Making The Error Controlling Algorithm of Observable Operator Models Constructive*

Ming-Jie Zhao[+,†] Herbert Jaeger[+] Michael Thon[+]

[+] Jacobs University Bremen gGmbH, Bremen, Germany.

[†] Graz University of Technology, Graz, Austria.

Abstract

Observable operator models (OOMs) are a class of models for stochastic processes which properly subsumes the class that can be modelled by finite-dimensional hidden Markov models (HMMs). One of the main advantages of OOMs over HMMs is that they admit asymptotically correct learning algorithms. A series of learning algorithms has been developed, with increasing computational and statistical efficiency, whose recent culmination was the *error controlling* (EC) algorithm developed by the first author. The EC algorithm is an iterative, asymptotically correct algorithm which yields (and minimizes) an assured upper bound on the modelling error. The runtime is faster by at least one order of magnitude than EM-based HMM learning algorithms, and yields significantly more accurate models than the latter. Here we present yet a significant improvement of the EC algorithm, the *constructive error controlling* (CEC) algorithm. CEC inherits from EC the main idea of minimizing an upper bound on the modelling error, but is constructive where EC needs iterations. As a consequence, we obtain further gains in learning speed without loss in modelling accuracy.

Keywords: Observable operator model, learning algorithm, constructive

*The final version of this article will appear in Neural Computation.

†Corresponding author.

1 Introduction

Observable operator models (Jaeger, 2000b) are a mathematical model class for stochastic processes which generalizes *hidden Markov models* (HMMs) (Bengio, 1999) and can be represented in a structually similar matrix formalism, with the crucial difference that the range of model parameters is extended from nonnegative to arbitrary real numbers. This relaxation on the parameter range endows OOMs with linear algebraic properties that are not available for HMMs. These properties engender a basic scheme for learning OOMs that is constructive and asymptotically correct. The statistical efficiency of this basic scheme, however, depends crucially on the design of two auxiliary matrices which are now named the *characterizer* and the *indicator* (Zhao and Jaeger, 2009) and will be denoted by C and Q, respectively, throughout this paper.

To optimize the design of the matrices C and Q is one of the major streams in OOM research. The early learning algorithms (Jaeger, 1998, 2000a,b) relied on simple heuristics for creating C and Q; and the ensuing poor statistical efficiency made OOMs learnt in this way no match for HMMs trained with the expectation-maximization (EM) algorithm (Dempster et al., 1977). In (Kretzschmar, 2003), the author identified the crucial role of the matrices C and Q in the basic scheme and provided a first learning algorithm whose rationale was to optimize their design. Since then, the importance and the challenge of optimizing C and Q became increasingly better understood.

After Kretzschmar's lead work, two theoretically justified and computationally accessible optimality criteria for designing C and Q have been worked out, resulting in the two current state-of-the-art instantiations of the basic learning scheme of OOMs, the *efficiency sharpening* (ES) algorithm (Jaeger et al., 2005) and the *error controlling* (EC) algorithm (Zhao and Jaeger, 2009). Both ES and EC implement iterative schemes to optimize C and Q. Their computational and statistical efficiencies are by and large similar (and outperform EM-based HMM learning), with maybe a small advantage for EC (Zhao and Jaeger, 2009). The underlying ideas for the two are, however, interestingly and fundamentally different.

The perspective of ES is *statistical*. This algorithm aims at minimizing the effects of stochastic variation in C and Q. An algebraic characterization of minimal variance of the estimator is exploited to iteratively improve C and Q toward the goal of minimizing

the variance of the overall algorithm.

The EC algorithm, adopting a complementary view, concentrates on *algebraic* properties of the learning equations. This algorithm optimizes C and Q by exploiting algebraic laws characterizing the condition of matrices, which in turn is related to an upper bound of the relative error in model parameters (or, for short, *modeling error*). As a side effect, this focus on matrix conditions affords a much improved numerical stability of EC over ES.

Although the basic OOM learning scheme is constructive, neither ES nor EC have this attractive property, for both of them employ an iterative procedure to compute the auxiliary matrices C and Q. In this article, following the road of EC, we derive a new upper bound of modeling error which is tighter than the one utilized by the EC algorithm; and we propose a constructive method for minimizing this upper bound. Moreover, this constructive algorithm for optimizing C and Q is globally optimal, whereas the iterative methods employed by EC and ES are only locally optimal. Inserting this analytical minimizer into the basic learning scheme, we obtain an improved version of EC. The resulting *constructive error controlling* (CEC) algorithm is the first learning algorithm for OOMs which is (i) constructive (hence, fast); (ii) asymptotically correct (the hallmark of all OOM learning algorithms); and (iii) statistically efficient. This combination of properties recommends CEC, in our view, as a potentially eminently useful technique.

We perceive this work as a follow-up to (Zhao and Jaeger, 2009). While self-contained, this paper is therefore more condensed in some expository parts than its precursor, and it may be helpful to consult the latter when more background or detail is desired. We shall first briefly re-introduce the EC algorithm in Section 2, clarifying the problem to be addressed and fixing notation. Section 3 describes the derivation of CEC in detail, which includes a novel (tighter) upper bound of modeling error, an analytical minimizer of this error bound and the overall CEC algorithm. We then assess the performance of the CEC algorithm in comparison with EC and ES on the same modeling tasks as those from (Zhao and Jaeger, 2009) and document the empirical results in Section 4. Finally, we summarize the paper and draw some conclusions in Section 5.

2 The Error Controlling Algorithm

Let $O = \{a^1, a^2, \ldots, a^\ell\}$ be a finite set of observable values (an alphabet) and let $(X_n)_{n\in\mathbb{N}}$ be a stochastic process with all $X_n \in O$ and initial joint probabilities:

$$\{\Pr(X_1 = a_1, \ldots, X_n = a_n) =: P(a_1 a_2 \ldots a_n) \ : \ n \in \mathbb{N}, a_i \in O\}\,.$$

Following the notation convention of (Zhao and Jaeger, 2009), we use small letters with a bar to denote finite-length sequences of symbols from O, e.g., $\bar{a} = a_1 \ldots a_n$; and O^* to denote the set of all such finite sequences, including the empty sequence ϵ. The above family of joint probabilities can then be shortly rewritten as $\{P(\bar{a})\}_{\bar{a}\in O^*}$, with the agreement that $P(\epsilon) = 1$.

A (finite-dimensional) OOM for the process (X_n) is a triple $(\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ of the m-dimensional Euclidean space $\mathbb{R}^m$, an O-indexed family $\{\tau_a\}_{a\in O}$ of square matrices of order m, called *observable operators*, and an initial vector $\boldsymbol{w}_0 \in \mathbb{R}^m$ such that, for any finite sequence $\bar{a} = a_1 a_2 \ldots a_n \in O^*$, it holds that

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

where $\mathbf{1}_m$ denotes the m-dimensional column vector of units and $\tau_{\bar{a}}$ the product $\tau_{a_n} \cdots \tau_{a_2} \tau_{a_1}$ of observable operators associated with the sequence $\bar{a}$ (notice the reversal of order in indexing!).

We consider the following OOM learning task: assuming the process (X_n) is stationary, ergodic and has initial distribution $P(\bar{a})$ determined by (1), the objective is to estimate an OOM $(\mathbb{R}^m, \{\hat{\tau}_a\}_{a\in O}, \hat{\boldsymbol{w}}_0)$ of (X_n) from one of its finite initial realizations (say $\bar{s} = s_1 s_2 \ldots s_T$) which can be used to approximately reproduce the distribution of (X_n), i.e., $P(\bar{a}) \approx \mathbf{1}_m^\mathsf{T} \hat{\tau}_{\bar{a}} \hat{\boldsymbol{w}}_0$. A general constructive algorithm (i.e., the basic scheme) for this learning task is outlined as follows (Jaeger, 2000b; Jaeger et al., 2005).

1. For some sufficiently large $k \in \mathbb{N}$ take two sets $\{\bar{a}_j\}_{j=1}^K$ and $\{\bar{b}_i\}_{i=1}^K$ of *basic strings* ($\bar{a}_j$'s are called *indicative strings* and $\bar{b}_i$'s *chararcteristic strings*) which enumerate O^k, the set of all sequences over O of length k — so $K = \ell^k$; and construct the $K \times K$ *normalized counting matrices* $\underline{V}$ and $\underline{W}_a$ ($\forall a \in O$) with their (i, j)-th entry determined by

$$[\underline{V}]_{ij} = \frac{\#_{\text{but last}}(\bar{a}_j \bar{b}_i)}{T - 2k}\,; \qquad [\underline{W}_a]_{ij} = \frac{\#(\bar{a}_j a \bar{b}_i)}{T - 2k}\,, \tag{2}$$

where $(T-2k)$ is the normalization factor and $\bar{a}_j\bar{b}_i$ ($\bar{a}_j a\bar{b}_i$, respectively) is the concatenation of $\bar{a}_j$ (and a, resp.) and $\bar{b}_i$; for any $\bar{a} \in O^*$, $\#(\bar{a})$ denotes the occurrence number of $\bar{a}$ in $\bar{s} = s_1 s_2 \dots s_T$ and $\#_{\text{but last}}(\bar{a})$ its occurrence number in $s_1 s_2 \dots s_{T-1}$.

2. Design two auxiliary matrices: the *characterizer* $C \in \mathbb{R}^{m\times K}$ and the *indicator* $Q \in \mathbb{R}^{K\times m}$, such that $\mathbf{1}_m^{\mathsf{T}} C = \mathbf{1}_K^{\mathsf{T}}$ and the $m \times m$ matrix $C\underline{V}Q$ is invertible.

3. Obtain an (asymptotically correct) estimate of an OOM for the process (X_n) by computing observable operators by $\hat{\tau}_a = (C\underline{W}_a Q)(C\underline{V}Q)^{-1}$ and an initial vector by solving the equations $\mathbf{1}_m^{\mathsf{T}}\hat{\boldsymbol{w}}_0 = 1$ and $(\sum_{a\in O}\hat{\tau}_a)\hat{\boldsymbol{w}}_0 = \hat{\boldsymbol{w}}_0$.

A historical remark: In (Kretzschmar, 2003), Kretzschmar derived the learning equations $\hat{\tau}_a = (C\underline{W}_a Q)(C\underline{V}Q)^{-1}$ and, noting the importance of the matrix $(C\underline{V}Q)$ in the above basic learning scheme, suggested that the design of C and Q should make the condition number $\kappa = \|C\underline{V}Q\|_2 \cdot \|(C\underline{V}Q)^{-1}\|_2$ minimal, where, for any matrix A, $\|A\|_2$ denotes its spectral norm, i.e., the largest singular value of A. However, this algebraic criterion, while seeming to be reasonable at first glance, is actually misguided in the sense that it is always possible to obtain $\kappa = 1$, the smallest possible value of κ, by first creating an *arbitrary* $C \in \mathbb{R}^{m\times K}$ such that $\mathbf{1}_m^{\mathsf{T}} C = \mathbf{1}_K^{\mathsf{T}}$ and that $C\underline{V}$ has full rank m, and then putting $Q = (C\underline{V})^{\dagger}$, the pseudo-inverse of $(C\underline{V})$: with C and Q constructed in this way, we know $C\underline{V}Q = I_m$ (the identity matrix of order m) and hence $\kappa = 1$.

An alternative algebraic criterion for optimizing the matrices C and Q, which falls not prey to this fallacy on this direction, has been developed in previous work of the authors. It is based on the following two findings: firstly, if the *probability matrices* (the "true" normalized counting matrices) $\underline{V}^*$ and $\underline{W}_a^*$'s:

$$[\underline{V}^*]_{ij} = P(\bar{a}_j\bar{b}_i)\,; \qquad [\underline{W}_a^*]_{ij} = P(\bar{a}_j a\bar{b}_i)\,, \qquad (i,j = 1,2,\dots,K)$$

are known, then the true OOM of the underlying process can be reconstructed via the equations $\tau_a = (C\underline{W}_a^* Q)(C\underline{V}^* Q)^{-1}$ (Jaeger et al., 2005); secondly, as proven in Proposition 3 of (Zhao and Jaeger, 2009), the relative error in estimated observable operators $\hat{\tau}_a$ and the statistical error in counting matrices $\underline{V}$ and $\underline{W}_a$ are governed by the inequality

$$\frac{\|\tau - \hat{\tau}\|}{\|\tau\|} \leqslant \|C\| \cdot \|Q(C\underline{V}Q)^{-1}\| \cdot (\|\underline{E}_V\| + \ell \cdot \|\underline{E}_W\|)\,, \tag{3}$$

where, for any matrix A, $\|A\|$ denotes its Frobenius norm $\|A\| := \sqrt{\mathrm{tr}(A^\mathsf{T}A)}$; τ is the *tall matrix* formed by stacking all τ_a's one above another: $\tau = [\tau_{a^1}; \ldots; \tau_{a^\ell}]$ (in Matlab's notation); and $\hat{\tau}$, $\underline{W}$ and $\underline{W}^*$ are tall matrices constructed respectively from $\hat{\tau}_a$'s, $\underline{W}_a$'s and $\underline{W}^*_a$'s by the same stacking scheme; $\underline{E}_V := \underline{V} - \underline{V}^*$ and $\underline{E}_W := \underline{W} - \underline{W}^*$ are the error matrices of $\underline{V}$ and $\underline{W}$, respectively; and (recall that) ℓ is the alphabet size.

Inequality (3) is the theoretical foundation of the EC algorithm, from which one immediately sees that one possible way to control the modeling error $\frac{\|\tau-\hat{\tau}\|}{\|\tau\|}$ is to minimize the quantity $\kappa_1 = \|C\| \cdot \|Q(C\underline{V}Q)^{-1}\|$. This criterion, together with the previously mentioned constraint $\mathbf{1}_m^\mathsf{T} C = \mathbf{1}_K^\mathsf{T}$, forms the optimization problem

$$\min_{C\in\mathbb{R}^{m\times K}, Q\in\mathbb{R}^{K\times m}} \{\kappa_1 = \|C\| \cdot \|Q(C\underline{V}Q)^{-1}\| \ : \ \mathbf{1}_m^\mathsf{T} C = \mathbf{1}_K^\mathsf{T}\} \tag{4}$$

for obtaining optimal C and Q. In the EC algorithm, this problem is solved by a simple numerical method: one randomly creates an $m \times K$ matrix $C = C^{(0)}$ with column sums 1, and iteratively updates Q and C by

$$Q^{(t)} = (C^{(t-1)}\underline{V})^\dagger\,; \qquad C^{(t)} = (I_m - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^\mathsf{T})(\underline{V}Q^{(t)})^\dagger + \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_K^\mathsf{T}\,, \tag{5}$$

until some stopping criterion is met, say, $\kappa_1^{(t-1)} - \kappa_1^{(t)} < \delta$ for some predefined number $\delta > 0$. It was shown in (Zhao and Jaeger, 2009) that $\{\kappa_1^{(t)}\}_{t=1,2,\ldots}$ forms a decreasing sequence with $\kappa_1^{(t)} \geqslant 0$; and so the convergence of $(C^{(t)}, Q^{(t)})$ to a local optimum is guaranteed.

In sum, the error controlling algorithm presented in (Zhao and Jaeger, 2009) is actually an instantiation of the basic learning scheme in which the auxiliary matrices C and Q are optimized by the iterative procedure specified by (5). In the next section we derive an improved version of EC in which the iterative procedure (5) is replaced by a globally optimal analytical solution to an optimization problem that is very similar to (4).

3 The Constructive Error Controlling Algorithm

Although the basic learning scheme is constructive, the EC algorithm is not, since it involves an iterative procedure to optimize C and Q. In this section a constructive method for computing the matrices C and Q will be described, yielding an improved version

of EC which, to the authors' knowledge, is the first constructive learning algorithm of OOMs. To this end, we introduce another upper bound of the relative error $\frac{\|\tau-\hat{\tau}\|}{\|\tau\|}$, which is tighter than (3). We state this upper bound as a proposition.

Proposition 1 *Following the notations that have been introduced, we have*

$$\frac{\|\tau-\hat{\tau}\|}{\|\tau\|} \leqslant \|C\| \cdot \|Q(C\underline{V}Q)^{-1}\|_2 \cdot (\|\underline{E}_V\|_2 + \ell \cdot \|\underline{E}_W\|_2)\,. \tag{6}$$

Proof: From the proof to Proposition 3 of (Zhao and Jaeger, 2009), we get

$$\tau - \hat{\tau} = (\tau C\underline{E}_V - C_{\text{big}}\underline{E}_W) \cdot Q(C\underline{V}Q)^{-1}\,, \tag{7}$$

where $C_{\text{big}} := \text{diag}\{C, \ldots, C\}$ (ℓ copies of C); — using (3) and the definition of $\underline{E}_V$ and $\underline{E}_W$, one can also directly verify this identity. To establish (6), we invoke a well-known matrix inequality (see, e.g., (Golub and Loan, 1996)):

$$\|AB\| \leqslant \min\{\|A\| \cdot \|B\|_2,\ \|A\|_2 \cdot \|B\|\}\,, \tag{8}$$

It follows from (7) and (8) that

$$\|\tau - \hat{\tau}\| \leqslant (\|\tau\| \cdot \|C\| \cdot \|\underline{E}_V\|_2 + \|C_{\text{big}}\| \cdot \|\underline{E}_W\|_2) \cdot \|Q(C\underline{V}Q)^{-1}\|_2\,.$$

But in Proposition 3 of (Zhao and Jaeger, 2009) we have already proven $\|\tau\| \geqslant \frac{1}{\sqrt{\ell}}$ and $\|C_{\text{big}}\| = \sqrt{\ell} \cdot \|C\|$; so the inequality (6) follows. □

Note that, for any matrix A, $\|A\|_2$ equals to the largest singular value of A; and $\|A\|$ is the square root of square sum of all singular values of A. So $\|A\|_2 \leqslant \|A\|$ and (6) indeed provides a tighter upper bound of $\frac{\|\tau-\hat{\tau}\|}{\|\tau\|}$ than (3).

Inequality (6) provides us with another algebraic criterion for computing the optimal C and Q that is very similar to (4), namely,

$$\min_{C\in\mathbb{R}^{m\times K}, Q\in\mathbb{R}^{K\times m}} \{\kappa_2 = \|C\| \cdot \|Q(C\underline{V}Q)^{-1}\|_2 \ :\ \mathbf{1}_m^{\mathsf{T}} C = \mathbf{1}_K^{\mathsf{T}}\}\,. \tag{9}$$

In (Zhao and Jaeger, 2009), the authors have demonstrated that adding the extra constraint $C\underline{V}Q = I_m$ to problem (4) will not change its minimizer. Similarly, here we can prove the optimization problem (9) is essentially equivalent to

$$\min_{C\in\mathbb{R}^{m\times K}, Q\in\mathbb{R}^{K\times m}} \{\kappa_2 = \|C\|\|Q\|_2 \ :\ \mathbf{1}_m^{\mathsf{T}} C = \mathbf{1}_K^{\mathsf{T}},\ C\underline{V}Q = I_m\}\,, \tag{10}$$

where the expression of κ_2 has been simplified by using the fact $C\underline{V}Q = I_m$. To see this, for any minimizer (C, Q) of (9) we put $C_0 = C$ and $Q_0 = Q(C\underline{V}Q)^{-1}$, then $C_0\underline{V}Q_0 = I_m$ and $\kappa_2(C, Q) = \kappa_2(C_0, Q_0)$; which means (C_0, Q_0) also minimizes (9) and, additionally, satisfies the constraint $C\underline{V}Q = I_m$.

Before introducing the constructive method for minimizing (10), we make some remarks on the problem (10) itself and the normalized counting matrices $\underline{V}$ and $\underline{W}_a$ obtained from the training sequence $\bar{s} = s_1 s_2 \ldots s_T$ via (2). Note that some of these remarks have already been presented in (Zhao and Jaeger, 2009), here we restate them in a clearer way to clarify the meaning of some notations that will be used later.

- It is easy to see that, in the learning equations $\hat{\tau}_a = (C\underline{W}_a Q)(C\underline{V}Q)^{-1}$, the normalized counting matrices can be replaced by the *raw* counting matrices $\underline{V}^{\text{raw}}$ and $\underline{W}_a^{\text{raw}}$'s, i.e., $\hat{\tau}_a = (C\underline{W}_a^{\text{raw}}Q)(C\underline{V}^{\text{raw}}Q)^{-1}$, where

$$[\underline{V}^{\text{raw}}]_{ij} = \#_{\text{but last}}(\bar{a}_j \bar{b}_i)\,; \qquad [\underline{W}_a^{\text{raw}}]_{ij} = \#(\bar{a}_j a \bar{b}_i)\,. \tag{11}$$

  It is also clear that replacing $\underline{V}$ by $\underline{V}^{\text{raw}}$ in problem (10) does not change its minimizer (C, Q), although the minimal value of κ_2 is now reduced by a factor of $(T - 2k)$.

- The order of the above raw counting matrices, $K = \ell^k$, increases exponentially with k; which at first glance "results in" that (i) only small values of k are feasible in the basic learning scheme and hence only short-term history context effects of the target process can be learnt and (ii) the optimization problem (10) becomes computationally inaccessible quickly. Nevertheless, firstly, (11) reveals that, given the sequence $\bar{s} = s_1 s_2 \ldots s_T$, the element sum of $\underline{V}^{\text{raw}}$ and $\sum_{a \in O} \underline{W}_a^{\text{raw}}$ are $(T - 2k)$, so for large k, the matrices $\underline{V}^{\text{raw}}$ and $\underline{W}_a^{\text{raw}}$ are actually very sparse; secondly, the learning equations reveals that, discarding zero-rows (columns) in $\underline{V}^{\text{raw}}$, the corresponding rows (columns) in $\underline{W}_a^{\text{raw}}$'s — we call the resulting matrices the *compressed counting matrices* and denote them by $\underline{V}^{\text{cmp}}$ and $\underline{W}_a^{\text{cmp}}$ — and the corresponding columns in C (rows in Q) will not change the estimation $\hat{\tau}_a$; thirdly, those "undiscarded" columns of C (rows of Q) can be efficiently optimized by solving a compressed version of (10), which will be introduced presently. Therefore, in practice we can set the length k of basic strings to be large enough to capture long-term properties of the underlying process.

- As has been pointed out in (Zhao and Jaeger, 2009), in a variant method to fix the basic strings $\{\bar{a}_j\}$ and $\{\bar{b}_i\}$, one can admit them to have different length k. The only condition that has to be observed when selecting basic strings is that $\{\bar{b}_i\}$ should form a "cover" of some set O^k, in the sense that for each string $\bar{c}$ from O^k there is a unique $\bar{b}_i$ which is a prefix of $\bar{c}$. Moreover, by using *compact suffix trees* (Gusfield, 1997) we can construct the compressed counting matrices $\underline{V}$ and $\underline{W}_a$ from the training sequence $\bar{s} = s_1 s_2 \dots s_T$ in linear time, i.e., $\mathcal{O}(T)$ flops (Ukkonen, 1995).

Now assume the compressed counting matrices $\underline{V}^{\text{cmp}}$ and $\underline{W}_a^{\text{cmp}}$ are of size $N \times M$ (note that typically $N, M \ll K$). By the above discussion we know

$$\hat{\tau}_a = (C^{\text{cmp}} \underline{W}_a^{\text{cmp}} Q^{\text{cmp}}) \cdot (C^{\text{cmp}} \underline{V}^{\text{cmp}} Q^{\text{cmp}})^{-1}\,, \tag{12}$$

where, as mentioned above, $C^{\text{cmp}} \in \mathbb{R}^{m \times N}$ (resp. $Q^{\text{cmp}} \in \mathbb{R}^{M \times m}$) is formed from $C \in \mathbb{R}^{m \times K}$ (resp. $Q \in \mathbb{R}^{K \times m}$) by discarding its columns (resp. rows) at positions corresponding to zero-rows (resp. zero-columns) in $\underline{V}^{\text{raw}}$. It remains to find a way to compute C^{cmp} and Q^{cmp} which is efficient in memory-space — (10) is certainly too inefficient to be directly used to compute C and Q and then extract C^{cmp} and Q^{cmp}. To simplify the notation, we move $\underline{V}^{\text{cmp}}$ to the upper-left area of $\underline{V}^{\text{raw}}$ (see (11)) by re-ordering the basic strings $\{\bar{a}_j\}$ and $\{\bar{b}_i\}$, making the other entries of $\underline{V}^{\text{raw}}$ all be zero. Correspondingly, the matrices C, Q can now be written as $C = [C^{\text{cmp}}, C_0]$ and $Q = [Q^{\text{cmp}}; Q_0]$ (in Matlab's notation), respectively, with $C_0 \in \mathbb{R}^{m \times (K-N)}$ and $Q_0 \in \mathbb{R}^{(K-M) \times m}$. The optimization problem (10) is therefore equivalent to [1]

$$\begin{array}{ll} \min & \kappa_2^2 = (\|C^{\text{cmp}}\|^2 + \|C_0\|^2) \cdot \|[Q^{\text{cmp}}; Q_0]\|_2^2 \\ \text{s. t.} & \mathbf{1}_m^{\mathsf{T}} C^{\text{cmp}} = \mathbf{1}_N^{\mathsf{T}}\,, \quad \mathbf{1}_m^{\mathsf{T}} C_0 = \mathbf{1}_{K-N}^{\mathsf{T}}\,, \quad C^{\text{cmp}} \underline{V}^{\text{cmp}} Q^{\text{cmp}} = I_m\,. \end{array} \tag{13}$$

By Proposition 2 (see below), we know $C_0 = \frac{1}{m} \mathbf{1}_m \mathbf{1}_{K-N}^{\mathsf{T}}$ and $Q_0 = 0$, which immediately follows that $\|C_0\|^2 = \frac{K-N}{m}$ and that $\|[Q^{\text{cmp}}; Q_0]\|_2^2 = \|Q^{\text{cmp}}\|_2^2$. We thus get the desired compressed version of (10):

$$\begin{array}{ll} \min & \kappa_2^2 = (\|C^{\text{cmp}}\|^2 + \frac{K-N}{m}) \cdot \|Q^{\text{cmp}}\|_2^2 \\ \text{s. t.} & \mathbf{1}_m^{\mathsf{T}} C^{\text{cmp}} = \mathbf{1}_N^{\mathsf{T}}\,, \quad C^{\text{cmp}} \underline{V}^{\text{cmp}} Q^{\text{cmp}} = I_m\,. \end{array} \tag{14}$$

[1] For any matrix A, $\|A\|^2$ is the square sum of its entries, so $\|[C^{\text{cmp}}, C_0]\|^2 = \|C^{\text{cmp}}\|^2 + \|C_0\|^2$.

An analytical solution to the problem (14)**.** To simplify the notations we first rewrite the problem but with the superscript $^{\text{cmp}}$ dropped:

$$\min_{C,Q}\{\kappa_2^2 = (\|C\|^2 + \tfrac{K-N}{m}) \cdot \|Q\|_2^2 \ :\ \mathbf{1}_m^\mathsf{T} C = \mathbf{1}_N^\mathsf{T},\, C\underline{V}Q = I_m\}\,, \tag{15}$$

where $\underline{V} \in \mathbb{R}^{N\times M}$ is just seen as some known matrix; and $C \in \mathbb{R}^{m\times N}$, $Q \in \mathbb{R}^{M\times m}$. Taking the (full-size) SVD of $\underline{V}$, say $\underline{V} = \underline{L} \cdot \underline{D} \cdot \underline{R}^\mathsf{T}$, and putting $\underline{C} = C\underline{L}$ and $\underline{Q} = \underline{R}^\mathsf{T} Q$, we get a simplified form (the matrix $\underline{V}$ now becomes the diagonal one $\underline{D}$) of (15):

$$\min_{C,Q}\{\kappa_2^2 = (\|\underline{C}\|^2 + \tfrac{K-N}{m}) \cdot \|\underline{Q}\|_2^2 \ :\ \mathbf{1}_m^\mathsf{T} \underline{C} = \mathbf{1}_N^\mathsf{T} \underline{L},\ \ \underline{C} \cdot \underline{D} \cdot \underline{Q} = I_m\}\,,$$

as applying a unitary transformation on a matrix will not change its spectral norm and Frobenius norm. Now assume that $\underline{V}$ has rank r, i.e., $\underline{D} = \mathrm{diag}\{D, 0\}$, where $D = \mathrm{diag}\{d_1, d_2, \ldots, d_r\}$ with $d_1 \geqslant d_2 \geqslant \ldots \geqslant d_r > 0$. Then $\underline{V}$ has the compact SVD $\underline{V} = L_1 D R_1^\mathsf{T}$, where L_1 and R_1 are tall matrices formed by extracting the first r columns from $\underline{L}$ and $\underline{R}$, respectively. Writing $\underline{L} = [L_1, L_2]$ and $\underline{R} = [R_1, R_2]$, and utilizing the fact that $\|[A, B]\|^2 = \|A\|^2 + \|B\|^2$, we expand the above optimization problem to

$$\begin{array}{ll} \min & \kappa_2^2 = (\frac{K-N}{m} + \|C_1\|^2 + \|C_2\|^2) \cdot \|[Q_1; Q_2]\|_2^2 \\ \text{s. t.} & \mathbf{1}_m^\mathsf{T} C_1 = \mathbf{1}_N^\mathsf{T} L_1\,,\ \mathbf{1}_m^\mathsf{T} C_2 = \mathbf{1}_N^\mathsf{T} L_2\,,\ C_1 D Q_1 = I_m\,, \end{array} \tag{16}$$

where $C_i := CL_i$ and $Q_i := R_i^\mathsf{T} Q$ $(i = 1, 2)$.

Proposition 2 *In* (16) *let the matrices C_1 and Q_1 be fixed, then $C_2 = \frac{1}{m}\mathbf{1}_m \mathbf{1}_N^\mathsf{T} L_2$ and $Q_2 = 0$ form a minimizer of the target κ_2^2.*

See Appendix A.1 for the proof. According to the above proposition, we can rewrite the problem (16) as

$$\begin{array}{ll} \min & \kappa_2^2 = (\|C_1\|^2 + \phi) \cdot \|Q_1\|_2^2 \\ \text{s. t.} & \mathbf{1}_m^\mathsf{T} C_1 = \boldsymbol{l}^\mathsf{T}\,,\ C_1 D Q_1 = I_m\,, \end{array} \tag{17}$$

where $\boldsymbol{l}^\mathsf{T} := \mathbf{1}_N^\mathsf{T} L_1$ and $\phi := \frac{K-N}{m} + \|\frac{1}{m}\mathbf{1}_m \mathbf{1}_N^\mathsf{T} L_2\|^2 = \frac{K-N}{m} + \frac{1}{m}\mathbf{1}_N^\mathsf{T} L_2 L_2^\mathsf{T} \mathbf{1}_N$. Here we should point out that, to get the problem (17) from (15), one actually needs only to compute the compact SVD $\underline{V} = L_1 D R_1^\mathsf{T}$. This is because we can compute the quantity ϕ from L_1 by

$$\phi = \tfrac{K-N}{m} + \tfrac{1}{m}\mathbf{1}_N^\mathsf{T} L_2 L_2^\mathsf{T} \mathbf{1}_N = \tfrac{K-N}{m} + \tfrac{1}{m}\mathbf{1}_N^\mathsf{T}(I_N - L_1 L_1^\mathsf{T})\mathbf{1}_N = \tfrac{1}{m}(K - \boldsymbol{l}^\mathsf{T}\boldsymbol{l})\,;$$

and retrieve the optimal C and Q from a solution (C_1, Q_1) to (17) via

$$C = [C_1, C_2][L_1, L_2]^\mathsf{T} = (C_1 - \tfrac{1}{m}\mathbf{1}_m \boldsymbol{l}^\mathsf{T})L_1^\mathsf{T} + \tfrac{1}{m}\mathbf{1}_m \mathbf{1}_N^\mathsf{T}\,,$$
$$Q = [R_1, R_2][Q_1; Q_2] = R_1 Q_1\,.$$

Proposition 3 *In the optimization problem* (17)*, if C_1 is fixed, then $Q_1 = (C_1 D)^\dagger$ is a minimizer of κ_2^2.*

See Appendix A.2 for the proof. Assume $C_1 D$ has the compact SVD $C_1 D = LSR^\mathsf{T}$, then $C_1 = LSR^\mathsf{T} D^{-1}$ and, by Proposition 3, $Q_1 = RS^{-1}L^\mathsf{T}$. For this setting of (C_1, Q_1), the second constraint $C_1 D Q_1 = I_m$ is automatically fulfilled; and the first constraint $\mathbf{1}_m^\mathsf{T} C_1 = \boldsymbol{l}^\mathsf{T}$ now becomes $\mathbf{1}_m^\mathsf{T} LSR^\mathsf{T} = \boldsymbol{l}^\mathsf{T} D$, which implies that, (1) the subspace (in $\mathbb{R}^r$) spanned by columns of R (the *range space* of R) contains the vector $D\boldsymbol{l}$, i.e., $R\boldsymbol{x} = D\boldsymbol{l}$ for some $\boldsymbol{x} \in \mathbb{R}^m$; (2) by multiplying R on the right to this equality, $\mathbf{1}_m^\mathsf{T} LS = \boldsymbol{l}^\mathsf{T} DR$.

Furthermore, in the compact SVD $C_1 D = LSR^\mathsf{T}$ let $\boldsymbol{r}_i$ be the i-th column of R and $\beta_i = \boldsymbol{r}_i^\mathsf{T} D^{-2} \boldsymbol{r}_i$; and write $S = \mathrm{diag}\{s_1, s_2, \ldots, s_m\}$ with $s_1 \geqslant s_2 \geqslant \ldots \geqslant s_m > 0$. Then, by brute-force computation, we obtain

$$\|C_1\|^2 = \textstyle\sum_{i=1}^m \beta_i s_i^2\,; \quad \|Q_1\|_2^2 = s_m^{-2}\,; \quad \kappa_2^2 = s_m^{-2}(\phi + \textstyle\sum_{i=1}^m \beta_i s_i^2)\,.$$

Now assume the matrix $R \in \mathbb{R}^{r\times m}$, which satisfies $R^\mathsf{T} R = I_m$ and $\mathbf{1}_m^\mathsf{T} LSR^\mathsf{T} = \boldsymbol{l}^\mathsf{T} D$, is already known, then the vector $\boldsymbol{l}^\mathsf{T} DR =: \boldsymbol{v}^\mathsf{T} = [v_1, v_2, \ldots, v_m]$ and the scalars $\beta_i = \boldsymbol{r}_i^\mathsf{T} D^{-2} \boldsymbol{r}_i$ are also known. Moreover, by the equality $\mathbf{1}_m^\mathsf{T} LS = \boldsymbol{l}^\mathsf{T} DR$, the vector $\boldsymbol{v}$ has the property

$$\textstyle\sum_{i=1}^m v_i^2 = \boldsymbol{v}^\mathsf{T}\boldsymbol{v} = \boldsymbol{l}^\mathsf{T} DRR^\mathsf{T} D\boldsymbol{l} = \mathbf{1}_m^\mathsf{T} LSR^\mathsf{T} D\boldsymbol{l} = \boldsymbol{l}^\mathsf{T} D^2 \boldsymbol{l}\,, \tag{18}$$

which is a constant independent of the choice of the matrix R.

Writing $\mathbf{1}_m^\mathsf{T} L =: \boldsymbol{u}^\mathsf{T} = [u_1, u_2, \ldots, u_m]$, we get $u_i s_i = v_i$ and so $u_i = v_i/s_i$ from the constraint $\mathbf{1}_m^\mathsf{T} LS = \boldsymbol{l}^\mathsf{T} DR = \boldsymbol{v}$. Since L is an orthogonal matrix, we know

$$\textstyle\sum_{i=1}^m (v_i/s_i)^2 = \sum_{i=1}^m u_i^2 = \boldsymbol{u}^\mathsf{T}\boldsymbol{u} = \mathbf{1}_m^\mathsf{T} LL^\mathsf{T} \mathbf{1}_m = \mathbf{1}_m^\mathsf{T}\mathbf{1}_m = m\,. \tag{19}$$

Note that, this is the only condition $\boldsymbol{u}$ should satisfy since we can choose L freely.

Summing up the above discussion, we get the expanded "scalar" version of (17):

$$\begin{aligned} \min \quad & \kappa_2^2 = s_m^{-2}(\phi + \textstyle\sum_{i=1}^m \beta_i s_i^2) \\ \text{s. t.} \quad & \textstyle\sum_{i=1}^m (v_i/s_i)^2 = m\,, \quad s_1 \geqslant s_2 \geqslant \ldots \geqslant s_m > 0\,. \end{aligned} \tag{20}$$

For each i, let $\alpha_i = s_i/s_m$, then $s_i = \alpha_i s_m$ and $\alpha_1 \geqslant \alpha_2 \geqslant \ldots \geqslant \alpha_m = 1$. From the equality $\sum_{i=1}^m (v_i/s_i)^2 = m$ we know $s_m^2 = \frac{1}{m}\sum_{i=1}^m (v_i/\alpha_i)^2$. Therefore,

$$\kappa_2^2 = \phi s_m^{-2} + \textstyle\sum_{i=1}^m \beta_i \alpha_i^2 = m\phi \cdot (\sum_{i=1}^m v_i^2 \alpha_i^{-2})^{-1} + \sum_{i=1}^m \beta_i \alpha_i^2 \,,$$

from which we immediately see that κ_2^2 reaches its minimal value

$$(\kappa_2^2)_{\min} = m\phi \cdot (\textstyle\sum_{i=1}^m v_i^2)^{-1} + \sum_{i=1}^m \beta_i = m\phi \cdot (\boldsymbol{l}^\mathsf{T} D^2 \boldsymbol{l})^{-1} + \sum_{i=1}^m \beta_i \tag{21}$$

when $\alpha_i = 1$, i.e., $s_i = s_m$ for all i. It then follows from (18) and (19) that

$$s_i = \sqrt{m^{-1} \cdot (\boldsymbol{l}^\mathsf{T} D^2 \boldsymbol{l})} = \tfrac{1}{\sqrt{m}} \|D\boldsymbol{l}\| \,, \qquad (i = 1, 2, \ldots, m) \,. \tag{22}$$

As the value of $\boldsymbol{l}^\mathsf{T} D^2 \boldsymbol{l}$ is independent of the matrix R, (21) indicates that, to minimize the target κ_2^2, the choice of R should make the sum $\sum_{i=1}^m \beta_i$ as small as possible. By the definition of β_i, we know $\sum_{i=1}^m \beta_i = \mathrm{tr}(R^\mathsf{T} D^{-2} R)$. We therefore get the following optimization problem to determine the matrix R:

$$\begin{array}{ll} \min & \sum_{i=1}^m \beta_i = \mathrm{tr}(R^\mathsf{T} D^{-2} R) \quad \text{where } R \in \mathbb{R}^{r\times m} \\ \text{s. t.} & R^\mathsf{T} R = I_m \,, \quad \exists\, \boldsymbol{x} \in \mathbb{R}^m \,(R\boldsymbol{x} = D\boldsymbol{l}) \,. \end{array} \tag{23}$$

Note that, if R is a minimizer of (23), so is RU, where U can be any orthogonal matrix of order m. We thus assume, without loss of generality, that the last column of R is $\boldsymbol{r}_m = D\boldsymbol{l}/\|D\boldsymbol{l}\|$ — the unit vector on the direction of $D\boldsymbol{l}$ (for vectors, their Frobenius norm equals to their Euclidean norm, so we can abuse our notation of $\|\cdot\|$) — since $D\boldsymbol{l}$ lies in the range space of R. Thus, the matrix R has the form $R = [HX, \boldsymbol{r}_m]$, where $H \in \mathbb{R}^{r\times(r-1)}$ is fixed and has columns forming a basis of the null space of $\boldsymbol{r}_m$, i.e., H can be any matrix with the properties $H^\mathsf{T} H = I_{r-1}$ and $\boldsymbol{r}_m^\mathsf{T} H = 0$ — see Appendix A.3 for a simple construction of such a matrix H; and X is a some $(r-1)$ by $(m-1)$ matrix satisfying $X^\mathsf{T} X = I_{m-1}$. Substituting the expression $R = [HX, \boldsymbol{r}_m]$ into (23), we get

$$\min\{\mathrm{tr}(X^\mathsf{T} H^\mathsf{T} D^{-2} H X) \;:\; X^\mathsf{T} X = I_{m-1}\} \,.$$

As is well known, this optimization problem has an analytical solution: the i-th column of X, denoted $\boldsymbol{x}_i$, is exactly the eigenvector of $H^\mathsf{T} D^{-2} H$ with respect to its i-th *smallest* eigenvalue λ_i, which, surprisingly, is equal to β_i — this can be proven in one line: $\beta_i = \boldsymbol{r}_i^\mathsf{T} D^{-2} \boldsymbol{r}_i = \boldsymbol{x}_i^\mathsf{T} H^\mathsf{T} D^{-2} H \boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i^\mathsf{T} \boldsymbol{x}_i = \lambda_i$.

It now remains only to find an orthogonal matrix L such that $\mathbf{1}_m^\mathsf{T} L = \boldsymbol{u}^\mathsf{T}$. With the matrix R constructed as above, we find that

$$\boldsymbol{v}^\mathsf{T} = \boldsymbol{l}^\mathsf{T} DR = \|D\boldsymbol{l}\| \cdot \boldsymbol{r}_m^\mathsf{T}[HX, \boldsymbol{r}_m] = \|D\boldsymbol{l}\| \cdot [0, \ldots, 0, 1]\,,$$

i.e., $v_i = 0$ for $i < m$ and $v_m = \|D\boldsymbol{l}\|$. By the relation $u_i s_i = v_i$ and (22) we know $\boldsymbol{u}^\mathsf{T} = [0, \ldots, 0, \sqrt{m}]$, so L can be any orthogonal matrix with $\frac{1}{\sqrt{m}}\mathbf{1}_m$ as its last column. For instance, we can set

$$L = \frac{1}{\sqrt{m}} \begin{bmatrix} (1+\sqrt{m})^{-1}\mathbf{1}_{m-1}\mathbf{1}_{m-1}^\mathsf{T} - \sqrt{m} I_{m-1} & \mathbf{1}_{m-1} \\ \mathbf{1}_{m-1}^\mathsf{T} & 1 \end{bmatrix}. \tag{24}$$

A more general construction of such matrices L is presented in Appendix A.3.

The constructive error controlling algorithm. All the above discussion on the problem (14) sums up to a constructive procedure for evaluating the optimal C^{cmp} and Q^{cmp} that minimizes the modeling error bound presented in the inequality (6). Inserting this procedure into the basic learning scheme, we get the *constructive error controlling* (CEC) algorithm, as shown in Algorithm 1, in which some Matlab-style notation is employed.

The time complexity of the CEC algorithm is dominated by (1) the collection of the compressed counting matrices, which, as discussed above, costs $\mathcal{O}(T)$ flops; and (2) the computation of the compact SVD $\underline{V}^{\text{cmp}} = L_1 D R_1^\mathsf{T}$, which, in the worst case where the matrix $\underline{V}^{\text{cmp}}$ has full rank $r = N$ (assume that $M = N$), amounts to $\mathcal{O}(N^3)$ flops (Golub and Loan, 1996). Thus, the total time complexity of CEC is about $\mathcal{O}(T + N^3)$. Here we recall that the EM algorithm for training HMMs and the ES algorithm both have time complexity $\mathcal{O}(nm^2T)$, where n is the number of EM- or ES-iterations. We therefore conclude that the CEC algorithm is faster than EM and ES, since $N \ll T$ is the typical case. The comparision between EC and CEC is more sophisticated. As analyzed in (Zhao and Jaeger, 2009), the time complexity of EC is about $\mathcal{O}(T + 2nmN^2)$, which is typically higher than that of CEC; and so EC is slower than CEC, as shown by our simulations. However, in cases where large (full-rank) counting matrices V^{cmp} are obtained, CEC may need more CPU time (mainly due to the SVD of V^{cmp}) than EC.

Algorithm 1: The constructive error controlling (CEC) algorithm

Input: - the training sequence $\bar{s} = s_1 s_2 \dots s_T$;

- the model dimension m;

- the length k of basic strings.

Procedure:

1. Extract the compressed counting matrices $\underline{V}^{\text{cmp}}$ and $\underline{W}_a^{\text{cmp}}$ from $\bar{s}$.

2. Evaluate the optimal C^{cmp} and Q^{cmp} as follows.

a. compute the compact SVD of $\underline{V}^{\text{cmp}}$: $\quad \underline{V}^{\text{cmp}} = L_1 D R_1^{\mathsf{T}}$;

b. let $\boldsymbol{l}^{\mathsf{T}} = \mathbf{1}_N^{\mathsf{T}} L_1, \quad v_m = \|D\boldsymbol{l}\|, \quad s_m = v_m/\sqrt{m}$;

c. let $\boldsymbol{r}_m = D\boldsymbol{l}/v_m, \quad H = \text{null}\,(\boldsymbol{r}_m^{\mathsf{T}})$ — or, cf. Appendix A.3;

d. let $[\beta_i, \boldsymbol{x}_i] = i\,\text{th-smallest-eigenpair}\,(H^{\mathsf{T}} D^{-2} H)_{i=1,2,\dots,m-1}$;

e. let $R = [H\boldsymbol{x}_1, \dots, H\boldsymbol{x}_{m-1}, \boldsymbol{r}_m], \quad L$ be as in (24) or (29);

f. let $C^{\text{cmp}} = (s_m L R^{\mathsf{T}} D^{-1} - \frac{1}{m}\mathbf{1}_m \boldsymbol{l}^{\mathsf{T}}) L_1^{\mathsf{T}} + \frac{1}{m}\mathbf{1}_m \mathbf{1}_N^{\mathsf{T}}, \quad Q^{\text{cmp}} = s_m^{-1} R_1 R L^{\mathsf{T}}$;

3. Estimate all $\hat{\tau}_a$'s by (12) and $\hat{\boldsymbol{w}}_0$ as in step 3. of the basic scheme.

Output: - the learnt model $(\mathbb{R}^m, \{\hat{\tau}_a\}_{a\in O}, \hat{\boldsymbol{w}}_0)$.

An efficient variant to CEC. We now introduce an efficient variant to the CEC algorithm. The time complexity $\mathcal{O}(T + N^3)$ of the above CEC algorithm can be reduced to $\mathcal{O}(T + mN^2)$ if we compute only the first m (the model dimension) singular triples of the matrix $\underline{V}^{\text{cmp}}$. That is, instead of the compact SVD $\underline{V}^{\text{cmp}} = L_1 D R_1^{\mathsf{T}}$, we use the first (largest) m singular values and left- (right-) singular vectors of $\underline{V}^{\text{cmp}}$ to form the matrix D, L_1 and R_1, respectively; and then estimate the OOM as in Algorithm 1. To distinguish the two versions of CEC we shall call the efficient variant presented here the CEC-B algrithm and the original one described by Algorithm 1 the CEC-A algorithm in the sequel.

The basic idea behind CEC-B is rather simple: the counting matrix $\underline{V}^{\text{cmp}}$ is in fact the statistical approximation of the probability matrix $\underline{V}^*$ (up to a counting factor), which should have rank m — recall that we have already assumed that the underlying process can be modelled by some m-dimensional OOM. Therefore, we can (and it is quite natural) first estimate $\underline{V}^*$ (the "true" counting matrix) from $\underline{V}^{\text{cmp}}$; then use $\underline{V}^*$ to design the optimal C and Q — which is what CEC-B does.

By the above discussion, one immediately sees that CEC-A and CEC-B should

have approximately the same performance, especially when the training dataset is large (since then $\underline{V}^{\text{cmp}} \approx \underline{V}^{*}$). We would however emphasize again that CEC-B is computationally cheaper than CEC-A.

We finish this section with an important remark on a painful issue in OOM theory, namely, the *negative probability problem* (Jaeger, 2000b). It refers to the fact that the model $(\mathbb{R}^m, \{\hat{\tau}_a\}_{a\in O}, \hat{\boldsymbol{w}}_0)$ learnt by existing OOM learning algorithms, including the CEC algorithm presented in this article, can produce "negative probabilities" $\hat{P}(\bar{a}) = \mathbf{1}_m^{\mathsf{T}} \hat{\tau}_{\bar{a}} \hat{\boldsymbol{w}}_0 < 0$ for some sequence $\bar{a} \in O^*$. For a long time, the authors and others attempted to find ways to eliminate this problem. Very recently, it was proven that it is *undecidable* to determine whether a given candidate set of observable operators leads to negative values on some strings (Wiewiora, 2008). This finding does not preclude the possibility to find nontrivial and practically useful sufficient conditions for non-negativity. In practice, the current state of the art relies on heuristic methods which essentially "repair" the OOM state vector by small correction terms when it would give negative "probability" values. These work well, e.g., the one presented in Appendix J of (Jaeger et al., 2005).

4 Empirical Results

In this section, we assess the performance of CEC-A,B in comparison with the EC and ES algorithms on three sets of symbolic sequence modeling tasks: the first two are repeated from (Zhao and Jaeger, 2009), namely, learning models for a quantized logistic system and a suite of seven *partially observable Markovian decision processes* (POMDPs) that have been used frequently as benchmark tasks in the literature; the third one is taken from (Jaeger et al., 2005): to train OOMs on Mark Twain's short story "The One Million Pound Bank-Note".

Modeling the symbolic logistic system. In this experiment we consider the task of modeling a dynamical system that consists of a continuous-valued iterated map followed by a quantizer which maps real numbers to discrete symbols. The dynamics of the continuous-valued procedure is governed by the logistic mapping $x_{n+1} =$

$4x_n(1-x_n)$, with initial values x_0 arbitrary selected from the initerval $(0,1)$. As is well known, in its attractor set $(0,1)$, this system shows strong chaotic behavior (with Lyapunov exponent $\lambda = \ln 2$). The attractor set $(0,1)$ is divided into $\ell = 16$ equidistant sub-intervals, yielding an alphabet of 16 symbols and a 16-output quantizer that converts the continuous-valued sequence (x_n) into a symbolic process (X_n).

We follow the experimental settings of (Zhao and Jaeger, 2009): (1) 20 sequences each of which has length $L = 30000$ are created by running the above quantized logistic system; (2) OOMs of dimension $m \in \{5, 10, \ldots, 30, 40, \ldots, 70\}$ are then estimated from each such sequence, using the EC, ES and CEC-A,B algorithms; (3) finally, we compute the normalized log-likelihood (NLL) of these learnt models $\mathfrak{A}$ on the other 19 sequences, as follows:

$$\mathrm{NLL}(\mathfrak{A}, S) := \frac{1}{S^{\#}} \sum_{\bar{a} \in S} \frac{\log_\ell P(\bar{a}|\mathfrak{A})}{\text{length of } \bar{a}}\,, \tag{25}$$

where ℓ is the alphabet size; S is the test dataset that consists of the other 19 (testing) sequences; and $S^{\#}$ is the number of sequences in S, which takes the value 19 in this experiment. Here are the settings of our simulation:

- When evaluating test NLLs of learnt OOMs, the heuristic method described in Appendix J of (Jaeger et al., 2005) is employed to address the negative probability problem, from which each of the EC, ES and CEC-A,B algorithms suffers.

- The length of characteristic/indicative strings is set to be $k = 5$.

- As in (Zhao and Jaeger, 2009), the ES algorithm is terminated after 10 iterations; and the estimated model with highest training NLL is selected as the final output of ES. But unlike as in (Zhao and Jaeger, 2009), ES is now started with a more carefully chosen initial model, so that it is numerically stable for all model dimensions.

As has been pointed out in (Zhao and Jaeger, 2009), in practice the quantity $\mathrm{NLL}(\mathfrak{A}, S)$ should assume values from -1 to $-H(X_n)$, where (X_n) is the underlying stochastic process that we want to model and $H(X_n)$ is its entropy rate (under the base-ℓ logarithm), defined by the formula

$$H(X_n) := -\lim_{k\to\infty} \frac{1}{k} \sum_{\bar{a} \in O^k} P(\bar{a}) \log_\ell P(\bar{a})\,, \tag{26}$$

where, as introduced at the beginning of Section 2, $P(\bar{a})$ is the (initial) probability distribution of the process (X_n). Specifically, it turns out that the quantized logistic process described above has entropy rate $H(X_n) = 0.25$ (Zhao and Jaeger, 2009), meaning that the quantized logistic process is indeed a stochastic process, although its continuous space motion is rather simple and deterministic (in the sense that we can accurately predict the next output x_{n+1} based on the current one x_n). Thus, in this experiment, a test NLL of $-H(X_n) = -0.25$ represents the upper limit that a learning algorithm can reach.

In sum, here our dataset contains 20 sequences of length 30K, from each of which four OOMs of dimensions $m \in \{5, 10, \ldots, 30, 40, \ldots, 70\}$ were estimated, by the EC, ES and CEC-A,B algorithm, respectively; the test NLL of these learnt models was then computed on the other 19 sequences. We therefore obtain

$$20^{(\text{nr. of sequences})} \times 10^{(\text{nr. of model dimensions})} \times 4^{(\text{nr. of algorithms})}$$

NLL-values after the simulation. We plotted in Fig. 1-a the average test NLLs of each OOM learning algorithm versus model dimensions; in Fig. 1-b the standard deviation of these test NLLs; and in Fig. 2 the total CPU-time needed for learning these models — all algorithms are programmed in Matlab (with C-mex functions embedded) and implemented on a Pentium-M 1.73 GHz laptop.

From the figures we see that (1) CEC yields slightly more accurate models than EC and ES; (2) for large model dimensions CEC is significantly faster than EC and ES; and (3) while CEC-B has nearly the same test NLLs as CEC-A (see the explanation below Fig. 1 for more detail), it is 2–3 times faster than CEC-A. Another important observation is that, for model dimensions $m \geqslant 40$, all models (under ES, EC, CEC) level out in their NLLs, which reach ≈ -0.261, -0.257, -0.257, respectively, with a small margin toward the theoretical optimum of -0.25.

POMDP benchmark suite. In the second experiment we compare the performance of ES, EC, and CEC on the task of learning models of seven POMDPs that were frequently used as benchmark tasks for assessing the performance of predictive state representations (PSRs) (Littman and Sutton, 2001) in the literature.

First we would like to point out the relationship between OOMs and PSRs. PSRs are a class of models for discrete, stochastic input-output systems, which generalizes par-

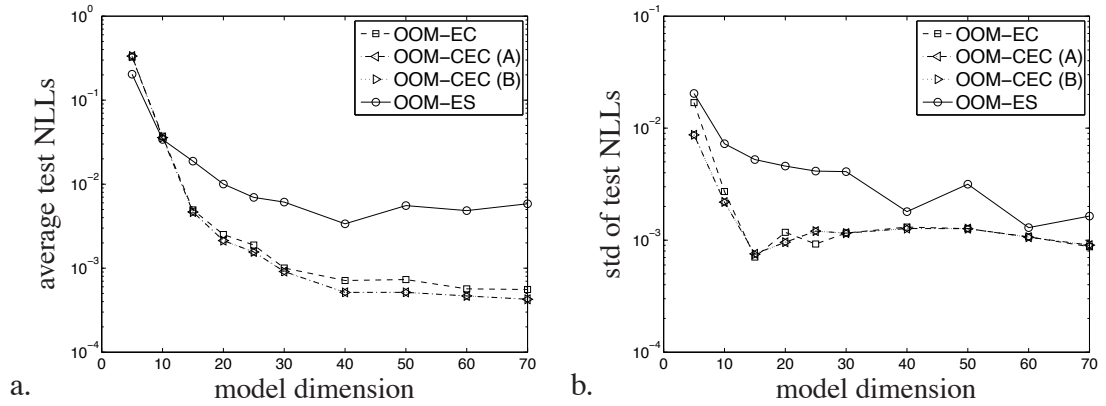

Figure 1: The test NLLs of learnt OOMs on the quantized logistic system: (a.) the average test NLLs — to make the difference more visible, here the y-axis is computed by "$y = -0.256 -$ average-NLL" and represented in logarithmic scale (note the number -0.256 is a little smaller than the previously introduced upper limit of test NLLs -0.25); (b.) the standard deviation of the test NLLs. *Note that, in this experiment (also in the second and the third experiments), models learnt by CEC-A and CEC-B have nearly the same quality — the relative difference in test NLL or one-step prediction error (see* (27)*) are less than* 0.1% *on all datasets. As a consequence, in Figures 1, 3 and 5 the curves for CEC-A and CEC-B are (almost) identical.*

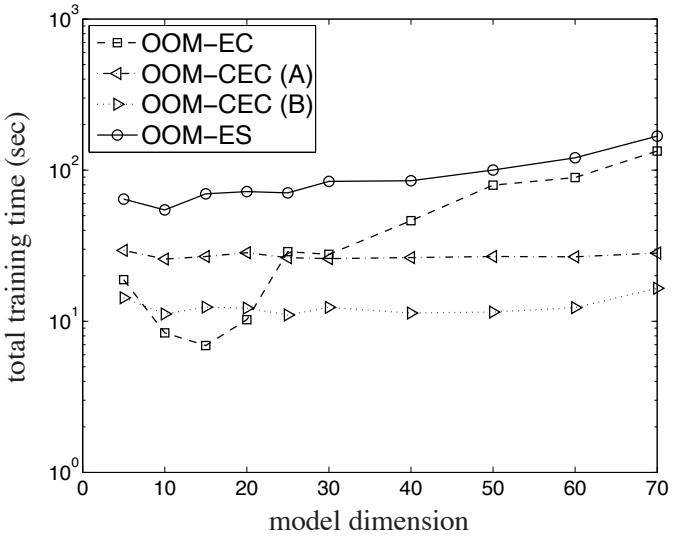

Figure 2: The training CPU-time of the ES, EC and CEC-A,B algorithms on the quantized logistic system.

tially observable Markov decision processes (POMDPs) like OOMs generalizes 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). In (Zhao and Jaeger, 2009) we demonstrated that ES and EC strongly outperform known PSR learning algorithms. These results are not repeated here. We remark furthermore that extending the ES, EC, and CEC algorithms to input-output OOMs constitutes a current line of research in our group [2] .

The seven target POMDPs are taken from a public web source [3] ; they include "bridge", "cheese", "maze4x3", "network", "paint", "shuttle" and "tiger". We use the same experimental settings as (Zhao and Jaeger, 2009), for more detail we refer the reader to this precursor paper. We use notations that are common in the PSR literature.

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

- For each of the above mentioned POMDP domains, we train OOMs on (input-output) training sequences of varying lengths $N = 10^3$–$10^{7.3}$ using the CEC-A or B (or 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 learn an OOM from the sequence $s^1s^2 \ldots s^N$ (with the length of basic strings set to be $k = 2$).

- The quality of learnt models are measured by their average one-step prediction error E on testing sequences of length $N = 10^4$ (the format that has been used

[2]Note to reviewers: by the time of a possible publication of this paper, we may be able to refer to a publication here.

[3]Tony's POMDP page, `http://www.cs.brown.edu/research/ai/pomdp/`, 1999.

for assessing the main PSR algorithms):

$$E = \frac{1}{N}\sum_{i=0}^{N-1}\frac{1}{|O|}\sum_{o\in O}\left[P(o|h_i a^{i+1}) - \hat{P}(o|h_i a^{i+1})\right]^2 , \tag{27}$$

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

$$\hat{P}(o|h_i a^{i+1}) = \frac{\hat{P}(a^{i+1}o|h_i)}{\hat{P}(a^{i+1}|h_i)} = \frac{\hat{P}(a^{i+1}o|h_i)}{\sum_{o'\in O}\hat{P}(a^{i+1}o'|h_i)} . \tag{28}$$

In Fig. 3 we plotted the average one-step prediction error E (y-axis) of the ES, EC and CEC-A,B algorithms on the seven POMDP domains versus training sequence length (x-axis); and in Fig. 4 the training CPU-time costed by these algorithms for each training length. We collect the main findings in the following list.

- The four OOM learning algorithms all show a "near-log-linear" relationship between the length N of training sequences and the prediction error E — that is, $\log E \approx \alpha - \beta \log N$, or, equivalently, $E \approx e^{\alpha} N^{-\beta} = E_0 N^{-\beta}$ ($E_0, \beta > 0$) — in all domains except for the "cheese" domain. This partly demonstrates that OOM learning is (1) asymptotically correct (since $E \to 0$ when $N \to \infty$) and (2) statistically efficient (since the convergence of E to 0 is fast), as has been claimed in our previous OOM references.

- In the "cheese" domain, the prediction error E does not decrease any more when $N \geqslant 10^5$. One possible reason is that the basic 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 3-h shows the empirical results, from which we see the desired property of OOM learning.

- The two CEC algorithms and the EC algorithm have nearly the same one-step prediction error. But for small training dataset CEC is faster than EC; whereas for large dataset the difference in speed becomes invisible. The reason is that, when the training sequence length T is large (compared to the size N of counting matrices), both CEC and EC have time complexity dominated by $\mathcal{O}(T)$; and when

T is small, CEC has the time complexity $\mathcal{O}(mN^2)$ or $\mathcal{O}(N^3)$, smaller than that of EC which is $\mathcal{O}(2nmN^2)$.

- Neither CEC nor EC outperforms ES in all domains: they win in the "network", "paint" and "tiger" domains, but are no match for ES in the "cheese" and "maze4x3" domains; and the four algorithms show essentially the same performance in the "shuttle" domain. But both CEC and EC are about 10 times faster than ES, especially for large training datasets.

- "Bridge" is an interesting domain, in which CEC/EC is inferior to ES when $N \leqslant 10^5$, but when N becomes larger ($\geqslant 3 \times 10^5$), CEC/EC starts to yield a smaller prediction error than ES (In fact, other domains also show the similar phenomenon, which is however less obvious as in the "bridge" domain). This on the one hand illustrates that, asymptotically, CEC/EC converges faster than ES ($\beta_{\text{CEC}} > \beta_{\text{ES}}$); and on the other hand reveals that in some domains CEC/EC might be not so statistically efficient as ES when the training dataset is small.

- In the "bridge" domain, the prediction error of EC jumps to a high level at $T = 10^7$, and ES shows a similar jump in the "paint" domain when $T = 10^6$; whereas CEC always get smoothly decreasing prediction error. This partially illustrates that CEC are numerically more stable than EC and ES.

It should be noticed that, in the "maze4x3" domain (see Fig. 4-c), the training CPU-time for EC and CEC-A,B show a significant drop when the training data size increases from $T = 10^{5.67}$ to $T = 10^6$. — This is not a coincidence, because we repeated the whole experiment for 10 times, with the matlab's `rand()` function initialized by different seeds; and always got similar results. As the authors can see, one possible reason for this is that, for training sequences of length $\geqslant 10^6$, the counting matrices are already very near to the "true" probability matrices and so have (numerical) rank equaling to the model dimension $m = 11$; this might speed up the SVD of the counting matrix $\underline{V}^{\text{cmp}}$ — only the first 11 singular values and vectors need to be computed, and reduce the number of EC-iterations needed for optimizing the two auxiliary matrices C and Q — one sees that in the ideal case where the counting matrix has rank m, only one EC-iteration is needed.

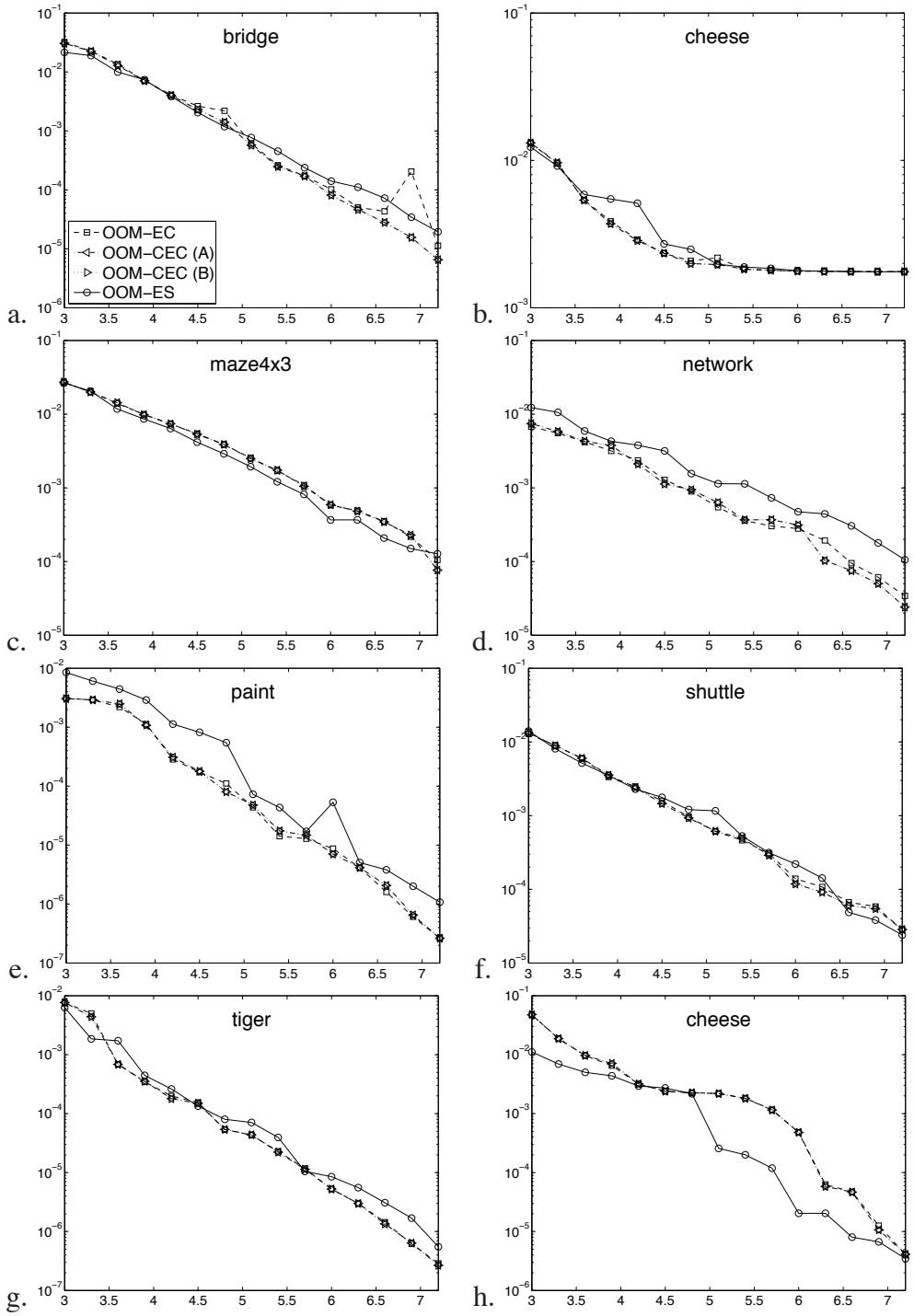

Figure 3: The average one-step prediction error of the ES, EC and CEC-A,B algorithms on seven POMDP benchmark problems (x-axis: ($\log_{10}$ of) length of training sequences; y-axis: average one-step predition error). Panel **h.** is re-run of **b.** with different parameter settings, see text.

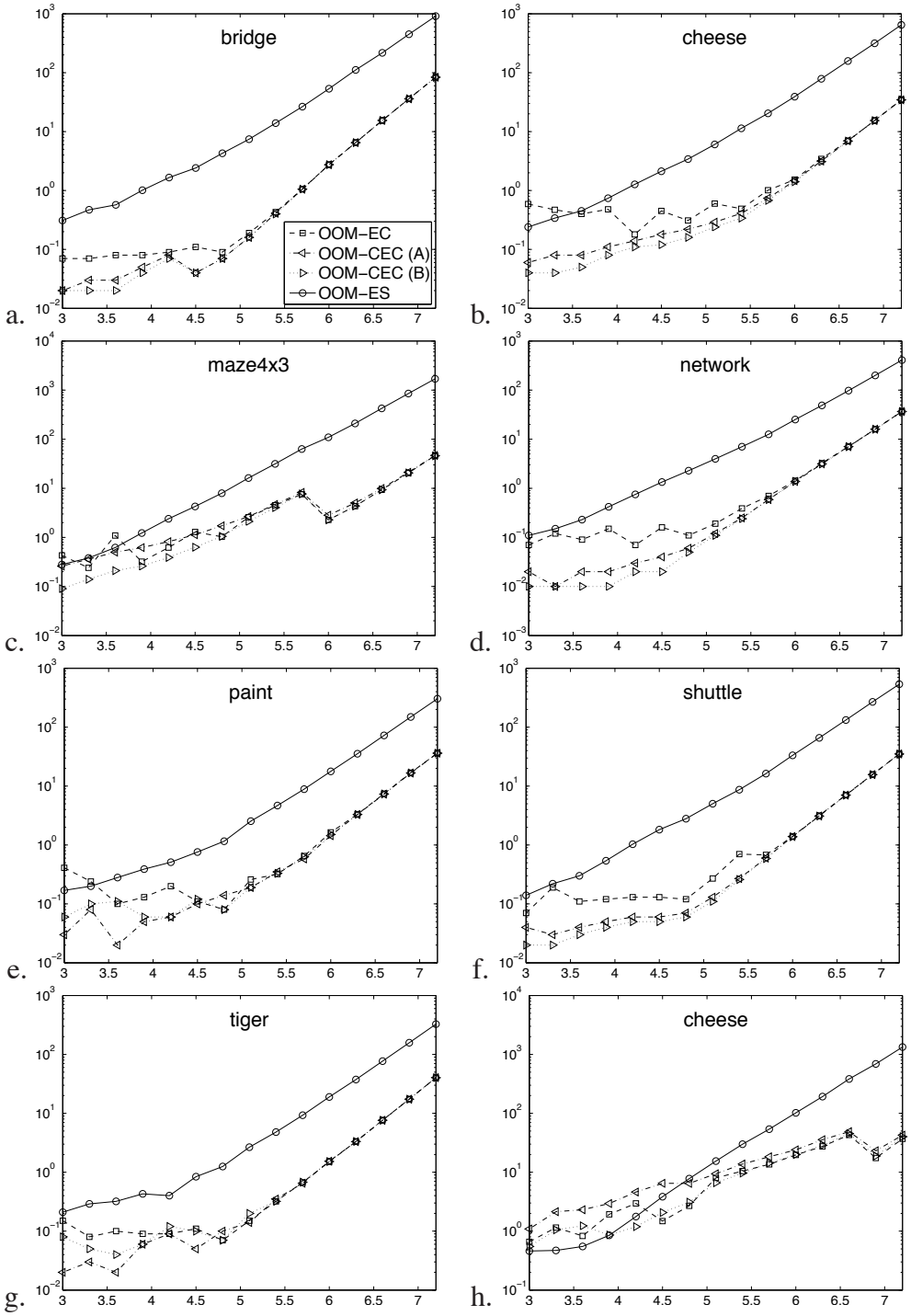

Figure 4: The training time costed by the ES, EC and CEC-A,B algorithms on seven POMDP benchmark problems (x-axis: ($\log_{10}$ of) length of training sequences; y-axis: training CPU-time). Panel **h.** is re-run of **b.** with different parameter settings, see text.

Modeling text sources. In the third experiment OOMs will be taken to model a text source, the short story "The 1,000,000 Pound Bank-Note" written by Mark Twain. To simplify the task we first preprocessed the text string by deleting all special characters except the blank, changing capital letters to small ones; which gave us an alphabet of 27 symbols: `a`, `b`, ..., `z` plus the blank. We then sorted the resulting string sentence-wise, obtaining two substrings of roughly equal length (21042 and 20569 symbols, respectively) that were used as training and test sequences.

As before, we estimated OOMs of dimensions $\{5, 10, \ldots, 50, 60, \ldots, 100, 120, 150\}$ from the training sequence by the EC, ES and CEC-A,B algorithms, respectively; and then computed the training and test NLLs of each learnt OOM, as shown in Fig. 5-a,b. Here the length of characteristic/indicative strings is set to be $k = 2$. We also plotted the training time of each learning algorithm versus model dimension in Fig. 6. From these figures we see that OOMs learnt by the four algorithms have nearly the same training and test NLLs (ES is a litter better for low dimensional models, but becomes worse when model dimension increases). Among the four algorithms CEC-B is the fastest one: it is about 2 times faster than CEC-A, which is in turn 10–100 times faster than ES and EC.

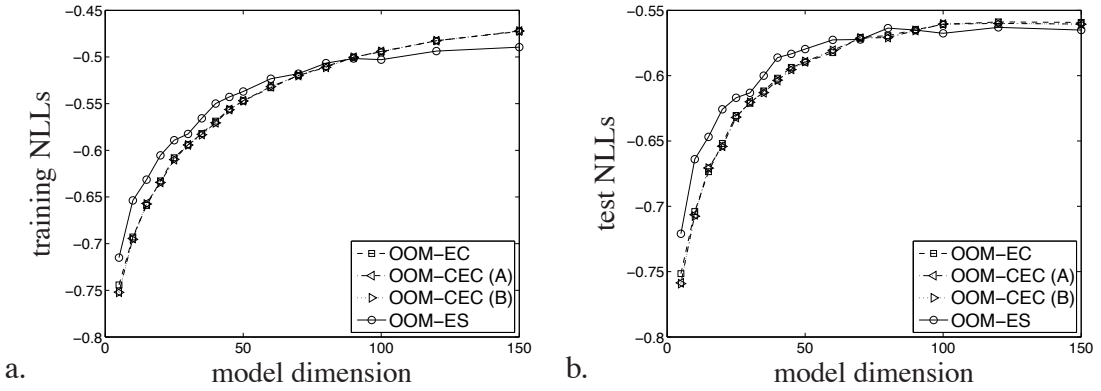

Figure 5: The training (a.) and test (b.) NLLs of learnt OOMs on the short fiction "The 1,000,000 Pound Bank-Note".

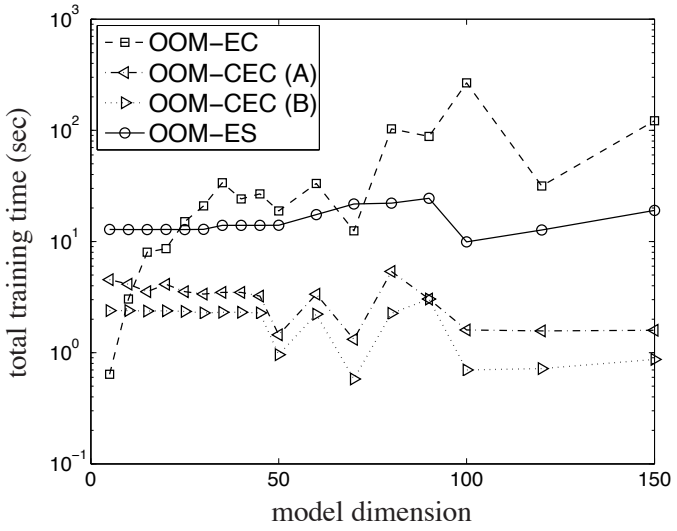

Figure 6: The training time of the ES, EC and CEC-A,B algorithms on the short fiction "The 1,000,000 Pound Bank-Note".

5 Conclusions and Discussion

We have derived the constructive error controlling algorithm (CEC-A), the first constructive algorithm for learning OOMs. We also proposed an efficient variant of CEC-A, called CEC-B. Our numerical experiments reveal that

- as expected, the two CEC algorithms indeed improve over their predecessor, the EC algorithm, in most domains: CEC usually learn more accurate models (at least not worse) than EC; when learning high dimensional models, CEC are much faster than EC;
- compared with the ES algorithm, both CEC algorithms are very much faster (especially when learning models from large datasets), while still showing comparable test accuracy;
- for large datasets, the two CEC algorithms have nearly the same performance; but for small datasets, CEC-B is about 2–3 times faster than CEC-A.

We conclude with some open questions:

- Like the EC algorithm, currently the CEC algorithm is given the desired model dimension and a (fixed) basic string length as parameters. This will have to be automated in the future; also, the CEC algorithm should be generalized to become,

like the ES algorithm, based on variable-length subsequence counting statistics, instead of the fixed-length statistics of the current version.

- As pointed out in (Zhao and Jaeger, 2009), a mathematical understanding of the relationships between the algebraic and the statistical aspects of EC/CEC vs. ES, or of the algebraic working principle vs. the statistical effects, is a theoretically intriguing question.

- As mentioned at the end of Section 3, all OOM learning algorithms (and the OOM theory itself) suffer from the negative probability problem. As the undecidability result of (Wiewiora, 2008) shows, this is an intrinsically hard problem in the current *linear* framework of OOMs. In response to this issue, the first author is developing and investigating *nonlinear* variants of OOMs that are free from this negativity problem by design (Zhao and Jaeger, 2007).

A Proofs and Algorithms

A.1 Proof of Proposition 2

If C_1 and Q_1 are fixed, then, as C_2 and Q_2 are independent of each other, we can split the problem (16) into two simpler problems, namely,

$$\min_{C_2}\{\|C_2\|^2 \ : \ \mathbf{1}_m^\mathsf{T} C_2 = \mathbf{1}_N^\mathsf{T} L_2\} \quad \text{and} \quad \min_{Q_2} \|[Q_1; Q_2]\|_2 \,.$$

Assume the i-th column of L_2 is $\boldsymbol{b}_i$ and denote by $\boldsymbol{y}_i$ the i-th column of C_2, then the above first problem is equivalent to $\min_{\boldsymbol{y}_i}\{\|\boldsymbol{y}_i\| \ : \ \mathbf{1}_m^\mathsf{T}\boldsymbol{y}_i = \mathbf{1}_N^\mathsf{T}\boldsymbol{b}_i\}$, $(\forall\, i)$, since $\|C_2\|^2 = \sum_i \|\boldsymbol{y}_i\|^2$. By the Cauchy-Schwarz inequality, we know $|\mathbf{1}_m^\mathsf{T}\boldsymbol{y}_i| \leqslant \|\mathbf{1}_m\| \cdot \|\boldsymbol{y}_i\|$, i.e., $\|\boldsymbol{y}_i\| \geqslant \frac{1}{\sqrt{m}}|\mathbf{1}_N^\mathsf{T}\boldsymbol{b}_i|$; with equality if and only if $\boldsymbol{y}_i = \alpha\mathbf{1}_m$, which, together with the constraint $\mathbf{1}_m^\mathsf{T}\boldsymbol{y}_i = \mathbf{1}_N^\mathsf{T}\boldsymbol{b}_i$, implies that $\boldsymbol{y}_i = (\frac{1}{m}\mathbf{1}_N^\mathsf{T}\boldsymbol{b}_i)\mathbf{1}_m$. It then follows that $C_2 = \frac{1}{m}\mathbf{1}_m\mathbf{1}_N^\mathsf{T}L_2$ is the unique minimizer of the first problem.

To prove that $Q_2 = 0$ minimizes $\|[Q_1; Q_2]\|_2$, we will use an alternative definition of spectral norm, viz $\|A\|_2 = \max\{\|A\boldsymbol{x}\| \ : \ \|\boldsymbol{x}\| = 1\}$. Let $\boldsymbol{x}$ be such that $\|\boldsymbol{x}\| = 1$ and $\|[Q_1; 0]\|_2 = \|[Q_1; 0]\boldsymbol{x}\|$, then $\|[Q_1; Q_2]\|_2^2 \geqslant \|[Q_1; Q_2]\boldsymbol{x}\|^2 = \|[Q_1\boldsymbol{x}; Q_2\boldsymbol{x}]\|^2 \geqslant \|[Q_1\boldsymbol{x}; 0]\|^2 = \|[Q_1; 0]\boldsymbol{x}\|^2 = \|[Q_1; 0]\|_2^2$ and so $\|[Q_1; Q_2]\|_2 \geqslant \|[Q_1; 0]\|_2$, which is what we want to prove. □

A.2 Proof of Proposition 3

Assume $C_1 D$ has the compact SVD $C_1 D = LSR^\mathsf{T}$, then $L^\mathsf{T} L = I_m$ and the constraint $C_1 D Q_1 = I_m$ now reads $LSR^\mathsf{T} Q_1 = I_m$. It follows that $SR^\mathsf{T} Q_1 L = L^\mathsf{T} LSR^\mathsf{T} Q_1 L = L^\mathsf{T} L = I_m$ and so $R^\mathsf{T} Q_1 L = S^{-1}$, which in turn implies that $RR^\mathsf{T} Q_1 LL^\mathsf{T} = RS^{-1} L^\mathsf{T} = (C_1 D)^\dagger$. This identity, together with the facts that $\|L\|_2 = \|R\|_2 = 1$; $\|A\|_2 = \|A^\mathsf{T}\|_2$; and $\|AB\|_2 \leqslant \|A\|_2 \cdot \|B\|_2$, implies that $\|(C_1 D)^\dagger\|_2 \leqslant \|Q_1\|_2$, which completes the proof. □

A.3 Simple Construction of Two Othogonal Matrices

Here we consider the following simple problems in linear algebra: given two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ with unit norm, (1) how to (efficiently) construct an orthogonal matrix $L \in \mathbb{R}^{n \times n}$ such that $L\boldsymbol{x} = \boldsymbol{y}$? (2) how to find a basis of the null space of $\boldsymbol{x}^\mathsf{T}$, i.e., to find a matrix $H \in \mathbb{R}^{n \times (n-1)}$ satisfying $H^\mathsf{T} H = I_{n-1}$ and $\boldsymbol{x}^\mathsf{T} H = 0$? As one can easily see, such L and H are not unique, so we add an extra condition on L, namely, it should be as close to I_n as possible, in the sense that $\|L - I_n\|$ is minimized. As $L^\mathsf{T} L = I_n$, we have $\|L - I_n\|^2 = 2\,\mathrm{tr}(I_n - L)$. Thus, by brute-force computation,

$$L = \begin{cases} I_n - \dfrac{(\boldsymbol{y} - \boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x})^\mathsf{T}}{1 - \boldsymbol{x}^\mathsf{T} \boldsymbol{y}} & \text{if } \boldsymbol{x} \neq \boldsymbol{y}\,, \\ I_n & \text{if } \boldsymbol{x} = \boldsymbol{y}\,. \end{cases} \tag{29}$$

In the above formula, putting $\boldsymbol{y} = [0, \ldots, 0, 1]^\mathsf{T}$ and extracting the first $(n-1)$ columns of L, we get the matrix H.

Acknowledgement: The results reported in this paper were achieved when the first author worked under a grant (JA 1210/1-1&2) from the Deutsche Forschungsgemeinschaft (DFG).

References

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

Bowling, M., McCracken, P., James, M., Neufeld, J., and Wilkinson, D. (2006). Learning

predictive state representations using non-blind policies. In *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.

Golub, G. H. and Loan, C. F. v. (1996). *Matrix computations*. Johns Hopkins University Press, 3rd edition.

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. (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.-J., 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.

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

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.

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

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

Wiewiora, E. W. (2008). *Modeling probability distributions with predictive state representation*. PhD thesis, University of California, San Diego.

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 *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 *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.

Zhao, M.-J. and Jaeger, H. (2009). A bound on modeling error in observable operator models and an associated learning algorithm. *Accepted by Neural Computation*, 00(0):00–00.