Modeling and learning continuous-valued stochastic processes with OOMs

Herbert Jaeger
GMD – German National Research Center
for Information Technology

June 8, 2000

Abstract. Observable operator models (OOMs) are a generalization of hidden Markov models (HMMs). They support a fast, novel learning algorithm. Standardly, OOMs model discrete-valued ("symbol") processes. This report shows how OOMs for continuous-valued processes can be defined, and how the learning algorithm can be extended.

key words: stochastic processes, observable operator models

Zusammenfassung. Observable Operator Modelle (OOMs) sind eine Verallgemeinerung von Hidden Markov Modellen (HMMs). Sie sind mit einem neuartigen, schnellen Lernalgorithmus ausgerüstet. Die bisher bekannten OOMs modellierten diskretwertige Prozesse. In diesem Report werden OOMs und der Lernalgorithmus auf kontinuierlichwertige Prozesse erweitert.

Stichwörter: Stochastische Prozesse, observable operator models

1 Introduction

In this report it is shown how observable operator models (OOMs) and their learning algorithm can be generalized from discrete-valued to continuous-valued processes. It is assumed that the reader is familiar with discrete OOMs. Introductions can be found e.g. in [9] [7].

We start with a crash review of discrete observable operator models (OOMs) for fixing terminology and notation. OOMs model distributions of stochastic processes very much like hidden Markov models do. Consider a discrete-time stochastic process $(\Omega, \mathcal{F}, P, (X_t)_{t \in \mathbb{N}})$ (or shortly, X) with values in a finite set $E = \{a_1, \ldots, a_n\}$, i.e., a symbol process. An OOM of X is a linear algebra structure $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a \in E}, w_0)$, (where $\mathbb{R}^m$ is its *state space*; for every $a \in E$, the *observable operator* $\tau_a : \mathbb{R}^m \to \mathbb{R}^m$ is a linear map rendered by a matrix, and $w_0 \in \mathbb{R}^m$ is a *starting vector*), such that the probabilities of finite sequences can be calculated by iterated matrix multiplications, as follows:

$$P(X_0 = a_{i_0}, \ldots, X_k = a_{i_k}) = \mathbf{1}\, \tau_{a_{i_k}} \cdots \tau_{a_{i_0}} w_0, \tag{1}$$

where $\mathbf{1}$ is the (row) vector $(1, \ldots, 1)$ of length m. We will use shorthand $P(a_{i_0} \ldots a_{i_k})$ or even $P(\bar{a})$ to denote these and similar probabilities.

The vector $\tau_{a_{i_k}} \cdots \tau_{a_{i_0}} w_0 / \mathbf{1}\, \tau_{a_{i_k}} \cdots \tau_{a_{i_0}} w_0$ is the *state vector* obtained in the OOM after the generation of a sequence $a_{i_0} \ldots a_{i_k}$. Note that it is normalized to unit component sum. It is also written as $w_{a_{i_0} \ldots a_{i_k}}$. It follows from (1) that the conditional probability $P(b_{i_{k+1}} \ldots b_{i_{k+l}} \mid a_{i_0} \ldots a_{i_k})$ to observe a sequence $b_{i_{k+1}} \ldots b_{i_{k+l}}$ after a prior history of $a_{i_0} \ldots a_{i_k}$, is equal to $\mathbf{1}\, \tau_{b_{i_{k+l}}} \cdots \tau_{b_{i_{k+1}}} w_{a_{i_0} \ldots a_{i_k}}$.

Two OOMs are *equivalent* when they yield the same distribution according to (1). An OOM $\mathcal{A}$ is *minimal-dimensional* if every OOM $\mathcal{B}$ which is equivalent to $\mathcal{A}$ has a state space dimension not smaller than that of $\mathcal{A}$. Given some OOM $\mathcal{A}$, a minimal-dimensional, equivalent OOM can be effectively constructed. Two minimal-dimensional OOMs $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a \in E}, w_0), \mathcal{A}' = (\mathbb{R}^m, (\tau'_a)_{a \in E}, w'_0)$ are equivalent if and only if there exists a matrix ϱ whose column vectors each sum to unity, and which renders $\mathcal{A}$ and $\mathcal{A}'$ conjugate, i.e., $\forall a \in E : \varrho \tau_a = \tau'_a \varrho$ and $\varrho w_0 = w'_0$. Historically, it has been the quest for a similar equivalence theorem for HMMs [2] that led to the stepwise discovery and elaboration of OOMs (called *linearly dependent processes* in [5] and *generalized HMMs* in [10]). The vector space dimension of a minimal-dimensional OOM that describes X is called the *dimension of X*. This notion of dimension of a process can be generalized to arbitrary-valued (e.g., continuous-valued), discrete or continuous-time processes [9] [8].

If $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a \in E}, w_0)$ is a minimal-dimensional OOM of X, then m is called the *dimension of the process* X. The dimension of a process is a fundamental characteristic of its stochastic complexity (see [9] for a probability theoretic account of this dimension).

If $\mathcal{A}$ is a minimal-dimensional OOM of X, then X is strictly stationary if and only if $\mu w_0 = w_0$, where $\mu = \sum_{a \in E} \tau_a$.

The state vectors of an OOM cannot in general be interpreted as probability distributions, as it is possible with HMMs. However, an *interpretable* version of OOMs can be obtained as follows [6]. Let $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a \in E}, w_0)$ be an m-dimensional OOM. Select some $\kappa \geq 1$, and some partition of $E^\kappa = A_1 \cup \cdots \cup A_m$ into m disjoint, non-empty subsets A_i (i.e., A_i consists of observation sequences of length κ). Then, in general (some pathological selections of sets A_i excepted) one can construct a conjugate mapping ϱ and use it to transform $\mathcal{A}$ into an equivalent OOM $\mathcal{A}' =: \mathcal{A}(A_1, \ldots, A_m) = (\mathbb{R}^m, (\varrho \tau_a \varrho^{-1})_{a \in E}, \varrho w_0)$, which has the following property. Let $w_{a_{i_0} \ldots a_{i_k}} = (x_1, \ldots, x_m)$ be the state vector obtained after an initial sequence $a_{i_0} \ldots a_{i_k}$. Then the j-th component x_j of the state vector is equal to the probability that the event A_j is observed after the initial history $a_{i_0} \ldots a_{i_k}$:

$$x_j = P(A_j \,|\, a_{i_0} \ldots a_{i_k}), \tag{2}$$

where $P(A_j \mid a_{i_0} \ldots a_{i_k})$ denotes $\sum_{b_{j_1} \ldots b_{j_\kappa} \in A_j} P(b_{j_1} \ldots b_{j_\kappa} \mid a_{i_0} \ldots a_{i_k}) = \sum_{b_{j_1} \ldots b_{j_\kappa} \in A_j} \mathbf{1} \, \tau_{b_{j_\kappa}} \cdots \tau_{b_{j_1}} w_{a_{i_0} \ldots a_{i_k}}$. In other words, state vectors w that occur in an *interpretable* OOM $\mathcal{A}(A_1, \ldots, A_m)$ can be interpreted as the probability distribution of the *characteristic events* A_j to be observed after the OOM is in the state w. By consequence, state vectors in an interpretable OOMs always fall into the non-negative hyperplane $H^{\geq 0} = \{(x_1, \ldots, x_m) \in \mathbb{R}^m \mid x_i \geq 0, x_1 + \cdots x_m = 1\}$. Figure 1 depicts this hyperplane for $m = 3$ and shows state diagrams of three different OOMs, all of which had three observable events $E = \{a, b, c\}$ and were interpretable w.r.t. the singleton characteristic events $A_1 = \{a\}, A_2 = \{b\}, A_3 = \{c\}$.

In an interpretable OOM $\mathcal{A}(A_1, \ldots, A_m)$, it holds that

$$\tau_{a_l} \cdots \tau_{a_1} w_0 = (P(a_1 \ldots a_l A_1), \ldots, P(a_1 \ldots a_l A_m))^\mathsf{T}, \tag{3}$$

where $\cdot^\mathsf{T}$ denotes transpose. (3) leads directly to a constructive estimation of OOMs from data. We describe it here for stationary processes.

Assume that a (long) path S of length N has been generated by some unknown, m-dimensional, stationary OOM $\mathcal{A}$. Let $\mathcal{A}(A_1, \ldots, A_m)$ be an interpretable version of $\mathcal{A}$. The learning task considered here is to (i) estimate the dimension m from S; and (ii) estimate an interpretable OOM

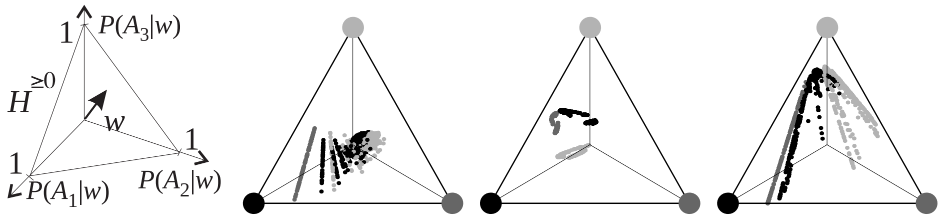

Figure 1: From left to right: the admissible state hyperplane $H^{\geq 0}$ in an interpretable OOM, and three exemplary state diagrams obtained from runs of length 1000. Three grayscales in these diagrams correspond to three operators/events. For details see text.

$\tilde{\mathcal{A}}(A_1, \ldots, A_m)$ from S such that $\tilde{\mathcal{A}}(A_1, \ldots, A_m)$ converges to $\mathcal{A}(A_1, \ldots, A_m)$ (in some matrix norm) almost surely as $N \to \infty$.

Task (1) can be solved by adapting a criterium used in numerical linear algebra for deciding the rank of a noisy matrix. The procedure is detailed out in [7]. We omit this step here and assume that m has been correctly estimated.

Estimating an OOM basically means to construct its observable operators, i.e., linear maps $\tilde{\tau}_a : \mathbb{R}^m \to \mathbb{R}^m$. Such a linear map is defined by the values it takes on m linearly independent argument vectors. That is, in order to obtain an estimate $\tilde{\tau}_a$, we have to procure estimates $\tilde{u}_1, \ldots, \tilde{u}_m$ of m linearly independent vectors $u_1, \ldots, u_m$, and estimates $\tilde{u}'_1, \ldots, \tilde{u}'_m$ of the images $u'_1, \ldots, u'_m$ of $u_1, \ldots, u_m$ under τ_a. Equation (3) points out how this can be achieved in an interpretable OOM $\mathcal{A}(A_1, \ldots, A_m)$: for u_1, take $w_0 = (P(A_1), \ldots, P(A_m))^{\mathsf{T}}$. An estimate $\tilde{u}_1$ of w_0 can be estimated from S by an obvious frequency count. Then, (3) tells us that $u'_1 = \tau_a u_1 = (P(aA_1), \ldots, P(aA_m))^{\mathsf{T}}$, which can likewise be estimated by a frequency count. For u_2, select some other vector, e.g. $\tau_a w_0$ (which we already have estimated), and estimate it and its image under τ_a again by a frequency count.

This is the basic idea. If it is systematically assembled, one obtains an asymptotically correct estimation procedure that has a time complexity of $O(N + nm^3)$.

In order to make this estimation algorithm work in practice, two auxiliary procedures have to be carried out. The first is the selection of characteristic events. This has to be done with some care, because the selection greatly affects the convergence rate of the estimation procedure. [7] presents some rules-of-thumb. Secondly, the estimation procedure sometimes may return a set of matrices which is no valid OOM, i.e. if one uses these matrices to compute probabilities according to (1), negative "probability" values may be obtained for some sequences. This problem calls for a renormalization procedure which transforms the estimated matrices into the nearest valid OOM (work in progress).

2 OOMs for continuous-valued processes

Basic OOMs, as described in the Introduction, characterize distribution of discrete-valued processes. However, many interesting systems are observed through continuous-valued (but discrete-time) measurements in some measurement interval $I \subseteq \mathbb{R}^p$. One way to cope with such processes would be to discretize the observation space and use discrete-valued OOMs.

A more satisfactory approach is to use continuous-valued OOMs, defined as follows. Let $(\Omega, \mathcal{F}, P, (Y_t)_{t\in\mathbb{N}})$ be a process with values in the continuous measurable space $(I, \mathcal{B})$, where $\mathcal{B}$ is the Borel-σ-algebra on I. An (finite-dimensional) OOM of Y is a structure $\mathcal{C} = (\mathbb{R}^m, (\tau_J)_{J\in\mathcal{B}}, w_0)$, which allows to calculate the distribution of Y according to the following variant of (1):

$$P(Y_0 \in J_0, \ldots, Y_k \in J_k) = \mathbf{1}\, \tau_{J_k} \cdots \tau_{J_0} w_0. \tag{4}$$

In [8] it is shown how an OOM (not necessarily finite-dimensional) can be constructed for every continuous-valued process. The contribution of this article is to describe a way in which the uncountably many observable operators $(\tau_J)_{J\in\mathcal{B}}$ of a continuous-valued OOM can be "blended" from finitely many observable operators of a basic OOM, and how this basic OOM can be identified from continuous-valued data. The basic idea of *blended* OOMs is illustrated in Figure 2 and formalized in Definition 1.

Definition 1 *Let $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in E}, w_0)$ be a discrete-valued OOM, where $E = \{a_1, \ldots, a_n\}$. Let $I \subseteq \mathbb{R}^p$ be an interval. For $a \in E$ let $\nu_a : I \to \mathbb{R}$ be a probability density on I. We call the ν_a* membership functions. *For every $x \in I$, define*

$$\tau_x = \sum_{a\in E} \nu_a(x)\tau_a, \tag{5}$$

and for every $J \in \mathcal{B}$ define

$$\tau_J = \int_J \tau_x d\mu, \tag{6}$$

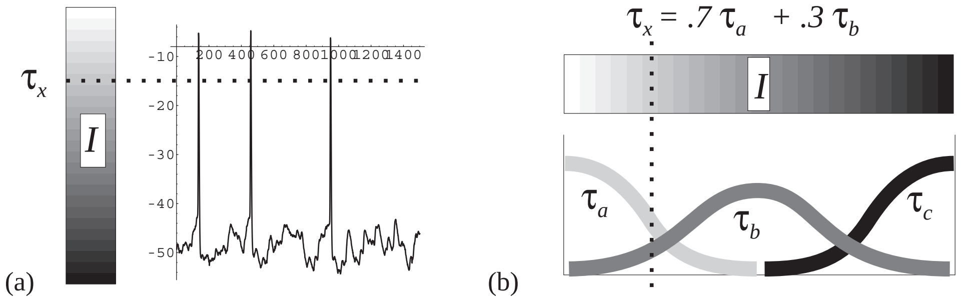

Figure 2: (a) A process with observation values in a one-dimensional interval I. A value x is modeled by an observable operator τ_x. (b) A real-number-indexed operator τ_x is linearly combined from finitely many symbolically indexed operators (here: τ_a, τ_b) via membership functions.

where μ is the Borel measure on I. Then $\mathcal{C} = (\mathbb{R}^m, (\tau_J)_{J\in\mathcal{B}}, w_0)$ is the (continuous-valued) OOM blended *from $\mathcal{A}$ via the membership functions $(\nu_a)_{a\in E}$.*

Blended OOMs can be used as sequence generators for the process whose distribution is given by (4):

Proposition 1 *Let $\mathcal{C} = (\mathbb{R}^m, (\tau_J)_{J\in\mathcal{B}}, w_0)$ be an OOM blended from $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in E}, w_0)$ via $(\nu_a)_{a\in E}$. Use $\mathcal{C}$ to generate sequences $x_0x_1x_2\ldots \in I^{\mathbb{N}}$ inductively, as follows. Compute a probability density $\nu_0 = \sum_{a\in E}(\mathbf{1}\,\tau_a w_0)\,\nu_a$ on I. Randomly select $x_0 \in I$ according to this density. Put $w_1 = \tau_{x_0}w_0/(\mathbf{1}\,\tau_{x_0}w_0)$. To select x_1, repeat previous steps, using state vector w_1 instead of w_0. Iterate ad infinitum. The process whose paths are obtained according to this algorithm is distributed according to (4).*

Furthermore, if $\mathcal{A}'$ is equivalent to $\mathcal{A}$ and $\mathcal{C}'$ is blended from $\mathcal{A}'$ via the same membership functions $(\nu_a)_{a\in E}$, then the process distribution described by $\mathcal{C}'$ is the same as the one described by $\mathcal{C}$. Finally, if the discrete-valued process described by $\mathcal{A}$ is stationary, then the continuous-valued process described by $\mathcal{C}$ is stationary, too. (The proof is given in the Appendix).

It is easy to see that blending is invariant w.r.t. equivalence of the underlying basic OOM. I.e., if $\mathcal{C}$ is blended from $\mathcal{A}$ via $(\nu_a)_{a\in E}$, and $\mathcal{A}'$ is equivalent to $\mathcal{A}$, and $\mathcal{C}'$ is blended from $\mathcal{A}'$ via $(\nu_a)_{a\in E}$, then $\mathcal{C}$ and $\mathcal{C}'$ generate the same process.

Blended OOMs are related to continuous-valued HMMs in the following way.

Proposition 2 *Let $\mathcal{H}$ be an ordinary HMM with values in $E = \{a_1, \ldots, a_n\}$ and hidden states $S = \{s_1, \ldots, s_m\}$. Let $I \subset \mathbb{R}^p$ be an interval, and for every $a \in E$, let ν_a be a probability density on I. Let $P(a \mid s)$ be the probability that $\mathcal{H}$ emits a from hidden state s. For every $s \in S$ define a probability density $\nu_s = \sum_{a \in E} P(a \mid s)\nu_a$. Define a continuous-valued process by letting the hidden Markov process emit $x \in I$ from state s according to ν_s. This process is the same as the one that is obtained when (i) $\mathcal{H}$ is formulated as an OOM $\mathcal{A}$, and (ii) this OOM $\mathcal{A}$ and the membership functions ν_a are used to generate a process according to Proposition 1.*

A precise statement of (i) and a proof are given in the Appendix. Note that the class of continuous-valued HMMs constructed in this way is what is called *semi-continous* HMMs in the HMM literature [1]. However, the emission densities of semi-continuous HMMs are usually mixtures of Gaussians, whereas here we allow mixtures of arbitrary density functions.

Example. For a basic OOM, we take the "probability clock" OOM $\mathcal{A} = (\mathbb{R}^3, \{\tau_a, \tau_b\}, w_0)$, which is described in detail in [9]. The most conspicuous property of the two-symbol-process generated by this OOM is that the conditional probabilities $P(a \mid a^t)$ to observe an a after a prior history of t consecutive a's is an oscillation (in t). Note that such a probability oscillation cannot be modeled by HMMs (they can only approximate it by sampling the time axis with hidden states, cf. [7]). Figure 3 (a) shows the fifty most frequently visited states in an interpretable version $\mathcal{A}(\{aa\}, \{ab\}, \{ba, bb\})$, which actually lie on an ellipse (a fact which is closely related with the probability oscillation). Figure 3 (b) depicts the evolution of $P(a \mid a^t)$.

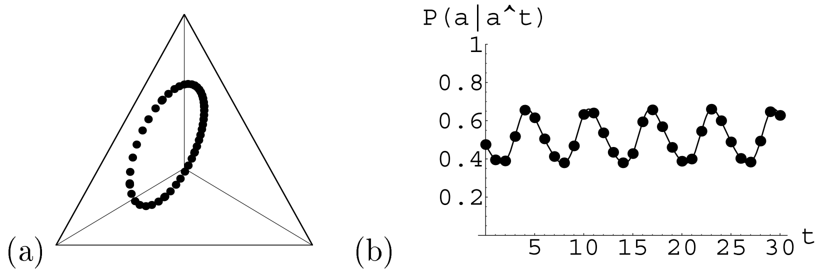

Figure 3: The probability clock. (a) "Fingerprint" of the probability clock. The 50 most frequently visited states are shown. (b) The probability oscillation.

From $\mathcal{A}$ we construct a continuous-valued process with values in $[0, 1]$ by blending τ_a, τ_b via the membership functions $\nu_a(x) = 2-2x$ and $\nu_b(x) = 2x$, to obtain a continuous-valued OOM $\mathcal{C} = (\mathbb{R}^3, (\tau_J)_{J \in \mathfrak{B}}, w_0)$. Figure 4 (a) shows a path generated by this blended OOM, which on visual inspection looks very much like an i.i.d. process with a uniform distribution. This impression is corroborated by the autocorrelation plot (Fig. 4(b)), which reveals that the process is almost a pure white noise process. This is mirrored by the (equivalent) fact that the OOM states all fall into a small cluster (Fig. 4(c)), which implies that the conditional distributions $P(\cdot\,|\,\text{prior history})$ vary only little with prior histories, i.e. the process is almost memoryless.

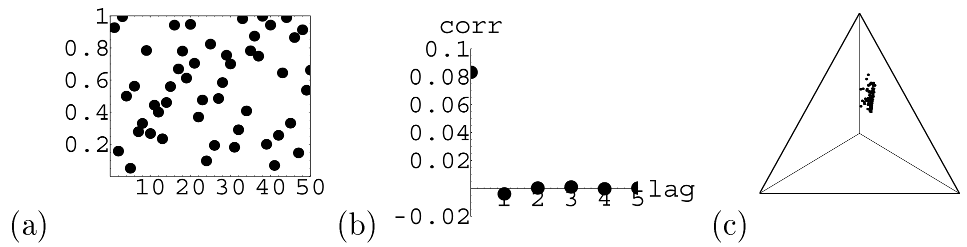

Figure 4: The process blended from the probability clock. (a) A 50-step path of the blended process. (b) Autocorrelation diagram. (c) Fingerprint of the blended OOM obtained from a 100 step run.

3 Blended OOMs and their underlying discrete OOMs

The symbolic process X generated by a basic OOM $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in E}, w_0)$ is connected to the process Y generated by an OOM $\mathcal{C} = (\mathbb{R}^m, (\tau_J)_{J\in\mathcal{B}}, w_0)$ blended from $\mathcal{A}$ via $(\nu_a)_{a\in E}$, by virtue of the following *transfer function* T:

Proposition 3 *Let $E = \{a_1, \ldots, a_n\}$. Define a function $T : \mathbb{R}^n \to \mathbb{R}^n$ by*

$$T(y_1, \ldots, y_n) = (\int_I \nu_{a_1}(x) \sum_{i=1}^n y_i \nu_{a_i}(x)\ d\mu(x), \ldots, \int_I \nu_{a_n}(x) \sum_{i=1}^n y_i \nu_{a_i}(x)\ d\mu(x)). \tag{7}$$

T has the following properties:

1. *T is linear.*
2. *$(\langle \nu_{a_i}, \nu_{a_j}\rangle)_{i,j}$, where $\langle \nu_{a_i}, \nu_{a_j}\rangle$ denotes the inner product $\int_I \nu_{a_i}(x)\ \nu_{a_j}(x)\ d\mu(x)$, is a matrix representation of T.*

3. *T is invertible if and only if the density functions $(\nu_a)_{a\in E}$ are linearly independent.*

4. *$T(P(X_0 = a_1), \ldots, P(X_0 = a_n)) = (E[\nu_{a_1} \circ Y_0], \ldots, E[\nu_{a_n} \circ Y_0])$, where $E[\nu_{a_i} \circ Y_0]$ is the expected value of the random variable obtained through concatenating Y_0 with ν_{a_i}.*

The proof is given in the Appendix. Point *4.* establishes a connection between X_0 and Y_0. We proceed to extend this connection to length k sequences $X_0, \ldots, X_{k-1}$ and $Y_0, \ldots, Y_{k-1}$. The idea is to partition both processes into blocks of length k, and re-use Proposition 3 for the blocked process. We first describe the partition:

Proposition 4 *Let $\mathcal{A}$ and $\mathcal{C}$ be as above. Let $k \geq 2$. For every length k sequence $a_{i_0} \ldots a_{i_{k-1}} = \bar{a} \in E^k$ define a function $\nu_{\bar{a}} : I^k \to \mathbb{R}$ by $\nu_{\bar{a}}(x_0, \ldots, x_{k-1}) = \nu_{a_{i_0}}(x_0) \cdot \ldots \cdot \nu_{a_{i_{k-1}}}(x_{k-1})$. Then (1) $\nu_{\bar{a}}$ is a probability density. Define $\tau_{\bar{a}} = \tau_{a_{i_{k-1}}} \circ \cdots \circ \tau_{a_{i_0}}$. Let $\mathcal{A}^k = (\mathbb{R}^m, (\tau_{\bar{a}})_{\bar{a}\in E^k}, w_0)$. $\mathcal{A}^k$ describes a k-blocked version X^k of X in an obvious analogy of Eq. (1), and it holds that $P(X_0^k = a_{0,i_0} \ldots a_{0,i_{k-1}}, \ldots, X_{l-1}^k = a_{l-1,i_0} \ldots a_{l-1,i_{k-1}}) = P(X_0 = a_{0,i_0}, \ldots, X_{l\cdot k-1} = a_{l-1,i_{k-1}})$. In analogy to Definition 1, define $\tau_{x_0 \ldots x_{k-1}} = \sum_{\bar{a}\in E^k} \nu_{\bar{a}}(x_0 \ldots x_{k-1})\, \tau_{\bar{a}}$ and for $\bar{J} = J_0 \times \cdots \times J_{k-1}$ define $\tau_{\bar{J}} = \int_{\bar{J}} \tau_{\bar{x}} d\mu(\bar{x})$ (the $\bar{\ }$ notation should be self-explanatory). Define $\mathcal{C}^k = (\mathbb{R}^m, (\tau_{\bar{J}})_{\bar{J}\in\mathcal{B}^k}, w_0)$. Then (2) $\mathcal{C}^k$ defines a distribution of a process Y^k with values in $\mathbb{R}^k$ by $P(Y_0^k \in \bar{J}_0, \ldots, Y_l^k \in \bar{J}_l) = \mathbf{1}\, \tau_{\bar{J}_l} \cdots \tau_{\bar{J}_0} w_0$. Furthermore, (3) this distribution is the same as the one of the process obtained when $\mathcal{C}^k$ is used as a generator similar to the procedure from Proposition 1. Finally, if $T^k : \mathbb{R}^{n^k} \to \mathbb{R}^{n^k}$ denotes the analog for the blocked process of T described in Proposition 3, it holds (4) that $T^k = T \otimes \cdots \otimes T$, where the right hand side is an outer (Kronecker) product with k factors.*

The proof is given in the Appendix. Note that T^k is invertible iff T is invertible, i.e., iff the density functions $(\nu_a)_{a\in E}$ are linearly independent. Furthermore, note that the following k-block version of Prop. 3(*4*) holds:

$$\begin{aligned} T(P(X_0^k = a_1 \ldots a_1), P(X_0^k = a_1 \ldots a_1 a_2), \ldots, P(X_0^k = a_n \ldots a_n)) &= \\ = (E[\nu_{a_1 \ldots a_1} \circ Y_0^k], \ldots, E[\nu_{a_n \ldots a_n} \circ Y_0^k]). \quad (8) \end{aligned}$$

A standard operation on a discrete OOM $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in E}, w_0)$, where $E = \{a_1, \ldots, a_n\}$, is to join two observations, say, a_1 and a_2, into an observation b with the semantics "a_1 or a_2". The corresponding OOM is $\mathcal{A}' = (\mathbb{R}^m, \{\tau_b, \tau_{a_3}, \ldots, \tau_{a_n}\}, w_0)$, where $\tau_b = \tau_{a_1} + \tau_{a_2}$. We conclude this section with a description of a similar construction for blended OOMs, and how that relates to the underlying discrete OOMs. This (technical) result is needed in the learning procedure.

Proposition 5 *Let the continuous-valued OOM $\mathcal{C} = (\mathbb{R}^m, (\tau_J)_{J\in\mathcal{B}}, w_0)$ be blended from $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in E}, w_0)$ via membership functions $(\nu_a)_{a\in E}$, and let Y be the corresponding continuous-valued process. Let $\alpha = \mathbf{1}\tau_{a_1}w_0, \beta = \mathbf{1}\tau_{a_2}w_0$. Define a probability density function*

$$\nu_b = \frac{\alpha\nu_{a_1} + \beta\nu_{a_2}}{\alpha+\beta}. \tag{9}$$

Rename $\{b, a_3, \ldots, a_n\}$ to $E' = \{b_1, \ldots, b_{n-1}\}$. Construct a transfer function $T' = (\langle \nu_{b_i}, \nu_{b_j}\rangle)_{i,j=1,\ldots,n-1}$. Let Y' be the continuous-valued process obtained through blending from $\mathcal{A}'$ via membership functions $\nu_{b_1}, \ldots, \nu_{b_{n-1}}$. Then it holds that

$$\begin{aligned} T'(P(X_0 = a_1) + P(X_0 = a_2), P(X_0 = a_3), \ldots, P(X_0 = a_n)) &= \\ = \ (E[\nu_{b_1} \circ Y_0'], \ldots, E[\nu_{b_{n-1}} \circ Y_0']) & \\ = \ (E[\nu_{b_1} \circ Y_0], \ldots, E[\nu_{b_{n-1}} \circ Y_0]). & \end{aligned} \tag{10}$$

The proof is given in the Appendix.

4 Learning continuous-valued OOMs

We solve the following learning task. Let $S = x_0 \ldots x_N$ be a path of a continuous-valued, ergodic, strictly stationary process Y generated by an unknown OOM $\mathcal{C} = (\mathbb{R}^m, (\tau_J)_{J\in\mathcal{B}}, w_0)$, which in turn is blended from an unknown, discrete-valued, minimal-dimensional OOM $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a\in E}, w_0)$ of a process X, via known membership functions $(\nu_a)_{a\in E}$, which are linearly independent. The task is to derive from S an asymptotically (in N) correct estimate $\tilde{\mathcal{A}}$ of $\mathcal{A}$.

Our approach is to translate, using the transfer function, information about Y contained in S into information pertaining to $\mathcal{A}$, and use the known learning algorithm for discrete-valued OOMs to obtain an estimate $\tilde{\mathcal{A}}$ of $\mathcal{A}$ from this information. An estimate $\tilde{\mathcal{C}}$ of $\mathcal{C}$ is then trivially obtained from $\tilde{\mathcal{A}}$ by blending via $(\nu_a)_{a\in E}$.

The learning procedure both (i) fixes an appropriate model size, such that teaching data are neither under-exploited nor overfitted, and (ii) calculates a model estimate.

The learning proceeds in three major steps. In the first step, certain statistics of the teaching data S are collected and arranged in matrices. The size of these matrices is (roughly) selected as large as needed to ensure that information in S be not under-exploited. In the second, fine-tuning step, these matrices are stepwise reduced in size, until they have a size that guarantees that S is not overfitted. The third and final step is an application of the standard learning procedure for discrete OOMs.

The learning procedure will be illustrated by a probability clock example[1]. The OOM $\mathcal{C}$ blended from the probability clock (as described in section 2) was run for $N = 2,000,000$ time steps to yield a path S. The task is to recover from S the original probability clock. Throughout the following description of the learning process, we will refer to this example.

4.1 First major step

Overview.. We start with the continuous-valued observation sequence S and compute from it $n^l \times n^l$ matrices $\tilde{V}_{\text{bl}} = (\tilde{E}[\nu_{\bar{b}_j \bar{a}_i} \circ Y_0^{2l}])_{i,j=1,\dots,n^l}$ and $W_{a,\text{bl}} = (\tilde{E}[\nu_{\bar{b}_j a \bar{a}_i} \circ Y_0^{2l+1}])_{i,j=1,\dots,n^l}$, where $a \in E$, n is the cardinality of the set $E = \{a_1, \dots, a_n\}$ of observations of the underlying discrete OOM $\mathcal{A}$, l is an integer that must be selected properly (which is the essential task in the first step), $\bar{a}_i$ and $\bar{b}_j$ $(i, j = 1, \dots, n^l)$ each are alphabetical enumerations of E^l, and $\tilde{E}[\nu_{\bar{b}_j \bar{a}_i} \circ Y_0^{2l}]$ is an estimate of $E[\nu_{\bar{b}_j \bar{a}_i} \circ Y_0^{2l}]$ derived from S. (Subscript "$_{\text{bl}}$" stands for "blended".)

The task of selecting l has the following significance. In later steps, the observable operator matrices will be computed from (among others) $\tilde{V}_{\text{bl}}$, and the resulting model dimension will be the dimension k of $\tilde{V}_{\text{bl}}$. Therefore, if k is selected too big, the resulting estimated model $\tilde{\mathcal{A}}$ will overfit data. Conversely, if k is too small, data will be under-exploited. In the first step, the matrix $\tilde{V}_{\text{bl}}$ has some dimension $k = n^l$. While in the second major step k is fine-tuned, in the first step k is adjusted on a coarse scale by varying l. This is done by initializing l with the smallest possible value of $l = 1$, testing whether an n^1-dimensional model under-exploits data, if yes, increment l by 1, test again, etc. until data are not under-exploited.

Substep 1.1. `Initialize` $l = 1$.

Substep 1.2. `Test whether a` n^l`-dimensional model under-exploits data. If yes, increment` l `by` 1 `and repeat. If not, go to major step 2.`

[1] A self-explaining implementation of the learning procedure written in Mathematica, and its application to the probability clock example, is available at the author's homepage at www.gmd.de/People/Herbert.Jaeger/Resources.html.

We describe this step in detail. First, from S we compute the matrix $\tilde{V}_{\mathrm{bl}} = (\tilde{E}[\nu_{\bar{b}_j \bar{a}_i} \circ Y_0^{2l}])_{i,j=1,\ldots,n^l}$, by exploiting the ergodic theorem for strictly stationary processes, (aka strong law of large numbers for strictly stationary processes [3]). It states that

$$\lim_{M\to\infty} \frac{1}{M+1} \sum_{n=0}^{M} \nu_{\bar{b}_j \bar{a}_i} Y_n^{2l}(\omega) = E[\nu_{\bar{b}_j \bar{a}_i} \circ Y_0^{2l}] \tag{11}$$

with probability 1 for every $\bar{b}_j \bar{a}_i \in E^{2l}$. Therefore, we obtain asymptotically correct estimates of $E[\nu_{\bar{b}_j \bar{a}_i} \circ Y_0^{2l}]$ from S by putting $\tilde{E}[\nu_{\bar{b}_j \bar{a}_i} \circ Y_0^{2l}] = \frac{1}{N-2l+1} \sum_{i=0}^{N-2l} \nu_{b_j^1}(x_i) \ldots \nu_{b_j^l}(x_{i+l-1}) \nu_{a_i^1}(x_{i+l}) \ldots \nu_{a_i^l}(x_{i+2l-1})$, where $\bar{b}_j = b_j^1 \ldots b_j^l$, $\bar{a}_i = a_i^1 \ldots a_i^l$, and x_i is the i-th observation in S.

We describe in more detail the construction of $\tilde{V}_{\mathrm{bl}}$. Actually, instead of the matrix $\tilde{V}_{\mathrm{bl}}$ we construct a vector $\tilde{v}_{\mathrm{bl}} = (\tilde{E}[\nu_{\bar{d}_1} \circ Y_0^k], \ldots, \tilde{E}[\nu_{\bar{d}_{n^k}} \circ Y_0^k])$, where $k = 2l$ and $\bar{d}_1, \ldots, \bar{d}_{n^k}$ is the alphabetical enumeration of E^{2l}. The vector $\tilde{v}_{\mathrm{bl}}$ contains the same entries as the matrix, and will be used in later steps.

Let ToDigits be a function that assigns to the numbers $1, \ldots, n^k$ the length-$2l$ vectors of the n-adic digits (where digit 0 is replaced by n), i.e. the vectors $(1, \ldots, 1, 1), (1, \ldots, 1, 2), \ldots, (n, \ldots, n)$, and let subscript $v[i]$ denote the i-th component of a vector v. Using this auxiliary function, compute $\tilde{v}_{\mathrm{bl}}$ from S with the following procedure:

```
$\tilde{v}_{\mathrm{bl}} = (0, \ldots, 0)$; /* initialization; vector length $n^k$ */
For $p = 1, \ldots, N-2l+1$ Do
    For $q = 1, \ldots, n^k$ Do
        $(\mathrm{obs}_1, \ldots, \mathrm{obs}_{2l}) = (x_p, \ldots, x_{p+2l-1})$;
        $(\mathrm{digit}_1, \ldots, \mathrm{digit}_{2l}) = (\mathrm{ToDigits}(q)[1], \ldots, \mathrm{ToDigits}(q)[2l])$;
        $\tilde{v}_{\mathrm{bl}[q]} = \tilde{v}_{\mathrm{bl}[q]} + \nu_{a_{\mathrm{digit}_1}}(\mathrm{obs}_1) \cdot \ldots \cdot \nu_{a_{\mathrm{digit}_{2l}}}(\mathrm{obs}_{2l})$;
$\tilde{v}_{\mathrm{bl}} = (N-2l+1)^{-1} \cdot \tilde{v}_{\mathrm{bl}}$;
```

The time complexity of this construction is $O(Nkn^k)$ for a straightforward execution of the described procedure. Note however that if the absolute increments $|x_{i+1} - x_i|$ have a small bound and if the membership functions overlap only with neighboring membership functions (i.e., the transfer matrix is a band matrix) – both conditions will often be satisfied –, then a majority of the entries of $\tilde{v}_{\mathrm{bl}}$ can be put to zero right away.

The entries of $\tilde{v}_{\mathrm{bl}}$ are rearranged into the matrix $\tilde{V}_{\mathrm{bl}}$.

We now explain how the matrix $\tilde{V}_{\mathrm{bl}}$ can be used to decide whether the final model estimate under-exploits data.

In step 2.2, a $k \times k$ invertible matrix $\tilde{V}$ will be obtained from $\tilde{V}_{\text{bl}}$ by a (nonlinear) transformation which never increases (and typically preserves) rank. $\tilde{V}$ is a main input for the learning procedure for discrete OOMs, which will be carried out in major step 3. The dimension k of $\tilde{V}$ then becomes the dimension of the process described by the estimated model. Matrices $\tilde{V}$ can be constructed in different dimensions k. The greater k, the greater detail in the training data can be captured by model estimates. Thus, k must be selected just right: if it is too small, data are under-exploited; if too big, the resulting models will overfit data.

Overfitting and under-exploitation are both connected with the *numerical rank* of $\tilde{V}$. The numerical rank of a matrix is smaller or equal to its usual rank; intuitively, it is the rank that is justified under the assumption that the matrix is a noisy sample. Technically, the numerical rank r of a matrix is the number of its singular values that are *significantly* greater than zero.

When one constructs a sequence of matrices $\tilde{V}_k$ of increasing size $k \times k$ ($k = 1, 2, \ldots$), one will find that for small k the numerical rank r of $\tilde{V}_k$ is equal to k (i.e., $\tilde{V}_k$ is numerically full-rank). For $k > k_0$ however, one will find that the numerical rank of $\tilde{V}_k$ is smaller than k, i.e., $\tilde{V}_k$ is numerically rank-deficient. The right dimension k is obtained when $\tilde{V}_k$ is numerically full rank (this means that data are not overfitted), and $\tilde{V}_{k+1}$ is numerically rank-deficient (which means that $\tilde{V}_k$ does not under-exploit data).

Deciding the numerical rank of a noisy matrix is a standard task in numerical linear algebra, and a number of techniques are available (see [4] for algorithms). In [7] a particularly simple one is applied to the matrix $\tilde{V}$ in a task of estimating discrete-valued OOMs.

In the present case, however, it would be incorrect to test $\tilde{V}$ for its numerical rank, since it is only indirectly connected to the training data. These are directly mirrored in $\tilde{V}_{\text{bl}}$ instead. Since $\tilde{V}$ is connected to $\tilde{V}_{\text{bl}}$ by a non-rank-increasing transformation, the justifiable rank of $\tilde{V}$ (i.e., the number of singular values which are significantly greater than zero, given data) is bounded by the numerical rank of $\tilde{V}_{\text{bl}}$ (typically, this bound is strict since the transformation from $\tilde{V}_{\text{bl}}$ to $\tilde{V}$ is typically rank-preserving). Therefore, we have to test whether $\tilde{V}_{\text{bl}}$ has full numerical rank.

In order to check whether $\tilde{V}_{\text{bl}}$ has full numerical rank, we apply the test criterium introduced in [7], which we repeat here for conveniance. First the matrix $\tilde{V}_{\text{bl}}$ is normalized such that the sum of its entries becomes N (the size of training data). Then, its singular values $s_1, \ldots, s_k$ are computed (in descending order). We accept the hypothesis that $s_k > 0$ if $s_k > \epsilon \parallel \tilde{V}_{\text{bl}} \parallel_\infty$, where the ∞-norm of a matrix (ζ_{ij}) is $\max_i \sum_j \mid \zeta_{ij} \mid$. The parameter ϵ is equal to $1/d^2 \sum_{i,j} \alpha_{ij}$, where

$$\alpha_{i,j} = \begin{cases} \sqrt{\tilde{V}_{\text{bl}}(i,j)(1-\tilde{V}_{\text{bl}}(i,j)/N)}/\tilde{V}_{\text{bl}}(i,j), & \text{if } \tilde{V}_{\text{bl}}(i,j) > 0 \\ 0 & \text{if } \tilde{V}_{\text{bl}}(i,j) \leq 0 \end{cases} \tag{12}$$

In our probclock learning example, the singular values of the $k = n^1 = 2$-dimensional matrix $\tilde{V}_{\text{bl}}$ are approximately $(1000270, 17389)$, and the cutoff criterium $\epsilon \parallel \tilde{V}_{\text{bl}} \parallel_\infty$ is 1245. Therefore, for $l = 1$ we would obtain a 2-dimensional model which certainly would not overfit data, but possibly might be under-exploiting.

In order to exclude this possibility, we increment to $l = 2$. We then obtain singular values of the 4-dimensional matrix $\tilde{V}_{\text{bl}}$ of $(500421, 8828, 2441, 0.23)$ with a cutoff of 1394. This means that $\tilde{V}_{\text{bl}}$ has numerical rank 3, therefore is numerically rank deficient, and a 4-dimensional model would overfit data. Thus, we fix l at a value of 2 and proceed to the second major step.

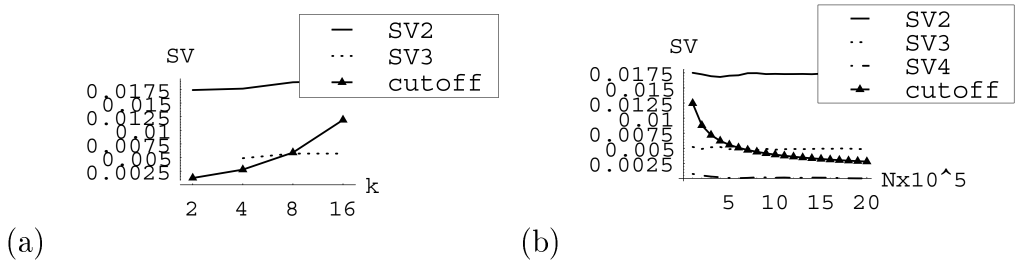

Figure 5: Determining an appropriate model dimension. For explanation see text. (a) Second and third largest singular values and cutoff obtained for matrix dimensions $k = 2, 4, 8, 16$ (i.e., $l = 1, 2, 3, 4$) for the full length run (2 M steps). SVs and cutoff are normalized such that the largest SV becomes 1 (not shown). (b) Normalized singular values for $k = 4$, calculated from matrices $\tilde{V}_{\text{bl}}$ obtained from initial sequences of increasing length. – Note that after normalization, singular values maintain approximately constant mutual ratios in (a) and (b); however, the cutoff grows with l (in (a)) and falls with N (in (b)).

Fig. 5 (a) depicts the second and third singular values of $k = n^l$-dimensional matrix $\tilde{V}_{\text{bl}}$ for $l = 1, 2, 3, 4$. For $l = 1 (k = 2)$, two SV's are obtained, of which only the second is shown. It is far above the cutoff, which indicates that a 2-dimensional model would be warranted by data by a far margin. For $k = 4$ we find that a 3-dimensional model is justified by data, although by just a small margin. For $k = 8$ and $k = 16$, the cutoff criterium justifies only

2-dimensional models. The reason is that the information from S becomes dispersed over many more matrix entries as k grows; the "noise to signal" ratio of the individual matrix entries grows worse; accordingly, the numerical rank deteriorates.

Fig. 5 (b) illustrates the development of singular values of $\tilde{V}_{\text{bl}}$ (case $l = 2$) and the cutoff, when the learning procedure is applied to teaching data of increasing size N. Note the the original process is 3-dimensional, which implies that the 4th singular value in the true matrix $V_{\text{bl}} = (E[\nu_{\bar{b}_j \bar{a}_i} \circ Y_0^{2l}])_{i,j=1,\ldots,n^l}$ is zero. Accordingly the 4th SV goes to zero in the Figure as N grows. Because the cutoff crosses the 3rd singular value at about $N = 700,000$, a model dimension of 3 would be justified by data of that or greater size.

4.2 Second major step: fine-tuning the model dimension

We enter this step with a model dimension $k = n^l$ which would lead to an overfitting model. The dimension is now decreased by steps of -1 until a numerically full rank matrix $\tilde{V}_{\text{bl}}$ is obtained, which leads to a model that is neither overfitting nor under-exploiting.

Substep 2.1. `Compute the transfer matrix` T.

Since the membership functions ν_{a_i} are givens, computing the transfer matrix $T = (\langle \nu_{a_i}, \nu_{a_j} \rangle)_{i,j=1,\ldots,n}$ is straightforward. The time complexity depends on various contingent factors (e.g., whether the integrals have an explicit solution or must be approximated numerically; in the latter case, desired precision; how many entries can be simply put to zero because the membership functions don't overlap). Be this as it may, this step will typically be much cheaper than the previous substeps.

Substep 2.2. `Compute` n^l`-dimensional matrices` $\tilde{V}$ `and` $\tilde{W}_a$.

These matrices are defined as $\tilde{V} = (\tilde{P}(\bar{b}_j \bar{a}_i))_{i,j=n^l}$ and (for every $a \in E$) $\tilde{W}_a = (\tilde{P}(\bar{b}_j a \bar{a}_i))_{i,j=1,\ldots,n^l}$, where $\bar{b}_j$ and $\bar{a}_i$ $(i, j = 1, \ldots, n^l)$ each are alphabetic enumerations of E^l, and $\tilde{P}(\bar{b}_j \bar{a}_i), \tilde{P}(\bar{b}_j a \bar{a}_i)$ are estimates of the probabilities $P(\bar{b}_j \bar{a}_i), P(\bar{b}_j a \bar{a}_i)$ of sequences $\bar{b}_j \bar{a}_i, \bar{b}_j a \bar{a}_i$ in the process X generated by the discrete OOM $\mathcal{A}$.

The entries $\tilde{P}(\bar{b}_j \bar{a}_i)$ of $\tilde{V}$ can be obtained from the vector $\tilde{v}_{\text{bl}}$ by an application of Eq. (8). Observe that the transfer function T is invertible according to Proposition 3. Thus, one can construct T^{-1}. Eq. (8) demands that we apply $(T \otimes \cdots \otimes T)^{-1}$ ($2l$ factors) to $\tilde{v}_{\mathrm{bl}}$. It holds that $(T \otimes \cdots \otimes T)^{-1} = T^{-1} \otimes \cdots \otimes T^{-1}$. The calculation of $T^{-1} \otimes \cdots \otimes T^{-1}(\tilde{v}_{\mathrm{bl}})$ can be done incrementally (without explicitly calculating $T^{-1} \otimes \cdots \otimes T^{-1}$) with a complexity of $O(kn^{k+1})$ (where $k = 2l$).

The entries $\tilde{P}(\bar{b}_j a \bar{a}_i)$ of all $\tilde{W}_a$ ($a \in E$) can be obtained in a similar way from the length $k' = n^{2l+1}$ vector $\tilde{w}_{\mathrm{bl}} = (\tilde{E}[\nu_{\bar{d}_1} \circ Y_0^{k'}], \ldots, \tilde{E}[\nu_{\bar{d}_{n^{k'}}} \circ Y_0^{k'}])$, where $\bar{d}_1, \ldots, \bar{d}_{n^{k'}}$ is the alphabetical enumeration of $E^{k'}$. The vector $\tilde{w}_{\mathrm{bl}}$ is calculated from S in a similar way as was $\tilde{v}_{\mathrm{bl}}$ (step 1.2) with a cost of $O(Nk'n^{k'})$. The cost incurred by the subsequent application of $T^{-1} \otimes \cdots \otimes T^{-1}$ is $O(k'n^{k'+1})$.

We conclude the description of this substep by explaining why the rank of $\tilde{V}$ is less or equal to the rank of $\tilde{V}_{\mathrm{bl}}$ (here we refer to the "mathematical" matrix rank, not the numerical rank). This can be seen as follows. Since $T \otimes \cdots \otimes T$ has $2l$ factors, it holds that $T \otimes \cdots \otimes T = T' \otimes T'$, where $T' = T \otimes \cdots \otimes T$ (l factors). Let $k = n^l$. Let U be the linear subspace in $\mathbb{R}^k$ spanned by the column vectors $v_1, \ldots, v_k$ of $\tilde{V}_{\mathrm{bl}}$. T' is invertible because T is. Therefore, the subspace $T'U$ spanned by the vectors $T'v_i$ has the same dimension as U. For $i = 1, \ldots, k$ consider the vector $u_i = \sum_{j=1}^{i} T'(i,j) T' v_i$. Obviously $u_i \in T'U$. But u_i is just the i-th column vector of $\tilde{V}$. Thus, $\mathrm{rank}(\tilde{V}) \leq \mathrm{rank}(\tilde{V}_{\mathrm{bl}})$. Furthermore, a closer inspection of this line of argument reveals that the two ranks will be equal except for carefully designed special cases.

Substep 2.3. `Reduce dimension of` $\tilde{V}_{\mathrm{bl}}$ `by` 1 `such that the numerical rank is optimized.`

In this step two rows and two columns in $\tilde{V}_{\mathrm{bl}}$ (and $\tilde{V}$, respectively) are joined, thus reducing the matrix dimension.

Rows and columns should be selected such that after joining them, the resulting matrix $\tilde{V}'_{\mathrm{bl}}$ is optimally conditioned in the sense that if we tested it again like in substep 1.2, the numerical rank should be as big as possible. One way to achieve this optimality would be to test out all possible combinations. However, this is computationally expensive. A principled yet cheap method for optimal selection is not known.

Currently we use a greedy local optimization that with each row (or column) joining step maximizes a certain volume. Maximizing this volume is connected to an optimal conditioning of numerical rank in an intuitively plausible way.

We first explain the case of joining rows. We try out every possible combination i, j of two rows, compute for each such combination a certain volume $\mathrm{v}ol(i,j)$, and finally select those two rows where this volume was found maximal. Concretely, this is done as follows. Let $\tilde{v}^i_{\text{bl}}, \tilde{v}^j_{\text{bl}}$ in $\tilde{V}_{\text{bl}}$ be a pair of rows in the $k \times k$ matrix $\tilde{V}_{\text{bl}}$. We first join them according to the joining operation described further below, obtaining a $(k-1) \times k$ matrix $\tilde{V}_{\text{rowJoined}}$. We also join the corresponding rows in $\tilde{V}$ to obtain $\tilde{V}_{\text{rowJoined}}$; here we use simple addition as a joining operation. For each row vector of $\tilde{V}_{\text{rowJoined}}$ we compute the component sum, thus obtaining $k-1$ weights $\gamma_1, \ldots, \gamma_{k-1}$. We then multiply the $k-1$ row vectors of $\tilde{V}_{\text{rowJoined}}$ with these weights, obtaining $k-1$ weighted vectors. The desired volume v$ol(i,j)$ is the $k-1$-dimensional volume of the polyhedron spanned by these weighted vectors.

The same is done for columns.

Let i, j be the pair of row indices where this volume was found maximal, and m, n the pair of column indices.

The actual joining in $\tilde{V}_{\text{bl}}$ is not a simple addition but a pairwise weighted sum, where the weights stem from the matrix $\tilde{V}$. The theoretical justification for this operation will be explained after we have described the operation itself. We first join rows. Let $\tilde{v}^i_{\text{bl}} = (\alpha_1, \ldots, \alpha_k), \tilde{v}^j_{\text{bl}} = (\beta_1, \ldots, \beta_k), v^i = (\gamma_1, \ldots, \gamma_k), v^j = (\delta_1, \ldots, \delta_k)$, where v^i, v^j are the ith and jth rows in $\tilde{V}$. Compute a new vector $w = (\theta_1, \ldots, \theta_k)$, where $\theta_\kappa = (\gamma_\kappa \alpha_\kappa + \delta_\kappa \beta_\kappa)/(\gamma_\kappa + \delta_\kappa)$. In $\tilde{V}_{\text{bl}}$ replace $\tilde{v}^i_{\text{bl}}$ by w and delete $\tilde{v}^j_{\text{bl}}$, to obtain a $(k-1) \times k$ matrix $\tilde{V}^*_{\text{bl}}$. In $\tilde{V}$, simply add v_j to v_i and delete v_j, to obtain a $(k-1) \times k$ matrix $\tilde{V}^*$. Repeat an analog procedure for the columns m, n of $\tilde{V}^*_{\text{bl}}$ and $\tilde{V}^*$, to obtain the final $(k-1) \times (k-1)$ matrices $\tilde{V}'_{\text{bl}}$ and $\tilde{V}'$.

This joining procedure has a simple semantics, which is revealed by Proposition 5. For an explanation, assume that the first two rows and columns are joined. Before joining, the matrix $\tilde{V}$ of our running example looks like this:

$$\tilde{V} = \begin{pmatrix} \tilde{P}(aaaa) & \tilde{P}(abaa) & \tilde{P}(baaa) & \tilde{P}(bbaa) \\ \tilde{P}(aaab) & \tilde{P}(abab) & \tilde{P}(baab) & \tilde{P}(bbab) \\ \tilde{P}(aaba) & \tilde{P}(abba) & \tilde{P}(baba) & \tilde{P}(bbba) \\ \tilde{P}(aabb) & \tilde{P}(abbb) & \tilde{P}(babb) & \tilde{P}(bbbb) \end{pmatrix}. \tag{13}$$

Now assume that the first two rows and columns have been joined (i.e., added) in $\tilde{V}$. This gives

$$\tilde{V}' = \begin{pmatrix} \tilde{P}(aEaE) & \tilde{P}(baaE) & \tilde{P}(bbaE) \\ \tilde{P}(aEba) & \tilde{P}(baba) & \tilde{P}(bbba) \\ \tilde{P}(aEbb) & \tilde{P}(babb) & \tilde{P}(bbbb) \end{pmatrix}, \tag{14}$$

where $E = \{a, b\}$ is the observation of a or b. Now assume that instead of using the original 2-symbol discrete probability clock $\mathcal{A}$ to construct a blended OOM $\mathcal{C}$, we started with a discrete 9-symbol process X'^4 with symbols $E' = \{aEaE, baaE, bbaE, aEba, baba, bbba, aEbb, babb, bbbb\}$, described by an OOM $\mathcal{A}' = (\mathbb{R}^3, (\tau_e)_{e \in E'}, w_0)$, where $\tau_{aEaE} = \tau_{aaaa} + \tau_{aaab} + \tau_{abaa} + \tau_{abab}$, etc. ($\tau_{abab}$ is shorthand for $\tau_b \circ \tau_a \circ \tau_b \circ \tau_a$). Actually, X'^4 is nothing but a 4-blocked version of X, in which additionally we coarsen the observation resolution by joining certain observations. Now assume furthermore that from $\mathcal{A}'$ we construct a blended process Y'^4 via membership functions ν_e derived from the membership functions $\nu_{\bar{a}}$ of the 4-blocked version X^4 of X (as described in Prop. 4). For instance, ν_{aEba} would be equal to $(P(aaba)\nu_{aaba} + P(abba)\nu_{abba})/(P(aaba) + P(abba))$. Then, Prop. 5 tells us that the expectations $E[\nu_e \circ Y_0'^4]$ would be the same as $E[\nu_e \circ Y_0^4]$. But the entries of $\tilde{V}'_{\text{bl}}$, as constructed above, are estimates of these $E[\nu_e \circ Y_0^4]$.

Thus, in a nutshell, the adding of rows and columns in $\tilde{V}$ and the simultaneous pairwise weighted joining of corresponding rows and columns in $\tilde{V}_{\text{bl}}$ can be interpreted as a simultaneous coarsening of the description of the blocked process X^{2l} and its sibling Y^{2l}. This justifies repeated applications of the numerical rank estimation in further iterates of the second major step.

Substep 2.4 `Reduce dimension of` $\tilde{W}_{a,\text{bl}}, \tilde{W}_a$ `analogically.`

Take the row indices i, j and column indices n, m computed in the previous step, and repeat the joining procedure for every pair $\tilde{W}_{a,\text{bl}}, \tilde{W}_a$, where rows and columns in $\tilde{W}_{a,\text{bl}}$ are joined with the weighted addition method (weights taken from $\tilde{W}_a$), and rows and columns in $\tilde{W}_a$ are joined by simple addition. This results in $(k-1) \times (k-1)$ matrices $\tilde{W}'_{a,\text{bl}}, \tilde{W}'_a$.

The costs for substeps 2.3 and 2.4 are negligible compared to earlier steps.

Substep 2.5 `Test for full numerical rank.`

Put $k := k - 1; \tilde{V} := \tilde{V}'; \tilde{W}_a := \tilde{W}'; \tilde{V}_{\text{bl}} := \tilde{V}'_{\text{bl}}; \tilde{W}_{a,\text{bl}} := \tilde{W}'_{a,\text{bl}}$. Repeat the test described in substep 1.2 for the new $\tilde{V}_{\text{bl}}$. If it is found to have full numerical rank, go to major step 3; else repeat 2.4 and 2.5.

In our experience, the simple heuristics of joining rows / columns according to the maximum volume heuristics works well when the ratio between the dimension n^l of the initial matrix $\tilde{V}_{\text{bl}}$ and its numerical rank is not too big when step 2 is entered. If this is not the case, a local and greedy strategy like this may yield unsatisfactory results. Instead, one should perform some clustering analysis on the rows (columns, resp.), and join rows (columns, resp.) that fall into the same cluster. Principled strategies remain to be worked out.

In our example, the second major step was entered with 4-dimensional matrices $\tilde{V}, \tilde{W}_a, \tilde{V}_{\mathrm{bl}}, \tilde{W}_{a,\mathrm{bl}}$. After the first dimension reduction step, the cutoff criterium found the numerical rank to be 3 (singular values 667453, 10794, 3557; cutoff 1375). Rows (3,4) and columns (2,4) were joined according to the outcome of the maximum volume tests. These pairings actually are the ones that intuitively suggest themselves as optimal, since (in the terminology of discrete OOMs) this yields indicative events $\{aa\}, \{ab, bb\}, \{ba\}$ and characteristic events $\{aa\}, \{ab\}, \{ba, bb\}$, which are the only natural choices considering that the event "b" resets the probability clock (see [9] for more about this).

For a rough assessment of the quality of this joining strategy, 3-dimensional matrices $\tilde{V}, \tilde{W}_a$ were computed for all possible combinations of row / column pairings. This makes a total of 36 different outcomes. From each of these a probability clock was re-constructed (as described below in step 3) and compared to the original probability clock. This comparison was done by computing the weighted absolute prediction error ε for length 5 sequences as $\varepsilon = \sum_{\bar{a} \in E^5} P(\bar{a}) \mid P(\bar{a}) - \tilde{P}(\bar{a}) \mid$. It turned out that the estimate that resulted from the row / column selection (3,4), (2,4) was indeed a good choice: the error for this model was $\varepsilon = 0.000572$, which was the fourth best model possible (best model had error $\varepsilon = 0.000432$, worst had $\varepsilon = 0.103$). Fig. 6 shows the cumulative distribution of errors from all models.

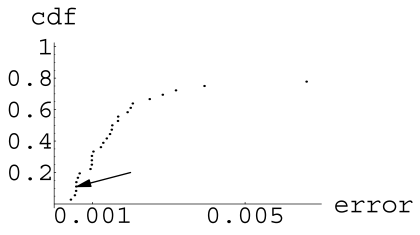

Figure 6: Cumulative distribution function of errors from the best 28 out of 36 possible 3-dimensional model estimates. Arrow marks the solution singled out by the row/column joining strategy driven by greedy local maximization of correlation. For explanation see text.

A similar assessment was also carried out for models obtained from ten smaller training samples of size $N = 200,000$ each. As in our running example, for each of these ten trials, 36 possible matrices $\tilde{V}, \tilde{W}_a$ were computed, transformed into estimates of the probability clock, and their prediction errors ε_i ($i = 1, \ldots, 36$) were computed. One of these prediction errors, ε_{i_0}, belonged to the model that was computed with the volume maximization strategy. For each of the trials, the ranking of ε_{i_0} in the (ordered) sequence of the ε_i was determined. These rankings turned out to be 3,3,1,1,1,1,1,4,3,1 (mean 1.9); i.e., in more than half of the trials, the volume maximization strategy found the best possible model.

In this assessment, the volume maximization strategy was only applied to the task of reducing matrix size from 4×4 to 3×3. The Mathematica demo implementation contains an example of reductions from 8×8 to 3×3, where this heuristics also works to satisfaction; and an example of a reduction from 16×16 to 3×3, where it yields a clearly suboptimal result.

It is clear that the strategy of major steps 1 and 2 is not the most efficient one could think of. Consider, for instance, the case of $n = 10$ membership functions and assume that the proper model dimension would be 11. Then major step 1 would require from us to construct 100×100 matrices, which would then have to be reduced again to 11×11 matrices in major step 2 by increments of -1. It would be much more effective, e.g., to have a method to *increase* matrix dimension from 10 to 11 in step 2 by *splitting* rows and columns, instead of decreasing dimension from 100 to 11 by our joining procedure. However, developing such methods remains a goal for future investigations.

4.3 Third major step: computing the discrete model estimate

The third major step uses only the matrices $\tilde{V}, \tilde{W}_a$ ($a \in E$) computed so far. This step is just the core step of the learning procedure for discrete-valued OOMs. From $\tilde{V}, \tilde{W}_a$ an estimate $\tilde{\mathcal{A}} = (\mathbb{R}^k, (\tilde{\tau}_a)_{a \in E}, \tilde{w}_0)$ of $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a \in E}, w_0)$ is obtained by putting $\tilde{\tau}_a = \tilde{W}_a \tilde{V}^{-1}$ (for all $a \in E$) and $\tilde{w}_0 = \tilde{V}\mathbf{1}^\mathsf{T}$. This step has been explained in detail elsewhere (e.g., in [9], [7]). Its time complexity is $O(nk^3)$.

The overall learning algorithm is asymptotically correct in the following sense. Let the teaching sequences S be generated by an OOM blended from a minimal-dimensional discrete OOM $\mathcal{A} = (\mathbb{R}^m, (\tau_a)_{a \in E}, w_0)$. Then, as the length N of sequences S goes to infinity, it holds that (i) the dimension k of the estimated OOM $\tilde{\mathcal{A}}$ stabilizes at m with probability 1 (see [6], Prop. 7 for the reason why) and (ii) if both $\mathcal{A}$ and its estimate $\tilde{\mathcal{A}}$ are converted into interpretable equivalents $\mathcal{A}(A_1, \ldots, A_m)$ and $\tilde{\mathcal{A}}(A_1, \ldots, A_m)$, the operator matrices that make up $\tilde{\mathcal{A}}(A_1, \ldots, A_m)$ converge to the corresponding matrices in $\mathcal{A}(A_1, \ldots, A_m)$ with probability one in any matrix norm.

Comparing the costs of all substeps, we find that the total cost is dominated by the contribution $O(Nk'n^{k'})$ from the calculation of $\tilde{w}_{\text{bl}}$ in substep 2.2, where N is size of data, $k' = 2l + 1$ is the maximal length of the data collection window used for collecting statistics from S, and n is the number of observable symbols in the discrete process. This is a worst-case complexity which is greatly reduced when many of the membership functions are non-overlapping and the sequence S features sufficiently bounded increments.

4.4 Assessing the quality of the learnt blended probability clock

We conclude this section by discussing the outcome of the learning procedure in our running example, i.e., the probability clock estimated from a blended process path of length 2,000,000.

The model $\tilde{\mathcal{A}}$ estimated from the full length sample S was transformed into an interpretable version $\tilde{\mathcal{A}}(A_1, A_2, A_3) = \mathcal{A}(\{aa\}, \{ab\}, \{ba, bb\})$, which allowed a direct comparison with the original, interpretable probability clock $\mathcal{A}(A_1, A_2, A_3)$. It turned out that the matrix parameters were estimated with an average absolute relative error of 2.20 %. This rather notable inaccuracy (given the enormous size of training sample) results from the fact that this particular blended process was almost white noise and contained only thinly diluted information about the underlying probability clock.

Fig. 9(a) shows the error made when $\tilde{\mathcal{A}}$ is put to the task of calculating the probabilities of sequences of length $N = 10, 20, 30, 40, 50$. For each N, 500 sequences $S_{N,i}$ were generated by the original probability clock, and their true probabilities $P(S_{N,i})$ and estimated probabilities $\tilde{P}(S_{N,i})$ according to the learnt model were calculated. The diagram shows the mean absolute percentual error $e_N = 1/500 \sum_{i=1}^{500} 100 \cdot \text{abs}(P(S_{N,i}) - \tilde{P}(S_{N,i})/P(S_{N,i}))$. Probabilities of length 50 sequences were miscalculated, on average, by 22 per cent. Given the long-range effects of the probability oscillation, this is quite satisfactory.

An inspection of the probability oscillation according to the learnt model $\tilde{\mathcal{A}}(A_1, A_2, A_3)$ reveals the major source of estimation error (Fig. 8(a)). It is a faulty estimation of probabilities of long uninterrupted sequences of a-s (any occurence of a b would "reset" the learnt model properly). While the oscillatory nature of these probabilities has been captured in principle, for sequences longer than about 15 consecutive a-s, the estimated oscillation gets out of hand. Even negative "probabilities" are then claimed by the model, which thereby reveals itself as being not a valid OOM (i.e., the nonnegativity

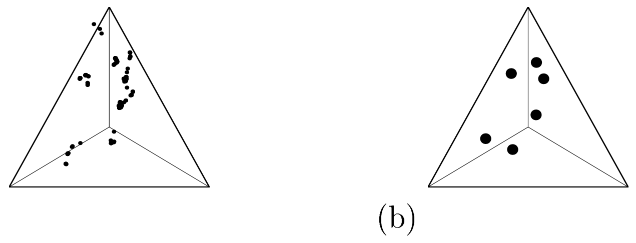
Figure 7: (a) A fingerprint of the estimated model. States from a 1000 step run are plotted. (b) For comparison, the 6 most frequently visited states of the original probability clock.

condition in the definition of OOMs [9] is violated). An equivalent observation is that the fingerprint (Fig. 7(a)) features a state that falls out of the admissible triangular region.

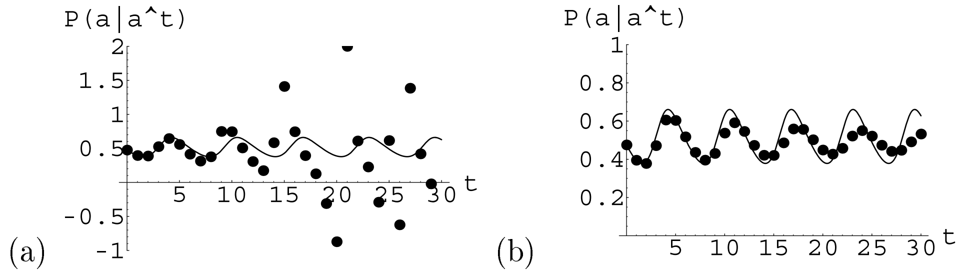

Figure 8: (a) The probability oscillation according to the estimated model (dots) vs. the original one (solid line). (b) Same as (a), but for a model estimated from another training sample ($N = 400,000$).

This disturbing finding needs an explanation. The operator τ_a in the original probability clock is a pure rotation of $\mathbb{R}^3$. In terms of dynamical systems theory, a rotation mapping is not structurally stable: any disturbance will change it qualitatively. Either it will become a mapping that has a single global attractor – or a repellor. Estimating this rotation from data can be seen as a disturbance. In cases where this disturbance leads to a repellor, the estimated probability clock features an "undamped" probability oscillation like in Fig. 8(a). In cases where an attractor is obtained, the probability oscillation would be damped and die out eventually, as in Fig. 8(b).

The possibility to obtain models that are no valid OOMs is a major nuisance for theory. Fortunately, for practical purposes there exists a simple workaround. First transform the learnt model into an interpretable one (as we did here to get $\tilde{\mathcal{A}}(A_1, A_2, A_3)$). It has the property that all state vectors $w_{\bar{a}}$ that occur in the course of probability calculations must be non-negative. If the learnt model is not a valid OOM, negative vector components will (rarely) arise. If they are obtained, one puts them to zero (and renormalize the vector to a unit component sum) and continues with the calculation.

Clearly what would be needed here is a method to "renormalize" an invalid (pseudo-) OOM to the nearest valid one. However, such a method is not known.

We proceed with an inspection of the prediction accuracy for the process Y achieved by the learnt model. For simplicity, we divide the interval $[0, 1]$ into two intervals $I_1 = [0, .5]$ and $I_2 = [.5, 1.0]$, and consider the resulting symbolic process Z with symbols I_1, I_2 (i.e., $Z_t = I_1$ iff $Y_t \in I_1$). From the original discrete probability clock $\mathcal{A}$ we compute operators τ_{I_1}, τ_{I_2} as specified in Eq. (6). Similarly, we compute operators $\tilde{\tau}_{I_1}, \tilde{\tau}_{I_2}$ from the estimated probability clock $\tilde{\mathcal{A}}(A_1, A_2, A_3)$. Thus, we obtain a two-symbol OOM $\mathcal{D} = (\mathbb{R}^3, \{\tau_{I_1}, \tau_{I_2}\}, w_0)$, which describes Z, plus its estimate $\tilde{\mathcal{D}} = (\mathbb{R}^3, \{\tilde{\tau_{I_1}}, \tilde{\tau}_{I_2}\}, w_0)$. Mean prediction errors e_N were computed as before for the discrete probability clock. Fig. 9(b) shows the results. It turns out the true probabilities and their estimates differ, on average, only for approximately 1.7% even for length 50 sequences.

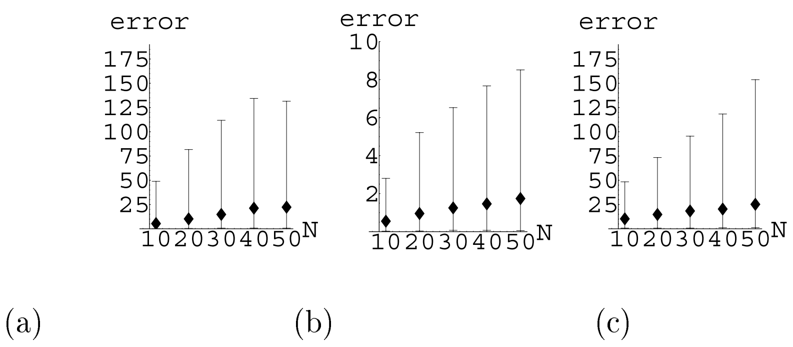

Figure 9: Average percentual error when model estimates were put to calculate probabilities of length k sequences. (a) Errors made by the estimate $\tilde{\mathcal{A}}$ of the discrete probability clock. (b) Errors for the two-interval process described by $\mathcal{D}$. (c) Errors obtained when the two-interval process was approximated as an i.i.d. process. Refer to text for details. Error bars indicate intervals into which 95 % of obtained single-sequence errors fall. Note different scalings.

One might suspect that this high precision of probability estimates is due to the almost white noise nature of the process, a circumstance that facilitates probability estimation. To check this possibility, the probabilities of test sequences were estimated with a pure white noise model by $\tilde{P}(S_{N,i}) = P(I_1)^\alpha P(I_2)^\beta$, where α, β are the number of occurences of I_1, I_2 in $S_{N,i}$. Fig. 9(c) shows the errors that result from this estimation strategy. For length 50 sequences, they are about 25 %. On the one hand, this highlights again how close this process comes to white noise, but on the other hand, it also demonstrates how much is gained in precision by the OOM estimation.

Out of curiosity, we cast a brief look on the fingerprints of the two-interval-OOM $\mathcal{D}$ and its estimate $\tilde{\mathcal{D}}$. Both OOMs were transformed into interpretable versions $\mathcal{D}(A_1, A_2, A_3)$ and $\tilde{\mathcal{D}}(A_1, A_2, A_3)$, where $A_1 = \{I_1 I_1\}, A_2 = \{I_1 I_2\}, A_1 = \{I_2 I_1, I_2 I_2\}$. Fig. 10(a) shows the fingerprint obtained from a 1000-step run of $\mathcal{D}(A_1, A_2, A_3)$, of which Fig. 10(b) is a closeup. Finally, Fig. 10(c) is the equivalent closeup for $\tilde{\mathcal{D}}(A_1, A_2, A_3)$. Two things are worth mentioning. First, the covered region in state space is very small, which means that regardless of prior history the conditional future distributions are all very similar. Second, while they occupy little space, the state plots are nonetheless highly structured; in fact, they have inherited essential features of the probclock's dynamics. This means that, although minuscule in quantity, the qualitative phenomenology of this process is rich; specifically, there are strong long-range memory effects. A comparison of Fig. 10(b) and (c) illustrates how well this qualitative phenomenology has been captured by the learning procedure, and explains the high estimation precision reported above. From the experience of this example, one is tempted to call the blended OOM learning procedure a “noise microscope”!

5 Discussion

One way to view blended OOMs is as a twofold generalization of semicontinuous HMMs, in that (i) mixtures of arbitrary emission densities are used instead of mixtures of Gaussians; and in that (ii) the underlying discrete model is an OOM instead of a HMM. Blended OOMs can be learnt from data with a constructive, asymptotically correct estimation procedure, which additionally offers the convenience of a automatic overfitting vs. underexploitation negotiation.

Although these facts are promising, one must bear in mind that OOM theory is young and was not yet put to real-world applications. Furthermore, two important issues remain to be worked out for the discrete OOM learning algorithm. The first is a principled treatment of the fine-tuning step within the learning algorithm. The goal is to join those rows and columns that

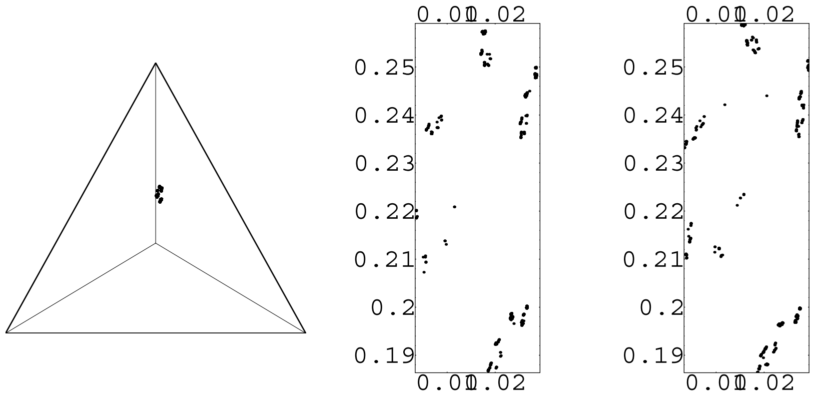

Figure 10: (a) State plot for $\mathcal{D}(A_1, A_2, A_3)$. (b) Closeup of (a). (c) Similar closeup for $\tilde{\mathcal{D}}(A_1, A_2, A_3)$. See text for details.

render the resulting matrices maximally informative about the underlying process. This is highly relevant in practice, specifically if one wishes to achieve a fully automated learning. Currently, only heuristic methods are available. The second open issue is that the learning algorithm may return invalid OOMs. This is not too worrisome for applications, since the prediction powers of the (possibly) invalid OOMs appear to be unimpaired. But from a theoretical point of view this is very unsatisfactory, and current work is devoted to solving this problem.

6 Appendix: Proofs

6.1 Proof of Proposition 1

We first show that the function ν_t obtained at time step t (where $t = 0, 1, 2, \ldots$) is a probability density. It suffices to demonstrate that $(\mathbf{1}\,\tau_{a_1} w_t, \ldots, \mathbf{1}\,\tau_{a_n} w_t)$ is a probability vector. The components of this vector sum to unity since obviously $\mathbf{1} w_t = 1$ and since in $\mathcal{A}$ it holds that $\mathbf{1}(\sum_{a \in E} \tau_a) w_t = \mathbf{1} w_t$. Thus, we must show that $\mathbf{1}\,\tau_a w_t \geq 0$. According to Proposition 6 from [9], there exists a convex cone $K \subset \mathbb{R}^m$, such that (i) $\mathbf{1}\,v \geq 0$ for all $v \in K$, (ii) $w_0 \in K$, (iii) $\tau_a K \subseteq K$ for all $a \in E$. We show that (a) $w_t \in K$ and that (b) $\mathbf{1}\,\tau_a w_t \geq 0$ for all t, by induction on t. (a) is true for $t = 0$ by virtue of (ii), and (b) follows from (i) and (iii). Assume $w_{t-1} \in K$ and $\mathbf{1}\,\tau_a w_{t-1} \geq 0$ for all a. This implies that the prescription in the Proposition to select x_{t-1} is well-defined. Exploiting (iii) again, and the fact that according to (5) $\tau_{x_{t-1}}$

is a linear combination from $\tau_{a_1}, \ldots, \tau_{a_n}$ with non-negative coefficients, by an argument of convexity it follows that $\tau_{x_{t-1}} w_{t-1} \in K$. Therefore, $w_t \in K$. (b) follows immediately using (iii).

Having shown that every ν_t is a probability density, it is clear that the generation procedure is well-defined and implicitly defines a process $(\Omega, \mathcal{F}, P, (Y_t)_{t\in\mathbb{N}})$ with values in I. We now show that the distribution of this process can be calculated with (4). Let $\nu_{x_0 \ldots x_{k-1}}$ be the probability distribution obtained in the above generation procedure at time k after a path $x_0 \ldots x_{k-1}$ has been generated (where $k \geq 1$), and let ν_0 be as in Proposition 1. It holds that

$$\begin{aligned} P(Y_0 \in J_0, \ldots, Y_k \in J_k) &= \\ &= \int_{J_0} \nu_0(x_0) \int_{J_1} \nu_{x_0}(x_1) \cdots \int_{J_k} \nu_{x_0 \ldots x_{k-1}}(x_k) \; d\mu(x_k) \ldots d\mu(x_0), \quad (15) \end{aligned}$$

where μ is the Borel measure on I. We construct an explicit representation of $\nu_{x_0 \ldots x_{k-1}}$. Let $w_{x_0 \ldots x_k} = \tau_{x_k} \cdots \tau_{x_0} w_0 / (\mathbf{1} \tau_{x_k} \cdots \tau_{x_0} w_0)$ be the state vector obtained after generating $x_0 \ldots x_k$ (where $k \geq 0$), and let w_0 be as in Proposition 1. We first isolate a term $\tau_{x_{k-1}} w_{x_0 \ldots x_{k-2}}$ in $\nu_{x_0 \ldots x_{k-1}}$:

$$\begin{aligned} \nu_{x_0 \ldots x_{k-1}}(x_k) &= \\ &= \sum_{a\in E} (\mathbf{1} \tau_a w_{x_0 \ldots x_{k-1}}) \; \nu_a(x_k) \\ &= \frac{1}{\mathbf{1} \tau_{x_{k-1}} w_{x_0 \ldots x_{k-2}}} \sum_{a\in E} \mathbf{1} \tau_a \tau_{x_{k-1}} w_{x_0 \ldots x_{k-2}} \; \nu_a(x_k) \\ &= \frac{1}{\mathbf{1} \sum_{a\in E} \nu_a(x_{k-1}) \tau_a w_{x_0 \ldots x_{k-2}}} \sum_{a\in E} \mathbf{1} \tau_a \tau_{x_{k-1}} w_{x_0 \ldots x_{k-2}} \; \nu_a(x_k) \\ &= \frac{1}{\sum_{a\in E} \nu_a(x_{k-1}) \mathbf{1} \tau_a w_{x_0 \ldots x_{k-2}}} \sum_{a\in E} \mathbf{1} \tau_a \tau_{x_{k-1}} w_{x_0 \ldots x_{k-2}} \; \nu_a(x_k) \\ &= \frac{1}{\nu_{x_0 \ldots x_{k-2}}(x_{k-1})} \; \mathbf{1} \; (\sum_{a\in E} \nu_a(x_k) \tau_a) \circ \tau_{x_{k-1}} w_{x_0 \ldots x_{k-2}}. \quad (16) \end{aligned}$$

The term $\tau_{x_{k-1}} w_{x_0 \ldots x_{k-2}}$ can be expressed in terms of $\tau_{x_{k-2}} w_{x_0 \ldots x_{k-3}}$:

$$\begin{aligned} \tau_{x_{k-1}} w_{x_0 \ldots x_{k-2}} &= \\ &= \frac{1}{\mathbf{1} \tau_{x_{k-2}} (w_{x_0 \ldots x_{k-3}})} \; \tau_{x_{k-1}} \tau_{x_{k-2}} (w_{x_0 \ldots x_{k-3}}) \end{aligned}$$

$$\begin{aligned} &= \frac{1}{\nu_{x_0 \ldots x_{k-3}}(x_{k-2})}\, \tau_{x_{k-1}} \tau_{x_{k-2}} (w_{x_0 \ldots x_{k-3}}) \\ &= \frac{1}{\nu_{x_0 \ldots x_{k-3}}(x_{k-2})}\, (\sum_{a \in E} \nu_a(x_{k-1}) \tau_a) \circ \tau_{x_{k-2}} (w_{x_0 \ldots x_{k-3}}). \qquad (17) \end{aligned}$$

This directly leads to a recursion scheme for expressions $\tau_{x_{k-i}} w_{x_0 \ldots x_{k-(i+1)}}$. Applying this recursion to (16), observing that $\tau_{x_0} w_0 = (\sum_{a \in E} \nu_a(x_0) \tau_a) w_0$, yields

$$\begin{aligned} &\nu_{x_0 \ldots x_{k-1}}(x_k) = \\ &= \frac{1}{\nu_{x_0 \ldots x_{k-2}}(x_{k-1})} \cdot \cdots \cdot \frac{1}{\nu_0(x_0)} \cdot \\ &\quad \cdot \mathbf{1}\, (\sum_{a \in E} \nu_a(x_k) \tau_a) \circ (\sum_{a \in E} \nu_a(x_{k-1}) \tau_a) \circ \cdots \circ (\sum_{a \in E} \nu_a(x_0) \tau_a)\ w_0. \quad (18) \end{aligned}$$

Inserting (18) in (15) gives

$$\begin{aligned} &P(Y_0 \in J_0, \ldots, Y_k \in J_k) = \\ &= \int_{J_0} 1 \int_{J_1} 1 \cdots \int_{J_k} \mathbf{1}\, (\sum_{a \in E} \nu_a(x_k) \tau_a) \circ \\ &\quad \circ (\sum_{a \in E} \nu_a(x_{k-1}) \tau_a) \circ \cdots \circ (\sum_{a \in E} \nu_a(x_0) \tau_a)\ w_0\ d\mu(x_k) \ldots d\mu(x_0) \\ &= \mathbf{1} \sum_{a_0 \ldots a_k \in E^{k+1}} (\int_{J_0} \cdots \int_{J_k} (\nu_{a_k}(x_k) \tau_{a_k}) \circ \cdots \circ (\nu_{a_0}(x_0) \tau_{a_0})\ d\mu(x_k) \ldots d\mu(x_0))\ w_0 \\ &= \mathbf{1} \sum_{a_0 \ldots a_k \in E^{k+1}} (\int_{J_k} \nu_{a_k}(x_k) \tau_{a_k} d\mu(x_k) \circ \cdots \circ \int_{J_0} \nu_{a_0}(x_0) \tau_{a_0} d\mu(x_0))\ w_0 \\ &= \mathbf{1}\, (\int_{J_k} \sum_{a \in E} \nu_a(x_k) \tau_a\ d\mu(x_k) \circ \cdots \circ \int_{J_0} \sum_{a \in E} \nu_a(x_0) \tau_a\ d\mu(x_0))\ w_0 \\ &= \mathbf{1}\ \tau_{J_k} \circ \cdots \circ \tau_{J_0}\ w_0. \end{aligned}$$

Proving the statement that one obtains equivalent continuous-valued OOMs $\mathcal{C}, \mathcal{C}'$ when their discrete-valued parents $\mathcal{A}, \mathcal{A}'$ are equivalent is a routine exercise. It remains to prove the stationarity statement. Without loss of generality we may assume that $\mathcal{A}$ is minimal-dimensional. In that case, stationarity is equivalent to the fact that $\mu w_0 = w_0$, where $\mu = \sum_{a \in E} \tau_a$. Observe that $\mu = \sum_{a \in E} \tau_a \int_I \nu_a(x) dx = \int_I \sum_{a \in E} \nu_a(x) \tau_a dx = \int_I \tau_x dx = \tau_I$. Thus, w_0 is an invariant vector of τ_I. By Proposition 9 from [8] it follows that the process described by $\mathcal{C}$ is stationary.

6.2 Proof of Proposition 2

First we render precise point (i) from the proposition. Let M be the $m \times m$ Markov transition matrix of $\mathcal{H}$, and let $\mu = M^\top$ be its transpose. For $a \in E$ let o_a be the $m \times m$ diagonal matrix with entry $P(a \mid s_i)$ at place (i,i). Let $w_0 \in \mathbb{R}^m$ be the initial distribution of $\mathcal{H}$. Define an OOM $\mathcal{A} = (\mathbb{R}^m, (\mu \circ o_a)_{a \in E}, w_0)$. It describes the same process as $\mathcal{H}$ (for a proof see [9]).

Now we prove that the continuous-valued process generated by $\mathcal{H}$ by emission through ν_s from state s, is the same as the process generated by the OOM obtained from blending $\mathcal{A}$ w.r.t. the membership functions ν_a. It suffices to show that if $\mathcal{A}$ is taken through a state sequence $w_0, w_{x_0}, w_{x_0 x_1}, \ldots$ when an observation sequence $x_0, x_1, \ldots$ is emitted, and if $\mathcal{H}$ generates the same sequence, then the discrete state distribution vectors obtained in $\mathcal{H}$ is $w_0, w_{x_0}, w_{x_0 x_1}, \ldots$, too. We proceed by induction. Both $\mathcal{H}$ and $\mathcal{A}$ start with w_0 by definition. Assume that at time step $t \geq 1$, the state distribution vector of $\mathcal{H}$ is $w_{x_0 \ldots x_{t-1}}$, and that x_t is emitted. We calculate the new state distribution vector $w_{x_0 \ldots x_t}$ obtained in $\mathcal{H}$ (notation: subscript $v[i]$ denotes the i-th entry of a vector):

$$\begin{aligned}
w_{x_0 \ldots x_t} &= \begin{pmatrix} P(s_1 \,|\, w_{x_0 \ldots x_{t-1}}, x_t) \\ \vdots \\ P(s_m \,|\, w_{x_0 \ldots x_{t-1}}, x_t) \end{pmatrix} = \\
&= \mu \begin{pmatrix} w_{x_0 \ldots x_{t-1}}[1] \sum_{a \in E} o_a[1] \nu_a(x_t) \\ \vdots \\ w_{x_0 \ldots x_{t-1}}[m] \sum_{a \in E} o_a[m] \nu_a(x_t) \end{pmatrix} \cdot \Big(\sum_{i=1,\ldots,m} w_{x_0 \ldots x_{t-1}}[i] \sum_{a \in E} o_a[i] \nu_a(x_t) \Big)^{-1} \\
&= \mu \circ \Big(\sum_{a \in E} \nu_a(x_t) o_a \Big)(w_{x_0 \ldots x_{t-1}}) \cdot \Big(\mathbf{1} \sum_{a \in E} \nu_a(x_t) o_a(w_{x_0 \ldots x_{t-1}}) \Big)^{-1} \\
&= \frac{\sum_{a \in E} \nu_a(x_t) \tau_a(w_{x_0 \ldots x_{t-1}})}{\mathbf{1}\, \mu \sum_{a \in E} \nu_a(x_t) o_a(w_{x_0 \ldots x_{t-1}})} = \frac{\tau_{x_t}(w_{x_0 \ldots x_{t-1}})}{\mathbf{1}\, \tau_{x_t}(w_{x_0 \ldots x_{t-1}})} \\
&= w_{x_0 \ldots x_t}. \quad \square
\end{aligned}$$

6.3 Proof of Proposition 3

Points *1.* and *2.* are trivial. *3.* follows from *2.*: it is a general property of n-dimensional Hilbert spaces that n vectors $x_1, \ldots, x_n$ are linearly independent if and only if the matrix $(\langle x_i, x_j \rangle)$ is regular. To see *4.*, observe that $P(X_0 = a_i) = \mathbf{1}\, \tau_{a_i} w_0$. This implies $\sum_{i=1}^n P(X_0 = a_i)\, \nu_{a_i}(x) = \nu_0$, where ν_0 is as in Proposition 1. As a consequence we have

$$\begin{aligned}
T(P(X_0 = a_1), \ldots, P(X_0 = a_n)) &= \\
&= (\int_I \nu_{a_1}(x)\ \nu_0(x)\ dx, \ldots, \int_I \nu_{a_n}(x)\ \nu_0(x)\ dx) \\
&= (E[\nu_{a_1} \circ Y_0], \ldots, E[\nu_{a_n} \circ Y_0]). \quad \square
\end{aligned}$$

6.4 Proof of Proposition 4

Point (1) is an instance of the general fact that the product of probability measures is a probability measure. (2) and (3) follow from the fact that the mapping $\tau_{x_0 \ldots x_{k-1}}$, as defined in the Proposition, is equal to the concatenation $\tau_{x_{k-1}} \circ \cdots \circ \tau_{x_0}$ of observable operators from $\mathcal{C}$. Finally, in order to show (4), let $p(i)[l]$ be the l-th digit (from the left) in the n-adic representation of i (filled with leading zeros to length k). Then

$$\begin{aligned}
T^k &= \\
&= (\langle \nu_{a_{p(i-1)[1]+1}} \cdots \nu_{a_{p(i-1)[k]+1}},\ \nu_{a_{p(j-1)[1]+1}} \cdots \nu_{a_{p(j-1)[k]+1}} \rangle)_{i,j=1,\ldots,n^k} \\
&= \int_{I^k} \nu_{a_{p(i-1)[1]+1}} \cdots \nu_{a_{p(i-1)[k]+1}}(x_0, \ldots, x_{k-1}) \ \cdot \\
&\quad \cdot \nu_{a_{p(j-1)[1]+1}} \cdots \nu_{a_{p(j-1)[k]+1}}(x_0, \ldots, x_{k-1})\ d(x_0, \ldots, x_{k-1}) \\
&= \int_I \nu_{a_{p(i-1)[1]+1}}(x_0) \nu_{a_{p(j-1)[1]+1}}(x_0)\ dx_0 \cdot \quad \cdots \\
&\quad \cdots \quad \cdot \int_I \nu_{a_{p(i-1)[k]+1}}(x_{k-1}) \nu_{a_{p(j-1)[k]+1}}(x_{k-1})\ dx_{k-1} \\
&= (\langle \nu_{a_{p(i-1)[1]+1}}, \nu_{a_{p(j-1)[1]+1}} \rangle \cdots \langle \nu_{a_{p(i-1)[k]+1}}, \nu_{a_{p(j-1)[k]+1}} \rangle)_{i,j=1,\ldots,n^k} \\
&= T \otimes \cdots \otimes T. \quad \square
\end{aligned}$$

6.5 Proof of Prop. 5

The first "=" is an instance of Prop. 3(*4*). To see why the second "=" holds, let $\nu_0 = \sum_{a \in E} (\mathbf{1}\, \tau_a w_0) \nu_a$, (as in Prop. 1), and let $\nu_0' = \sum_{b \in E'} (\mathbf{1}\, \tau_b w_0) \nu_b$ be the analog for Y'. It is easy to see that $\nu_0 = \nu_0'$. Use this to conclude $E[\nu_{b_1} \circ Y_0'] = \int_I \nu_{b_1}(x) \nu_0'(x) dx = \int_I \nu_{b_1}(x) \nu_0(x) dx = E[\nu_{b_1} \circ Y_0]$. The other vector entries are likewise pairwise equal.

References

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

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

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

[4] G.H. Golub and Ch.F. van Loan. *Matrix Computations, Third Edition.* The Johns Hopkins University Press, 1996.

[5] H. Ito. *An algebraic study of discrete stochastic systems.* Phd thesis, Dpt. of Math. Engineering and Information Physics, Faculty of Engineering, The University of Tokyo, Bunkyo-ku, Tokyo, 1992. ftp'able from http://kuro.is.sci.toho-u.ac.jp:8080/english/D/.

[6] H. Jaeger. Observable operator models II: Interpretable models and model induction. Arbeitspapiere der GMD 1083, GMD, Sankt Augustin, 1997. http://www.gmd.de/People/ Herbert.Jaeger/Publications.html.

[7] H. Jaeger. Discrete-time, discrete-valued observable operator models: a tutorial. GMD Report 42, GMD, Sankt Augustin, 1998. http://www.gmd.de/People/ Herbert.Jaeger/Publications.html.

[8] H. Jaeger. Characterizing distributions of stochastic processes by linear operators. GMD Report 62, German National Research Center for Information Technology, 1999. http://www.gmd.de/publications/report/0062/.

[9] H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, (6), 2000. Draft version at http://www.gmd.de/People/Herbert.Jaeger/Publications.html.

[10] D.R. Upper. *Theory and algorithms for Hidden Markov models and Generalized Hidden Markov models.* Phd thesis, Univ. of California at Berkeley, 1997.