Making the Error-Controlling Algorithm of Observable Operator Models Constructive

Ming-Jie Zhao
mingjie.zhao@gmail.com
Jacobs University Bremen gGmbH, Bremen 28759, Germany, and
Graz University of Technology, Graz A-8010, Austria

Herbert Jaeger
h.jaeger@jacobs-university.de
Michael Thon
m.thon@jacobs-university.de
Jacobs University Bremen gGmbH, Bremen 28759, Germany

Observable operator models (OOMs) are a class of models for stochastic processes that properly subsumes the class that can be modeled 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 that yields (and minimizes) an assured upper bound on the modeling error. The run time 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 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 modeling error but is constructive where EC needs iterations. As a consequence, we obtain further gains in learning speed without loss in modeling accuracy.

1 Introduction

Observable operator models (Jaeger, 2000b) are a mathematical model class for stochastic processes that generalizes hidden Markov models (HMMs) (Bengio, 1999) and can be represented in a structurally 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
Constructive Error-Controlling Algorithm for OOMs

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 \textit{characterizer} and the \textit{indicator} (Zhao, Jaeger, & Thon, 2009) and will be denoted by $C$ and $Q$, respectively, throughout this letter.

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, 2000b) relied on simple heuristics for creating $C$ and $Q$, and the ensuing poor statistical efficiency made OOMs learned in this way no match for HMMs trained with the expectation-maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977). Kretzschmar (2003) 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 $Q$ have become increasingly better understood.

After Kretzschmar’s lead work, two theoretically justified and computationally accessible optimality criteria for designing $C$ and $Q$ were 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, Jaeger, & Thon, 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 perhaps a small advantage for EC (Zhao, Jaeger, & Thon, 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 has this attractive property, for both employ an iterative procedure to compute the auxiliary matrices $C$ and $Q$. In this letter, following the road of EC, we derive a new upper bound of modeling error that is tighter than the one used 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 that is constructive (hence, fast),
asymptotically correct (the hallmark of all OOM learning algorithms), and 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, Jaeger, and Thon (2009). While self-contained, this letter 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 first briefly reintroduce 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, Jaeger, and Thon (2009) and document the empirical results in section 4. Finally, we summarize the letter 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:

\[
\{ \Pr(X_1 = a_1, \ldots, X_n = a_n) =: P(a_1a_2 \ldots a_n) : n \in \mathbb{N}, a_i \in O \}.
\]

Following the notation convention of Zhao, Jaeger, and Thon (2009), we use lowercase letter 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 \( \epsilon \). The above family of joint probabilities can then be 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 \( (\mathbb{R}^m, \{\tau_a\}_{a \in O}, \mathbf{w}_0) \) 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 \( \mathbf{w}_0 \in \mathbb{R}^m \) such that for any finite sequence \( \bar{a} = a_1a_2 \ldots a_n \in O^* \), it holds that

\[
P(\bar{a}) = \mathbf{1}_m^T \tau_{a_n} \cdots \tau_{a_2} \tau_{a_1} \mathbf{w}_0 =: \mathbf{1}_m^T \tau_{\bar{a}} \mathbf{w}_0 ,
\]

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 equation 2.1, the objective is to estimate an OOM \( (\mathbb{R}^m, \{\hat{\tau}_a\}_{a \in O}, \hat{\mathbf{w}}_0) \) of \( (X_n) \) from one of its finite initial realizations (say, \( \bar{s} = s_1s_2 \ldots s_T \)), which
can be used to approximately reproduce the distribution of \((X_n)\), that is, \(P(\tilde{a}) \approx 1^n_k \tilde{t}_a \tilde{\omega}_0\). A general constructive algorithm (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 \(\{\tilde{a}_j\}_{j=1}^K\) and \(\{\tilde{b}_i\}_{i=1}^K\) of basic strings (\(\tilde{a}_j\)’s are called indicative strings and \(\tilde{b}_i\)’s characteristic 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 \(V^*\) and \(W^*\) \((\forall a \in O)\) with their \((i, j)\)th entry determined by

\[
[V^*]_{ij} = \frac{\#\text{but last}(\tilde{a}_j \tilde{b}_i)}{T - 2k}; \\
[W^*]_{ij} = \frac{\#(\tilde{a}_j \tilde{a} \tilde{b}_i)}{T - 2k},
\]

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

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 \(1^m_m C = 1^1_k\) and the \(m \times m\) matrix \(CVQ\) is invertible.

3. Obtain an (asymptotically correct) estimate of an OOM for the process \((X_n)\) by computing observable operators by \(\tilde{t}_a = (CW^*_a Q)(CVQ)^{-1}\) and an initial vector by solving the equations \(1^m_m \tilde{\omega}_0 = 1\) and \((\sum_{a \in O} \tilde{t}_a) \tilde{\omega}_0 = \tilde{\omega}_0\).

We begin with a historical remark. Kretzschmar (2003) derived the learning equations \(\tilde{t}_a = (CW^*_a Q)(CVQ)^{-1}\) and, noting the importance of the matrix \((CVQ)\) in the above basic learning scheme, suggested that the design of \(C\) and \(Q\) should make the condition number \(\kappa = \|CVQ\|_2 \cdot \|(CVQ)^{-1}\|_2\) 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 apparently reasonable at first glance, is actually misguided in the sense that it is always possible to obtain \(\kappa = 1\), the smallest possible value of \(\kappa\), by first creating an arbitrary \(C \in \mathbb{R}^{m \times K}\) such that \(1^m_m C = 1^1_k\) and that \(CV\) has full rank \(m\), and then putting \(Q = (CV)^\dagger\), the pseudo-inverse of \((CV)\): with \(C\) and \(Q\) constructed in this way, we know \(CVQ = I_m\) (the identity matrix of order \(m\)) and hence \(\kappa = 1\).

An alternative algebraic criterion for optimizing the matrices \(C\) and \(Q\), which does not fall prey to this fallacy on this direction, has been developed in our previous work. It is based on two findings. First, if the probability matrices (the “true” normalized counting matrices) \(\bar{V}^*\) and \(\bar{W}^*\)'s,

\[
[V^*]_{ij} = P(\tilde{a}_j \tilde{b}_i); \\
[W^*]_{ij} = P(\tilde{a}_j \tilde{a} \tilde{b}_i), \quad (i, j = 1, 2, \ldots, K),
\]

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

\[
\frac{\| \tau - \hat{\tau} \|}{\| \tau \|} \leq \| C \| \cdot \| Q(CVQ)^{-1} \| \cdot (\| E_V \| + \ell \cdot \| E_W \|),
\]

where, for any matrix \( A, \| A \| \) denotes its Frobenius norm \( \| A \| := \sqrt{\text{tr}(A^T A)} \); \( \tau \) is the tall matrix formed by stacking all \( \tau_a \)'s one above another:

\[
\tau = [\tau_1; \ldots; \tau_\ell],
\]

in Matlab’s notation; and \( \hat{\tau}, W_a \), and \( W^* \) are tall matrices constructed, respectively, from \( \tau_a \)'s, \( W_a \)'s, and \( W^* \)'s by the same stacking scheme; \( E_V := V - V^* \) and \( E_W := W - W^* \) are the error matrices of \( V \) and \( W \), respectively; and (recall that) \( \ell \) is the alphabet size.

Inequality 2.3 is the theoretical foundation of the EC algorithm, from which one immediately sees that one possible way to control the modeling error \( \| \tau - \hat{\tau} \| \) is to minimize the quantity \( \kappa_1 = \| C \| \cdot \| Q(CVQ)^{-1} \| \). This criterion, together with constraint \( 1_m^T C = 1_K^T \), forms the optimization problem,

\[
\min_{C \in \mathbb{R}^{m \times K}, Q \in \mathbb{R}^{K \times m}} \left\{ \kappa_1 = \| C \| \cdot \| Q(CVQ)^{-1} \| : 1_m^T C = 1_K^T \right\},
\]

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

\[
Q^{(t)} = (C^{(t-1)}V)^\dagger; \quad C^{(t)} = \left( I_m - \frac{1}{m} 1_m 1_m^T \right) (VQ^{(t)})^\dagger + \frac{1}{m} 1_m 1_K^T,
\]

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

In sum, the error-controlling algorithm presented in Zhao, Jaeger, and Thon (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 equation 2.5. In the next section, we derive an improved version of EC in which the iterative procedure 2.5 is replaced by a globally optimal analytical solution to an optimization problem that is very similar to equation 2.4.

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 $Q$ is described, yielding an improved version of EC that, to our knowledge, is the first constructive learning algorithm of OOMs. To this end, we introduce another upper bound of the relative error $\frac{\|\hat{\tau} - \tau\|}{\|\tau\|}$, which is tighter than equation 2.3. We state this upper bound as a proposition.

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

$$\frac{\|\tau - \hat{\tau}\|}{\|\tau\|} \leq \|C\| \cdot \|Q(C V Q)^{-1}\|_2 \cdot (\|E_V\|_2 + \ell \cdot \|E_W\|_2). \quad (3.1)$$

**Proof.** From the proof to proposition 3 of Zhao, Jaeger, and Thon (2009), we get

$$\tau - \hat{\tau} = (\tau C E_V - C_{\text{big}} E_W) \cdot Q(C V Q)^{-1}, \quad (3.2)$$

where $C_{\text{big}} := \text{diag}(C, \ldots, C)$ ($\ell$ copies of $C$); using equation 2.3 and the definition of $E_V$ and $E_W$, one can also directly verify this identity. To establish inequality 3.1, we invoke a well-known matrix inequality (see, e.g., Golub & Loan, 1996):

$$\|AB\| \leq \min\{\|A\| \cdot \|B\|_2, \|A\|_2 \cdot \|B\|\}, \quad (3.3)$$

It follows from equations 3.2 and 3.3 that

$$\|\tau - \hat{\tau}\| \leq (\|\tau\| \cdot \|C\| \cdot \|E_V\|_2 + \|\tau\| \cdot \|E_W\|_2) \cdot \|Q(C V Q)^{-1}\|_2.$$

But in proposition 3 of Zhao, Jaeger, and Thon (2009), we have already proven $\|\tau\| \geq \frac{1}{\sqrt{\ell}}$ and $\|C_{\text{big}}\| = \sqrt{\ell} \cdot \|C\|$, so inequality 3.1 follows.

Note that for any matrix $A$, $\|A\|_2$ equals the largest singular value of $A$, and $\|A\|$ is the square root of the square sum of all singular values of $A$. So $\|A\|_2 \leq \|A\|$, and equation 3.1 indeed provides a tighter upper bound of $\frac{\|\tau - \hat{\tau}\|}{\|\tau\|}$ than equation 2.3.

Inequality 3.1 provides us with another algebraic criterion for computing the optimal $C$ and $Q$ that is very similar to equation 2.4:

$$\min_{C \in \mathbb{R}^{m \times K}, Q \in \mathbb{R}^{K \times m}} \left\{ \kappa_2 = \|C\| \cdot \|Q(C V Q)^{-1}\|_2 \right. : \left. 1^T_m C = 1^T_K \right\}. \quad (3.4)$$

Zhao, Jaeger, and Thon (2009) demonstrated that adding the extra constraint $C V Q = I_m$ to problem 2.4 will not change its minimizer. Similarly,
here we can prove the optimization problem 3.4 is essentially equivalent to

$$\min_{C \in \mathbb{R}^{m \times K}, Q \in \mathbb{R}^{K \times m}} \left\{ \kappa_2 = \|C\|_2 \|Q\|_2 : 1_m^T C = 1_K^T, C V Q = I_m \right\},$$  

(3.5)

where the expression of $\kappa_2$ has been simplified by using $C V Q = I_m$.

To see this, for any minimizer $(C, Q)$ of problem 3.4, we put $C_0 = C$ and $Q_0 = Q(C V Q)^{-1}$, then $C_0 V Q_0 = I_m$ and $\kappa_2(C, Q) = \kappa_2(C_0, Q_0)$, which means $(C_0, Q_0)$ also minimizes the quantity $\kappa_2$ and, additionally, satisfies the constraint $C V Q = I_m$.

Before introducing the constructive method for minimizing equation 3.5, we make some remarks on problem 3.5 and the normalized counting matrices $V$ and $W_a$ obtained from the training sequence $\bar{s} = s_1 s_2 \ldots s_T$ via equation 2.2. Note that some of these remarks have already been presented in Zhao, Jaeger, and Thon (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 $\hat{\tau}_a = (C W_a Q)(C V Q)^{-1}$, the normalized counting matrices can be replaced by the raw counting matrices $V_{\text{raw}}$ and $W_{a_{\text{raw}}}$'s, that is, $\hat{\tau}_a = (C W_{a_{\text{raw}}} Q)(C V_{\text{raw}} Q)^{-1}$, where

$$[V_{\text{raw}}]_{ij} = \#\text{but last}(\bar{a}_j \bar{b}_i); \quad [W_{a_{\text{raw}}}]_{ij} = \#(\bar{a}_j a \bar{b}_i).$$  

(3.6)

It is also clear that replacing $V$ by $V_{\text{raw}}$ in problem 3.5 does not change its minimizer $(C, Q)$, although the minimal value of $\kappa_2$ 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 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 learned, and the optimization problem 3.5 becomes computationally inaccessible quickly. Nevertheless, first, equation 3.6 reveals that, given the sequence $\bar{s} = s_1 s_2 \ldots s_T$, the element sums of $V_{\text{raw}}$ and $\sum_{a \in O} W_{a_{\text{raw}}}$ are $(T - 2k)$, so for large $k$, the matrices $V_{\text{raw}}$ and $W_{a_{\text{raw}}}$ are actually very sparse; second, the learning equations reveal that, discarding zero rows (columns) in $V_{\text{raw}}$, the corresponding rows (columns) in $W_{a_{\text{raw}}}$'s (we call the resulting matrices the compressed counting matrices and denote them by $V_{\text{cmp}}$ and $W_{a_{\text{cmp}}}$) and the corresponding columns in $C$ (rows in $Q$) will not change the estimation $\hat{\tau}_a$; third, those “undiscarded” columns of $C$ (rows of $Q$), can be efficiently optimized by solving a compressed version of equation 3.5, 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 Zhao, Jaeger, and Thon (2009) pointed out, 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 \( \{ \tilde{b}_i \} \) should form a “cover” of some set \( O^k \), in the sense that for each string \( \tilde{c} \) from \( O^k \), there is a unique \( \tilde{b}_i \), which is a prefix of \( \tilde{c} \). Moreover, by using compact suffix trees (Gusfield, 1997), we can construct the compressed counting matrices \( V \) and \( W_a \) from the training sequence \( \bar{s} = s_1 s_2 \ldots s_T \) in linear time, that is, \( O(T) \) flops (Ukkonen, 1995).

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

\[
\tilde{t}_a = (C_{\text{cmp}} W_a^{\text{cmp}} Q^{\text{cmp}}) \cdot (C_{\text{cmp}} V_{\text{cmp}} Q^{\text{cmp}})^{-1},
\]

(3.7)

where \( C_{\text{cmp}} \in \mathbb{R}^{m \times N} \) (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 \( V_{\text{raw}} \). It remains to find a way to compute \( C_{\text{cmp}} \) and \( Q_{\text{cmp}} \), which is efficient in memory space. Equation 3.5 is too inefficient to be directly used to compute \( C \) and \( Q \) and then extract \( C_{\text{cmp}} \) and \( Q_{\text{cmp}} \). To simplify the notation, we move \( V_{\text{cmp}} \) to the upper-left area of \( V_{\text{raw}} \) (see equation 3.6) by reordering the basic strings \( \{ \tilde{a}_j \} \) and \( \{ \tilde{b}_i \} \), making the other entries of \( V_{\text{raw}} \) be zero. Correspondingly, the matrices \( C, Q \) can now be written as \( \overline{C} = [C_{\text{cmp}}, C_0] \) and \( \overline{Q} = [Q_{\text{cmp}}; Q_0] \) (in Matlab’s notation), respectively, with \( C_0 \in \mathbb{R}^{n \times (K-N)} \) and \( Q_0 \in \mathbb{R}^{(K-M) \times m} \). The optimization problem 3.5 is therefore equivalent to

\[
\min \kappa_2^2 = (\|C_{\text{cmp}}\|^2 + \|C_0\|^2) \cdot \|Q_{\text{cmp}}; Q_0\|_2^2
\]

s. t. \( 1^T_m C_{\text{cmp}} = 1^T_N, \quad 1^T_m C_0 = 1^T_{K-N}, \quad C_{\text{cmp}} V_{\text{cmp}} Q_{\text{cmp}} = I_m. \) (3.8)

By proposition 2 (see below), we know \( C_0 = \frac{1}{m} 1_m 1^T_{K-N} \) 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 problem 3.5:

\[
\min \kappa_2^2 = \left(\|C_{\text{cmp}}\|^2 + \frac{K-N}{m}\right) \cdot \|Q_{\text{cmp}}\|_2^2
\]

s. t. \( 1^T_m C_{\text{cmp}} = 1^T_N, \quad C_{\text{cmp}} V_{\text{cmp}} Q_{\text{cmp}} = I_m. \) (3.9)

### 3.1 An Analytical Solution to Problem 3.9.

To simplify the notations, we first rewrite the problem but with the superscript \( \text{cmp} \) dropped:

\[
\min_{C, Q} \left\{ \kappa_2^2 = \left(\|C\|^2 + \frac{K-N}{m}\right) \cdot \|Q\|_2^2 : \quad 1^T_m C = 1^T_N, \quad C V^T Q = I_m \right\},
\]

(3.10)

\[\overset{1}{\text{For any matrix } A, \| A \|^2 \text{ is the square sum of its entries, so } |||C_{\text{cmp}}, C_0|||^2 = \|C_{\text{cmp}}\|^2 + \|C_0\|^2.}\]
where $V \in \mathbb{R}^{N \times M}$ is 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 $V$, say, $V = L \cdot D \cdot R^\top$, and putting $\underline{C} = CL$ and $\underline{Q} = R^\top Q$, we get a simplified form (the matrix $V$ now becomes the diagonal one $D$) of equation 3.10:

$$\min_{C, Q} \left\{ \kappa_2^2 = \left( \|C\|_2^2 + \frac{K - N}{m} \right) \cdot \|Q\|_2^2 : \mathbf{1}_m^\top C = \mathbf{1}_N^\top L, \ C \cdot D \cdot Q = I_m \right\},$$

as applying a unitary transformation on a matrix will not change its spectral norm and Frobenius norm. Now assume that $V$ has rank $r$—$\underline{D} = \text{diag} \{D, 0\}$—where $D = \text{diag} \{d_1, d_2, \ldots, d_r\}$ with $d_1 \geq d_2 \geq \cdots \geq d_r > 0$. Then $V$ has the compact SVD $V = L_1 DR_1^\top$, where $L_1$ and $R_1$ are tall matrices formed by extracting the first $r$ columns from $L$ and $R$, respectively. Writing $L = [L_1, L_2]$ and $R = [R_1, R_2]$, and utilizing the fact that $\|[A, B]\|_2^2 = \|A\|_2^2 + \|B\|_2^2$, we expand the above optimization problem to

$$\min \kappa_2^2 = \left( \frac{K - N}{m} + \|C_1\|_2^2 + \|C_2\|_2^2 \right) \cdot \|[Q_1; Q_2]\|_2^2$$

s. t. $\mathbf{1}_m^\top C_1 = \mathbf{1}_N^\top L_1, \ \mathbf{1}_m^\top C_2 = \mathbf{1}_N^\top L_2, \ C_1 D Q_1 = I_m$, \hspace{1cm} (3.11)

where $C_i := CL_i$ and $Q_i := R_i^\top Q (i = 1, 2)$.

**Proposition 2.** In problem 3.11, let the matrices $C_1$ and $Q_1$ be fixed. Then $C_2 = \underline{\mathbf{1}}_m^\top \underline{\mathbf{1}}_N^\top L_2$ and $Q_2 = 0$ form a minimizer of the target $\kappa_2^2$.

See section A.1 for the proof. According to the above proposition, we can rewrite problem 3.11 as

$$\min \kappa_2^2 = (\|C_1\|_2^2 + \phi) \cdot \|Q_1\|_2^2$$

s. t. $\mathbf{1}_m^\top C_1 = I_\tau, \ C_1 D Q_1 = I_m$, \hspace{1cm} (3.12)

where $I_\tau := \mathbf{1}_N^\top L_1$ and $\phi := \frac{K - N}{m} + \frac{1}{m} \mathbf{1}_m^\top \mathbf{1}_N^\top L_2 L_2^\top \mathbf{1}_N^\top = \frac{K - N}{m} + \frac{1}{m} \mathbf{1}_N^\top (I_N - L_1 L_1^\top) \mathbf{1}_N$.

Here we should point out that to get problem 3.12 from 3.10 requires only computing the compact SVD $V = L_1 D R_1^\top$. This is because we can compute the quantity $\phi$ from $L_1$ by

$$\phi = \frac{K - N}{m} + \frac{1}{m} \mathbf{1}_N^\top L_2 L_2^\top \mathbf{1}_N = \frac{K - N}{m} + \frac{1}{m} \mathbf{1}_N^\top (I_N - L_1 L_1^\top) \mathbf{1}_N = \frac{1}{m}(K - I_\tau I);$$
and retrieve the optimal $C$ and $Q$ from a solution $(C_1, Q_1)$ to problem 3.12 by

$$C = [C_1, C_2][L_1, L_2]^T = \left(C_1 - \frac{1}{m}1_m1_m^T\right)L_1^T + \frac{1}{m}1_m1_n^T,$$

$$Q = [R_1, R_2][Q_1; Q_2] = R_1Q_1.$$

**Proposition 3.** *In the optimization problem 3.12, if $C_1$ is fixed, then $Q_1 = (C_1D)^\dagger$ is a minimizer of $\kappa_2^2$. 

See section A.2 for the proof. Assume $C_1D$ has the compact SVD $C_1D = LSR\dagger$, then $C_1 = LSR\dagger L^{-1}$ and, by proposition 3, $Q_1 = RS^{-1}L\dagger$. For this setting of $(C_1, Q_1)$, the second constraint $C_1DQ_1 = I_m$ is automatically fulfilled, and the first constraint $1_mC_1 = I\dagger$ now becomes $1_mLSR\dagger = I\dagger D$, which implies that (1) the subspace (in $\mathbb{R}^r$) spanned by columns of $R$ (the range space of $R$) contains the vector $DL$, that is, $Rx = DL$ for some $x \in \mathbb{R}^m$; and (2) by multiplying $R$ on the right to this equality, $1_m^T L = I\dagger DR$.

Furthermore, in the compact SVD $C_1D = LSR\dagger$, let $r_i$ be the $i$th column of $R$ and $\beta_i = r_i^TD^{-2}r_i$ and write $S = \text{diag}\{s_1, s_2, \ldots, s_m\}$ with $s_1 \geq s_2 \geq \cdots \geq s_m > 0$. Then, by brute force computation, we obtain

$$\|C_1\|^2 = \sum_{i=1}^{m}\beta_is_i^2; \quad \|Q_1\|^2 = s_m^{-2}; \quad \kappa_2^2 = s_m^{-2}\left(\phi + \sum_{i=1}^{m}\beta_is_i^2\right).$$

Now assume the matrix $R \in \mathbb{R}^{r \times m}$, which satisfies $R^\dagger R = I_m$ and $1_m^T LSR\dagger = I\dagger D$, is already known. Then the vector $I\dagger DR =: v^T = [v_1, v_2, \ldots, v_m]$ and the scalars $\beta_i = r_i^TD^{-2}r_i$ are also known. Moreover, by the equality $1_m^T L = I\dagger DR$, the vector $v$ has the property

$$\sum_{i=1}^{m}v_i^2 = v^T v = I\dagger DRR^\dagger DI = 1_m^T LSR^\dagger DI = I\dagger D^2I, \quad (3.13)$$

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

Writing $1_m^T L =: u^T = [u_1, u_2, \ldots, u_m]$, we get $u_is_i = v_i$ and so $u_i = v_i/s_i$ from the constraint $1_m^T L = I\dagger DR = v$. Since $L$ is an orthogonal matrix, we know

$$\sum_{i=1}^{m}(v_i/s_i)^2 = \sum_{i=1}^{m}u_i^2 = u^T u = 1_m^T LLL^\dagger 1_m = 1_m^T 1_m = m. \quad (3.14)$$

Note that this is the only condition $u$ should satisfy since we can choose $L$ freely.
Summing up the above discussion, we get the expanded “scalar” version of problem 3.12:

$$
\begin{align*}
\min & \quad \kappa_2^2 = s_m^{-2} \left( \phi + \sum_{i=1}^{m} \beta_i s_i^2 \right) \\
\text{s. t.} & \quad \sum_{i=1}^{m} (v_i/s_i)^2 = m, \quad s_1 \geq s_2 \geq \cdots \geq s_m > 0.
\end{align*}
$$

(3.15)

For each $i$, let $\alpha_i = s_i/s_m$. Then $s_i = \alpha_i s_m$ and $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m = 1$. 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,

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

from which we immediately see that $\kappa_2^2$ reaches its minimal value

$$
(\kappa_2^2)_{\min} = m\phi \cdot \left( \sum_{i=1}^{m} v_i^2 \right)^{-1} + \sum_{i=1}^{m} \beta_i = m\phi \cdot (I^T D^2 I)^{-1} + \sum_{i=1}^{m} \beta_i
$$

(3.16)

when $\alpha_i = 1$—$s_i = s_m$ for all $i$. It then follows from equations 3.13 and 3.14 that

$$
s_i = \sqrt{m^{-1} \cdot (I^T D^2 I)} = \frac{1}{\sqrt{m}} \| DI \|, \quad (i = 1, 2, \ldots, m).
$$

(3.17)

As the value of $I^T D^2 I$ is independent of the matrix $R$, equation 3.16 indicates that to minimize the target $\kappa_2^2$, the choice of $R$ should make the sum $\sum_{i=1}^{m} \beta_i$ as small as possible. By the definition of $\beta_i$, we know $\sum_{i=1}^{m} \beta_i = \text{tr}(R^T D^{-2} R)$. We therefore get the following optimization problem to determine the matrix $R$:

$$
\begin{align*}
\min & \quad \sum_{i=1}^{m} \beta_i = \text{tr}(R^T D^{-2} R) \quad \text{where } R \in \mathbb{R}^{r \times m} \\
\text{s. t.} & \quad R^T R = I_m, \quad \exists x \in \mathbb{R}^m \ (Rx = DI).
\end{align*}
$$

(3.18)

Note that if $R$ is a minimizer of problem 3.18, 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 $r_m = DI/\| DI \|$—the unit vector on the direction of $DI$ (for vectors, their Frobenius norm equals their Euclidean norm, so we can abuse our notation of $\| \cdot \|$)—since $DI$ lies in the range space of $R$. Thus, the matrix $R$ has the form $R = [HX, r_m]$, where $H \in \mathbb{R}^{r \times (r-1)}$ is fixed and
has columns forming a basis of the null space of \( r_m \); that is, \( H \) can be any matrix with the properties \( H^T H = I_{r-1} \) and \( r_m^T H = 0 \) (see section A.3 for a simple construction of such a matrix \( H \)); and \( X \) is a some \((r-1)\) by \((m-1)\) matrix satisfying \( X^T X = I_{m-1} \). Substituting the expression \( R = [H X, r_m] \) into equation 3.18, we get

\[
\min \{ \text{tr}(X^T H^T D^{-2} H X) : X^T X = I_{m-1} \}.
\]

As is well known, this optimization problem has an analytical solution: the \( i \)th column of \( X \), denoted \( x_i \), is exactly the eigenvector of \( H^T D^{-2} H \) with respect to its \( i \)th smallest eigenvalue \( \lambda_i \), which, surprisingly, is equal to \( \beta_i \). This can be proven in one line:

\[
\beta_i = r_i^T D^{-2} r_i = x_i^T H^T D^{-2} H x_i = \lambda_i x_i^T x_i = \lambda_i.
\]

It now remains only to find an orthogonal matrix \( L \) such that \( 1_m^T L = u^T \). With the matrix \( R \) constructed as above, we find that

\[
v^T = I^T D R = \|D\| \cdot r_m^T [H X, r_m] = \|D\| \cdot [0, \ldots, 0, 1],
\]

that is, \( v_i = 0 \) for \( i < m \) and \( v_m = \|D\| \). By the relation \( u_i s_i = v_i \) and equation 3.17, we know \( u^T = [0, \ldots, 0, \sqrt{m}] \), so \( L \) can be any orthogonal matrix with \( 1_m^T L \) as its last column. For instance, we can set

\[
L = \frac{1}{\sqrt{m}} \left[ (1 + \sqrt{m})^{-1} 1_{m-1}^T 1_{m-1}^T - \sqrt{m} 1_{m-1}^T 1_m^T \right].
\]

A more general construction of such matrices \( L \) is presented in appendix A.3.

3.2 The Constructive Error-Controlling Algorithm. The discussion on problem 3.9 sums up to a constructive procedure for evaluating the optimal \( C_{\text{cmp}} \) and \( Q_{\text{cmp}} \) that minimizes the modeling error bound presented in inequality 3.1. Inserting this procedure into the basic learning scheme, we get the following constructive error-controlling (CEC) algorithm, in which some Matlab-style notation is employed:

**Algorithm 1:** Constructive Error-Controlling Algorithm

**Input:**
- Training sequence \( \bar{s} = s_1 s_2 \ldots s_T \)
- Model dimension \( m \)
- Length \( k \) of basic strings

**Procedure**
1. Extract the compressed counting matrices \( V_{\text{cmp}} \) and \( W_{\alpha}^{\text{cmp}} \) from \( \bar{s} \).
2. Evaluate the optimal \( C_{\text{cmp}} \) and \( Q_{\text{cmp}} \) as follows.
   a. Compute the compact SVD of \( V_{\text{cmp}} \):
      \( V_{\text{cmp}} = L_1 D R_1^T \).
   b. Let \( I^T = 1_N L_1 \), \( v_m = \|D\| \), \( s_m = v_m / \sqrt{m} \).
   c. Let \( r_m = D / v_m \), \( H = \text{null}(r_m^T) \) — or, cf. section A.3.
3. Let $[\beta_i, x_i] = i$th-smallest-eigenpair $(H^T D^{-2} H)_{i=1,2,...,m-1}$.
4. Let $R = [Hx_1, \ldots, Hx_{m-1}, r_m]$, $L$ be as in equation 3.19 or A.1.
5. Let $C^{cmp} = (s_m L R^T D^{-1} - \frac{1}{m^2} m_1^2 T N_1) L_1^T + \frac{1}{m} m_1^T N_1$, $Q^{cmp} = s_m^1 R_1 R L^T$.

3. Estimate all $\hat{\tau}_a$'s by equation 3.7 and $\hat{w}_0$ as in step 3 of the basic scheme.

Output: The learned model $(\mathbb{R}^m, \{\hat{\tau}_a\}_{a \in \mathcal{O}}, \hat{w}_0)$.

The time complexity of the CEC algorithm is dominated by the collection of the compressed counting matrices, which costs $O(T)$ flops, and the computation of the compact SVD $V^{cmp} = L_1 D R_1^T$, which, in the worst case where the matrix $V^{cmp}$ has full rank $r = N$ (assume that $M = N$), amounts to $O(N^3)$ flops (Golub & Loan, 1996). Thus, the total time complexity of CEC is about $O(T + N^3)$. Here we recall that the EM algorithm for training HMMs and the ES algorithm have time complexity $O(nm^2 T)$, 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 comparison between EC and CEC is more sophisticated. As analyzed in Zhao, Jaeger, and Thon (2009), the time complexity of EC is about $O(T + 2nmN^2)$, which is typically higher than that of CEC, 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.

3.3 An Efficient Variant to CEC. We now introduce an efficient variant to the CEC algorithm. The time complexity $O(T + N^3)$ of the CEC algorithm can be reduced to $O(T + mN^2)$ if we compute only the first $m$ (the model dimension) singular triples of the matrix $V^{cmp}$. That is, instead of the compact SVD $V^{cmp} = L_1 D R_1^T$, we use the first (largest) $m$ singular values and left- (right-) singular vectors of $V^{cmp}$ 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 call the efficient variant presented here the CEC-B algorithm 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 $V^{cmp}$ is in fact the statistical approximation of the probability matrix $V^*$ (up to a counting factor), which should have rank $m$. Recall that we have already assumed that the underlying process can be modeled by some $m$-dimensional OOM. Therefore, we can (and it is quite natural) first estimate $V^*$ (the “true” counting matrix) from $V^{cmp}$ and then use $V^*$ 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 data set is large (since then $V^{cmp} \approx V^*$). We would, however, emphasize again that CEC-B is computationally cheaper than CEC-A.

We conclude this section with an important remark on a painful issue in OOM theory: the negative probability problem (Jaeger, 2000b).
Constructive Error-Controlling Algorithm for OOMs 3473

refers to the fact that the model \((\mathbb{R}^m, \{\tilde{r}_a\}_{a \in O}, \tilde{w}_0)\) learned by existing OOM learning algorithms, including the CEC algorithm presented in this letter, can produce negative “probabilities” \(\tilde{P}(\bar{a}) = 1^T \tilde{r}_a \tilde{w}_0 < 0\) for some sequence \(\bar{a} \in O^*\). For a long time, we 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 of finding nontrivial and practically useful sufficient conditions for nonnegativity. In practice, the current state of the art relies on heuristic methods that essentially “repair” the OOM state vector by small correction terms when it would give negative “probability” values. These work well (see the one presented in appendix J in 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, Jaeger, and Thon (2009): 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 is from (Jaeger et al., 2005): to train OOMs on Mark Twain’s short story, “The One Million Pound Bank-Note.”

4.1 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 that 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 interval \((0, 1)\). 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 subintervals, 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, Jaeger, and Thon (2009): (1) 20 sequences, each of which has length \(L = 30,000\), are created by running the 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) we compute the normalized log-likelihood (NLL) of these learned models \(\mathfrak{A}\) on the other 19 sequences as follows:

\[
\text{NLL}(\mathfrak{A}, S) := \frac{1}{\# S} \sum_{\bar{a} \in S} \frac{\log_{\ell} \tilde{P}(\bar{a} | \mathfrak{A})}{\text{length of } \bar{a}},
\] (4.1)
where \( \ell \) is the alphabet size, \( S \) is the test data set 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 learned 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, Jaeger, and Thon (2009), the ES algorithm is terminated after 10 iterations, and the estimated model with the highest training NLL is selected as the final output of ES. But unlike as in Zhao, Jaeger, and Thon (2009), ES is now started with a more carefully chosen initial model, so that it is numerically stable for all model dimensions.

As Zhao, Jaeger, and Thon (2009) have pointed out, in practice, the quantity \( \text{NLL}(\mathbf{A}, S) \) 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-\( \ell \) logarithm), defined by the formula

\[
H(X_n) := -\lim_{k \to \infty} \frac{1}{k} \sum_{\bar{a} \in \mathcal{A}^k} P(\bar{a}) \log_\ell P(\bar{a}),
\]

(4.2)

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, Jaeger, & Thon, 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 data set contains 20 sequences of length 30,000, 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 learned 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 Figure 1a the average test NLLs of each OOM learning algorithm versus model dimensions, in Figure 1b the standard deviation of these test NLLs, and in Figure 2 the total CPU time needed for learning these models. All algorithms are
Figure 1: Test NLLs of learned OOMs on the quantized logistic system: (a) The average test NLLs. To make the difference more visible, the y-axis is computed by $y = -0.256 - \text{average-NLL}$ and represented in logarithmic scale (note that 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 learned by CEC-A and CEC-B have nearly the same quality. The relative difference in test NLL or one-step prediction error (see equation 4.3) is less than 0.1% on all data sets. As a consequence, in Figures 1, 3, and 5, the curves for CEC-A and CEC-B are almost identical.
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 Figure 1 caption for more detail), it is two to three times faster than CEC-A. Another important observation is that for model dimensions $m \geq 40$, all models (under ES, EC, and CEC) level out in their NLLs, which reach $\approx -0.261, -0.257, -0.257$, respectively, with a small margin toward the theoretical optimum of $-0.25$.

4.2 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 & Sutton, 2001) in the literature.

First, we point out the relationship between OOMs and PSRs. PSRs are a class of models for discrete, stochastic input-output systems, which generalizes partially observable Markov decision processes (POMDPs) like OOMs generalize HMMs. PSRs have been partly inspired by OOMs but have developed a representation format for input-output processes 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 & Bowling, 2005; Wolfe, James, & Singh, 2005; Bowling, McCracken, James, Neufeld, & Wilkinson, 2006; Rudary & Singh, 2006; Wolfe, 2006; Wingate & Singh, 2007). Zhao, Jaeger, and Thon (2009) 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.

The seven target POMDPs are taken from a public Web source. They are Bridge, Cheese, Maze4x3, Network, Paint, Shuttle, and Tiger. We use the same experimental settings as Zhao, Jaeger, and Thon (2009) and for more detail refer readers to that article. We use notations 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 $a^1 o^1 a^2 o^2 \ldots a^N o^N$, 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 $N = 10^3 - 10^7$ using the CEC–A or B (or EC, ES) algorithm, as follows. We (naively) regard each combination $(a^i o^i)$ 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 $s^1 s^2 \ldots s^N$ (with the length of basic strings set to be $k = 2$).
- The quality of learned models is measured 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):

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

(4.3)

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 $h_i = a^1 o^1, \ldots, a^i o^i$ and the action $a^{i+1}$. For learned OOMs (over $S = A \times O$), we compute the above predicting probabilities $\hat{P}(o|h_i a^{i+1})$ by

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

(4.4)

In Figure 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 Figure 4 the training CPU

---

Figure 3: 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 prediction error). Panel h is a rerun of panel b with different parameter settings. See the text.
Figure 4: Training time of the ES, EC, and CEC-A,B algorithms on seven POMDP benchmark problems (x-axis: (log₁₀ of) length of training sequences; y-axis: training CPU time). Panel h is a rerun of panel b with different parameter settings. See the text.
time cost by these algorithms for each training length. These are the main findings:

- 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 except the Cheese domain. This partly demonstrates that OOM learning is asymptotically correct (since $E \to 0$ when $N \to \infty$) and 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 anymore when $N \geq 10^5$. 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 3h 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 a small training data set, CEC is faster than EC, whereas for a large data set, 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 $O(T)$, and when $T$ is small, CEC has the time complexity $O(mN^2)$ or $O(N^3)$, smaller than that of EC, which is $O(2nmN^2)$.

- 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. 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 data sets.

- Bridge is an interesting domain, in which CEC/EC is inferior to ES when $N \leq 10^5$, but when $N$ becomes larger ($\geq 3 \times 10^5$), CEC/EC starts to yield a smaller prediction error than ES (In fact, other domains show a similar phenomenon, which is, however, less obvious, as in the Bridge domain). This illustrates that asymptotically, CEC/EC converges faster than ES ($\beta_{\text{CEC}} > \beta_{\text{ES}}$) but also reveals that in some domains, CEC/EC might not be as statistically efficient as ES when the training data set 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 noted that in the Maze4x3 domain (see Figure 4c), the training CPU times 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 experiment 10 times, Matlab’s `rand()` function initialized by different seeds and always got similar results. One possible reason for this is that for training sequences of length $\geq 10^6$, the counting matrices are already very near to the true probability matrices and so have (numerical) rank equal to the model dimension $m = 11$. This might speed up the SVD of the counting matrix $V^{cmp}$—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 $Q$. In the ideal case where the counting matrix has rank $m$, only one EC iteration is needed.

### 4.3 Modeling Text Sources

The third experiment models a text source—“The 1,000,000 Pound Bank-Note,” a short story written by Mark Twain. To simplify the task, we 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 (21,042 and 20,569 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 learned OOM, as shown in Figures 5a and 5b. Here the length of characteristic or indicative strings is set as $k = 2$. We also plotted the training time of each learning algorithm versus the model dimension in Figure 6. From these figures, we see that OOMs learned by the four algorithms have nearly the same training and test NLLs (ES is a bit 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 to 100 times faster than ES and EC.

### 5 Conclusion and Discussion

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

- As expected, the two CEC algorithms improve over their predecessor, the EC algorithm, in most domains. CEC usually learns more accurate models (at least not worse) than EC. When learning high-dimensional models, CEC are much faster than EC.
Figure 5: Training (a) and test (b) NLLs of learned OOMs in "The 1,000,000 Pound Bank-Note."
Figure 6: Training times of the ES, EC, and CEC-A,B algorithms on "The 1,000,000 Pound Bank-Note."

- Compared with the ES algorithm, both CEC algorithms are much faster (especially when learning models from large data sets), while still showing comparable test accuracy.
- For large data sets, the two CEC algorithms have nearly the same performance, but for small data sets, CEC-B is about two to three 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 Zhao, Jaeger, and Thon (2009) pointed out, a mathematical understanding of the relationships between the algebraic and the statistical aspects of EC/CEC versus ES, or of the algebraic working principle versus 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 a difficult problem in the current linear framework of OOMs. In response to this issue, M.-J. Z. is developing and investigating nonlinear...
variants of OOMs that are free from this negativity problem by design (Zhao & Jaeger, 2007).

Appendix 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 problem 3.11 into two simpler problems:

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

Assume the \( i \)th column of \( L_2 \) is \( b_i \) and denote by \( y_i \) the \( i \)th column of \( C_2 \). Then the above first problem is equivalent to \( \min_{y_i} \{ \|y_i\| : \mathbf{1}_m^T y_i = \mathbf{1}_N^T b_i \} \), (\( \forall i \)), since \( \|C_2\|^2 = \sum_i \|y_i\|^2 \). By the Cauchy-Schwarz inequality, we know 
\[ |\mathbf{1}_m^T y_i| \leq \|\mathbf{1}_m\| \cdot \|y_i\|, \]
that is, \( \|y_i\| \geq \frac{1}{\sqrt{m}} |\mathbf{1}_m^T b_i| \); with equality if and only if 
\[ y_i = \alpha \mathbf{1}_m, \]
which, together with the constraint \( \mathbf{1}_m^T y_i = \mathbf{1}_N^T b_i \), implies that 
\[ y_i = (\frac{1}{\sqrt{m}} \mathbf{1}_m) b_i \mathbf{1}_m. \]
It then follows that \( C_2 = \frac{1}{m} \mathbf{1}_m \mathbf{1}_N^T L_2 \) 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: \( \|A\|_2 = \max\{\|Ax\| : \|x\| = 1\} \). Let \( x \) be such that \( \|x\| = 1 \) and \( \|Q_1: 0\|_2 = \|\{Q_1; 0\}x\| \), then \( \|Q_1: Q_2\|_2^2 \geq \|\{Q_1; Q_2\}x\|^2 = \|\{Q_1; 0\}\|^2 = \|\{Q_1; 0\}\|_2^2 \) and so \( \|\{Q_1; Q_2\}\|_2 \geq \|\{Q_1; 0\}\|_2 \), which is what we want to prove.

A.2 Proof of Proposition 3. Assume \( C_1 D \) has the compact SVD \( C_1 D = L S R^T \). Then \( L^T L = I_m \) and the constraint \( C_1 D Q_1 = I_m \) now reads \( L S R^T Q_1 = I_m \). It follows that \( S R^T Q_1 L = L^T L S R^T Q_1 L = L^T L = I_m \), and so \( R^T Q_1 L = S^{-1} \), which in turn implies that \( R^T Q_1 L L^T = R S^{-1} L^T = (C_1 D)^\dagger \). This identity, together with the facts that \( \|L\|_2 = \|R\|_2 = 1, \|A\|_2 = \|A^T\|_2, \) and \( \|AB\|_2 \leq \|A\|_2 \cdot \|B\|_2 \), implies that \( \|(C_1 D)^\dagger\|_2 \leq \|Q_1\|_2 \), which completes the proof.

A.3 Simple Construction of Two Orthogonal Matrices. Here we consider the following simple problems in linear algebra: given two vectors \( x, y \in \mathbb{R}^n \) with unit norm, (1) how to (efficiently) construct an orthogonal matrix \( L \in \mathbb{R}^{n \times n} \) such that \( L x = y \) and (2) how to find a basis of the null space of \( x^T \), that is, to find a matrix \( H \in \mathbb{R}^{n \times (n-1)} \) satisfying \( H^T H = I_{n-1} \) and \( x^T H = 0 \)? As one can easily see, \( L \) and \( H \) are not unique, so we add an extra condition on \( L \): it should be as close to \( I_n \) as possible, in the sense that \( \|L - I_n\| \) is minimized. As \( L^T L = I_n \), we have \( \|L - I_n\|^2 = 2 \text{tr}(I_n - L) \).
Thus, by brute force computation,

\[
L = \begin{cases} 
I_n - \frac{(y - x)(y - x)\top}{1 - x\top y} & \text{if } x \neq y, \\
I_n & \text{if } x = y.
\end{cases}
\]  
(A.1)

In the above formula, putting \( y = [0, \ldots, 0, 1]\top \) and extracting the first \((n - 1)\) columns of \( L \), we get the matrix \( H \).

Acknowledgments

The results reported in this article were achieved when the first author worked under a grant (JA 1210/1-1 & 2) from the Deutsche Forschungsgemeinschaft (DFG). We also thank two anonymous reviewers for their very careful and constructive review.

References


Received October 2, 2008; accepted April 2, 2009.