $\tau_{\bar{a}} w_0 = (P(\bar{a}B_1) \cdots P(\bar{a}B_m))^\top$,

where $P(\bar{a}B)$ denotes $\sum_{\bar{b}\in B} P(\bar{a}\bar{b})$. The proof is trivial.

Most often interpretable OOMs are used in a context when they are minimal-dimensional, but sometimes it is useful to generalize the notion by dropping the requirement of minimal-dimensionality. An n-dimensional OOM of an m-dimensional process is called interpretable w.r.t. $B_1, \ldots, B_n$ if the analog of Eq. (23) holds. An n-dimensional OOM with operators τ_a can be made interpretable by putting $\tau'_a = \varrho\,\tau_a\varrho^\dagger$, where again $\varrho = (\mathbf{1}\tau_{B_i} e_j)$ and $\varrho^\dagger$ is the pseudo-inverse of ϱ. A special case that we will need to consider later on is obtained when the B_i are all singletons. We introduce some special concepts for this case:

Definition 6 *Let (X_n) be an m-dimensional process over an observation alphabet O. Fix some $k \in \mathbb{N}, k > 0$, put $\kappa = |O|^k$ and let $\bar{b}_1, \ldots, \bar{b}_\kappa$ be the alphabetical enumeration of O^k. Then these sequences $\bar{b}_i$ are the* characteristic sequences of length k *for (X_n) if m "indicative" sequences $\bar{a}_1, \ldots, \bar{a}_m$ exist that make the $\kappa \times m$ matrix $V = (P(\bar{b}_i|\bar{a}_j))$ regular. The minimal k for which such sequences $\bar{a}_1, \ldots, \bar{a}_m$ exist is the* characterizing length *of (X_n).*

We list two properties of characteristic sequences (the simple proof is left to the reader):

Proposition 9 *Let $\bar{b}_i$ be characteristic sequences of $(X_n)_{n\geq 0}$ of length k and let $\kappa = |O|^k$.*

1. *If $\mathcal{A} = (\mathbb{R}^n, (\tau_a)_{a\in O}, w_0)$ is some (not necessarily minimal-dimensional) OOM for $(X_n)_{n\geq 0}$, then the $\kappa \times n$ matrix $\pi_\mathcal{A}$ that has as its i-th row $\mathbf{1}\tau_{\bar{b}_i}$ maps states w of $\mathcal{A}$ to $\pi_\mathcal{A} w = (P(\bar{b}_1|w) \cdots P(\bar{b}_\kappa|w)^\top$.*

2. *The characterizing length k_0 of (X_n) is the minimal length of characteristic events for (X_n) and vice versa.*

3. *The characterizing length k_0 is less or equal than $m - 1$.*

Here are some observations concerning interpretable OOMs:

- If an m-dimensional OOM $\mathcal{A}$ has been learnt from empirical data, and one chooses disjoint events $B_1, \ldots, B_m$ at random, it is generically the case that some sequences $\bar{a}_1, \ldots, \bar{a}_m$ exist such that the matrix $(P(B_i \mid \bar{a}_j))_{1\leq i,j\leq m}$ is nonsingular. The reason is that the matrix composed from rows $(\mathbf{1}\tau_{B_i})$ is a random matrix and as such generically non-singular. Generally speaking, for arbitrary events $B_1, \ldots, B_m$ being characteristic is the rule, not an exceptional circumstance.

- A given OOM can be transformed into many different equivalent, interpretable OOMs depending to the choice of characteristic events.

- Interpretability yields a very useful way to visualize the state dynamics of an OOM. To see how, first consider the case where the OOM dimension is 3. Interpretable states, being probability vectors, are non-negative and thus lie in the intersection of the positive orthant of $\mathbb{R}^3$ with the hyperplane $H = \{x \in \mathbb{R}^3 \mid \mathbf{1}x = 1\}$. This intersection is a triangular surface. Its corners mark the three unit vectors of $\mathbb{R}^3$. This triangle can be conveniently used as a plotting canvas. Figure 2 shows three "fingerprint" plots of states obtained from generating runs of three different synthetic 3-dimensional OOMs (see Appendix C for details) over an observation alphabet of size 3, which were made interpretable w.r.t. the the same three characteristic events. The states are colored with three colors depending on which of the three operators was used to produce this state. A similar graphical representation of states was first introduced in [39] for HMMs.

When one wishes to plot states of interpretable OOMs with dimension $m > 3$, one can join some of the characteristic events, until three merged events are left, and create plots as explained above.

- If one has several non-equivalent OOMs over the same alphabet O, making them interpretable w.r.t. to a common set of characteristic events is useful for *comparing* them in a meaningful way. This has been done for the three OOMs plotted in Figure 2. Their observable operators depended on a control parameter α which was slightly changed over the three OOMs.

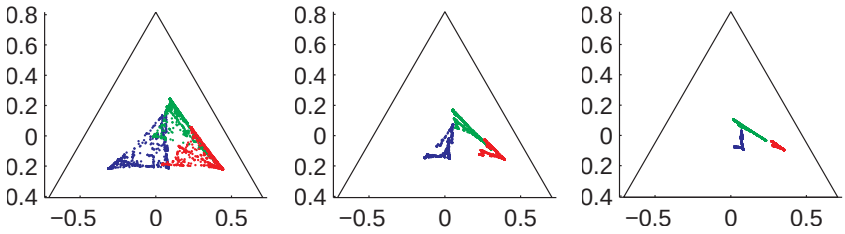

Figure 2: State dynamics "fingerprints" of three related interpretable OOMs. For details see text.

8 The Basic Learning Algorithm

We shall address the following learning task. Assume that a realization $S = a_0 a_1 \cdots a_N$ of a stationary, m-dimensional process (X_n) is given, that is, S is generated by some OOM $\mathcal{A}$ of (minimal) dimension m. We assume that m is known but otherwise $\mathcal{A}$ unknown. We wish to induce from S an estimate $\hat{\mathcal{A}}$ of $\mathcal{A}$ in the sense that the distribution characterized by $\hat{\mathcal{A}}$ comes close to the distribution characterized by $\mathcal{A}$ (the hat $\hat{\cdot}$ will be used throughout this chapter for denoting estimates).

We first collect some observations concerning the unknown generator $\mathcal{A}$. We may assume that $\mathcal{A}$ is interpretable w.r.t. characteristic events $B_1, \ldots, B_m$. Then the principle of learning OOMs emerges from the following observations:

- Proposition 10(*2*) can be used to procure argument-value pairs for the operator τ_a $(a \in O)$ by exploiting

$$\begin{aligned} \tau_a((P(\bar{a}B_1) \cdots P(\bar{a}B_m))^\top) &= \tau_a(\tau_{\bar{a}} w_0) \\ &= \tau_{\bar{a}a} w_0 \\ &= (P(\bar{a}aB_1) \cdots P(\bar{a}aB_m))^\top. \end{aligned} \quad (27)$$

  Such argument-value pairs are vectors that are made from probability values.

- A linear operator on $\mathbb{R}^m$ is determined by any m argument-value pairs provided the arguments are linearly independent.
- Probabilities of the kind $P(\bar{a}B_i)$ that make up the argument-value pairs in (27) can be estimated from the training string S through the relative frequencies $\hat{P}_S$ of the event $\bar{a}B_i$:

$$\hat{P}_S(\bar{a}B_i) = \frac{\text{number of ocurrences of words } \bar{a}\bar{b} \text{ (where } \bar{b} \in B_i) \text{ within } S}{N - \mid \bar{a}B_i \mid +1}, \tag{28}$$

where $\mid \bar{a}B_i \mid$ denotes the length of $\bar{a}$ plus the length of the sequences in B_i.

Thus the blueprint for estimating an OOM $\hat{\mathcal{A}}$ from S is clear:

1. Choose characteristic events $B_1, \ldots, B_m$ and *indicative sequences* $\bar{a}_1, \ldots, \bar{a}_m$ such that the matrix $\hat{V} = \big(\hat{P}_S(\bar{a}_j B_i)\big)_{i,j=1,\ldots,m}$ is non-singular (this matrix contains in its columns m linearly independent argument vectors for the operators τ_a).
2. For each $a \in O$, collect the corresponding value vectors in a matrix $\hat{W}_a = \big(\hat{P}_S(\bar{a}_j a B_i)\big)_{i,j=1,\ldots,m}$.
3. Obtain an estimate for τ_a by

$$\hat{\tau}_a = \hat{W}_a \, \hat{V}^{-1}. \tag{29}$$

If the process (X_n) is ergodic, the estimates $\hat{P}_S(\bar{a}_j B_i), \hat{P}_S(\bar{a}_j a B_i)$ converge with probability 1 to the correct probabilities as the sample size N grows to infinity. This implies that the estimated $\hat{\tau}_a$ will converge to the operators of the true data generator $\mathcal{A}$, assuming that $\mathcal{A}$ is interpretable w.r.t. the characteristic events $B_1, \ldots, B_m$ used in the learning procedure. In other words, the learning algorithm is asymptotically correct.

The statistical efficiency of the algorithm can be improved if instead of using indicative *sequences* $\bar{a}_j$ one uses indicative *events* A_j that partition O^l into m non-empty, disjoint subsets. Then $\hat{V} = \big(\hat{P}_S(A_j B_i)\big)_{i,j=1,\ldots,m}$ and $\hat{W}_a = \big(\hat{P}_S(A_j a B_i)\big)_{i,j=1,\ldots,m}$. If this is done, counting information from *every* subword of S of length $|A_j B_i|$ enters the model estimation, whereas when indicative sequences are used, only those subwords beginning with an indicative sequence are exploited.

A computational simplification of this basic algorithm is obtained if one uses in (29) the raw counting matrices

$$\begin{aligned} V^{\text{raw}} &= \big(\text{count no. of event } A_j B_i \text{ in } S_{\text{short}} = a_0 \ldots a_{N-1}\big)_{i,j=1,\ldots,m}, \\ W_a^{\text{raw}} &= \big(\text{count no. of event } A_j a B_i \text{ in } S\big)_{i,j=1,\ldots,m}. \end{aligned} \tag{30}$$

It is easy to see that $W_a^{\text{raw}} \, (V^{\text{raw}})^{-1} = \hat{W}_a \, \hat{V}^{-1}$.

The counting matrices can be gleaned in a single sweep of a window of length $\mid A_j B_i \mid$ across S, and the computation of (29) incurs $O(m^3)$ flops. This makes the overall computational cost of the algorithm $O(N + m^3)$.

Note that while the obtained models $\hat{\mathcal{A}}$ converge to an interpretable OOM with increasing sample size, it is not the case that a model obtained from a finite training sample is interpretable w.r.t. the characteristic events chosen for learning.

The statistical efficiency (model variance) of this basic algorithm depends crucially on the choice of characteristic and indicative events. This can be seen immediately from the basic learning equation (29). Depending on the choice of these events, the matrix $\hat{V}$ will have a high or low condition number, that is, its inversion will magnify estimation errors of $\hat{V}$ to a high or low extend, which in turn means a high or low model variance. Several methods of determining characteristic and indicative events that lead to a low condition number of $\hat{V}$ have been devised. The first of these methods is documented in [34]; another will be presented later in this chapter (it is documented in Appendix H).

We assumed here that the *correct* model dimension m is known beforehand. Finding the correct model dimension is however an academic question. Real-life processes will hardly ever have a finite dimension. The problem in practical applications is instead to find a model dimension that gives a good compromise in the bias-variance dilemma. The model dimension m should be chosen (i) large enough to enable the model to capture all the properties of the distribution that are statistically revealed in S, and in the meantime (ii) small enough to prevent overfitting.

Negotiating this compromise can be affected by the standard techniques of machine learning, for instance cross-validation. But OOM theory suggest a purely algebraic approach to this problem. The key is the matrix $\hat{V}$. Roughly speaking, if it has a low condition number and can thus be stably inverted, model variance will be low and overfitting is avoided. Quantitative bounds on model variance, as well as an algebraic method for finding good characteristic events (of a more general kind than introduced here) that minimize the condition number of $\hat{V}$ for a given model dimension can be found in [34].

While the basic learning algorithm is conceptually transparent and computationally cheap, it has two drawbacks that make it ill-suited for applications:

1. Even with good characteristic and indicative events for a small condition number of $\hat{V}$, the statistical efficiency of the basic algorithm has turned out to be inferior to HMMs estimated via the EM algorithm. The reason is that the EM algorithm implicitly exploits the statistics of arbitrarily long substrings in S, whereas our OOM learning algorithm solely exploits the statistics of substrings of length $|A_j B_i|$.

2. The models returned by this learning method need not be valid OOMs. The non-negativity condition *3* of Definition 2 is often violated by the "OOMs" computed via this method.

In Sections 10 to 13 of this chapter, the first of these problems will be completely solved. The second problem will remain unsolved, but practical working solutions will be presented.

9 History, Related Work, and Ramifications

Hidden Markov models (HMMs) [2] of stochastic processes have been investigated in mathematics under the name of "functions of Markov chains" long before they became a popular tool in speech processing and engineering. A basic mathematical question was to decide when two HMMs are equivalent, i.e. describe the same distribution [20]. This problem was tackled by framing HMMs within a more general class of stochastic processes, nowadays termed *linearly dependent processes* (LDPs). Deciding the equivalence of HMMs amounts to characterise HMM-describable processes as LDPs. This strand of research [4; 7; 8; 9; 21; 15; 16; 17] came to a successful conclusion in [24], where equivalence of HMMs was characterized algebraically, and a decision algorithm was provided. That article also gives an overview of the work done in this area up to writing time.

The results from [24] were further elaborated in [1], where for the first time matrix representations with negative entries appeared, called "generalized hidden Markov models". The algebraic characterization of HMM equivalence could be expressed more concisely than in the original paper [24].

All of this work on HMMs and LDPs was mathematically oriented and did not bear on the practical question of learning models from data.

In 1997, the concept of OOMs was introduced in [25], including the basic learning algorithm [26]. Independently a theory almost identical to the OOM theory presented here was developed in [40]. The only difference is that in that work characteristic *sequences* were utilized for learning instead of characteristic events, which renders the algorithm a bit more complicated.

Unconnected to all of these developments, the idea of describing the observables of a stochastic process as update operators was carried out in [22] within a very general mathematical framework. However, it was not perceived that these operators can be assumed to be linear.

Recently we have witnessed a growing interest in observable operator models in the field of optimal decision making / action selection for autonomous agents. Under the name of *predictive state representations* (PSRs) and with explicit connections made to OOMs, a generalization of partially observable Markov decision processes (POMDPs, e.g. [33]) is being explored (e.g. [35; 31], try Google on "predictive state representation" to find more). PSRs can be seen as a version of OOMs that models systems with input. Such input-output OOMs (including a variant of the basic learning algorithm) were first described in [27].

Since their discovery, OOMs have been investigated in the group of the first author. The most notable results are (i) matrix OOMs for continuous-valued processes, including a version of the basic learning algorithm [28], (ii) a general

OOM theory for stochastic processes (non-stationary, continuous-time, with arbitrary observation sets) including an algebraic characterization of general processes which reveals fascinating structural similarities between the formalism of quantum mechanics and OOMs, (iii) a first solution to the problem of finding characteristic events that optimize statistical efficiency, including bounds on model variance [34], and (iv) the introduction of suffix tree representations for the training string as a tool to improve statistical efficiency [37] (more about this later). Much effort was spent and wasted on the non-negativity problem; for the time being we put this at rest. Hopefully, new developments in linear algebra will ultimately help to resolve this issue [12].

Ongoing work in our group focusses on online learning algorithms, heuristics for ascertaining non-negativity of model-predicted probabilities (more in later sections), and the investigation of *quadratic* OOMs which arise from replacing the basic equation (8) by $P(\bar{a}) = (\sigma\tau_{\bar{a}}w_0)^2$. Non-negativity is clearly a non-issue in quadratic OOMs, which is the prime motivation for considering them, and the basic learning algorithm is easily carried over; however, it is currently not clear which processes can be characterized by quadratic OOMs. Finally, in a PhD project by Alexander Schönhuth at the University of Cologne, learning algorithms for non-stationary processes are being developed.

10 Overview of the ES Algorithm

We have seen that the basic OOM learning algorithm has limited statistical efficiency

1. because only the statistics of substrings of some (small) fixed length are entered in the estimation algorithm, thus much information contained in the training data is ignored, and

2. because it is unclear how to choose the characteristic/indicative events optimally, thus the information that enters the algorithm becomes further degraded by agglomerating it into possibly badly adapted collective events.

Both obstacles can be overcome:

1. Using a suffix tree representation of the training sequence, one can exploit characteristic/indicative sequences of all possible lengthes simultaneously. Instead of exploiting a mere m argument-value pairs, the number of used argument-value pairs is in the order of the training data size.

2. We can get rid of characteristic and indicative events altogether. They will only be used for the estimation of an initial model $\hat{\mathcal{A}}^{(0)}$, from which a sequence $\hat{\mathcal{A}}^{(1)}, \hat{\mathcal{A}}^{(2)}, \ldots$ of better models is iteratively obtained without using such events at all. The model improvement is driven by a novel learning principle whose main idea is to use the model $\hat{\mathcal{A}}^{(n)}$ for improving the statistical efficiency of the estimation procedure yielding $\hat{\mathcal{A}}^{(n+1)}$. We call this the principle of *efficiency sharpening* (ES).

11 The ES Principle: Main Idea and a Poor Man's ES Learning Algorithm

This is the main section of this chapter. We derive the underlying ideas behind the ES principle, present an elementary instance of an ES-based learning algorithm and finish with a little simulation study.

The core of the ES principle is to use in each iteration a new set of characteristic events that yields an estimator with a better statistical efficiency. However, a very much generalized version of such events is used:

Definition 7 *Let $\mathcal{A} = (\mathbb{R}^n, (\tau_a)_{a\in O}, w_0)$ be a (not necessarily minimal-dimensional) OOM of an m-dimensional process (X_n). Let $k \in \mathbb{N}$. A function $c : O^k \to \{r \in \mathbb{R}^n \,|\, \mathbf{1}r = 1\}$ is a* characterizer *of $\mathcal{A}$ (of length k) if*

$$\forall \bar{a} \in O^* : \quad w_{\bar{a}} = \sum_{\bar{b}\in O^k} P(\bar{b}\,|\,\bar{a})\; c(\bar{b}), \tag{31}$$

If convenient, we will identify c with the matrix $C = [c(\bar{b}_1)\cdots c(\bar{b}_\kappa)]$, where $\bar{b}_1, \ldots, \bar{b}_\kappa$ is the alphabetical enumeration of O^k.

It is clear that C is a characterizer for $\mathcal{A}$ if and only if every state $w_{\bar{a}}$ of $\mathcal{A}$ can be written as

$$w_{\bar{a}} = C\,(P(\bar{b}_1|\bar{a})\cdots P(\bar{b}_\kappa|\bar{a}))^\top, \tag{32}$$

where $\bar{b}_1, \ldots, \bar{b}_\kappa$ is the alphabetical enumeration of O^k. The characteristic events introduced in Section 7 can be regarded as a special characterizer: if $\mathcal{A}$ is interpretable w.r.t. characteristic events $B_1, \ldots, B_n$ of length k, and if $\bar{b} \in B_i$ then define $c(\bar{b})$ as the vector of dimension n that is zero everywhere except at position i. The two conditions from the above definition are easily checked. In matrix form, this gives the *characteristic event characterizer* (apologies for the loopy terminology)

$$\begin{aligned} C_{B_1,\ldots,B_m} \quad &= \quad (c_{ij})_{\substack{i=1,\ldots,m \\ j=1,\ldots,\kappa}} \\ &\text{where} \\ c_{ij} \quad &= \quad \begin{cases} 1, & \text{if} \quad \bar{b}_j \in B_i \\ 0, & \text{else.} \end{cases} \end{aligned} \tag{33}$$

We proceed by investigating other characterizers.

Proposition 10 *Let κ and $\bar{b}_1, \ldots, \bar{b}_\kappa$ be as in Definition 7. Given an m-dimensional process (X_n), then an $n \times \kappa$ matrix C whose columns sum to 1 is a characterizer of some n-dimensional OOM for (X_n) if and only if there exist m sequences $\bar{a}_j$ such that the $n \times m$ product matrix $W = CV$ of C and the $\kappa \times m$ matrix $V = (P(\bar{b}_i\,|\,\bar{a}_j))$ has rank m.*

The proof is in Appendix D. Now consider two equivalent, minimal-dimensional OOMs $\mathcal{A}$, $\mathcal{A}'$ which are related by $\tau'_a = \varrho\tau_a\varrho^{-1}$ (cf. Eq. (25)). Then it holds that

Proposition 11 *if C is a characterizer of $\mathcal{A}$, then $\varrho \circ C$ is a characterizer of $\mathcal{A}'$,*

because the states $w'_{\bar{a}}$ of $\mathcal{A}'$ are equal to the transforms $\varrho w_{\bar{a}}$ of the respective states of $\mathcal{A}$. A given minimal-dimensional OOM (of dimension m) has many finite characterizers of length k if $\kappa > m$; if $\kappa = m$ then the characterizer is unique. This is detailed out in the following corollary to Proposition 10 (proof in Appendix E):

Proposition 12 *Let C_0 be a characterizer of length k of a minimal-dimensional OOM $\mathcal{A}$. Let κ and V be as in Proposition 10. Then C is another characterizer of length k of $\mathcal{A}$ if and only if it can be written as $C = C_0 + G$, where*

$$G = [g_1, \cdots, g_{m-1}, -\Sigma_{i=1,\ldots,m-1} g_i]^\top, \tag{34}$$

where the g_i are any vectors from $\ker V^\top$.

An important type of characterizers is obtained from the states of reverse OOMs, that is, OOMs for the time-reversed process. We now describe in more detail the time reversal of OOMs. Given an OOM $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ with an induced probability distribution $P_{\mathcal{A}}$, its *reverse* OOM $\mathcal{A}^r$ is characterized by a probability distribution $P_{\mathcal{A}^r}$ satisfying

$$\forall\, a_0 \cdots a_n \in O^* : \quad P_{\mathcal{A}}(a_0 \cdots a_n) = P_{\mathcal{A}^r}(a_n \cdots a_0). \tag{35}$$

The reverse OOM can be computed from the forward OOM observing the following fact, whose proof is in Appendix F:

Proposition 13 *If $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ is an OOM for a stationary process, and w_0 has no zero entry, then $\mathcal{A}^r = (\mathbb{R}^m, (D\tau_a^\top D^{-1})_{a\in O}, w_0)$ is a reverse OOM to $\mathcal{A}$, where $D = diag(w_0)$ is a diagonal matrix with w_0 on its diagonal.*

Because from an m-dimensional matrix OOM for the "forward" process an m-dimensional matrix OOM for the reverse process can be constructed and vice versa, it follows that the process dimension of the forward process equals the process dimension of the reverse process.

When discussing "forward" and reverse OOMs of a process at the same time, using shorthand notations of the kind $P(\bar{b}_i \mid \bar{a}_j)$ easily leads to confusion. We fix the following conventions:

1. The character "b" and string shorthands $\bar{b}$ always denote symbols/substrings that follow symbols/substrings denoted by character "a" and string shorthands $\bar{a}$ — *"after" with respect to the forward time direction.*

2. We use P to denote probabilities for the forward process and P^r for the reverse process.

3. When using indices i, j for alphabetical enumerations for words $\bar{b}_i, \bar{a}_j$, the enumeration is carried out in the forward direction, even if we denote reverse probabilities. For example, if $O = \{0, 1, 2\}$, and if $\bar{a}_j, \bar{b}_j$ are each the alphabetical enumerations of O^2, and if τ_a, τ_a^r are the observable operators for a forward and a reverse OOM of a process, then $\bar{a}_6 = 12$, $\bar{b}_2 = 01$ and $\mathbf{1}\tau_1\tau_0\tau_2\tau_1 w_0 / \mathbf{1}\tau_2\tau_1 w_0 = P(\bar{b}_2 \mid \bar{a}_6) = P(X_2 = 0, X_3 = 1 \mid X_0 = 1, X_1 = 2) = P^r(X_2 = 0, X_3 = 1 \,|\, X_0 = 1, X_1 = 2) = P^r(\bar{b}_2 \,|\, \bar{a}_6) = \mathbf{1}\tau_1^r\tau_2^r\tau_0^r\tau_1^r w_0^r / \mathbf{1}\tau_1^r\tau_2^r w_0^r$.

4. Likewise, when using $\bar{a}$ as an index to denote a concatenation of operators, the forward direction is always implied for interpreting $\bar{a}$. For example, $\tau_{01} = \tau_1\tau_0$ and $\tau_{01}^r = \tau_0^r\tau_1^r$.

The states of a reverse OOM obtained after sufficiently long reverse words make a characterizer of a forward OOM for the process:

Proposition 14 *Let the dimension of (X_n) be m and let $\mathcal{A}^r = (\mathbb{R}^m, (\tau_a^r)_{a\in O}, w_0)$ be a reverse OOM for (X_n) that was derived from a forward OOM $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in O}, w_0)$ as in Proposition 13. Let k_0 be the characterizing length of (X_n), let $k \geq k_0$, and let $\kappa = |O^k|$. Then the following two statements hold:*

1. *$C = [w_{\bar{b}_1}^r \cdots w_{\bar{b}_\kappa}^r]$ is a characterizer of an OOM $\mathcal{A}'$ for (X_n).*

2. *The states $w'_{\bar{a}}$ of $\mathcal{A}'$ are related to the states $w_{\bar{a}}$ of $\mathcal{A}$ by the transformation $w'_{\bar{a}} = \varrho w_{\bar{a}}$, where $\varrho = C\pi_{\mathcal{A}}$. If in addition $w_0 = (1/m \cdots 1/m)^\top$, then furthermore $\varrho = R^\top R$. The matrices $\pi_{\mathcal{A}}$ and R are*

$$\pi_{\mathcal{A}} = \begin{pmatrix} \mathbf{1}\tau_{\bar{b}_1} \\ \vdots \\ \mathbf{1}\tau_{\bar{b}_\kappa} \end{pmatrix}, \quad R = \pi_{\mathcal{A}}\ diag\ ((m\,P(\bar{b}_1))^{-1/2} \cdots (m\,P(\bar{b}_\kappa))^{-1/2}). \tag{36}$$

The proof can be found in Appendix G. The proposition implies that $\varrho^{-1}C = (C\pi_{\mathcal{A}})^{-1}C =: C_{\mathcal{A}}^r$ is a characterizer for the original forward OOM $\mathcal{A}$. $C_{\mathcal{A}}^r = [\varrho^{-1}w_{\bar{b}_1}^r \cdots \varrho^{-1}w_{\bar{b}_\kappa}^r]$ is the characterizer obtained from the reverse OOM $\varrho^{-1}\mathcal{A}^r = (\mathbb{R}^m, (\varrho^{-1}\tau_a^r\varrho)_{a\in O}, w_0)$, so we may note for later use that every OOM $\mathcal{A}$ has a *reverse characterizer* $C_{\mathcal{A}}^r$ that is made from the states of a suitable reverse OOM.

Among all characterizers of OOMs $\mathcal{A}$ for (X_n), the reverse characterizers minimize a certain measure of variance, an observation which is the key to the ES learning principle. We enter the presentation of this core finding by describing some variants of the basic learning algorithm from Section 8.

In the basic learning algorithm from Eqn. 29, an estimate $\hat{\tau}_a$ of an m-dimensional OOM was determined from m estimated argument-value pairs for τ_a, which were sorted in the columns of an $m \times m$ matrix $\hat{V} = (\hat{P}(\bar{a}_j B_i))$ (containing the argument vectors) and another $m \times m$ matrix $\hat{W}_a = (\hat{P}(\bar{a}_j a B_i))$ (containing the values), by $\hat{\tau}_a = \hat{W}_a\hat{V}^{-1}$. It is clear that this is equivalent to

$$\hat{\tau}_a = \Big(\hat{P}(aB_i|\bar{a}_j)\Big)_{i,j=1,\ldots,m} \Big(\hat{P}(B_i|\bar{a}_j)\Big)^{-1}_{i,j=1,\ldots,m}. \tag{37}$$

The choice of m indicative sequences $\bar{a}_j$ is arbitrary and has the additional drawback that in estimating the argument-value matrices from a training string, only a fraction of the data enters the model estimation - namely, only the counting statistics of substrings beginning with one of the m chosen indicative sequences. The information contained in the data is better exploited if we use *all* indicative sequences $\bar{a}_1, \ldots, \bar{a}_\kappa \in O^k$, which yields two $m \times \kappa$ matrices containing the argument and the value vectors, requires the use of the pseudoinverse $\dagger$ instead of the matrix inverse, and turns (37) into

$$\hat{\tau}_a = \Big(\hat{P}(aB_i|\bar{a}_j)\Big)_{\substack{i=1,\ldots,m \\ j=1,\ldots,\kappa}} \Big(\hat{P}(aB_i|\bar{a}_j)\Big)^{\dagger}_{\substack{i=1,\ldots,m \\ j=1,\ldots,\kappa}}. \tag{38}$$

Let $\underline{V} = \big(P(\bar{b}_i|\bar{a}_j)\big)_{i,j=1,\ldots,\kappa}$ be the matrix of all conditional probabilities of length k sequences $\bar{b}$ given length k sequences $\bar{a}$, where i, j index the alphabetical enumeration of O^k (we will always use underlined symbols like $\underline{V}$ to denote "big" matrices of size $\kappa \times \kappa$), and let $\hat{\underline{V}}$ be the estimate of $\underline{V}$ obtained from the training string through the obvious counting procedure. Likewise, let $\underline{W}_a = \big(P(a\bar{b}_i|\bar{a}_j)\big)_{i,j=1,\ldots,\kappa}$ and $\hat{\underline{W}}_a$ its estimate. Then (38) is easily seen to be equivalent to

$$\hat{\tau}_a = C_{B_1,\ldots,B_m} \hat{\underline{W}}_a \, (C_{B_1,\ldots,B_m} \hat{\underline{V}})^{\dagger}. \tag{39}$$

Instead of the characteristic event characterizer $C_{B_1,\ldots,B_m}$ one may use any characterizer C, which gives us the following learning equation:

$$\hat{\tau}_a = C \hat{\underline{W}}_a \, (C \hat{\underline{V}})^{\dagger} \quad \text{for any characterizer } C. \tag{40}$$

It follows from Prop. 12 that all characterizers $C + G$, where G is any $m \times \kappa$ matrix with zero column sums and $G\underline{V} = \mathbf{0}$ yield the same state vectors as C, which entails

$$\tau_a = C \underline{W}_a \, C\underline{V}^{\dagger} = (C + G)\underline{W}_a \, ((C + G)\underline{V})^{\dagger} \tag{41}$$

for any such G. Finally, we observe that if ϱ is an OOM transformation as in Prop. 6, and if C is a characterizer for some OOM $\mathcal{A}$ and ϱC a characterizer for $\mathcal{A}'$ (cf. Prop. 11), then it is irrelevant whether we use C or ϱC in the learning equation 40, because the estimated OOMs will be equivalent via ϱ:

$$\begin{array}{ll} \text{If} & \hat{\tau}_a = C \hat{\underline{W}}_a \, (C \hat{\underline{V}})^{\dagger} \\ \text{and} & \hat{\tau}'_a = \varrho C \hat{\underline{W}}_a \, (\varrho C \hat{\underline{V}})^{\dagger} \\ \text{then} & \varrho \hat{\tau}_a \varrho^{-1} = \hat{\tau}'_a. \end{array} \tag{42}$$

After Prop. 14 we remarked that every OOM $\mathcal{A}$ has a reverse characterizer $C^r_{\mathcal{A}}$, and Prop. 11 informs us how transforming OOMs via transformations ϱ is

reflected in transforming their characterizers with ϱ. Together with Prop. 12 and Eqn. (41) we can draw the following overall picture:

- Call two characterizers *equivalent* if they characterize the same OOM. Then the equivalence class of all characterizers of an OOM $\mathcal{A}$ can be written as $C^r_{\mathcal{A}} + G$, where G is any matrix as described above.
- We know empirically that different choices of characteristic events (and hence, different characterizers) yield models of different quality when used in the learning equation (40). In order to study such sensitivity of learning w.r.t. choice of characterizers, (42) informs us that we may restrict the search for "good" characterizers to a single equivalence class.
- Concretely, we should analyze the quality of model estimates when G is varied in
$$\hat{\tau}_a = (C^r + G)\underline{\hat{W}}_a \, ((C^r + G)\underline{\hat{V}})^\dagger \tag{43}$$
for some reverse characterizer C^r whose choice (and hence, choice of equivalence class) is irrelevant.

In order to explain the ES principle, we concentrate of the role of $(C^r + G)\underline{\hat{V}}$ in this learning equation. We can make the following two observations:

- The variance of models estimated via (43) is determined by the variance of $(C^r + G)\underline{\hat{V}}$ across different training sequences. We may ignore the role of variance in $(C^r + G)\underline{\hat{W}}_a$ because either the condition of $(C^r + G)\underline{\hat{V}}$ is significantly larger than one, in which case variance in this matrix becomes magnified through the pseudoinverse operation in (43) and the overall variance of (43) becomes dominated by the variance of $(C^r + G)\underline{\hat{V}}$. Or, the condition of this matrix is close to one, in which case the variance of both $((C^r + G)\underline{\hat{V}})^\dagger$ and $(C^r + G)\underline{\hat{W}}_a$ will be approximately the same due to the similar makeup of $\underline{\hat{V}}$ and $\underline{\hat{W}}_a$, and again we may focus on $(C^r + G)\underline{\hat{V}}$ alone. (For a detailed analysis of these issues see [34]).
- The j-th column in the correct matrix $(C^r + G)\underline{V}$ is the state $w_{\bar{a}_j}$ of an OOM characterized by $(C^r + G)$. This is also the expectation of the j-th column $\hat{\mathbf{v}}_j$ in estimates $(C^r + G)\underline{\hat{V}}$. This column $\hat{\mathbf{v}}_j$ can be computed from the training string S as follows:
  1. Initialize $\hat{\mathbf{v}}_j = \mathbf{0}$.
  2. Sweep an observation window of length $2k$ across S. Whenever the windowed substring begins with $\bar{a}_j$, showing $\bar{a}_j\bar{b}_i$, add the i-th column $(C^r + G)(:, i)$ of $(C^r + G)$ to $\hat{\mathbf{v}}_j$.
  3. When the sweep is finished, normalize $\hat{\mathbf{v}}_j$ to unit component sum.

  We can interpret each additive update of $\hat{\mathbf{v}}_j$ in 2. as adding a stochastic approximation $(C^r + G)(:, i)$ of $w_{\bar{a}_j}$ to $\hat{\mathbf{v}}_j$. The variance of $\hat{\mathbf{v}}_j$ will thus grow monotonically with the mean stochastic approximation error. Considering

the entire matrix $(C^r + G)\underline{\hat{V}}$ with all its columns, we see that its variance is monotonically tied to the expected stochastic approximation error

$$\xi_G = \sum_{i,j=1}^{\kappa} P(\bar{a}_i \bar{b}_j) \, \|w_{\bar{a}_i} - (C^r + G)(:, j)\|^2. \tag{44}$$

Looking for statistically efficent model estimations via (43) we thus must ask which choice of G makes ξ_D minimal. Here is the main result of this report:

Proposition 15

$$\arg\min_G \xi_G = \mathbf{0}, \tag{45}$$

that is, the reverse characterizer C^r itself minimizes, within its equivalence class, the variance of the argument matrix $(C^r + G)\underline{\hat{V}}$. The proof (by M. Zhao) is in Appendix H. We would like to point out again that it is irrelevant *which* reverse characterizer (and hence, which equivalence class of characterizers) is used; all reverse characterizers yield equivalent models.

The normalizing step 3. is in fact redundant. Just as in the original learning method (cf. Eqn. 30) we may just as well use the "raw" counting matrices $\underline{V}^{\text{raw}} = (\#\bar{a}_j \bar{b}_i)$ and $\underline{W}_a^{\text{raw}} = (\#\bar{a}_j \, a \, \bar{b}_i)$ in place of the normalized matrices $\underline{\hat{V}}$ and $\underline{\hat{W}}_a$ in (43), saving one normalization operation.

This finding suggests an iterative learning procedure, with the goal of developing a sequence of characterizers that approaches a reverse characterizer, as follows:

1. **Learning task.** Given: a training sequence S of length N over an observation alphabet O of size α, and a desired OOM model dimension m.

2. **Setup.** Choose a characterizing length k (we found that the smallest k satisfying $\kappa = \alpha^k \geq m$ often works best). Construct the $\kappa \times \kappa$ counting matrices $\underline{V}^{\text{raw}} = (\#\bar{a}_j \bar{b}_i)$ and $\underline{W}_a^{\text{raw}} = (\#\bar{a}_j \, a \, \bar{b}_i)$.

3. **Initial model estimation**. To get started, use the basic learning algorithm from Section 8 once. Choose characteristic events $B_1, \ldots, B_m$ and code them in the characteristic event characterizer $C_{B_1,\ldots,B_m}$ (Eqn. (33)). The characteristic events should be chosen such that $C_{B_1,\ldots,B_m}\underline{V}^{\text{raw}}$ has a good condition number. A greedy heuristic algorithm for this purpose, which works very well, is detailed out in Appendix H. Compute an initial model $\hat{\mathcal{A}}^{(0)}$ through $\hat{\tau}_a^{(0)} = C_{B_1,\ldots,B_m}\underline{W}_a^{\text{raw}}(C_{B_1,\ldots,B_m}\underline{V}^{\text{raw}})^\dagger$. The starting state $\hat{w}_0^{(0)}$ can either be computed as the eigenvector to the eigenvalue 1 of the matrix $\hat{\mu}^{(0)} = \sum_{a \in O} \hat{\tau}_a^{(0)}$, or equivalently as the vector of rowsums of $C_{B_1,\ldots,B_m}\underline{V}^{\text{raw}}$, normalized to unit component sum.

4. **ES iteration**. Assume that $\hat{\mathcal{A}}^{(n)}$ is given. Compute its reverse $\hat{\mathcal{A}}^{r\,(n)}$ and the reverse characterizer $\hat{C}^{(n+1)} = \left(\hat{w}_{\bar{b}_1}^{r\,(n)} \cdots \hat{w}_{\bar{b}_\kappa}^{r\,(n)}\right)$. Compute a new model $\hat{\mathcal{A}}^{(n+1)}$ through $\hat{\tau}_a^{(n+1)} = \hat{C}^{(n+1)}\underline{W}_a^{\text{raw}}(\hat{C}^{(n+1)}\underline{V}^{\text{raw}})^\dagger$. The starting

state $\hat{w}_0^{r\,(n+1)}$ can again be computed as the normalized rowsum vector of $\hat{C}^{(n+1)}\underline{V}^{\mathrm{raw}}$ or from $\hat{\mu}^{(n+1)}$.

5. **Termination**. A standard termination criterium would be to calculate the log-likelihood of each model $\hat{\mathcal{A}}^{(n)}$ on S and stop when this appears to settle on a plateau, which is typically the case after 2 to 5 iterations.

The rationale behind the iteration step is that if some model $\hat{\mathcal{A}}^{(n+1)}$ comes closer to the true model than the previous one, then the resulting estimated reverse characterizer $\hat{C}^{(n+1)}$ will come closer to a version of the true reverse characterizer, thereby yielding an estimator with lower variance, which in turn *on average* will yield an even better model, etc. This idea motivated calling the entire approach "efficiency sharpening" (ES). We like to call this particular algorithmic instantiation of the ES principle the "poor man's" ES algorithm because it is simple, cheap, and suboptimal – the latter because it exploits only the statistics of substrings of length $2k$. We will soon see how one can do better in this respect. Here are two optional embellishments of the poor man's algorithm:

- In each iteration, the model $\hat{\mathcal{A}}^{(n)}$ can be transformed into an equivalent one that is interpretable w.r.t. the characteristic events used for the initial model estimation, before it is used in the iteration. This has, in principle, no effect on the procedure: a sequence of models each equivalent to the corresponding member in the original sequence of models will be obtained. The benefit of having interpretable models is cosmetical and diagnostic: one can produce state plots for each model which are visually comparable.

- The computational cost per iteration is dominated by computing the pseudo-inverse of $\hat{C}^{(n+1)}\hat{V}^{(0)}$. If this matrix is not too ill-conditioned (rule of thumb: with a condition number below 1e10 one is on the safe side when using double precision arithmetics), one may employ the well-known [e.g., 14], computationally much cheaper Wiener-Hopf equation to compute the desired least-square solution $\hat{\tau}_a^{(n+1)}$ to $(\hat{C}^{(n+1)}\underline{V}^{\mathrm{raw}})^\top X^\top = (\hat{C}^{(n+1)}\underline{W}_a^{\mathrm{raw}})^\top$.

A technical point not directly related to the ES principle: If one uses $\underline{W}_a^{\mathrm{raw}}, \underline{V}^{\mathrm{raw}}$ as suggested here, the pseudoinverse (which minimizes MSE of the obtained argument-value mapping) leads to a solution that disproportionally emphasizes the influence of argument-value pairs that represent a relatively small "mass of evidence" in the sense that the corresponding argument-value pairs in $\underline{V}^{\mathrm{raw}}$ and $\underline{W}_a^{\mathrm{raw}}$ have a small mass. If the j-th column of these raw matrices is normalized through dividing by the square root of the total weight of the j-th column of $\underline{V}^{\mathrm{raw}}$ (instead of division by the raw total weight), one obtains $\hat{W}_a^{(0)}, \hat{V}^{(0)}$ that under the $\hat{\tau}_a = \hat{W}_a\hat{V}^\dagger$ operation behave as if the argument-value pair $(\hat{P}(\bar{b}_1|\bar{a}_j)\cdots\hat{P}(\bar{b}_\kappa|\bar{a}_j)), (\hat{P}(a\bar{b}_1|\bar{a}_j)\cdots\hat{P}(a\bar{b}_\kappa|\bar{a}_j))$ would occur in $\underline{\hat{W}}_a^{(0)}, \underline{\hat{V}}^{(0)}$ multiple times with a multiplicity proportional to $\hat{P}(\bar{a}_j)$. This more properly

reflects the "mass of evidence" represented in each argument value pair and should be preferred. We omitted this reweighting above for expository reasons. We conclude this section with a little demonstration of the poor man's algorithm at work. The training sequences were obtained from running a randomly created HMM with 4 states and 3 output symbols for 1000 steps; test sequences were 10,000 steps long. The random creation of Markov transition and emission probabilities was biased towards a few high probabilities and many low ones. The reason for doing so is that if the HMM probabilities were created from a uniform distribution, the resulting processes would typically be close to i.i.d. — only Markov transition and emission matrices with relatively many low and a few high probabilities have enough structure to give "interesting" processes. Hundred train/test sequence pairs from different HMM generators were used to train and test 100 OOMs of dimension 3 with the poor man's algorithm, employing two versions where the raw counting matrices were normalized through division with the column sums (variant A, corresponding to Eqn. (43)) and through division with the square root of the column sums (variant B).

For comparison, HMMs with 3 states were trained with the Baum-Welch algorithm. For HMM training we used a public domain implementation of Baum-Welch written by K. P. Murphy (http://www.cs.ubc.ca/∼murphyk/Software/HMM/hmm.html). The Baum-Welch algorithm was run for at most 100 iterations and stopped earlier when the ratio of two successive training log-likelihoods dropped below 5e-5. Only a single Baum-Welch run was executed per data set with the HMM initialization offered by Murphy's software package. On average Baum-Welch used 40.8 iterations.

Our findings are collected in Figure 3. Here is a summary of observations of interest:

- On average, we see a rapid development of training and testing log-likelihoods to a plateau, with the first iteration contributing the bulk of model improvement. A closer inspection of the individual learning runs (not shown here) however reveals a large variability.

- Interesting things happen to the condition number of the argument matrices $C\underline{\hat{V}}$ (or their square-root normalized correlates in version B). The first iteration on average leads to a significant decrease of it, steering the learning process into a region where the matrix inversion magnifies estimation error to a lesser degree and thus improves statistical efficiency by an additional mechanism different from the ES mechanism proper. We must concede that all phenomena around this important condition number are not well understood.

- The initial model estimates with the basic learning algorithm are, on average, already quite satisfactory (for variant B, they match the final Baum-Welch outcome). This is due to the heuristic algorithm for finding characteristic events, which not only in this suite of experiments worked to our complete satisfaction.

- Compared to the Baum-Welch trained HMMs, the training log-likelihood of OOMs is higher by about 1%, reflecting the greater expressiveness of OOMs and/or the fact that our learning algorithm cannot be trapped in the local optima. In contrast, the OOM test log-likelihood is significantly lower. This reflects the fact that for this particular kind of data, HMMs possess a built-in bias which prevents these models from overfitting.
- Variant B leads to better training log-likelihoods than variant A. Especially the initial models are superior.
- Even the averaged curves exhibit a non-monotonic development of likelihoods. Inspection of the individual runs would reveal that sometimes the likelihood development is quite bumpy in the first three steps. This point is worth some extra consideration. The ES principle does not root in a concept of iteratively minimizing training error, as Baum-Welch does (and most other machine learning algorithms do). In fact, the ES principle comes with no guaranteed mechanism of convergence whatsoever. The ES algorithm only "tries" to find an estimator of better statistical efficiency in each iteration, but there is no guarantee that on a given dataset, an estimator of higher efficiency will produce a model with higher likelihood.
- The state fingerprints plotted in Figure 3 have been derived from models that were interpretable w.r.t. the characteristic events used in the initial model computation. The plots exhibit some states which fall outside the triangular region which marks the non-negative orthant of $\mathbb{R}^3$. Whenever we witness states outside this area in an interpretable OOM, we see an invalid OOM at work, that is, the non-negativity condition *3* from Definition 2 is violated. It is unfortunately the rule rather than the exception that trained OOMs are invalid. This is cumbersome in at least two respects. First, if one uses such pseudo-OOMs for computing probabilities of sequences, one may get negative values. Second, and even more critically in our view, invalid OOMs are inherently instable (not detailed out here), that is, if they are used for generating sequences, the states may explode. A fundamental solution to this problem is not in sight (cf. the concluding remarks in Section 6). We can offer only a heuristic stabilizing mechanism that affords non-exploding states and non-negative probabilities when an invalid OOM is run, at the expense of slightly "blurred" probabilities computed by such stabilized OOMs. This mechanism is described in Appendix J. We used it in this suite of learning experiments for determining the training and testing log-likelihoods of learned OMMs.

12 Essentials of Suffix Trees

We now proceed to describe an improved instantiation of the ES learning principle, using suffix trees to represent state sequences. In this section we recapitulate the basic concepts of suffix trees.

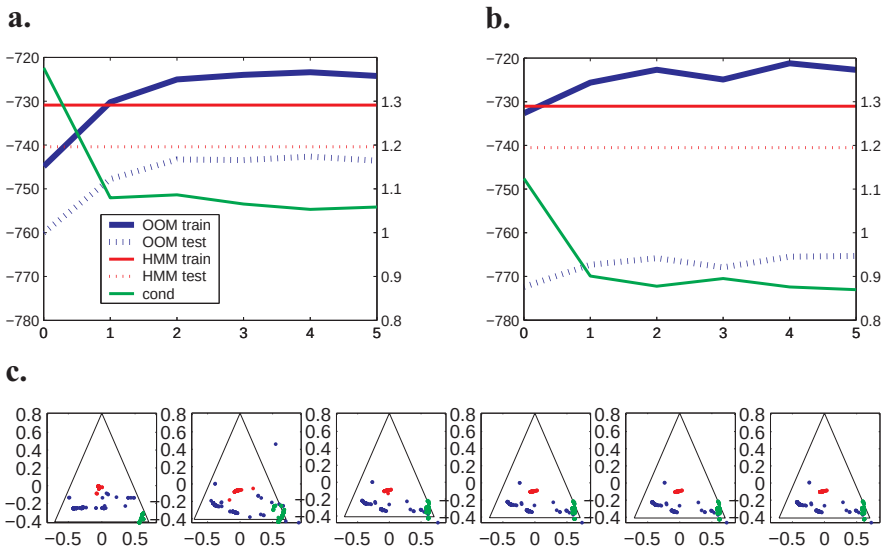

Figure 3: **a.** Training and testing log-likelihoods (scale on left y axis) of variant A trained OOMs plotted against the iteration number. The test log-likelihoods were divided by 10 because test strings were longer by this factor, to render them directly comparable with training log-likelihoods. The plot shows the average over 100 training experiments. For comparison, the final train/test log-likelihoods of Baum-Welch trained HMMs is shown by straight horizontal lines. Iteration 0 corresponds to the initial model estimation. In addition, the average of $\log_{10}$ of condition numbers of the matrices $C\underline{\hat{V}}$ are shown (scale on the right y axis). **b.** A similar plot for variant B. **c.** "Fingerprints" of the OOM models obtained in the successive iterations in one of the learning runs.

The *suffix tree* T [41] for a given string S provides a compact representation of all substrings of S while exposing the internal structure of S in an efficient data structure. Moreover, a (compact) suffix tree T can be constructed in linear time $\mathcal{O}(|\,S\,|)$. While the suffix tree is a simple enough data structure, linear-time construction algorithms are quite involved [19].

Formally, a suffix tree is a *trie* with additional properties. A *trie* T [18] — the name being derived from re*trie*val - is an ordered tree where edges are labelled with strings over an alphabet O such that no two edges from some node k of T to its children have labels beginning with the same symbol $a \in O$. Thus we may speak of the *path* to a node k of T, defined as the concatenation of all edge labels encountered when traveling from the root to k, and we may identify the node k with $\mathsf{path}(k) \in O^*$. The set of words $\mathsf{words}(T) \subset O^*$ represented by a tree is defined by

$$\bar{a} \in \mathsf{words}(T) \quad \Longleftrightarrow \quad \exists k \in T:\ \bar{a} = \mathsf{path}(k)\bar{b} \tag{46}$$

where $\bar{b}$ is a prefix of an outgoing edge of k. Then, a suffix tree T_S of a string S is a trie that contains all substrings of S:

$$\mathsf{words}(T_S) = \{\bar{a} \in O^* : \bar{a} \text{ is a substring of } S\}. \tag{47}$$

In general there will be more than one trie satisfying the above condition. However if we additionally require that all nodes in T_S are either leaves or have at least two children, the suffix tree of S is uniquely determined (up to ordering). It is called the *compact* suffix tree of S. Compact tries were introduced historically under the name *Patricia trees* [36]. Figure 4 illustrates the concept with a compact suffix tree for the string *cocoa*.

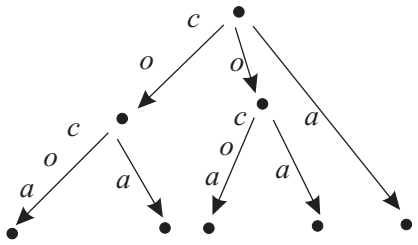

Figure 4: The compact suffix tree for *cocoa*. Note that a sentinel is not required because the symbol a appears in the string only at the terminal position and thus acts as a sentinel.

Sometimes it is desirable to have every suffix of S represented by a leaf. This can be enforced by appending a *sentinel* symbol $\$ \notin O$ at the end of S. Then the compact suffix tree for the extended string $S\$$ represents every suffix of $S\$$ and only the suffixes of $S\$$ as leaves. Finally, note that a compact suffix tree of a sequence of length N has at most $2N$ nodes.

13 A Suffix-Tree Based Version of ES

An obvious weakness of the poor man's ES algorithm is that the family of indicative sequences $(\bar{a}_i)_{1 \le \kappa} = O^k$ is not adapted to the training sequence S. Some of the $\bar{a}_i$ might not occur in S at all, entailing a useless computational overhead. Some others may occur only once or twice, yielding very poor estimates of probabilities through relative frequencies. On the other side of the spectrum, if some sequence $\bar{a}_i$ occurs very frequently, learning would clearly benefit from splitting it into α longer sequences by $\bar{a}_i \mapsto \{a\bar{a}_i | a \in O\}$. In this section we show how a compact suffix tree (ST) representation of S can be used to support a choice of indicative sequences which is matched to S. (We will henceforth drop the qualifier "compact", it is tacitly assumed throughout). The idea of representing variable-length "context" for prediction purposes in a suffix tree is known in the literature under various names, e.g. "prediction suffix trees" or "variable-order Markov chains" (concise overview in [5]).

Implementing suffix trees involves some tedious "indexery witchcraft". We therefore do not attempt a detailed documentation of our suffix-tree based EM implementation but instead outline the basic ideas, which are quite straightforward.

Let \$ be a sentinel symbol not in O, and let $\$S$ be S prepended by \$. We will still speak of $\$S$ as the training sequence. Suffix trees are brought into the learning game by observing that, first, the words of the ST $T_{(\$S)^r} = T_{S^r\$}$ of the *reverse* training sequence $(\$S)^r = S^r\$$ are the reverse substrings of $\$S$ (this is clear by definition of a ST). But second and more interestingly, it moreover holds that if

- k_1 is a node in $T_{S^r\$}$ and k_2 is a child node of node k_1,
- $\bar{c}_1 = \mathsf{path}(k_1)$ is the path to k_1 and $\bar{c}_1\bar{c}_2$ the path to k_2,
- $\bar{a}_1 = \bar{c}_1^r$, $\bar{a}_2 = \bar{c}_2^r$, $\bar{a}_2\bar{a}_1 = (\bar{c}_1\bar{c}_2)^r$ are the associated forward words,
- $\bar{a}_2 = \bar{a}_{21}\bar{a}_{22}$ is some split of $\bar{a}_2$,

then wherever $\bar{a}_{22}\bar{a}_1$ occurs in $\$S$, it occurs after $\bar{a}_{21}$.

This can be rephrased as follows. Let, for some word $\bar{a}$, $\mathsf{pos}(\bar{a})$ denote the set of all the position indices n in S such that the sequence from the beginning of S up to position n ends with $\bar{a}$. Furthermore, for some node k of $T_{S^r\$}$, let $\mathsf{pos}(k) = \mathsf{pos}((\mathsf{path}(k))^r)$ be the set of positions in the forward sequence $\$S$ associated with the reverse path of k. Then, if we reuse the notations from the above bullet list, and if $\bar{a}_2 = a_1 \cdots a_l$, then

$$\begin{aligned} \mathsf{pos}(a_1 a_2 \cdots a_l\, \bar{a}_1) &= \mathsf{pos}(a_2 a_3 \cdots a_l\, \bar{a}_1) \\ &\cdots \\ &= \mathsf{pos}(a_l\, \bar{a}_1) \\ &\subsetneq \mathsf{pos}(\bar{a}_1). \end{aligned} \tag{48}$$

Now think of reverse versions $\bar{a}$ of words $\bar{c}$ from $T_{S^r\$}$ as candidates for indicative sequences. If $\mathsf{pos}(\bar{a}) = \mathsf{pos}(\bar{a}')$, then clearly it makes no sense to collect continuation statistics of the type $\#\bar{a}\bar{b}$ both for $\bar{a}$ and $\bar{a}'$, because they are identical. Therefore, the nodes of $T_{S^r\$}$ correspond to potential indicative sequences that are distinguishable within S w.r.t. their continuations in S, and we may ignore all words $\bar{a}$ whose reverse does not end in a node of $T_{S^r\$}$. This is the basic idea of suffix-tree based ES: use as indicative sequences all the words whose reverse correspond to nodes in $T_{S^r\$}$.

We now turn to reverse characterizers. An analysis of the poor man's algorithm reveals that, given a reverse OOM with states $w_{\bar{b}}^r$, we constructed estimates of $w_{\bar{a}}$ through

$$\hat{w}_{\bar{a}} = \frac{\sum_{\bar{b}\in O^k} w_{\bar{b}}^r * (\text{number of occurences of } \bar{a}\bar{b} \text{ in } O)}{\text{number of occurences of } \bar{a} \text{ in } O}. \tag{49}$$

If we were to copy this idea identically for use with suffix-tree managed indicative sequences $\bar{a}$, we would have to collect statistics for continuations by all

$\bar{b} \in O^k$, for all our indicative sequences $\bar{a}$. Furthermore, in doing so we would of course have to fix k and thus ignore information provided in S by continuations longer than k. A stronger idea suggests itself here. Let $S = a_1 \dots a_N$. Instead of precalculating all $w^r_{\bar{b}}$ and collecting the necessary continuation statistics, we simply run the reverse OOM on the reverse training sequence once, obtaining a state sequence $w^r_\varepsilon = (w^r_0, w^r_{a_N}, w^r_{a_{N-1}a_N}, \dots, w^r_S)$. Reversing in time and renaming yields a state sequence that is more convenient for our purposes, by putting

$$(c_0, c_1, \dots, c_N) := (w^r_S, \dots, w^r_{a_{N-1}a_N}, w^r_{a_N}, w^r_0). \tag{50}$$

We interpret c_n as a stochastic approximation to $w_{\bar{a}_n}$, in the following sense. Consider the limit case of $N \to \infty$. Assume that for a right-infinite sequence $\bar{b}_\infty = b_1 b_2 \dots$ the limes $c_{\bar{b}_\infty} = \lim_{l\to\infty} w^r_{b_1 \cdots b_l}$ exists almost surely (we conjecture that for reverse ergodic processes this always holds). Let $P_{\bar{a}_n}$ be the conditional probability distribution over the set of right-infinite continuations $\bar{b}_\infty$ of $\bar{a}_n$. Then the family $(c_{\bar{b}_\infty})_{\bar{b}_\infty \in O^\infty}$ can be regarded as an (infinite) characterizer by setting

$$w_{\bar{a}} = \int c_{\bar{b}_\infty} dP_{\bar{a}} \tag{51}$$

for all $\bar{a} \in O^*$. Because S is finite, interpreting c_n as a stochastic approximation to $w_{\bar{a}_n}$ via (51) will incur some inaccuracy, which however will be negligible for all n that are not very close to the right end of S. All of this entitles us to change the poor man's strategy (49) to this rich woman's version:

$$\hat{w}_{\bar{a}} = \frac{1}{|\mathsf{pos}(\bar{a})|} \sum_{n \in \mathsf{pos}(\bar{a})} c_n. \tag{52}$$

Finally, observe that $T_{S^r\$}$ has $N+1$ leaf nodes, each corresponding uniquely to one position in $\$S$. That is, for a leaf node k, $\mathsf{pos}(k) = \{n\}$, where n is the position within $\$S$ where the reverse path of k ends, started from the beginning of $\$S$. For an internal node k with children $k_1, \dots, k_x$ it holds that $\mathsf{pos}(k) = \bigcup_{i=1,\dots,x} \mathsf{pos}(k_i)$.

Now everything is in place for describing the suffix-tree based ES algorithm.

- **Task.** Given: a training sequence S of length N and a model dimension m. Wanted: an m-dimensional OOM.
- **Initialization.**

  Learn initial model. Learn an m-dimensional OOM $\hat{\mathcal{A}}^{(0)}$ using the basic learning algorithm, as in the poor man's algorithm.

  Construct $T_{S^r\$}$.

  Procure argument value pair mapping. Let k_{all} be the leaf that corresponds to the entire sequence. Allocate a map $f : T_{S^r\$} - \{k_{\text{all}}\} \times O \to T_{S^r\$} \cup \{0\}$ and initialize it by all zero values. For each node k except k_{all}, where the reverse path of k is $\bar{a}$, and for each $a \in O$, determine the highest node k' such that $(\bar{a}a)^r$ is a prefix of the path of k' (then $\mathsf{pos}(k') = \mathsf{pos}(\bar{a}a)$). Set $f(k, a) = k'$.

- **ES Iteration.** Input: $\hat{\mathcal{A}}^{(n)}$, $T_{S^r\$}$. Output: $\hat{\mathcal{A}}^{(n+1)}$.

  Procure $\hat{w}_{\bar{a}}$'s. (i) Compute the reverse OOM $\hat{\mathcal{A}}^{r\,(n)}$. (ii) Run it on the reverse training sequence to obtain $(c_0, c_1, \ldots, c_N)$ (use the "blurred" but stabilizing method for running potentially invalid OOMs that is detailed out in Appendix J). (iii) Sort these $N+1$ states into the leaf nodes of $T_{S^r\$}$, where c_n goes to the leaf node with $\mathsf{pos}(k) = \{n\}$. Formally, for leaves k set $C(k) = c_{\mathsf{pos}(k)}$. (iv) From the leaves upward, for some internal node k for whose children k' $C(k')$ has already been determined, set $C(k) = \sum_{k' \text{is a child of} k} C(k')$. Do this until all nodes have been covered. Then for all nodes k it holds that

$$C(k) = |\mathsf{pos}(\bar{a})|\ \hat{w}_{\bar{a}} = \sum_{n \in \mathsf{pos}(\bar{a})} c_n. \tag{53}$$

  where $\bar{a}$ is the reverse path of the node.

  Procure argument-value matrices $\hat{V}$ and $\hat{W}_a$. To obtain matrices $\hat{V}$ and $\hat{W}_a$ (each of size $m \times |T_{S^r\$}| - 1$) that play a similar role as we are accustomed to, go through all nodes k of the tree (except k_{all}), write $C(k)$ into the k-th column of $\hat{V}$ and $C(f(k,a))$ into the k-th column of $\hat{W}_a$.

  Reweigh. In analogy to the reweighing scheme described for the poor man's algorithm, divide each column k in $\hat{V}$ and all $\hat{W}_a$ by the square root of the k-th column sum of $\hat{V}$.

  Compute new model. Set $\hat{\tau}_a^{(n+1)} = W_a \hat{V}^\dagger$ and $\hat{w}_0^{(n+1)}$ as the eigenvector to the eigenvalue 1 of $\hat{\mu} = \sum_{a \in O} \hat{\tau}_a^{(n+1)}$ (normalized of course to unit component sum), obtaining $\hat{\mathcal{A}}^{(n+1)}$.

- **Termination.** Stop after a predetermined number of iterations or when training log-likelihood seems to saturate.

- **Optional tuning.** One may augment this algorithm in several ways:

  Make models interpretable. Transform each $\hat{\mathcal{A}}^{(n+1)}$ to an interpretable OOM before further use, using the characteristic events employed in the initial model estimation. This gives comparable "fingerprint" plots that are helpful in monitoring the algorithm's progress. More importantly, we found that such renormalization sometimes renders the algorithm more robust when otherwise the condition of V deteriorated over iterations.

  Use subsets of tree. When constructing $\hat{V}$ and $\hat{W}_a$, we may restrict ourselves to using only ST nodes that represent some minimal number of positions, guided by the intuition that nodes representing only very few string positions contain too unreliable stochastic approximations of forward states.

A nasty trick to improve stability. The algorithm depends on a matrix inversion (more correctly, a pseudoinverse computation). We sometimes experienced that the matrix $\hat{V}$ becomes increasingly ill-conditioned as iterations progress, completely disrupting the learning process when the ill-conditioning explodes. A powerful but brutal and ill-understood remedy is to transform the reverse OOM $\hat{\mathcal{A}}^{r\,(n)}$ (before using it) into a surely valid OOM by making its operator matrices all-nonnegative. Concretely, set all negative entries in the operator matrices of $\hat{\mathcal{A}}^{r\,(n)}$ to zero and renormalize columns by dividing them through the corresponding column sum of their sum matrix. To our surprise, often the model quality suffered only little from this dramatic operation. Purely intuitively speaking, we might say that learning process is forced to find a solution that is close to a HMM (where forward and reverse matrices are non-negative).

All ST related computations involved in this procedure can be effected in time linear of the ST size, which is at most $2N$. The main cost factors in a suffix-tree ES iteration are the computation of the reverse state sequence, which is $O(Nm^2)$, and the computation of the pseudo-inverse. To speed up the computation in cases where $\hat{V}$ is not too ill-conditioned, one may use the Wiener-Hopf solution instead, as described for the poor man's version of ES. Then the cost of calculating the operators from $\hat{V}$ and the $\hat{W}_a$'s is $O(\alpha m^2 N)$, which dominates the cost for obtaining the reverse state sequence. In practice we found runtimes per iteration that are somewhat shorter than a Baum-Welch iteration in the same task. However, for the total time of the algorithm we must add the time for the initial model estimation and the computation of the suffix tree. The latter in our implementation takes about 1 to 2 times the time of an ES iteration.

14 A Case Study: Modeling 1 Million Pound

The most demanding task that we tried so far was to train models on Mark Twain's short story "The 1,000,000 Bank-Note" (e.g., www.eastoftheweb.com/short-stories/UBooks/MilPou.shtml). We preprocessed the text string by deleting all special characters except the blank, changing capital letters to small caps, and coding letters by integers. This gave us 27 symbols: 1 (codes "a"), ..., 26 ("z"), 27 ("_"). The resulting string was sorted sentence-wise into two substrings of roughly equal length (21042 and 20569 symbols, respectively) that were used as training and test sequences.

The suffix-tree based learning algorithm was used with the "brutal" stabilizing method mentioned above, which was necessary for larger model dimensions. Five ES iterations besides the zero-th step of estimating an initial model were run for model dimensions 5, 10, 15,...,50, 60,..., 100, 120, 150. More details can be found in Appendix K.

For comparison, HMMs of the same dimensions were trained with K. Mur-

phy's Matlab implementation of Baum-Welch. HMMs were initialized using the default initialization offered by this implementation. Iterations were stopped after 100 iterations or when the ratio of two subsequent training log-likelihoods was smaller than 1∼1e-5, whichever occurred first (almost always 100 iterations were reached). Only a single run per model dimension was carried out; no overhead search methods for finding good local optima were invoked. Thus it remains unclear whether with more search effort, the HMM performance could have been significantly improved. In the light of the fact that across the different model dimension trials the HMM performance develops quite smoothly, this appears unlikely: if significantly better HMMs would exist and could be found by random search, then we would expect that due to the random HMM initialization the performance curve would look much more rugged.

Computations were done on a notebook PC with a Pentium 330 MHz processor in Matlab.

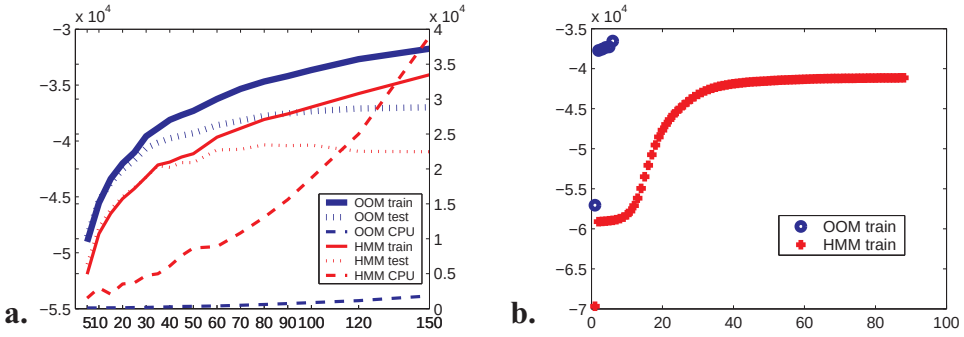

Figure 5: **a.** Training and test log-likelihoods (scale on left y axis) of ES trained OOMs and EM trained HMMs plotted against model dimension. In addition, the CPU time (in seconds, scale on right y axis) for both is shown. **b.** A closeup on the development of training log-likelihood for the learning of a 50-dimensional OOM and HMM. The EM algorithm here met its termination criterion in the 90th iteration. For details see text.

15 Discussion

The findings about the ES algorithm reported in this chapter are not older than 3 months at the time of writing and far from mature. We could not yet extensively survey the performance of the ES algorithm except for ad hoc tests with some synthetic data (discretized chaotic maps, outputs of FIR filters driven by white noise input, outputs of random recurrent neural networks, HMM generated sequences) and a few standard benchmark datasets (sunspot data, Melbourne meteorological data). In all cases, the behavior of ES was similar to the HMM and 1,000,000 Mio Pound datasets reported here: a rapid development of

models towards plateau training log-likelihoods (in 1 to 10 iterations, typically 3), followed by a jittery "hovering" at that plateau. Both training and testing likelihoods at the plateau level were superior to HMM performance (however, these were not optimized) except for HMM generated data, where HMM models can play out their natural bias for these data. Thus it is fair to say that learning algorithms based on suffix-tree ES clearly have the potential to yield more accurate models of stationary symbol processes, and in a much shorter runtime, than HMMs.

Although it might be tempting, it is misleading to see the ES algorithm as a variant of the EM algorithm. Both algorithms execute an iterative interplay between model re-estimation and state generation, but there are two fundamental differences between them:

- The estimator instantiated by each ES step, including the initial model estimator, is asymptotically correct. That is, if the process is in fact m-dimensional and m-dimensional models are trained, the modeling error would go to zero with increasing length of training data almost surely, right from the zero-th iteration. This is not the case with the EM algorithm.

- The training log-likelihood does not necessarily grow monotonically under ES, — but this behavior is constitutional for EM. The ES principle is not designed to monotonically improve any target quantity.

Contemplating the task of improving an asymptotically correct estimator, the natural target for improvement – actually the only one that we can easily think of – is the statistical efficiency of the estimator. This was the guiding idea that led us to the discovery of the learning methods described in this chapter, and it is borne out by the mathematical analysis presented in Section 11. However, we are not sure whether this is already the complete explanation of why and how our algorithms work. First, the role of the condition of the $\hat{V}$ matrix has to be clarified — if it is not well-conditioned, inverting $\hat{V}$ will magnify estimation errors contained in it and potentially invalidate the reasoning of Section 11. Interestingly, even when the condition of $\hat{V}$ deteriorates through ES iterations, we mostly find that the model quality increases nonetheless, up to a point where $\hat{V}$ becomes so ill-conditioned that numerical errors explode. We confess that this is hard to understand. Second, we have only provided an argument that the reverse characterizer obtained from the correct model yields a maximally efficient estimator. But it remains to be investigated in what sense the sequence of reverse characterizers constructed in ES runs moves toward this correct model; in other words, how the sequence of reverse characterizers is related to the gradient of ξ at points away from the minimum.

The main problem of current ES implementations is that learning runs sometimes become instable (condition of $\hat{V}$ explodes). This is related to the model dimension: while small dimensions never present difficulties in this respect, learning larger models becomes increasingly prone to instability problems. We currently perceive two sources for instability:

Data-inherent ill-conditioning. If the generating process has a lower dimension than the models one wants to learn, the matrix $\hat{V}$ *must* become ill-conditioned with increasing training data size. Now, real-life datasets will hardly come from any low-dimensional generator. But if we would investigate their limiting singular value spectrum (that is, the singuluar value spectrum of the matrices $V_k = (P(\bar{b}_i|\bar{a}_j))_{\bar{a},\bar{b}\in O^k}$ in the limes of $k \to \infty$) and find a rapidly thinning tail, then for all practical learning purposes such generators behave like low-dimensional generators, intrinsically leading to matrices V of low numerical rank.

Invalid OOMs. Running invalid reverse OOMs to create reverse characterizers is prone to sprinkle the obtained state sequence with outliers, which are hard to detect. Using such contaminated reverse characterizers to feed the input to linear regression task will even deteriorate the situation because the minimization of MSE will further magnify the impact of outliers — a familiar problem. This clearly happened in some of our more problematic (high model dimension) test runs.

HMM/EM learning does not rely on any matrix (pseudo-)inversion and does not suffer from the ensuing stability problems. Although we are fascinated by the promises of ES, for the time being we would prefer HMM/EM over OOM/ES in any safety-critical application. But we hope that the rich repertoire of numerical linear algebra will equip us with the right tools for resolving the stability issue. The rewards should be worth the effort.

Appendix A: Proof of Proposition 4(*4*)

We first show $\forall\ a \in O,\ w \in W\ \ \pi(\tau_a w) = t_a\ \pi(w)$. Let $w = \sum_{i=1,\ldots,d} \alpha_i w_{\bar{a}_i}$. Then

$$\begin{aligned}
\pi(\tau_a w) &= \pi\big(\sum_{i=1,\ldots,d} \alpha_i \tau_a w_{\bar{a}_i}\big) \\
&= \pi\big(\sum_{i=1,\ldots,d} \alpha_i\ \mathbf{1}\tau_a w_{\bar{a}_i} \frac{\tau_a w_{\bar{a}_i}}{\mathbf{1}\tau_a w_{\bar{a}_i}}\big) \\
&= \sum_{i=1,\ldots,d} \alpha_i\ P(a\,|\,\bar{a}_i) f_{\bar{a}_i a} \\
&= \sum_{i=1,\ldots,d} \alpha_i\ t_a\ f_{\bar{a}_i} = t_a \sum_{i=1,\ldots,d} \alpha_i f_{\bar{a}_i} \\
&= t_a\ \pi(w).
\end{aligned}$$

An iterated application of this finding yields the statement of the proposition.

Appendix B: Proof of Proposition 5

"⇒": Let $x \in \ker \pi$, $\bar{a} \in O^*$. Then $\mathbf{1}\tau_{\bar{a}}x = \sigma\pi(\tau_{\bar{a}}x) = \sigma t_{\bar{a}}\pi(x) = 0$ by Proposition 4 (*3.* and *4.)*.
"⇐":

$$\begin{aligned} &\forall \bar{a} \in O^* \;\; \mathbf{1}\tau_{\bar{a}}x = 0 \\ &\quad \rightarrow \quad \forall \bar{a} \in O^* \;\; \sigma t_{\bar{a}}\pi(x) = 0 \\ &\quad \rightarrow \quad \forall \bar{a} \in O^* \;\; (\pi(x))(\bar{a}) = 0 \\ &\quad \rightarrow \quad \pi(x) = \mathbf{0}. \end{aligned}$$

Appendix C: The Three OOMs from Figure 2

The plotted OOMs were obtained from HMMs over the alphabet $O = \{a, b, c\}$ as described in Section 3. The Markov transition matrix was

$$M = \begin{pmatrix} 1-2\alpha & \alpha & \alpha \\ \alpha & 0 & 1-\alpha \\ 0 & 1-\alpha & \alpha \end{pmatrix}, \tag{54}$$

where α was set to 0.1, 0.2 and 0.3 respectively for the three plotted OOMs. The symbol emission matrices O_a were

$$O_a = \mathrm{diag}(0.8 \;\; 0.1 \;\; 0.1),\; O_b = \mathrm{diag}(0.1 \;\; 0.8 \;\; 0.3),\; O_c = \mathrm{diag}(0.1 \;\; 0.1 \;\; 0.6) \tag{55}$$

for all HMMs. From these HMMs, OOMs were created that were interpretable w.r.t. the singleton characteristic events $A_1 = \{a\}, A_2 = \{b\}, A_3 = \{c\}$.

Appendix D: Proof of Proposition 10

To see the "if" direction, consider an n-dimensional OOM $\mathcal{A}$ for (X_n) whose states $w_{\bar{a}_j}$ $(j = 1, \ldots, m)$ are the columns of W and whose other states $w_{\bar{a}}$ are linear combinations of the $w_{\bar{a}_j}$ (whereby $\mathcal{A}$ is uniquely determined according to the insights from Section 5 — the column vectors of W span the "prediction-relevant" subspace V from Eq. (18)). It is a mechanical exercise to show that condition (31) holds. For the "only if" direction choose any m sequences $\bar{a}_j$ such that V has rank m. Then by the definition of a characterizer, W has the states $w_{\bar{a}_j}$ of the OOM characterized by c as its columns, which must be linearly independent because V has rank m.

Appendix E: Proof of Proposition 12

According to Proposition 10, every characterizer C of $\mathcal{A}$ must satisfy $V^\top C^\top = W^\top$. It is straightforward to derive that conversely, any C with unit column sums satisfying $V^\top C^\top = W^\top$ is a characterizer. Now any $m \times \kappa$ matrix C

satisfies $V^\top C^\top = W^\top$ if and only if $C = C_0 + D$, where the rows of D are in $\ker V^\top$. The additional requirement that the column sums of C must sum to unity is warranted by making the last row of D equal to the negative sum of the other rows.

Appendix F: Proof of Proposition 13

Recalling that D is a diagonal matrix with w_0 on its diagonal, the following transformations yield the claim:

$$\begin{aligned} P_{\mathcal{A}}(a_0 \cdots a_n) &= \mathbf{1}\tau_{a_n} \cdots \tau_{a_0} w_0 \\ &= w_0^\top \tau_{a_0}^\top \cdots \tau_{a_n}^\top \mathbf{1}^\top \\ &= w_0^\top D^{-1} D \tau_{a_0}^\top D^{-1} \cdots D \tau_{a_n}^\mathsf{T} D^{-1} D \mathbf{1}^\top \\ &= \mathbf{1} D \tau_{a_0}^\top D^{-1} \cdots D \tau_{a_n}^\top D^{-1} w_0 \\ &= P_{\mathcal{A}^r}(a_n \cdots a_0). \end{aligned}$$

Appendix G: Proof of Proposition 14

We first assume that C is a characterizer and show claim 2. According to Eq. (32) we have $w'_{\bar{a}} = C\,(P(\bar{b}_1|\bar{a}) \cdots P(\bar{b}_\kappa|\bar{a}))^\top$, which is equal to $C\,\pi_{\mathcal{A}}\,w_{\bar{a}}$ by Proposition 9 (1). To show that $\varrho = R^\top R$ (assuming now $w_0 = (1/m \cdots 1/m)^\top$), we consider for some $\bar{b} = b_1 \cdots b_k$ a column $c = \tau_{\bar{b}}^r w_0 \,/\, \mathbf{1}\tau_{\bar{b}}^r w_0 = \tau_{\bar{b}}^r w_0 / P(\bar{b})$ of C. Using the terminology from Proposition 13 and noting that $D = (1/m \cdots 1/m)^\top$, it can be re-written as follows:

$$\begin{aligned} P(\bar{b})c &= \tau_{b_1}^r \cdots \tau_{b_k}^r w_0 \\ &= D\tau_{b_1}^\top \cdots \tau_{b_k}^\top D^{-1}\, w_0 \\ &= 1/m \quad \tau_{b_1}^\top \cdots \tau_{b_k}^\top \mathbf{1}^\top \\ &= 1/m \quad \left(\mathbf{1}\tau_{b_k} \cdots \tau_{b_1}\right)^\top \;=\; 1/m \quad (\mathbf{1}\tau_{\bar{b}})^\top. \end{aligned}$$

Thus the i-th column of C equals the transpose of the i-th row of $\pi_{\mathcal{A}}$ up to a factor of $(mP(\bar{b}_i))^{-1}$. Splitting this factor into $(mP(\bar{b}_i))^{-1/2}\,(mP(\bar{b}_i))^{-1/2}$ and redistributing the splits over C and $\pi_{\mathcal{A}}$ yields the statement $\varrho = R^\top R$.

To show the first claim, first notice that C is a characterizer for $\mathcal{A}'$ if and only if $\tilde{\varrho}C$ is a characterizer for $\varrho\,\mathcal{A}'$ for some equivalence transformation $\tilde{\varrho}$ according to Propositions 6 and 11. Because a transformation $\tilde{\varrho}$ can always be found that maps w_0 on $(1/m \cdots 1/m)^\top$ (exercise), we may assume w.l.o.g. that $w_0 = (1/m \cdots 1/m)^\top$. Let

$$R = \left((mP(\bar{b}_1))^{-1/2}(\mathbf{1}\tau_{\bar{b}_1})^\top \;\cdots\; (mP(\bar{b}_\kappa))^{-1/2}(\mathbf{1}\tau_{\bar{b}_\kappa})^\top\right)^\top \tag{56}$$

as above. Define vectors $v_{\bar{a}} = C\pi_{\mathcal{A}} w_{\bar{a}} = R^\top\, R\, w_{\bar{a}}$. We want to show that the transformation $C\pi_{\mathcal{A}} = R^\top R$ is an OOM equivalence transformation according

to Proposition 6. It is easily checked that $C\pi_{\mathcal{A}}$ has the property 3 listed in Proposition 6. The critical issue is to show that $C\pi_{\mathcal{A}} = R^\top R$ is bijective, i.e. has rank m. We use that for any matrix A it holds that $\mathrm{rank}(A) = \mathrm{rank}(A^\top A)$. We see that $\mathrm{rank}(R^\top R) = \mathrm{rank}(R) = \mathrm{rank}((\mathbf{1}\tau_{\bar{b}_1})^\top \ \cdots \ (\mathbf{1}\tau_{\bar{b}_\kappa})^\top) = m$, where the last equality is due to the fact that the $\bar{b}_i$ are characterizing sequences. Thus, $C\pi_{\mathcal{A}} = R^\top R$ is an OOM equivalence transformation, and the vectors $v_{\bar{a}} = C\pi_{\mathcal{A}} w_{\bar{a}}$ are the states of an OOM equivalent to $\mathcal{A}$. But considering Eq. (32), this is just another way of saying that C is a characterizer.

Appendix H: Proof of Proposition 15

We first derive some conditions that the matrix G should satisfy. It follows from Prop. 12 that $\mathbf{1}_m G = \mathbf{0}$ and $G\underline{V} = \mathbf{0}$. As before, here we use $\mathbf{1}$ to denote the row vector of units, but with a subscript specifying its dimension. By the definition of $\underline{V}$ and $\pi_{\mathcal{A}}$ (cf. Eq. (36)), we have $\underline{V} = \pi_{\mathcal{A}} W$, where $W = [w_{\bar{a}_1} \cdots w_{\bar{a}_\kappa}]$. Because $\mathrm{rank} W = m$, it is clear that $G\underline{V} = \mathbf{0}$ if and only if $G\pi_{\mathcal{A}} = \mathbf{0}$. Thus, it suffices to show that $G = \mathbf{0}$ is a minimizer of the following optimization problem:

$$\begin{aligned} \min_{G} \quad & J(G) = \frac{1}{2} \sum_{i,j=1}^{\kappa} P(\bar{a}_i \bar{b}_j) \| w_{\bar{a}_i} - (C^r + G)(:, j) \|^2 , \\ \text{s.t.} \quad & \mathbf{1}_m G = \mathbf{0} , \quad G\pi_{\mathcal{A}} = \mathbf{0} . \end{aligned} \tag{57}$$

The target function $J(G)$ can be re-written as:

$$\begin{aligned} J(G) \ &= \ \frac{1}{2} \sum_{i,j=1}^{\kappa} P(\bar{a}_i \bar{b}_j) \left(\| w_{\bar{a}_i} \|^2 + \| (C^r + G)(:, j) \|^2 \right) \\ &\quad - \sum_{i,j=1}^{\kappa} P(\bar{a}_i \bar{b}_j) \Big\langle w_{\bar{a}_i}, (C^r + G)(:, j) \Big\rangle , \end{aligned} \tag{58}$$

where the pair $\langle x, y \rangle$ denotes the inner product of x and y. The second item on the r.h.s. of the above equality can be further simplified, as follows:

$$\begin{aligned} & \sum_{i,j=1}^{\kappa} P(\bar{a}_i \bar{b}_j) \Big\langle w_{\bar{a}_i}, (C^r + G)(:, j) \Big\rangle \\ = \ & \sum_{i=1}^{\kappa} \left\langle w_{\bar{a}_i}, \sum_{j=1}^{\kappa} P(\bar{a}_i \bar{b}_j) (C^r + G)(:, j) \right\rangle \\ = \ & \sum_{i=1}^{\kappa} \left\langle w_{\bar{a}_i}, P(\bar{a}_i) \sum_{j=1}^{\kappa} P(\bar{b}_j | \bar{a}_i) (C^r + G)(:, j) \right\rangle \\ = \ & \sum_{i=1}^{\kappa} \Big\langle w_{\bar{a}_i}, P(\bar{a}_i) (C^r + G) \underline{V}(:, i) \Big\rangle \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{\kappa} \left\langle w_{\bar{a}_i}, P(\bar{a}_i) w_{\bar{a}_i} \right\rangle \\
&= \sum_{i=1}^{\kappa} P(\bar{a}_i) \|w_{\bar{a}_i}\|^2 ,
\end{aligned}
$$

Assuming that the process is stationary, and substituting the above equality into Eq. (58), we get

$$
J(G) = \frac{1}{2} \sum_{j=1}^{\kappa} P(\bar{b}_j) \|(C^r + G)(:, j)\|^2 - \frac{1}{2} \sum_{i=1}^{\kappa} P(\bar{a}_i) \|w_{\bar{a}_i}\|^2 . \tag{59}
$$

Since the second item of Eq. (59) is irrelevant to G, minimizing $J(G)$ under the constraints (57) is a convex quadratic programming problem. So $G = \mathbf{0}$ is a minimizer of $J(G)$ if and only if it satisfies the following KKT system:

$$
\begin{aligned}
\frac{\partial J}{\partial G} = (C^r + G) D_p &= \mathbf{1}_m^\top \mu^\top + \lambda \pi_{\mathcal{A}}^\top , && (60) \\
\mathbf{1}_m G &= \mathbf{0} , && (61) \\
G \pi_{\mathcal{A}} &= \mathbf{0} , && (62)
\end{aligned}
$$

where $D_p = \mathrm{diag}\{P(\bar{b}_1), P(\bar{b}_2), \cdots, P(\bar{b}_\kappa)\}$; $\mu \in \mathbb{R}^\kappa$ and $\lambda \in \mathbb{R}^{m \times m}$ are Lagrange multipliers. By the definition of C^r (cf. the paragraph after Prop. 14, pp. 24), we have

$$
\begin{aligned}
C^r D_p &= \varrho^{-1} [P(\bar{b}_1) w^r_{\bar{b}_1}, \cdots, P(\bar{b}_\kappa) w^r_{\bar{b}_\kappa}] \\
&= \varrho^{-1} [\tau^r_{\bar{b}_1} w_0, \cdots, \tau^r_{\bar{b}_\kappa} w_0] \\
&= \varrho^{-1} [D \tau^\top_{\bar{b}_1} D^{-1} w_0, \cdots, D \tau^\top_{\bar{b}_\kappa} D^{-1} w_0] \\
&\quad \text{(cf. Prop. 13 and item 4 of the list thereafter)} \\
&= \varrho^{-1} D [\tau^\top_{\bar{b}_1} \mathbf{1}_m^\top, \cdots, \tau^\top_{\bar{b}_\kappa} \mathbf{1}_m^\top] \\
&= \varrho^{-1} D \pi_{\mathcal{A}}^\top , && (63)
\end{aligned}
$$

where $D = \mathrm{diag}(w_0)$ (cf. Prop. 13) and $\varrho = C \pi_{\mathcal{A}}$ (cf. Prop. 14). And it follows that $(G, \mu, \lambda) = (\mathbf{0}, \mathbf{0}, \varrho^{-1} D)$ is a solver of the KKT system (60)–(62).

By the above discussion, we conclude that $G = \mathbf{0}$ is a (global) minimizer of the target function $J(G)$; and is the unique minimizer if $P(\bar{b}_j) > 0$ $(j = 1, \cdots, \kappa)$.

Appendix I: Finding Good Characteristic Events

Given a training sequence $S = a_0 \ldots a_N$ over an alphabet O of size α, and given a desired length k of characteristic events and model dimension m, we use the following heuristic brute-force strategy to construct characteristic events $B_1, \ldots, B_m$ that in all our learning experiments rendered the matrix $\hat{V}$ or V^{raw} (cf. Eq. 30) reasonably well-behaved with respect to inversion, which is the prime requirement for success with the basic learning algorithm.

Let $\#\bar{a}$ denote the number of occurrences of some word $\bar{a}$ in S, let $\kappa = \alpha^k$ and let $(\bar{a}_j)_{1\leq j\leq\kappa}$ and $(\bar{b}_i)_{1\leq i\leq\kappa}$ both be the alphabetical enumeration of O^k. Start by constructing a $\kappa \times \kappa$ (often sparse) matrix $V_0^{\text{raw}} = (\#\bar{a}_j\bar{b}_i)$. Then it is clear that the matrix V^{raw} is obtained from V_0^{raw} by additively joining rows (to agglomerate characteristic sequences $\bar{b}$ into characteristic events B) and columns (to assemble indicative sequences $\bar{a}$ into indicative events A). We treat only the row-joining operations here; the column joining can be done simultaneously or separately in a similar fashion. So we consider a matrix sequence $V_0^{\text{raw}}, V_1^{\text{raw}}, \ldots, V_{\kappa-m-1}^{\text{raw}}$, where each matrix in the sequence is obtained from the previous by joining two rows. The last matrix $V_{\kappa-m-1}^{\text{raw}}$ then has size $m \times \kappa$; the characteristic sequences of the original rows from V_0^{raw} that are then collected in the i-th row of $V_{\kappa-m-1}^{\text{raw}}$ yield the desired characteristic events.

The intuitive strategy is to choose from V_n^{raw} for joining that pair of rows r_x, r_y that have the highest pairwise correlation $r_x/\|r_x\|(r_y/\|r_y\|)^\top$ among all pairs of rows in V_n^{raw}. This greedy strategy will (hopefully) result in characteristic events B that each comprise characteristic sequences $\bar{b}, \bar{b}'$ which are "prediction similar" in the sense that $P(\bar{b}|\bar{a}) \approx P(\bar{b}'|\bar{a})$ for all or most $\bar{a}$ – that is, joining $\bar{b}, \bar{b}'$ in B incurs a small loss of to-be-predicted distribution information. In addition we take care that the final characteristic events B_i are reasonably weight-balanced in the sense that $P(B_i) \approx P(B_{i'}) \approx 1/m$, in order to ensure that the estimation accuracy $\hat{P}_S(A_jB_i)$ is roughly similar for all entries of $\hat{V}$. Spelled out in more detail, we get the following joining algorithm:

1. **Initialization.** Construct V_0^{raw} and a normalized version V_0^{norm} thereof whose rows are either all zero (if the corresponding row in V_0^{raw} is zero) or have unit norm. For all rows of V_0^{raw} whose weight (sum of row entries) already exceeds N/m, put the corresponding row in V_0^{norm} to zero. These *finished* rows will thereby automatically become excluded from further joining. Set f to the number of finished rows. Furthermore, set the remaining mass Q of rows still open for joining to N minus the total entry sum of finished rows.

2. **Iteration.** V_n^{raw}, V_n^{norm} and f are given. If $f = m - 1$, jump to termination by joining all remaining unfinished rows. Else, compute the row correlation matrix $R = V_n^{\text{norm}}(V_n^{\text{norm}})^\top$ and choose the index (x, y) (where $y > x$) of the maximal off-diagonal entry in R for joining rows r_x, r_y by adding in V_n^{raw} the y-th row to the x-th and deleting the y-th row, obtaining V_{n+1}^{raw}. Normalize the summed row, replace the x-th row in V_n^{norm} by it and zero the y-th row in V_n^{norm}. If the entry sum of row x in V_{n+1}^{raw} exceeds $Q/(m-f)$, or if $m - f = \kappa - m - 1 - n$, increment f by one, zero the row x in V_n^{norm}, and decrement Q by the component sum of the x-th row in V_{n+1}^{raw}. The result of these operations on V_n^{norm} yields V_{n+1}^{norm}.

The computationally most expensive operation is $R = V_n^{\text{norm}}(V_n^{\text{norm}})^\top$. It can be effected by re-using the R from the previous step with $O(\kappa^2)$ floating point operations (the recursion will be easily found). All in all the theoretical cost of this algorithm is $O(\kappa^3) = O(\alpha^{3k})$, but for $\kappa/N > 1$ the concerned matrices

quickly become sparse, which could be exploited to greatly reduce the computational load. However, k should be chosen, if possible, such that $\kappa/N \leq 1$. The condition number of matrices $\hat{V}$ that we obtained in numerous experiments with natural and artificial data typically ranges between 2 and 50, which makes the algorithm very useful in practice for an initial model estimation with the basic OOM learning algorithm. Unfortunately it is theoretically not clarified what would be the best possible condition number among all choices of characteristic and indicative events; it may well be the case that the observed condition numbers are close to the optimum (or quite far away from it).

Appendix J: Running Invalid OOMs as Sequence Generators

Here is a modification of the sequence generation procedure described in Section 4, which allows us to use invalid OOMs and yet avoid negative probabilities. Notation from that section is re-used here without re-introduction. The basic idea is to check, at each generation step, whether the probability vector $\mathbf{p}$ contains negative entries, and if so, reset them to a predetermined, small positive margin (standard range: $0.001 \sim 0.01$), which is one of the three tuning parameters of this method. Furthermore, if the sum of negative entries in $\mathbf{p}$ falls below a significant setbackMargin (standard range: $-0.1 \sim -0.5$), indicating that the generation run is about to become instable, the generation is re-started setbackLength (typical setting: 2 or 3) steps earlier with the starting state w_0. Some care has to be taken that the resetting to margin leads to probability computations where the summed probability for all sequences of some length k is equal to 1. The method comes in two variants, one for generating random sequences, and the other for computing the probability of a given sequence S. The former has an additional step 3a in the description below. Detailed out, the n-th step using this method works as follows.

Input. Fixed parameters: margin, setbackMargin, setbackLength, size α of alphabet, observable operators τ_a, starting state w_0. Variables: the state w_{n-1}, and (if $n \geq$ setbackLength) the word $s = a_{n-\mathsf{setbackMargin}} \cdots a_{n-1}$ of previously processed symbols, and index i_{a_n} of current symbol a_n [*only if used probability computation mode*].

Output. State w_n, log-probability $L = \log(P(a_n|w_{n-1}))$, and new symbol a_n [*only if used in generation mode*] .

- **Step 1.** Compute $\mathbf{p} = \Sigma\, w_{n-1}$.
- **Step 2.** Compute $\delta = \sum_{i\in\{1,\ldots,\alpha\},\mathbf{p}(i)\leq 0}(\mathsf{margin} - \mathbf{p}(i))$; $p^+ = \sum_{i\in\{1,\ldots,\alpha\},\mathbf{p}(i)>0} \mathbf{p}(i)$; $p^- = p^+ - \delta$ and $\nu = p^-/p^+$.
- **Step 3.** *Check for potentially instable state, and act if necessary.*] If $\delta <$ setbackMargin and $n \leq$ setbackLength $+ 1$ [*we encounter a problematic state early in the process*], put $w_{n-1} = w_0$, recompute

$\mathbf{p} = \Sigma\, w_{n-1}$ and δ, p^+, p^-, ν as in step 2.
Else, if $\delta <$ setbackMargin [*we encounter a problematic state later in the process and restart the generation a few steps earlier*],
set $w = w_0$;
for $i = 1$ to setbackLength: $w = \tau_{s(i)} w$; $w = w/\mathbf{1}w$ [*we recompute the last few states from* w_0];
set $w_{n-1} = w$;
recompute $\mathbf{p} = \Sigma\, w_{n-1}$ and recompute δ, p^+, p^-, ν as in step 2.

- **Step 3a.** [*only executed in the generation variant*] Randomly choose a_n according to the probability vector $\mathbf{p}$; set i_{a_n} to its index in the alphabet.
- **Step 4.** [*update state and compute "blurred" probability of current symbol*]
If $\mathbf{p}(i_{a_n}) \leq 0$ [*current symbol would be assigned a negative probability*],
set $L = \log($margin$)$ and $w = \tau_{a_n} w_{n-1}$; $w_n = w/\mathbf{1}w$.
Else [*current symbol is O.K. but its probability has to be reduced to account for the added probability mass that might have been assigned to other symbols in this step*] set $L = \log(\nu \mathbf{p}(i_{a_n}))$ and $w = \tau_{a_n} w_{n-1}$; $w_n = w/\mathbf{1}w$.

Appendix K: Details of the 1,000,000 Mio Pound Learning Experiment

All OOMs computed during the ES iterations were invalid, so we employed the stabilizing method described in Appendix I for computing the requisite reverse state sequences. The same method was used to determine the log-likelihoods on the training and testing sequences. The settings (cf. Appendix I) that we used were margin $= 0.001$, setbackMargin $= 0.3$, setbackLength $= 2$. These settings were optimized by hand in preliminary tests.

Only such indicative sequences $\bar{a}$ were gleaned from the suffix tree that occurred at least 10 times in the training sequence.

This little study was carried out before the algorithm for finding good characteristic events described in Appendix H was available. Thus we used an inferior method for initial model estimation that we need not detail out here. Using the better method for initial model estimation would very likely have resulted in an improved overall performance (the high initial jump in model quality from the initial model to the first ES estimated model that appears in Figure 5b would disappear).

Acknowledgements

The results regarding the OOMs reported in Sections 1 through 9 of this chapter were obtained when the first author worked under a postdoctoral grant from the German National Research Center for Information Technology (GMD), now

Fraunhofer Institute for Autonomous Intelligent Systems. The first author feels deeply committed to its director Thomas Christaller for his unfaltering support.

References

[1] V. Balasubramanian. Equivalence and reduction of Hidden Markov models. A.I. Technical Report 1370, MIT AI Lab, 1993.

[2] Y. Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 2:129–162, 1999.

[3] A. Berman and R.J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Academic Press, 1979.

[4] D. Blackwell and L. Koopmans. On the identifiability problem for functions of finite Markov chains. *Annals of Mathematical Statistics*, 38:1011–1015, 1957.

[5] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The power of selective memory: Self-bounded learning of prediction suffix trees. Number 17 in Advances in Neural Information Processing Systems. MIT Press, 2004.

[6] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM-algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[7] S.W. Dharmadhikari. Functions of finite Markov chains. *Annals of Mathematical Statistics*, 34:1022–1032, 1963.

[8] S.W. Dharmadhikari. Sufficient conditions for a stationary process to be a function of a finite Markov chain. *Annals of Mathematical Statistics*, 34:1033–1041, 1963.

[9] S.W. Dharmadhikari. A characterization of a class of functions of finite Markov chains. *Annals of Mathematical Statistics*, 36:524–528, 1965.

[10] J.L. Doob. *Stochastic Processes*. John Wiley & Sons, 1953.

[11] R. Durbin, S. Eddy, A. Krogh, and G Mitchinson. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 2000.

[12] R. Edwards, J.J. McDonald, and M.J. Tsatsomeros. On matrices with common invariant cones with applications in neural and gene networks. *Linear Algebra and its Applications*, in press, 2004 (online version). preprint at http://www.math.wsu.edu/math/faculty/tsat/files/emt.pdf.

[13] R.J. Elliott, L. Aggoun, and J.B. Moore. *Hidden Markov Models: Estimation and Control*, volume 29 of *Applications of Mathematics*. Springer Verlag, New York, 1995.

[14] B. Farhang-Boroujeny. *Adaptive Filters: Theory and Applications*. Wiley, 1998.

[15] M. Fox and H. Rubin. Functions of processes with Markovian states. *Annals of Mathematical Statistics*, 39(3):938–946, 1968.

[16] M. Fox and H. Rubin. Functions of processes with Markovian states II. *Annals of Mathematical Statistics*, 40(3):865–869, 1969.

[17] M. Fox and H. Rubin. Functions of processes with Markovian states III. *Annals of Mathematical Statistics*, 41(2):472–479, 1970.

[18] E. Fredkin. Trie Memory. *Comm. ACM*, 3(9):490–499, September 1960.

[19] Robert Giegerich and Stefan Kurtz. From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction. *Algorithmica*, 19(3):331–353, 1997.

[20] E.J. Gilbert. On the identifiability problem for functions of finite Markov chains. *Annals of Mathematical Statistics*, 30:688–697, 1959.

[21] A. Heller. On stochastic processes derived from Markov chains. *Annals of Mathematical Statistics*, 36:1286–1291, 1965.

[22] M. Iosifescu and R. Theodorescu. *Random Processes and Learning*, volume 150 of *Die Grundlagen der mathematischen Wissenschaften in Einzeldarstellungen*. Springer Verlag, 1969.

[23] H. Ito. *An algebraic study of discrete stochastic systems*. Phd thesis, Dpt. of Math. Engineering and Information Physics, 1992.

[24] H. Ito, S.-I. Amari, and K. Kobayashi. Identifiability of hidden Markov information sources and their minimum degrees of freedom. *IEEE transactions on information theory*, 38(2):324–333, 1992.

[25] H. Jaeger. Observable operator models and conditioned continuation representations. Arbeitspapiere der GMD 1043, GMD, Sankt Augustin, 1997. http://www.faculty.iu-bremen.de/hjaeger/pubs/jaeger.97.oom.pdf.

[26] H. Jaeger. Observable operator models II: Interpretable models and model induction. Arbeitspapiere der GMD 1083, GMD, Sankt Augustin, http://www.faculty.iu-bremen.de/hjaeger/pubs/jaeger.97.oom2.pdf, 1997.

[27] H. Jaeger. Discrete-time, discrete-valued observable operator models: a tutorial. GMD Report 42, GMD, Sankt Augustin, 1998. http://www.faculty.iu-bremen.de/hjaeger/pubs/oom_tutorial.pdf.

[28] H. Jaeger. Modeling and learning continuous-valued stochastic processes with OOMs. GMD Report 42, GMD, Sankt Augustin, 1998. http://www.faculty.iu-bremen.de/hjaeger/pubs/jaeger.00.tr.contoom.pdf.

[29] H. Jaeger. Characterizing distributions of stochastic processes by linear operators. GMD Report 62, German National Research Center for Information Technology, 1999. http://www.faculty.iu-bremen.de/hjaeger/pubs/oom_distributionsTechRep.pdf.

[30] H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000. Draft version: http://www.faculty.iu-bremen.de/hjaeger/pubs/oom_neco00.pdf.

[31] M. James and S. Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *Proc. 21rst Int. Conf. Machine Learning*, pages 417–424, 2004. http://www.eecs.umich.edu/ baveja/Papers/resetPSRsdist.pdf.

[32] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1998.

[33] L.P. Kaelbling, M.L. Littman, and A.R.¡ Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[34] K. Kretzschmar. Learning symbol sequences with Observable Operator Models. GMD Report 161, Fraunhofer Institute AIS, 2003. http://omk.sourceforge.net/files/OomLearn.pdf.

[35] M. L. Littman, R. S. Sutton, and S. Singh. Predictive representation of state. In *Advances in Neural Information Processing Systems 14 (Proc. NIPS 01)*, pages 1555–1561, 2001. http://www.eecs.umich.edu/ baveja/Papers/psr.pdf.

[36] D. R. Morrison. PATRICIA - Practical Algorithm to Retrieve Information Coded Alphanumeric. *Journal of the ACM*, 15(4):514–534, October 1968.

[37] T. Oberstein. *Efficient Training of Observable Operator Models*. Master thesis, Köln University, 2002. http://omk.sourceforge.net/files/eloom.pdf.

[38] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann, San Mateo, 1990. Reprinted from Proceedings of the IEEE 77 (2), 257-286 (1989).

[39] R.D. Smallwood and E.J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[40] D.R. Upper. *Theory and algorithms for Hidden Markov models and Generalized Hidden Markov models*. Phd thesis, Univ. of California at Berkeley, 1997. http://www.santafe.edu/projects/CompMech/papers/TAHMMGHMM.html.

[41] P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th Symposium on Switching and Automata Theory*, pages 1–11, 1973.

[42] L.A. Zadeh. The concept of system, aggregate, and state in system theory. In L.A. Zadeh and E. Polak, editors, *System Theory*, volume 8 of *Inter-University Electronics Series*, pages 3–42. McGraw-Hill, New York, 1969.