A Bound on Modeling Error in Observable Operator Models and an Associated Learning Algorithm[*]

Ming-Jie Zhao[†] Herbert Jaeger Michael Thon
{m.zhao, h.jaeger, m.thon}@jacobs-university.de

September 17, 2008

Abstract

Observable operator models (OOMs) generalize hidden Markov models (HMMs) and can be represented in a structurally similar matrix formalism. The mathematical theory of OOMs gives rise to a family of constructive, fast and asymptotically correct learning algorithms, whose statistical efficiency however depends crucially on the optimization of two auxiliary transformation matrices. This optimization task is non-trivial; indeed, even formulating computationally accessible optimality criteria is not easy. Here we derive how a bound on the modeling error of an OOM can be expressed in terms of these auxiliary matrices, which in turn yields an optimization procedure for them, which finally affords us with a complete learning algorithm, the *error controlling* algorithm. Models learnt by this algorithm have an assured error bound on their parameters. The performance of this algorithm is illuminated by comparisons with two types of HMMs trained by the expectation-maximization (EM) algorithm, with the *efficiency sharpening* algorithm, another recently found learning algorithm for OOMs, and with predictive state representations (PSRs) (Littman and Sutton, 2001) trained by methods representing the state of the art in that field.

1 Introduction

Observable operator models are very general mathematical models of stochastic processes. The key element in OOMs is to identify a sequence of observations $(a_n)_{n=1,2,\ldots}$ with a sequence of linear operators $(\tau_{a_n})_{n=1,2,\ldots}$, called the *observable operators* of

[*]The final version of the article will appear in Neural Computation.
[†]Corresponding author.

the process. Any stochastic process whose observation space admits a Polish topology can be characterized as an OOM; this includes all discrete-time or continuous-time, stationary or nonstationary processes with discrete or continuous, univariate or multivariate observation spaces. The general mathematical theory of OOMs has been established in (Jaeger, 1999); far-reaching generalizations into the theory of ergodicity and quantum dynamics have been worked out in (Schönhuth, 2006; Faigle and Schönhuth, 2006, 2007).

In a machine learning context, it is the specific subclass of discrete-time, finite-valued OOMs — in other words, models of stochastic symbol sequences — that is of particular interest. If these OOMs are finite-dimensional, they can be represented by a matrix formalism which is structurally similar to *hidden Markov model* (HMM, (Rabiner, 1989; Bengio, 1999)) matrix representations. However, while OOMs and HMMs have deceptively similar-looking matrix formalisms, there is an important difference: the matrix entries in OOMs can be negative, whereas the analog entries in HMM matrix models must be non-negative. This has major consequences: first, OOMs properly include HMMs (there exist processes that can be described by OOMs but not by HMMs, but not vice versa (Jaeger, 2000b)); second, OOM matrix entries cannot be interpreted as probabilities. This may seem a drawback at first sight, but it is the key to a general treatment of OOMs by methods of linear algebra, and leads to constructive learning algorithms which are unencumbered by the non-negativity constraint in HMM representations. The theory of finite-dimensional OOMs, their matrix representations and basic learning algorithms are explained in detail in (Jaeger et al., 2005); a self-contained summary is given in Section 2 later in this article.

The learning algorithms for OOMs — of which by now there exist quite a number — all share the same basic structure:

1. Certain elementary counting statistics of the training sequence(s) are sorted into certain *counting matrices*.
2. These counting matrices are joined with certain *conditioning matrices* into linear matrix equations, called the *learning equations*.
3. Solving these equations yields estimates of the observable operators (as matrices), and hence an estimate of an OOM.

If certain trivial normalization constraints on the conditioning matrices are observed, this procedure implements an asymptotically correct estimator, whereas the variants of EM (Dempster et al., 1977) which are used in most HMM estimators do not have this property. Furthermore, steps 1 and 3 are computationally cheap; more precisely, the first step costs $\mathcal{O}(\textit{length of training sequence})$ flops and the third $\mathcal{O}(\textit{model dimension}^3)$. Steps 1 and 3 are simple and inevitable. Step 2, however, offers much room for design alternatives and optimization strategies, and it is this step which is responsible for the statistical efficiency of the overall algorithm. Specifically, the conditioning matrices (of which there are two; let us start calling them by the names by which they will later be formally introduced: the *characterizer* C and the *indicator* Q) can be essentially freely chosen (up to obvious normalization and non-degeneracy constraints). Variation in C and Q has dual effects on the *algebraic* conditioning of the resulting learning equation, and on the *statistical* properties of the estimator implemented by the procedure. Algebraic and statistical effects of C and Q are intimately connected, but not in an easily analyzable way.

The early learning algorithms (Jaeger, 2000a,b) relied on all-too-simple heuristics for creating C and Q; the ensuing poor statistical efficiency made OOMs learnt in this way no match for EM-trained HMMs. However, starting with (Kretzschmar, 2003), the challenge of optimizing C and Q moved into the focus of investigations and became increasingly better understood. One line of research culminated in the *efficiency sharpening* (ES) algorithm (Jaeger et al., 2005), which concentrates on the statistical effects of variation in C and Q. The ES algorithm exploits an algebraic characterization of minimal variance of the estimator, and iteratively improves C and Q toward the goal of minimizing the variance of the overall learning algorithm. While the algebraic conditioning of the learning equation is not directly controlled by the ES algorithm, and thus theoretically the attempts to improve statistical efficiency could be annihilated by unwanted algebraic side-effects, in practice it turns out that in the majority of cases, as the statistical efficiency is iteratively optimized, the algebraic conditioning is likewise improving. On synthetic and real-life datasets, the ES algorithm has starkly outperformed EM-trained HMMs in test error and speed.

In the present article, we follow the other road and concentrate on the algebraic properties of the learning equation. This approach has been explored in (Kretzschmar, 2003) and (Jaeger et al., 2005) and reaches a certain degree of maturity in this article. Our main theoretical result is a theorem which relates the algebraic modeling error (a distance measure between the assumed true observable operator matrices and the estimated ones) to algebraic properties of the conditioning matrices C and Q, in the form of an upper bound (Proposition 3). This theoretical insight gives rise to a learning strategy: optimize C and Q such that this bound is minimized. We present a computationally efficient iterative method for this optimization task; this is the main algorithmical contribution of this article. We have named the new algorithm the *error controlling* (EC) algorithm. The performance of the overall learning algorithm thus obtained is checked on some synthetic systems. It turns out that the quality of the EC algorithm comparable to that of the ES algorithm in test accuracy, but it is faster and, for high-dimensional models, appears to possess, in addition, better numerical stability.

We want to emphasize that both the ES and the EC algorithm are not actually learning algorithms per se, but rather *learning algorithm optimizing algorithms*. Both algorithms iteratively optimize C and Q; but in each iteration, one could use the current C and Q, plug them into the learning equation, and thereby implement a complete, asymptotically correct learning algorithm.

The paper is organized as follows. We start with a review of the basic terminology and theory of OOMs (Section 2). Section 3 is devoted to the EC algorithm, which includes the derivation of an upper bound of estimation error; a numerical method for minimizing this upper bound; and the overall procedure of EC. Numerical studies to assess the performance of the new algorithm in comparison with existing ones are documented in Section 4. Section 5 summarizes the paper and draws some conclusions.

2 The Basics of OOM Theory

The fundamental OOM equation. Let us consider the class of discrete-time stochastic processes with values from a finite set of observables (the *alphabet*) $O = \{a^1, a^2, \ldots, a^\ell\}$. As is well known, each such process $(X_n)_{n\in\mathbb{N}}$ is uniquely determined and completely described by the family of all finite initial joint probabilities:

$$\{\Pr(X_1 = a_1, \ldots, X_n = a_n) \ : \ n \in \mathbb{N}, a_i \in O\} \, .$$

Since joint probabilities such as $\Pr(X_1 = a_1, \ldots, X_n = a_n)$ and conditional probabilities of the form $\Pr(X_{n+1} = b_1, \ldots, X_{n+k} = b_k | X_1 = a_1, \ldots, X_n = a_n)$ will occur very often in this paper, it is convenient to introduce some shorthand notations for them and the related quantities. We shall use small letters with a bar to denote finite sequences of symbols from O, e.g., $\bar{a} = a_1 \ldots a_n$. The set of all such finite sequences $\bar{a}$ will be denoted by O^*, including the *empty sequence* ϵ. For any sequences $\bar{a} = a_1 \ldots a_n$ and $\bar{b} = b_1 \ldots b_k$, we shall write $P(\bar{a})$ or $P(a_1 \ldots a_n)$ for the joint probability $\Pr(X_1 = a_1, \ldots, X_n = a_n)$; and write $P(\bar{b}|\bar{a})$ or $P(b_1 \ldots b_k | a_1 \ldots a_n)$ for the conditional probability $\Pr(X_{n+1} = b_1, \ldots, X_{n+k} = b_k | X_1 = a_1, \ldots, X_n = a_n)$. With these shorthands, we can rewrite the above family of joint probabilities as $\{P(\bar{a}) \ : \ \bar{a} \in O^*\}$, with the agreement that $P(\epsilon) = 1$.

This paper only deals with a special class of discrete-time stochastic processes over the alphabet O, namely, *linearly dependent processes* (LDPs). LDPs are a proper superclass of the processes that can be characterized by finite-dimensional HMMs. The reader is referred to (Jaeger, 2000b) and the references therein for a detailed introduction to LDPs. According to OOM theory, any LDP (over the set O) specified by the initial probabilities $\{P(\bar{a}) \ : \ \bar{a} \in O^*\}$ can be described by some OOM $(\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$, a triple of the m-dimensional Euclidean space $\mathbb{R}^m$, an O-indexed family $\{\tau_a\}_{a\in O}$ of square matrices of order m and an initial vector $\boldsymbol{w}_0 \in \mathbb{R}^m$, via the fundamental equation of OOMs:

$$P(\bar{a}) = \mathbf{1}_m^{\mathsf{T}} \tau_{a_n} \cdots \tau_{a_2} \tau_{a_1} \boldsymbol{w}_0 =: \mathbf{1}_m^{\mathsf{T}} \tau_{\bar{a}} \boldsymbol{w}_0 \, , \quad (\forall \bar{a} = a_1 a_2 \ldots a_n \in O^*) \tag{1}$$

where $\mathbf{1}_m$ denotes the m-dimensional column vector of units and $\tau_{\bar{a}}$ the *reversed* product $\tau_{a_n} \cdots \tau_{a_2} \tau_{a_1}$ for any sequence $\bar{a} = a_1 a_2 \ldots a_n$ in O^*.

Equivalent OOMs and minimal OOMs. An LDP can be modeled by different OOMs through (1). So the whole family of OOMs can be divided into a number of *equivalence classes* such that OOMs in the same class describe the same process. An OOM is said to be *minimal* if it has the least model dimension m in its equivalence class. The following proposition gives us a complete algebraic characterization of the class of minimal OOMs and *stationary* minimal OOMs (i.e., minimal OOMs for stationary LDPs).

Proposition 1 *A triple* $(\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ *is a minimal OOM of some LDP if, and only if,*

(a) *the starting vector* $\boldsymbol{w}_0$ *has components sum* 1*, i.e.,* $\mathbf{1}_m^\mathsf{T}\boldsymbol{w}_0 = 1$*;*
(b) *the sum of all* τ_a*'s has column sums* 1*, i.e.,* $\mathbf{1}_m^\mathsf{T}\sum_{a\in O}\tau_a = \mathbf{1}_m^\mathsf{T}$*;*
(c) $\mathbf{1}_m^\mathsf{T}\tau_{\bar{a}}\boldsymbol{w}_0 \geqslant 0$ *for any sequence* $\bar{a} = a_1 \dots a_n$ *in* O^**;*
(d) *the two sets of vectors* $\{\tau_{\bar{a}}\boldsymbol{w}_0 \ : \ \bar{a} \in O^*\}$ *and* $\{\tau_{\bar{a}}^\mathsf{T}\mathbf{1}_m \ : \ \bar{a} \in O^*\}$ *both span the space* $\mathbb{R}^m$*.*

Furthermore, a minimal OOM $(\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ *describes a stationary LDP if and only if*

(e) $\mu\boldsymbol{w}_0 = \boldsymbol{w}_0$*, where* μ *is the sum of all* τ_a*'s,* $\mu := \sum_{a\in O}\tau_a$*.*

The reader is referred to the discussion in Section 14.5 (pp.426–428) of (Jaeger et al., 2005) for an indirect proof of this proposition and, more importantly, for an algebraic procedure for converting a given OOM to its equivalent minimal OOM. So in the sequel we may, and do, only consider minimal OOMs.

By definition, two OOMs are equivalent if they describe the same LDP. Here a natural and basic mathematical problem is to determine whether or not two given (minimal) OOMs are equivalent. By the definition of minimal OOMs, if two OOM models are of different dimensions, they are not equivalent. We can therefore assume the two OOMs whose equivalence we wish to ascertain both are minimal and have the same dimension. For this case, the answer to the equivalence problem is given in the following *equivalence theorem.*

Proposition 2 *Two minimal OOMs* $(\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ *and* $(\mathbb{R}^m, \{\phi_a\}_{a\in O}, \boldsymbol{u}_0)$ *are equivalent if and only if there exists a nonsingular matrix* $\varrho \in \mathbb{R}^{m\times m}$*, satisfying*

(a) $\varrho\boldsymbol{w}_0 = \boldsymbol{u}_0$*;*
(b) $\varrho\tau_a\varrho^{-1} = \phi_a$ *for all* $a \in O$*;*
(c) $\mathbf{1}_m^\mathsf{T}\varrho = \mathbf{1}_m^\mathsf{T}$*, i.e., the matrix* ϱ *has column sums* 1*.*

See Section 5 of (Jaeger, 2000b) for the proof. This proposition plays a central role in the derivation of the learning equation and the basic algorithm.

The learning equation. Equation (1) allows us to compute the probability distribution $\{P(\bar{a}) \ : \ \bar{a} \in O^*\}$ of a LDP from its OOM $(\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$. Next we discuss the reverse problem, viz. reconstructing the model parameters in the matrices τ_a and in the starting vector $\boldsymbol{w}_0$ from the distribution $\{P(\bar{a}) \ : \ \bar{a} \in O^*\}$. This can be done as follows.

Assume (X_n) is a LDP with probability distribution specified, via (1), by the minimal OOM $(\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$. We select two sets of sequences from O^*, say $\{\bar{a}_1, \bar{a}_2, \dots, \bar{a}_r\}$ and $\{\bar{b}_1, \bar{b}_2, \dots, \bar{b}_r\}$, called *indicative strings* and *characteristic strings*, respectively. In general, the number of indicative strings and of characteristic strings can be different, but to simplify the notation here we assume both sets have r ($r \gg m$) members. This will not affect the generality of the discussion below. Assume furthermore these strings are chosen such that the following two matrices

$$\Pi := \begin{bmatrix} \mathbf{1}_m^\top \tau_{\bar{b}_1} \\ \vdots \\ \mathbf{1}_m^\top \tau_{\bar{b}_r} \end{bmatrix}, \qquad \Phi := \begin{bmatrix} \tau_{\bar{a}_1} \boldsymbol{w}_0 & \dots & \tau_{\bar{a}_r} \boldsymbol{w}_0 \end{bmatrix} \tag{2}$$

both are of rank m — the condition (d) from Proposition 1 makes this possible; and such that Π has column sums 1: $\mathbf{1}_r^\top \Pi = \mathbf{1}_m^\top$. We further define the $r \times r$ *probability matrices* $\underline{V}$ and $\underline{W}_a$ with their (i,j)-th entry given by

$$\underline{V} := \left[P(\bar{a}_j \bar{b}_i)\right]_{i,j=1,2,\dots,r}, \qquad \underline{W}_a := \left[P(\bar{a}_j a \bar{b}_i)\right]_{i,j=1,2,\dots,r}, \tag{3}$$

where the observable a runs over the alphabet O; and $\bar{a}_j \bar{b}_i$ ($\bar{a}_j a \bar{b}_i$) denotes the concatenation of $\bar{a}_j$ (and a) and $\bar{b}_i$. By (1) one easily verifies that the matrices defined in (2) and (3) are related by $\underline{V} = \Pi\Phi$ and $\underline{W}_a = \Pi\tau_a\Phi$. Since Π, Φ both are of full rank m, we can construct $C \in \mathbb{R}^{m\times r}$ and $Q \in \mathbb{R}^{r\times m}$, called the *characterizer* and the *indicator* respectively, such that [1]

(i) $\mathbf{1}_m^\top C = \mathbf{1}_r^\top$, i.e., C has column sums 1;
(ii) $C\Pi$ and ΦQ both are nonsingular matrices (of order m).

Now let $\varrho = C\Pi$, then $\mathbf{1}_m^\top \varrho = \mathbf{1}_m^\top$ (as $\mathbf{1}_m^\top C = \mathbf{1}_r^\top$ and $\mathbf{1}_r^\top \Pi = \mathbf{1}_m^\top$) and

$$C\underline{W}_a Q = C\Pi\tau_a\Phi Q = \varrho\tau_a\varrho^{-1} C\Pi\Phi Q = (\varrho\tau_a\varrho^{-1})(C\underline{V}Q).$$

This equation together with the equivalence theorem (Proposition 2) allows us to reconstruct the whole equivalence class of the original OOM $(\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ from the underlying distribution, via the formula

$$\tilde{\tau}_a(C,Q) := \varrho\tau_a\varrho^{-1} = (C\underline{W}_a Q)(C\underline{V}Q)^{-1} \tag{4}$$

by using different *characterizer-indicator pairs* (CIPs) (C,Q). The formula (4) is called the *learning equation.*

The initial state $\boldsymbol{w}_0$ can be reconstructed in an analogous way. Let $\boldsymbol{v}_0$ be the r-dimensional column vector with i-th component $P(\bar{b}_i)$, then $\boldsymbol{v}_0 = \Pi\boldsymbol{w}_0$. By the equivalence theorem, the initial state of the OOM with observable operators as in (4) is given by $\tilde{\boldsymbol{w}}_0(C,Q) = \varrho\boldsymbol{w}_0 = C\Pi\boldsymbol{w}_0 = C\boldsymbol{v}_0$. But for stationary processes, one should calculate the initial state from the conditions (a) and (e) of Proposition 1, i.e., by solving the linear system $\{\mathbf{1}_m^\top \boldsymbol{w}_0 = 1,\ (\mu - I_m)\boldsymbol{w}_0 = \mathbf{0}\}$ [2] , where I_m is the

[1] Our definition of characterizers is a little different from that of (Jaeger et al., 2005).

[2] This is not an overdetermined system and so has at least one solution $\boldsymbol{w}_0$ (which in general is also unique), since the matrix $\mu - I_m = \sum_{a\in O}\tau_a - I_m$ has column sums 0 (see condition (b) of Proposition 1) and hence is of rank less than m (usually $m-1$). If the solutions $\boldsymbol{w}_0$ are not unique (very rarely), one may, e.g., pick one such $\boldsymbol{w}_0$ with minimal norm.

identity matrix of order m. To simplify the problem, in the following we always assume that the process we want to model is stationary and ergodic.

We point out two facts concerning the learning equation (4).

1. In (Jaeger et al., 2005), the learning equation has a different form: $\tilde{\tau}_a = (C\underline{W}_a)(C\underline{V})^\dagger$, where $(\cdot)^\dagger$ denotes matrix pseudo-inverse. This actually is a special case of (4), for we can get the former by putting $Q = (C\underline{V})^\dagger$ in the latter.
2. In the derivation of (4), a condition on the characteristic strings $\bar{b}_i$ is that they should make the matrix Π have column sums 1. This means $\bar{b}_i$'s should "cover" some O^k, the set of sequences of length k. For instance, assume $O = \{a, b\}$, then $\{a, ba, bba, bbb\}$ covers O^3 in the sense that a covers all strings $\{a**\} = \{aaa, aab, aba, abb\}$ and ba covers $\{ba*\} = \{baa, bab\}$; thus $\{a, ba, bba, bbb\}$ is admissible as the set of characteristic strings. It is not necessary for indicative strings to have this property.

The basic learning algorithm. In practical modeling tasks, the distribution of the process is unknown. The typical situation is that we only know one (or several) finite instantiations of the target process — the training sequence(s); and are required to estimate an OOM that approximately models the process. In this article, we will restrict ourselves to the following task: learn an OOM of known dimension m from a finite sequence $\bar{s} = s_1 s_2 \dots s_l$ which is assumed to be procured by a stationary and ergodic process.

For ergodic processes, the probability $P(\bar{a})$ of a certain sequence $\bar{a}$ can be (asymptotically) approximated by the following fraction:

$$\hat{P}(\bar{a}) = \frac{\text{number of occurrences of } \bar{a} \text{ in } \bar{s} = s_1 s_2 \dots s_l}{l + 1 - \text{the length of } \bar{a}} . \tag{5}$$

So one may approximate the probability matrices $\underline{V}$ and $\underline{W}_a$'s by

$$\underline{V}^* = \left[\hat{P}(\bar{a}_j \bar{b}_i)\right]_{i,j=1,2,\dots,r} , \qquad \underline{W}_a^* = \left[\hat{P}(\bar{a}_j a \bar{b}_i)\right]_{i,j=1,2,\dots,r} , \tag{6}$$

respectively. The matrices $\underline{V}^*$ and $\underline{W}_a^*$ are called (normalized) *counting matrices*, since they are obtained by counting the number of occurrences of specific strings in the training data.

The basic algorithm can now be summarized as follows.

1. Fix the indicative/characteristic strings $\{\bar{a}_j\}_{j=1,2,\dots,r}$ and $\{\bar{b}_i\}_{i=1,2,\dots,r}$.
2. Estimate the probability matrices from the sequence $\bar{s}$ as in (5) and (6).
3. Design an appropriate CIP (C, Q) in some way.
4. Evaluate an OOM by the learning equation (4) but with all probability matrices replaced by their approximation (6), i.e.,

$$\hat{\tau}_a = (C\underline{W}_a^* Q)(C\underline{V}^* Q)^{-1} . \tag{7}$$

As is well known in statistics, the counting matrices given by (6) converge with probability 1 to the corresponding probability matrices when the length of training data tends to infinity. Thus, the basic algorithm is asymptotically correct provide that the training sequence $\bar{s}$ were indeed procured by an m-dimensional minimal OOM in the first place. This means the target model can be perfectly recovered (up to its equivalence class) almost surely in the limit of inifinite-size training data.

A note on the counting matrices. Step 1 of the basic algorithm (which determines the counting matrices) leaves much freedom. For the selection of indicative/characteristic strings, a natural brute-force way is to take the alphabetical enumeration of O^k with k sufficiently large as $\{\bar{a}_j\}_{j=1,\ldots,r}$ and $\{\bar{b}_i\}_{i=1,\ldots,r}$. This from one point of view is also the best choice, for it collects the frequencies of all possible sequences of length $2k$ (in $\underline{V}^*$) and $2k+1$ (in $\underline{W}_a^*$'s). The major disadvantage of this method is resource inefficiency: it leads to very large ($r = \ell^k$) and very sparse counting matrices, since most strings from $\{\bar{a}_j\bar{b}_i\}_{i,j=1,2,\ldots,r}$ or $\{\bar{a}_j a\bar{b}_i\}_{i,j=1,2,\ldots,r}$ will never occur in the training sequence(s). Nevertheless, this obstacle can be overcome by using a *compact suffix tree* (CST) representation of the training sequence (Gusfield, 1997).

CSTs provide a compact representation of all sub-strings of a given sequence, say $\bar{s}$. They also serve as an efficient data structure for exposing the internal structure of $\bar{s}$. Moreover, given the sequence $\bar{s}$, the CST of $\bar{s}$ can be constructed in linear time $\mathcal{O}(|\bar{s}|)$, where $|\bar{s}|$ denotes the length of $\bar{s}$ (Ukkonen, 1995). For our particular case here, CSTs enable us to exploit characteristic/indicative strings of all possible lengths simultaneously; and to construct the *compressed* counting matrices (i.e., the above mentioned large sparse counting matrices but with all zero-columns/rows removed) efficiently.

An introduction to CSTs and how they are utilized to construct the counting matrices falls outside the scope of this paper, for which the reader is referred to (Jaeger et al., 2005). Here we only make two remarks to clarify the error controlling learning procedure.

1. In the EC algorithm, CSTs are used only in the first step as an efficient tool (linear time complexity) for constructing counting matrices.
2. The counting matrices obtained by CSTs are essentially identical to those obtained by the brute-force method: one counts all strings of given length(s) in the training sequence and then discards those zero columns/rows from the resulting (extremely sparse) matrices.

3 The Error Controlling Algorithm

Now assume that we have gotten the counting matrices $\underline{V}^*$ and $\underline{W}_a^*$ (from a finite training sequence). The major problem that remains is how to design a proper CIP (C, Q) for the estimation (7). An insightful choice should embody answers to the following questions:

- What is the influence on the final model quality of the statistical errors in counting matrices caused by finite training data?
- How can we control or minimize detrimental effects of statistical fluctuations in the counting matrices?

We solve this problem in two steps: first an upper bound of the estimation error of the basic algorithm is derived, then a numerical method for minimizing this error bound is introduced.

3.1 Error Analysis of The Basic Algorithm

To investigate the influence of statistical errors in the counting matrices $\underline{V}^*$ and $\underline{W}_a^*$ on the estimated model parameters τ_a, we stack all τ_a's one above another to form the *tall* matrix $\tau = [\tau_{a^1}; \ldots; \tau_{a^\ell}]$ (in Matlab's notation). In the same way we construct three more matrices $\hat{\tau}$, $\underline{W}$ and $\underline{W}^*$ from $\hat{\tau}_a$'s, $\underline{W}_a$'s and $\underline{W}_a^*$'s, respectively. Then the learning equation (4) and its perturbed version (7) can be rewritten as

$$\tau = (C_{\text{big}}\underline{W}Q)(C\underline{V}Q)^{-1}\,, \qquad \hat{\tau} = (C_{\text{big}}\underline{W}^*Q)(C\underline{V}^*Q)^{-1}\,, \tag{8}$$

with $C_{\text{big}} := \text{diag}\{C, \ldots, C\}$ (ℓ copies of C). We further define $\underline{E}_V := \underline{V} - \underline{V}^*$ and $\underline{E}_W := \underline{W} - \underline{W}^*$ to be the error matrices of $\underline{V}$ and $\underline{W}$, respectively. We now address the question of how we can quantify the influence of the perturbations $\underline{E}_V$ and $\underline{E}_W$ on the estimation $\hat{\tau}$? The answer to this question is summarized in the following proposition.

Proposition 3 *For any matrix A we use $\|A\|$ to denote its Frobenius norm $\|A\| := \sqrt{\text{tr}(A^\mathsf{T}A)}$. In (8) assume that $\|\underline{E}_V\| \leqslant \delta_V$, $\|\underline{E}_W\| \leqslant \delta_W$ and define the quantity $\kappa = \|C\| \cdot \|Q(C\underline{V}^*Q)^{-1}\|$. Then the estimation $\hat{\tau}$ obtained by the basic algorithm has the relative error $\frac{\|\tau-\hat{\tau}\|}{\|\tau\|} \leqslant \kappa(\delta_V + \ell\delta_W)$, where ℓ is the alphabet size.*

Proof: It follows from (8) and the definition of $\underline{E}_V$, $\underline{E}_W$ that

$$\begin{aligned}
\tau &= (C_{\text{big}}\underline{W}^*Q + C_{\text{big}}\underline{E}_WQ)(C\underline{V}^*Q + C\underline{E}_VQ)^{-1} \\
&= (C_{\text{big}}\underline{W}^*Q + C_{\text{big}}\underline{E}_WQ)(C\underline{V}^*Q)^{-1}(I_m + C\underline{E}_VQ(C\underline{V}^*Q)^{-1})^{-1} \\
&= [\hat{\tau} + C_{\text{big}}\underline{E}_WQ(C\underline{V}^*Q)^{-1}](I_m + C\underline{E}_VQ(C\underline{V}^*Q)^{-1})^{-1}\,,
\end{aligned}$$

which further implies $\qquad \tau + \tau C\underline{E}_VQ(C\underline{V}^*Q)^{-1} = \hat{\tau} + C_{\text{big}}\underline{E}_WQ(C\underline{V}^*Q)^{-1}$.
It is now clear that $\qquad \|\tau - \hat{\tau}\| \leqslant \kappa(\delta_V\|\tau\| + \delta_W \cdot \frac{\|C_{\text{big}}\|}{\|C\|})$.
But by the definition of C_{big}, $\frac{\|C_{\text{big}}\|}{\|C\|} = \sqrt{\ell}$; and by the condition (b) from Proposition 1, τ has column sums 1 and hence $\|\tau\| \geqslant \frac{1}{\sqrt{\ell}}$, with equality if and only if all entries of τ equal to $\frac{1}{m\ell}$. Thus, $\frac{\|\tau-\hat{\tau}\|}{\|\tau\|} = \kappa(\delta_V + \frac{\sqrt{\ell}}{\|\tau\|}\delta_W) \leqslant \kappa(\delta_V + \ell\delta_W)$. $\square$

Proposition 3 is the main theoretical result of this paper. It offers a deep insight into the relation between the error in counting matrices and the relative error in estimated model parameters, which procures the theoretical foundation of our EC algorithm. Intuitively, the error in counting matrices (measured by $\|\delta_V + \ell\delta_W\|$) will be magnified by a factor of at most κ in estimated models when the basic algorithm is employed. Thus, the quantity κ measures the robustness or stability of the learning algorithm and can be used as a criterion for choosing optimal CIPs. More concretely, in the EC algorithm we design the optimal CIP (C_0, Q_0) by solving the optimization problem

$$\text{minimize } \kappa = \|C\| \cdot \|Q(C\underline{V}^*Q)^{-1}\| \text{ subject to } \mathbf{1}_m^\mathsf{T} C = \mathbf{1}_r^\mathsf{T} \,. \tag{9}$$

As the quantity κ plays such an important role in the EC algorithm, we will give it a special name: the *robustness indicator*. The problem is now reduced to how can we minimize the robustness indicator efficiently, which will be discussed in the next subsection.

3.2 Minimizing The Robustness Indicator

This subsection introduces a fast numerical method for the optimization problem (9). First, we note that adding the extra constraint $C\underline{V}^*Q = I_m$ in (9) will not change the minimal value of κ. To see this, assume that in the problem (9) κ obtains the minimal value at the CIP (C_1, Q_1). Put $C_2 = C_1$ and $Q_2 = Q_1(C_1\underline{V}^*Q_1)^{-1}$, then one can easily verify that $\kappa(C_1, Q_1) = \kappa(C_2, Q_2)$ and that $C_2\underline{V}^*Q_2 = I_m$. So (C_2, Q_2) is a minimizer of κ which satisfies the constraint $C\underline{V}^*Q = I_m$. This means the problem (9) is essentially equivalent to

$$\min_{C,Q} \left\{ \|C\| \cdot \|Q\| \;:\; \mathbf{1}_m^\mathsf{T} C = \mathbf{1}_r^\mathsf{T} \,,\; C\underline{V}^*Q = I_m \right\} \,. \tag{10}$$

We shall use a *ping-pong* method to solve this equivalent problem: alternatively fix one of the matrices C and Q and calculate the other so that (10) is minimized. This results in two optimization problems with quadratic target and linear equality constraint(s), for either case (of C or Q being fixed). Such problems usually have a unique analytical solution that can be easily obtained by solving the corresponding Karush-Kuhn-Tucker (KKT) system (Kuhn and Tucker, 1951), as shown below.

If C is fixed such that $\mathbf{1}_m^\mathsf{T} C = \mathbf{1}_r^\mathsf{T}$, then the optimization problem (10) is reduced to $\min_Q\{\|Q\| : C\underline{V}^*Q = I_m\}$, which is further equivalent to

$$\min_{Q} \left\{ \tfrac{1}{2}\operatorname{tr}(Q^\mathsf{T} Q) \;:\; C\underline{V}^*Q = I_m \right\} \,, \tag{11}$$

since $\|Q\| = \sqrt{\operatorname{tr}(Q^\mathsf{T} Q)}$. Similarly, when Q is fixed and satisfies $\mathbf{1}_r^\mathsf{T}\underline{V}^*Q = \mathbf{1}_m^\mathsf{T}$, which is deduced from the facts $\mathbf{1}_r^\mathsf{T} = \mathbf{1}_m^\mathsf{T} C$ and $C\underline{V}^*Q = I_m$, we get

$$\min_{C} \left\{ \tfrac{1}{2}\operatorname{tr}(C^\mathsf{T} C) \;:\; \mathbf{1}_m^\mathsf{T} C = \mathbf{1}_r^\mathsf{T} \,,\; C\underline{V}^*Q = I_m \right\} \,. \tag{12}$$

Both (11) and (12) are convex quadratic programming problems, with their minimizer $Q_{\min}$ and $C_{\min}$ uniquely determined by the corresponding KKT system:

$$\begin{cases} Q &= (C\underline{V}^*)^{\mathsf{T}}\Lambda \\ I_m &= C\underline{V}^*Q \end{cases}, \qquad \begin{cases} C &= \Lambda(\underline{V}^*Q)^{\mathsf{T}} + \mathbf{1}_m\boldsymbol{\lambda}^{\mathsf{T}} \\ I_m &= C\underline{V}^*Q \\ \mathbf{1}_r^{\mathsf{T}} &= \mathbf{1}_m^{\mathsf{T}}C \end{cases},$$

respectively, where $\Lambda \in \mathbb{R}^{m\times m}$ and $\boldsymbol{\lambda} \in \mathbb{R}^r$ are Lagrange multipliers. It follows from the above equations that (by mechanical computation)

$$\begin{aligned} &\text{if } C \text{ is fixed,} \quad Q_{\min} = (C\underline{V}^*)^{\dagger}\,; \\ &\text{if } Q \text{ is fixed,} \quad C_{\min} = (I_m - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^{\mathsf{T}})(\underline{V}^*Q)^{\dagger} + \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_r^{\mathsf{T}}\,. \end{aligned} \tag{13}$$

We therefore get an iterative procedure for the problem (10), as outlined in Algorithm 1. Intuitively, the algorithm starts with an arbitrary characterizer C and iteratively calculates the new indicator Q and characterizer C by (13), until the robustness indicator κ converges. From the above discussion one easily sees that the sequence of κ's obtained by Algorithm 1 is monotonically decreasing (and lower bounded by 0), so the convergence of κ is guaranteed. Notice that, although each of the two minimizations in (13) is globally optimal, the overall iterative procedure, in general, converges only to a local minimum of κ. This, however, has no bearing on the asymptotic correctness of the resulting OOM estimation.

Algorithm 1: An iterative method for minimizing (10)

```
  Input: the counting matrix $\underline{V}^* \in \mathbb{R}^{r\times r}$, an initial characterizer $C^{(0)} \in \mathbb{R}^{m\times r}$,
         the termination threshold $\delta$ (typical value: $10^{-6}$–$10^{-4}$);
1 $C = (I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^{\mathsf{T}})C^{(0)} + \frac{1}{m}\mathbf{1}_m\mathbf{1}_r^{\mathsf{T}}$; // force $C$ to have column sums 1
2 $Q = (C\underline{V}^*)^{\dagger}$;
3 $\kappa = \|C\| \cdot \|Q\|$;
4 repeat
5     $\kappa_0 = \kappa$;
6     $C = (I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^{\mathsf{T}})(\underline{V}^*Q)^{\dagger} + \frac{1}{m}\mathbf{1}_m\mathbf{1}_r^{\mathsf{T}}$;
7     $Q = (C\underline{V}^*)^{\dagger}$;
8     $\kappa = \|C\| \cdot \|Q\|$;
9 until $\kappa_0 - \kappa < \delta$ ;
  Output: the minimizer $(C, Q)$ of the problem (10);
```

The computational complexity of Algorithm 1 is dominated by the evaluation of the pseudo-inverse of $C\underline{V}^*$ and $\underline{V}^*Q$. By their construction — C is randomly initialized and then Q, C are updated by (13), the two matrices $C\underline{V}^*$ and $\underline{V}^*Q$ are (typically) not too ill-conditioned; so here we can get the pseudo-inverses $(C\underline{V}^*)^{\dagger}$ and $(\underline{V}^*Q)^{\dagger}$ by solving the well-known Wiener-Hopf equations

$$(C\underline{V}^*)^{\dagger}(C\underline{V}^*)(C\underline{V}^*)^{\mathsf{T}} = (C\underline{V}^*)^{\mathsf{T}}\,, \qquad (\underline{V}^*Q)^{\mathsf{T}}(\underline{V}^*Q)(\underline{V}^*Q)^{\dagger} = (\underline{V}^*Q)^{\mathsf{T}}\,,$$

respectively, which is computationally much cheaper than calculating the pseudo-inverse directly. As $(C\underline{V}^*)(C\underline{V}^*)^\mathsf{T}$ and $(\underline{V}^*Q)^\mathsf{T}(\underline{V}^*Q)$ both are positive-definite and symmetric, we can use the Cholesky decomposition method to solve the above two Wiener-Hopf equations. Therefore, the complexity of solving one such Wiener-Hopf equation is $\mathcal{O}(mr^2 + \frac{5}{6}m^3 + m^2r)$. In more detail, computing $C\underline{V}^*$, $(C\underline{V}^*)(C\underline{V}^*)^\mathsf{T}$, the Cholesky decomposition $(C\underline{V}^*)(C\underline{V}^*)^\mathsf{T} = LL^\mathsf{T}$ and $(C\underline{V}^*)^\dagger = (C\underline{V}^*)^\mathsf{T}(L^\mathsf{T})^{-1}L^{-1}$ will cost $\mathcal{O}(mr^2)$, $\mathcal{O}(\frac{1}{2}m^3)$ (the resulting matrix is symmetric), $\mathcal{O}(\frac{1}{3}m^3)$ and $\mathcal{O}(m^2r)$ flops, respectively. Noting the fact that $r \gg m$, we conclude the total cost of Algorithm 1 is about $\mathcal{O}(2Kmr^2)$, where K is the number of iterations.

3.3 The Error Controlling Algorithm

Based upon the above discussion, we can now outline the EC algorithm for training OOMs, as in Algorithm 2. One sees an efficient variant to the algorithm based on the following two facts. First, as discussed at the end of Section 2, one may (and is recommended to) use the *compressed* counting matrices obtained from the CST of $\bar{s}$ in place of the corresponding normalized matrices $\underline{V}^*$ and $\underline{W}_a^*$'s. Second, in (5) the normalization of $\hat{P}(\bar{a})$ by the denominator $l + 1 - |\bar{a}|$ is actually unnecessary, provided that one keeps the same counting factor for all entries of $\underline{V}^*$ and $\underline{W}_a^*$. That is, instead of counting a string $\bar{a}$ in the whole training sequence $\bar{s} = s_1s_2\ldots s_l$, we would count it only in $s_1s_2\ldots s_{l-d}$, where d is a variable integer determined by the requirement that the denominator in (5) (now it becomes $l + 1 - d - |\bar{a}|$) are same for all strings exploited by the counting matrices.

Algorithm 2: The error controlling algorithm

```
  Input: the training sequence $\bar{s} = s_1s_2\ldots s_l$, the model dimension $m$, the
         length $k$ of characteristic/indicative strings;
1 construct the normalized counting matrices $\underline{V}^*$ and $\underline{W}_a^*$'s as in (6) with
  $\{\bar{a}_j\}_{j=1}^r$ and $\{\bar{b}_i\}_{i=1}^r$ both being an enumeration of $O^k$;
2 design the optimal CIP $(C, Q)$ using Algorithm 1;
3 for each $a \in O$, compute $\hat{\tau}_a = (C\underline{W}_a^*Q)(C\underline{V}^*Q)^{-1} = C\underline{W}_a^*Q$;
  // by Algorithm 1, we know $C\underline{V}^*Q = I_m$
4 evaluate the initial state $\hat{\boldsymbol{w}}_0$ from the equations $\mathbf{1}_m^\mathsf{T}\hat{\boldsymbol{w}}_0 = 1$ and
  $(\sum_{a\in O}\hat{\tau}_a)\hat{\boldsymbol{w}}_0 = \hat{\boldsymbol{w}}_0$;
  Output: the learnt OOM $(\mathbb{R}^m, \{\hat{\tau}_a\}_{a\in O}, \hat{\boldsymbol{w}}_0)$;
```

The major computational cost of the EC algorithm includes the design of the optimal CIP (C, Q) and the estimation $\hat{\tau}_a = C\underline{W}_a^*Q$. Hence, the overall complexity of EC is $\mathcal{O}((2K + \ell)mr^2)$, where K is the number of iterations in Algorithm 1 and ℓ the alphabet size.

We conclude with some important observations concerning the EC algorithm:

- In Algorithm 2, the length k of characteristic/indicative strings is fixed. This might be suboptimal since it exploits only the statistics of substrings of length

$2k$ (in $\underline{V}^*$) and $2k+1$ (in $\underline{W}_a^*$'s). To overcome this problem, one may either employ variable-length characteristic/indicative strings or increase the value of k. Note that using variable-length strings is essentially equivalent to increasing k (for we can fix k to be the maximal length of $\bar{a}_j$'s and $\bar{b}_i$'s).

- Larger values of k will not increase the complexity of the algorithm, for the compressed counting matrices actually will not grow too much when the value of k increases. So the efficiency of the EC algorithm is guaranteed. Moreover, we found in our numerical experiments that small k (k should satisfy $\ell^k \geqslant m$ or $\ell^{k-1} \geqslant m$) often works better. One possible reason for this is that for small k we can collect more (counting) data from training sequences and get counting matrices with smaller statistical error [3] .

- Like the ES algorithm, EC does not solve the so called *negative probability problem*: the condition (c) from Proposition 1 might be violated by the estimated model. In other words, the model learned by EC may assign "negative probabilities" $\hat{P}(\bar{a}) = \mathbf{1}_m^{\mathsf{T}}\hat{\tau}_{\bar{a}}\hat{\boldsymbol{w}}_0 < 0$ to some rare sequences $\bar{a} \in O^*$. However, this nuisance can be effectively avoided by using the heuristic method presented in Appendix J of (Jaeger et al., 2005).

4 Numerical Experiments

This section is intended to test the performance of the EC algorithm as described above through two sets of symbolic sequence modeling experiments: modeling a quantized logistic system and modeling several *partially observable Markov decision processes* (POMDPs) that have been extensively used as benchmark tasks in the literature [4] .

Modeling the symbolic logistic system. In the first set of experiments we shall learn OOMs from a complex dynamical system: the quantized logistic system, whose continuous space dynamics is characterized by the mapping $x_{n+1} = rx_n(1-x_n)$. For $r=4$ and initial values $x_0 \in (0,1)$, the continuous system shows strong chaotic behavior (with Lyapunov exponent $\lambda = \ln 2$) in the attractor set $(0,1)$ [5] . A partition of the interval $(0,1)$ into 16 equidistant sub-intervals yields an alphabet of 16 symbols and a 16-ary symbolization, converting the continuous-valued sequence (x_n) into a symbolic one.

The overall procedure of our simulation is as follows: (1) 20 sequences — each of length 30000 — are procured from the above symbolic logistic system; (2) from

[3] Theoretically speaking, there is a trade-off here, for small values of k may fail to discover long-term dependence in the process; and more investigations are needed to understand this problem thoroughly.

[4] We got these POMDPs from `http://www.cs.brown.edu/research/ai/pomdp/`.

[5] See, e.g., `http://tcode.tcs.auckland.ac.nz/~corpus/logistic.html`

each such sequence a matrix model (OOM or HMM) of dimension m is estimated; (3) the quality of the learnt model is then examined on the other 19 sequences. Here the quality of learnt models is measured by its *normalized log-likelihood* (NLL) on the testing data, which is defined by

$$\mathrm{NLL}(\mathfrak{A}, S) := \frac{1}{S^{\#}} \sum_{\bar{a} \in S} \frac{\log_{\ell} P(\bar{a}|\mathfrak{A})}{\text{length of } \bar{a}} \,. \tag{14}$$

In the above definition, ℓ is the alphabet size; $\mathfrak{A}$ represents the learnt model whose quality we want the examine; S is the test dataset; and $S^{\#}$ denotes the number of sequences in S — for the specific case here we have $S^{\#} = 19$.

One easily sees that $\mathrm{NLL}(\mathfrak{A}, S)$ may assume values from $-\infty$ to 0, and that higher NLL-values indicate better estimated models. Intuitively, the quantity ℓ^{NLL} represents the average probability that the learnt model predicts the next symbol correctly, which for a randomly-guessing machine takes the value $1/\ell$ (corresponding to $\mathrm{NLL} = -1$). On the other hand, for the "true" model $\mathfrak{A}^*$ that produces the data S, the quantity $\mathrm{NLL}(\mathfrak{A}^*, S)$ can be seen as a statistical approximation of the *negative* entropy rate of $\mathfrak{A}^*$ (under the base-ℓ logarithm):

$$-h(\mathfrak{A}^*) := \lim_{k \to \infty} \frac{1}{k} \sum_{\bar{a} \in O^k} P(\bar{a}|\mathfrak{A}^*) \log_{\ell} P(\bar{a}|\mathfrak{A}^*). \tag{15}$$

Thus, generally speaking, models estimated by a practical learning algorithm typically have NLL-values from -1 (a model worse than the randomly-guessing machine is unacceptable) to $-h(\mathfrak{A}^*)$ (one should not expect a model better than the true one).

In particular, here one should note that, while the continuous equation of the logistic motion is rather simple, the symbolic dynamics are not, with deep serial dependence among consequent symbols. In fact, according to Pesin's identity (Pesin, 1977), the Kolmogorov-Sinai entropy of the logistic map is equal to its Lyapunov exponent: $h_{\mathrm{KS}} = \ln 2$ (nats) $= 1$ (bit). Intuitively, this means even when the entire history of the process is known, there remains still on average 1 bit uncertainty in the next output. Therefore, under the logarithm of base $\ell = 16$, the test NLL of $-\log_{\ell} 2 = -0.25$ represents the upper limit that a learning algorithm can reach.

To assess the performance of the EC algorithm, we compared EC-learned OOMs with HMMs of the same dimension trained by the EM algorithm from the same dataset. There are two kinds of HMMs: the *state-emission* HMMs (SE-HMMs), in which the symbol is "emitted" by the hidden states; and the *transition-emission* HMMs (TE-HMMs), in which the symbol emitted at time t depends on the hidden states at times t and $t+1$ (Bourlard and Bengio, 2002). Note that, a TE-HMM can be seen as an OOM with nonnegative parameters; whereas a SE-HMM has fewer parameters than an OOM of the same dimension. So, to be fair, we shall compare the performance of OOMs with that of TE-HMMs. We also trained OOMs using the basic algorithm (with the matrices C and Q randomly created) and the efficiency sharpening algorithm, for comparison.

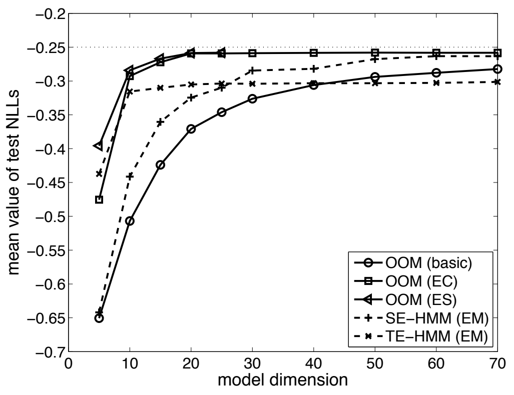
Before reporting the experimental results we need to clarify some universal settings applied in all experiments.

- As mentioned before, EC and ES might learn an invalid OOM from data in that it may produce "negative probabilities" $\hat{P}(\bar{a}) = \mathbf{1}_m^\top \hat{\tau}_{\bar{a}} \hat{\boldsymbol{w}}_0 < 0$ on some sequences $\bar{a} \in O^*$. So the heuristic method described in Appendix J of (Jaeger et al., 2005) is employed to evaluate test NLLs of learnt OOMs in all numerical experiments.

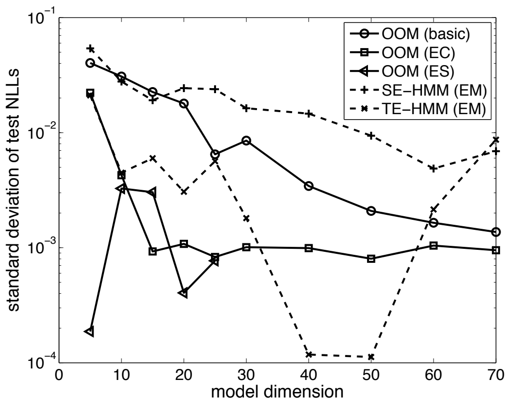
- For OOMs learning the length of characteristic/indicative strings is set to be $k = 5$. For the estimation of HMMs, the EM algorithm is stopped when the increasing of the training NLL-value between two subsequent EM-iterations is less than $\delta = 10^{-4}$ for consecutive 5 times.

- As in (Jaeger et al., 2005), the ES algorithm is terminated after 10 iterations (ES does not have an explicit termination condition); and the estimated model with highest training NLL (each ES-iteration creates an optimal characterizer C, which gives us the estimation $\hat{\tau}_a = (C\underline{W}_a^*)(C\underline{V}^*)^\dagger$; so there are 10 such models) is selected as the final output of ES.

In sum, in this set of experiments 20 sequences of length 30K were procured, from each of which three OOMs, one SE-HMM and one TE-HMM of dimensions $m \in \{5, 10, \ldots, 30, 40, \ldots, 70\}$ were learned; and the test NLL of the learnt models was then computed on the other 19 sequences. Collecting these test NLLs, we get naturally five 20×10 matrices of NLL-values. Figure 1 shows the mean value and the standard deviation of test NLLs of the learnt models. The total CPU-time (all algorithms are programmed in Matlab and implemented on a Pentium-M 1.73 GHz laptop) costed for learning these models is depicted in Fig. 2.



a.



b.

Figure 1: The mean value (a.) and the standard deviation (b.) of test NLLs of HMMs and OOMs on symbolic logistic sequences. The dotted black line in the left figure represents the upper bound of test NLLs.

We collected observations obtained from the results in the following list.

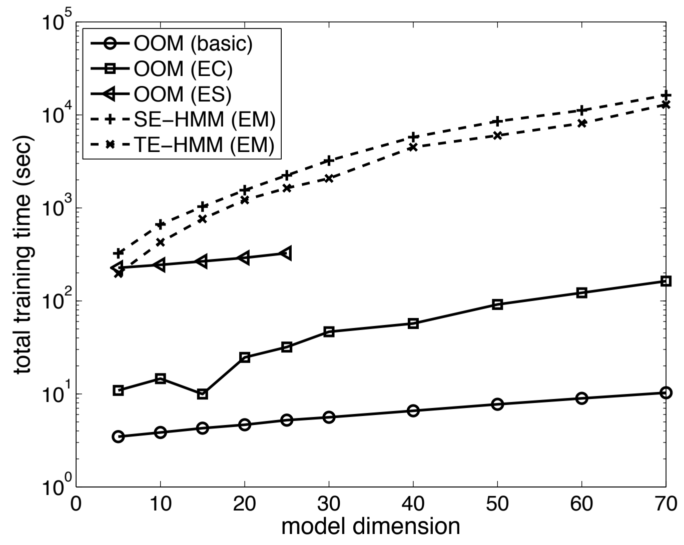


Figure 2: Total CPU-time for learning HMMs and OOMs.

- OOMs trained by EC and ES have nearly the same test NLLs (ES is slightly better) on all datasets and for all model dimensions (except those for which ES becomes instable, see below); which are significantly higher than that of OOMs trained by the basic algorithm, indicating that to design a proper CIP (C, Q) is indeed the key step of the OOM learning procedure and of crucial importance to get an efficient and robust algorithm. This is also confirmed by the considerable difference of the testing NLL-deviation between the EC/ES trained models and those learnt by the basic algorithm.

- For high model dimensions ($m \geqslant 30$), the ES algorithm becomes numerically instable, due to the bad condition of OOM's learning equation (4). This issue might be overcome by using a carefully designed initial characterizer C (e.g., one may consider using EC to get a proper initial model), which however is not the topic of this paper.

- OOMs trained by EC/ES show, compared to EM-trained HMMs, higher test NLLs for all model dimensions, reflecting the fact that stochastic processes that can be described by HMMs can also be captured by OOMs. Furthermore, with the increase of model dimension m, the test NLLs of EC/ES-trained OOMs quickly approaches (at $m = 20$) the best possible value -0.25, which is obtained only for higher-dimensional ($m \geqslant 60$) HMMs; reflecting the fact that OOMs usually provide more compact representations of stochastic processes than HMMs.

- While the NLL-deviation of EM-trained HMMs vary in a quite large range (from 10^{-4} to about 0.1), the same quantity of OOMs trained by EC and ES constantly remain at a low level ($\leqslant 0.01$) for most situations, demonstrating the statistical efficiency of EC and ES.

- Compared to EC, for low dimensional models ES might be statistically more efficient, as revealed by Fig. 1; however, as mentioned earlier, this slight advantage of ES could be quickly annihilated by its numerical instability when the model dimension m increases. In other words, EC is numerically more stable than ES, while still showing comparable efficiency.
- After getting nearly the best testing NLL-value (-0.25) at model dimension $m = 20$, the EC algorithm, contrary to most other machine learning algorithms, does not overfit the training data when the model dimension m increases. Remarkably, the test NLLs of EC stay stably at an almost constant (high) mean value and (low) deviation for $m \geqslant 20$.
- Last but not least, the EC algorithm is about 100 times faster than EM and 10 times faster than ES. This is however not entirely fair to ES, for in the experiments ES is forced to run 10 iterations. But still, even if we only run 2 ES-iterations, EC is faster than ES.

Modeling several POMDPs. In this set of experiments we want to compare the performance of EC/ES-learned OOMs to that of predictive state representations (PSRs) (Littman and Sutton, 2001) trained by methods that have recently been developed in that field.

First we would like to point out the relationship between OOMs and PSRs. PSRs are a class of models of discrete, stochastic input-output systems, which generalizes from partially observable Markov decision processes (POMDPs) like OOMs generalize from HMMs. PSRs have been partly inspired by OOMs but have developed a representation format for input-output processes which is different from the format of input-output OOMs described in (Jaeger, 1998). Developing learning algorithms and specialized variants of PSRs is an active area of research (McCracken and Bowling, 2005; Wolfe et al., 2005; Bowling et al., 2006; Rudary and Singh, 2006; Wolfe, 2006; Wingate and Singh, 2007). However, as shown by the empirical results of our experiment, all learning algorithms available today for PSRs are statistically inefficient (compared to OOMs learning). Therefor, it is desirable to extend the methods of the EC and ES algorithms to PSRs and input-output OOMs, which constitutes a current line of research in our group.

To make the comparison between PSRs and OOMs possible (note that PSRs are models for input-output systems, while OOMs are models for output-only systems), we use the same seven POMDPs as in (Wolfe et al., 2005) to produce training/test datasets; and follow the experimental settings therein. More concretely (from here on we shall use notations that are commonly used in the PSR literature),

- As is commonly done in the PSR field, the input policy is simply to choose an action a at each time step from a uniform random distribution (over the input alphabet A). So the training/test sequences have the form $a^1o^1a^2o^2\dots a^No^N$, a sequence of alternating actions $a^i \in A$ and observations $o^i \in O$.

- For each of the POMDP domains, we train OOMs on (input-output) training sequences of varying lengths ranging from $N = 10^3$ to $N = 10^7$ using the EC/ES algorithm, as follows. We (naively) regard each combination (a^io^i) of an action a^i and its immediate observation o^i as a single symbol s^i from the alphabet $S = A \times O$; and then train an OOM from the sequence $s^1s^2\dots s^N$ (the length of characteristic/indicative strings is set to be $k = 2$).

- The learnt models are evaluated by their average one-step prediction error E on testing sequences of length $N = 10^4$ (the format that has been used for assessing the main PSR algorithms):

$$E = \frac{1}{N}\sum_{i=0}^{N-1}\frac{1}{|O|}\sum_{o\in O}\left[P(o|h_ia^{i+1}) - \hat{P}(o|h_ia^{i+1})\right]^2 ,$$

  where P is the correct probability (computed using underlying POMDPs) and $\hat{P}$ the model prediction for the next observation given the testing history $h_i = a^1o^1\dots a^io^i$ and the action a^{i+1}. For learnt OOMs (over $S = A \times O$) we compute the above predicting probabilities $\hat{P}(o|h_ia^{i+1})$ by

$$\hat{P}(o|h_ia^{i+1}) = \frac{\hat{P}(a^{i+1}o|h_i)}{\hat{P}(a^{i+1}|h_i)} = \frac{\hat{P}(a^{i+1}o|h_i)}{\sum_{o'\in O}\hat{P}(a^{i+1}o'|h_i)} . \tag{16}$$

Note that such a comparison is already a little unfair to OOMs learning, since here we are using input-output systems as testing tasks while OOMs are output-only models and only a straightforward transformation (16) is applied.

The results are given in Fig. 3, in which the average one-step prediction error E (y-axis) is shown for increasing training sequence lengths (x-axis). The seven POMDP domains (from Fig. 3-a. to 3-g.) are named: "bridge", "cheese", "maze4x3", "network", "paint", "shuttle" and "tiger". As a comparison measure, we also presented the PSR learning performance in the figure. These PSR curves are taken from recent publications, which include: PSR-SH (suffix history), PSR-TD (temporal difference) and POMDP-EM are from (Wolfe et al., 2005); the "Gradient" method (only those results for training sequences of length 10^7 are available) is from (Singh et al., 2003); PSR-ODL (online discovery and learning) is from (McCracken and Bowling, 2005) and PSR-ADL (algorithm for discovery and learning, which is essentially a Monte Carlo method and very inefficient: the training sequence length here is much larger than 10^7) is from (James and Singh, 2004).

The main conclusion that we can draw from the figure is that basically the EC and ES algorithms both perform very well on all learning tasks when compared with PSR learning algorithms (especially in the domains "bridge", "maze4x3" and "shuttle"), illustrating that OOM learning is more efficient than PSR learning. In addition, several more detailed observations are of interest:

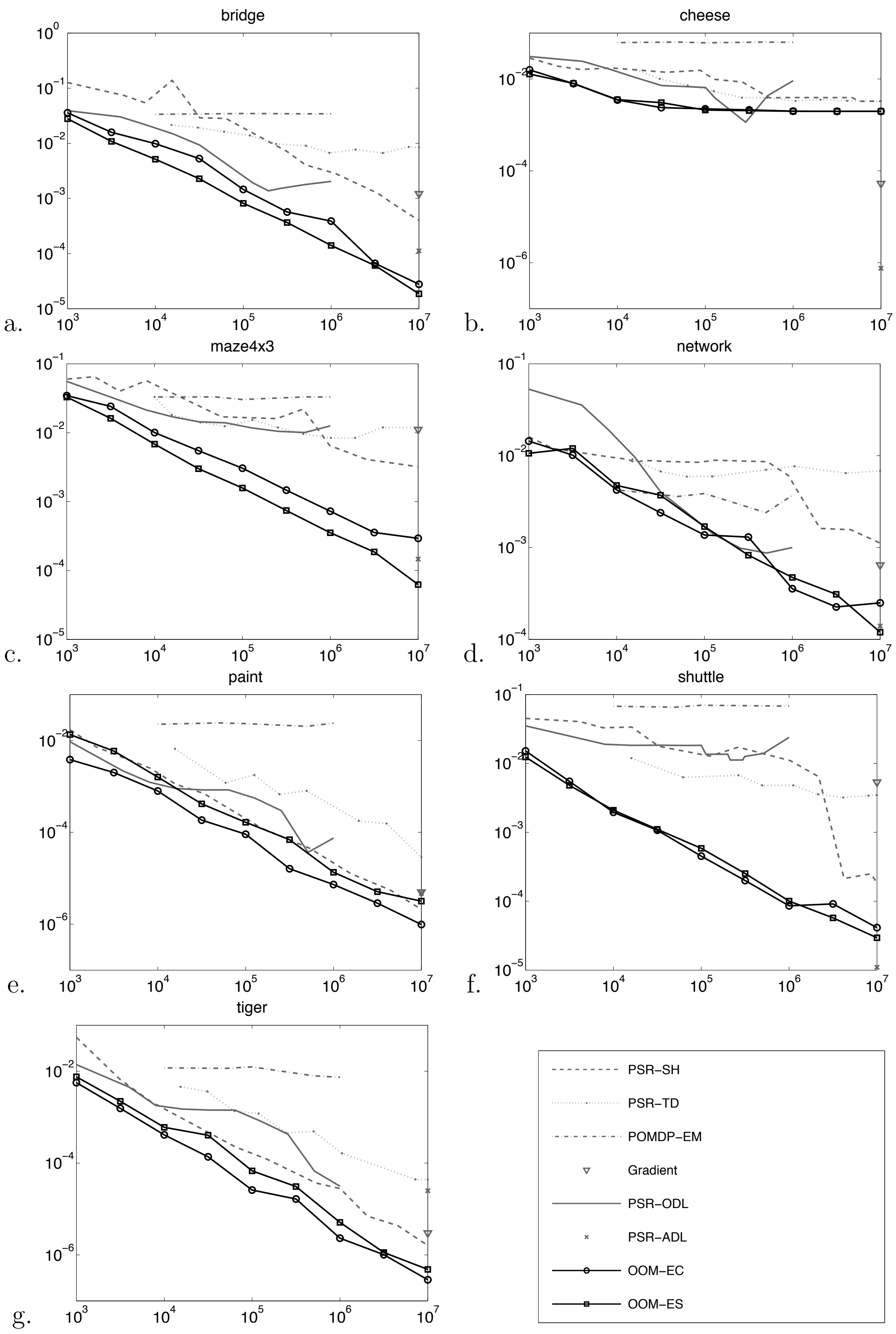


Figure 3: Empirical results of the EC/ES algorithm in comparison to various PSR learning algorithms on seven POMDP benchmark problems.

- Neither EC nor ES defeats the respective other in all domains (EC wins in the "paint" and "tiger" domains, but is not match for ES in the "bridge" and "maze4x3" domains; and they show essentially the same performance in the other three domains). Therefore, it is fair to say that EC and ES are at the same level when considering their statistical efficiency. However, we would emphasize again that ES is much slower than EC in the experiment.

- The OOM learning algorithms show the "near-log-linear" relation between the length N of training sequences and the prediction error E — which means $E \approx \alpha N^{-\beta}$ $(\alpha, \beta > 0)$ — in all domains except for the "cheese" domain. This partly demonstrates that EC/ES is (1) asymptotically correct (since $E \to 0$ when $N \to \infty$) and (2) statistically efficient (since the convergence of E to 0 is polynomially fast).

- In the "cheese" domain, the prediction error E does not decrease any more when $N \geqslant 10^5$. The (possible) reason is that the characteristic/indicative string length $k = 2$ is too small to capture all relevant statistics. We therefore increase the value of k by 1 and do the simulation again. Figure 4 shows the empirical results, from which we see the desired property of OOM learning.

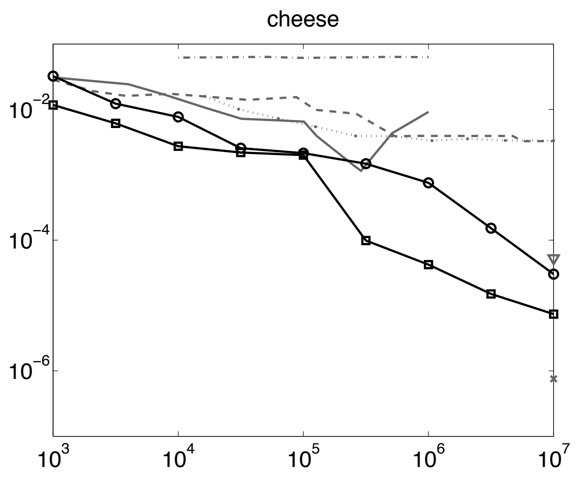


Figure 4: Learning results of the EC/ES algorithm in comparison to various PSR learning algorithms on the "cheese" domain with $k = 3$.

5 Conclusions and Future Work

We have derived the Error Controlling (EC) learning algorithm (or rather more correctly, the EC learning algorithm optimization algorithm) for OOMs, and demonstrated on synthetic learning tasks that

- compared to EM-trained HMMs, the resulting model test accuracy is higher, the model variance is much lower and the computing time is very much lower;

- compared to ES-trained OOMs, the EC algorithm shows superior stability (especially for higher model dimensions) with comparable test accuracy, at faster learning times;

- EC (and ES) compare very favourably to a choice of learning algorithms that have recently been developed in the PSR field.

Furthermore, we found indications that the EC algorithm enjoys a built-in robustness against overfitting.

In sum, we find that the EC algorithm recommends itself by a combination of (i) computational efficiency, (ii) statistical efficiency, (iii) numerical robustness and (iv) – tentatively – statistical robustness (in the sense of avoiding overfitting). Given that OOMs are more expressive than HMMs (i.e. can capture processes that HMMs can't, and often can capture a HMM-describable process with lower-dimensional models than HMMs (Jaeger, 2000b)), and given that HMMs are widely applied (e.g. in speech recognition (Jelinek, 1998) or biosequence analysis (Durbin et al., 2000)), we believe that the EC algorithm may turn out to be a door-opener for a new generation of efficient, robust, and expressive learning algorithm for stochastic symbol sequences. We are however aware of a number of open questions which need to be addressed before any comprehensive judgements can be made:

- Currently, the EC algorithm is given the desired model dimension and a (fixed) characteristic/indicative string length as parameters. This will need to be automated. A heuristic for automatically selecting these parameters based on purely algebraic criteria is under development.

- The EC algorithm should be generalized to become based on variable-length subsequence counting statistics, instead of the fixed-length statistics of the current version. This would presumably yield a further optimization of statistical efficiency.

- We currently pursue an approach to replace the iterative "ping-pong" procedure to solve the minimization (10) by a constructive one-step procedure; this would further speed up the EC algorithm.

- We mentioned in the introduction that the ES algorithm uses a *statistical* criterium to optimize the conditioning matrices C and Q, while the EC algorithm uses an *algebraic* criterium. Interestingly, the resulting statistical efficiency seems similiar if not identical (for the fixed-length counting statistics based versions of EC and ES). Likewise, while EC *functions algebraically* by pulling down a bound on parametric model error (in observable operator matrices), it *effects statistically* an optimization of test performance. We currently have no mathematical understanding of the relationships between the algebraic and the statistical aspects of ES vs. EC, or of the algebraic working principle vs. the statistical effects. Further mathematical analysis is needed. For the time being, we have to live (and can live well) with the existence of *two* algorithms (ES and EC) which have by and large similar performance characteristics (though not quite identical), but are derived from conceptually opposite sides. The existence of this twin pair of algorithms remains for us an intriguing challenge to understand better the connections between a statistical vs. an algebraic analysis of the system identification problem. Eventually we hope to arrive a theoretical unification of the two views. After all, the "heart" of the very OOM approach is to constitute the theory of stochastic processes as a subtheory of linear algebra. Here we want to point out that in (Jaeger, 1999) a purely algebraic characterizations of stochastic processes has already been given; however, this treatment did not include learning aspects.

- Similiarly, further analysis is needed to understand the apparent shieldedness of EC against overfitting.

- The algorithm should prove its usefulness on real-world application tasks (we are starting a collaboration in biosequence analysis).

- Finally, it would be very desirable to solve a general problem of all OOM learning algorithms, the *negativity problem*: the matrices which are computed by these algorithms may predict negative probabilities for certain events (which have true probabilities close to zero). Workarounds exist (Jaeger et al., 2005) (they have been used in the simulations reported here), but the only principled solution known today is to evade the problem altogether by considering a "quadratic" variant of OOMs where negative probabilities are precluded by design (Zhao and Jaeger, 2007).

Acknowledgement. The results reported in this article were achieved while the first author worked under a grant (JA 1210/1-1&2) from the Deutsche Forschungsgemeinschaft (DFG). The authors would also like to thank two anonymous reviewers for a very careful and constructive reviewing.

References

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

Bourlard, H. and Bengio, S. (2002). Hidden Markov models and other finite state automata for sequence processing. In Arbin, M. A., editor, *The handbook of brain theory and neural networks*. MIT Press, 2nd edition.

Bowling, M., McCracken, P., James, M., Neufeld, J., and Wilkinson, D. (2006). Learning predictive state representations using non-blind policies. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 129–136.

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

Durbin, R., Eddy, S., Krogh, A., and Mitchinson, G. (2000). *Biological sequence analysis: probabilistic models of proteins and nucleic acids.* Cambridge University Press.

Faigle, U. and Schönhuth, A. (2006). Quantum predictor models. *Electronic Notes in Discrete Mathematics*, 25:149–155.

Faigle, U. and Schönhuth, A. (2007). Asymptotic mean stationarity of sources with finite evolution dimension. *IEEE Transactions on Information Theory*, 53:2342–2348.

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press.

Jaeger, H. (1998). Discrete-time, discrete-valued observable operator models: a tutorial. GMD Report 42, GMD, Sankt Augustin.

Jaeger, H. (1999). Characterizing distributions of stochastic processes by linear operators. GMD Report 62, German National Research Center for Information Technology.

Jaeger, H. (2000a). Modeling and learning continuous-valued stochastic processes with OOMs. GMD Report 102, GMD, Sankt Augustin.

Jaeger, H. (2000b). Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398.

Jaeger, H., Zhao, M., Kretzschmar, K., Oberstein, T. G., Popovici, D., and Kolling, A. (2005). Learning observable operator models via the ES algorithm. In Haykin, S., Principe, J., Sejnowski, T., and McWhirter, J., editors, *New Directions in Statistical Signal Processing: from Systems to Brains*, chapter 20. MIT Press.

James, M. R. and Singh, S. (2004). Learning and discovery of predictive state representations in dynamical systems with reset. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 53–60.

Jelinek, F. (1998). *Statistical methods for speech recognition.* MIT Press.

Kretzschmar, K. (2003). Learning symbol sequences with observable operator models. GMD Report 161, Fraunhofer Institute AIS.

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. In *Proceedings of 2nd Berkeley Symposium*, pages 481–492. Berkeley: University of California Press.

Littman, M. L. and Sutton, R. S. (2001). Predictive representation of state. In *Advances in Neural Information Processing Systems*, volume 14, pages 1555–1561.

McCracken, P. and Bowling, M. (2005). Online discovery and learning of predictive state representations. In *Advances in Neural Information Processing Systems*, volume 18, pages 875–882.

Pesin, Y. B. (1977). Characteristic Lyapunov exponents and smooth ergodic theory. *Russ. Math. Surveys*, 32(4):55–114.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rudary, M. and Singh, S. (2006). Predictive linear-Gaussian models of controlled stochastic dynamical systems. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 777–784.

Schönhuth, A. (2006). *Diskretwertige stochastische Vektorräume (in German)*. PhD thesis, Faculty of Mathematics and Natural Sciences, University Köln.

Singh, S., Littman, M. L., Jong, N. K., Pardoe, D., and Stone, P. (2003). Learning predictive state representations. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 712–719.

Ukkonen, E. (1995). On-line construction of suffix trees. *Algorithmica*, 14(3):249–260.

Wingate, D. and Singh, S. (2007). On discovery and learning of models with predictive state representations of state for agents with continuous actions and observations. In *Procedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Wolfe, B. (2006). Predictive state representations with options. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 1025–1032.

Wolfe, B., James, M. R., and Singh, S. P. (2005). Learning predictive state representations in dynamical systems without reset. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 985–992.

Zhao, M.-J. and Jaeger, H. (2007). Norm observable operator models. Technical Report 8, Jacobs University Bremen.