MingJie Zhao Herbert Jaeger

The Error Controlling Algorithm for Learning OOMs

Technical Report No. 6
March, 2007

School of Engineering and Science

The Error Controlling Algorithm for Learning OOMs

MingJie Zhao Herbert Jaeger

Jacobs University Bremen
School of Engineering and Science
Campus Ring 6
28759 Bremen
Germany

E-Mail: m.zhao, h.jaeger@iu-bremen.de
http://www.iu-bremen.de/

Summary

Observable operator models (OOMs) are matrix models for describing stochastic processes. In this report we proposed a novel algorithm for learning OOMs from empirical data. Like other learning algorithms of OOMs, the presented algorithm is based upon the *learning equation*, a linear equation on observable operators involving two adjustable matrices P and Q. The algorithm designs P, Q so that an upper bound of the error between the estimated model and the "real" model is minimized. So we call it the *error controlling* (EC) algorithm.

One important feature of the EC algorithm is that it can be used online, where it updates the estimation of observable operators on each new observation via a set of RLS-like formulas [1] . By the linearity of the learning equation, we are able to prove the asymptotic correctness of the online learning procedure.

The main numerical problem of the EC algorithm is to control the condition of the learning equation *online* (or minimize it *offline*), leading to an optimization problem of special form. For this special problem we proposed an efficient method that searches for the *global* optimum alternatively on two orthogonal directions and call it the *orthogonal searching* (OS) algorithm. The performance of EC, in which the OS algorithm is used as the inner optimizer, is studied through some (offline and online) leaning examples.

[1] RLS means *recursive least squares* estimation.

Contents

1 Introduction

Hidden Markov models (HMMs) trained with the EM (Baum-Welch) algorithm [1] nowadays have become one of the standard tools for modeling stochastic systems and have been used in a wide range of applications such as speech recognition [9], control engineering [3], and biosciences [2] with large success.

Over the last few years, an alternative to HMMs has been developed, namely *observable operator models* (OOMs) [6]. According to the OOMs theory, the future conditional distributions of a stochastic process are seen as *predictive states*; and the stochastic process itself can be modelled by means of linear operators τ_a acting on these predictive states. HMMs can also be represented in a similar fashion using linear operators acting on hidden states. The main difference between HMMs and OOMs is that the matrices and state vectors in HMMs consist only of non-negative entries, while OOMs allow for negative entries. This difference gives OOMs three significant advantages over HMMs:

- The theory of OOMs is mathematically simple and elegant since only concepts from elementary linear algebra is required.
- OOMs are more expressive than HMMs in that stochastic processes which can be modelled by finite-dimensional HMMs can also be described by finite-dimensional OOMs but not vice versa.
- While the (iterative) EM algorithm for HMMs suffers from the problems of slow convergence and the presence of many local optima, a constructive, asymptotically consistent learning algorithm for OOMs can be derived from the *equivalence theorem* (cf. Section 2).

The basic theory of OOMs was set up in the last decade in [6], in which a general constructive procedure for learning OOMs from empirical data was also proposed. The performance of this general algorithm, however, heavily depends on the choice of *characteristic events* and *indicative events*, for which only some intuitive heuristics are obtained, see [6] for the detail. The first practical learning algorithm was developed by Kretzschmar in [8], where he considered different OOM representations of a stochastic process and proposed a systematic method for searching for representations that minimize the effects of estimation errors caused by finite training sequences. In this sense we call it the *robust learning algorithm*. We will briefly introduce this algorithm in Section 2.

In this technical report we proposed a novel algorithm for learning OOMs, the *error controlling* (EC) algorithm. Both the robust algorithm and the EC algorithm are based on the *learning equation*, a linear matrix equation involving several *probability matrices* whose elements are the probability masses on certain initial sequences, and two adjustable matrices P and Q. While the probability matrices are unavailable, they can be approximated by the *counting matrices* obtained from the training data by accumulating the occurrence number of the corresponding sequences. Since the training sequence is finite, this approximation will of course

introduce errors into the learning equation. So the basic idea here is to minimize or control the effect of such an approximation, that is, to make the learning equation well conditioned, by selecting appropriate P and Q. But the EC algorithm uses a different target function for evaluating the matrices P, Q from that used by the robust algorithm, which will be discussed later.

In a more recent paper [7], Jaeger argued that in many situations one can only gain a small data set containing incomplete information of the underlying process; so the statistical efficiency problem is more important than the above error controlling problem. He then proposed another criterion for selecting the matrices P and Q, from which an iterative learning algorithm of OOMs, called *efficiency sharpening* (ES) algorithm, was derived. The ES algorithm will also be introduced in Section 2 in brief, for the detailed procedure see [7].

Unlike the robust and ES algorithms, the EC algorithm can be used online. In fact, due to the linearity of the learning equation, a set of RLS-like formulae for updating the estimated model can be easily derived if the two adjustable matrices P and Q are fixed. An efficient online method for modifying P and Q, however, is demanded to minimize or control the condition of the learning equation *online* so that the estimation error will not be magnified by inverting an ill-conditioned matrix during the learning procedure. The condition of the learning equation can be measured via different matrix norms. Under the Frobenius norm, the above condition controlling problem is reduced to an optimization problem of special form, for which a new method that searches for the optimum iteratively on two orthogonal directions, and hence called the *orthogonal searching* (OS) algorithm, is proposed. The OS algorithm can also be used offline to *minimize* the condition of the learning equation, yielding the offline version of the EC algorithm.

The report is organized as follows. First the basics of OOMs theory, the robust algorithm and the ES algorithm are reviewed in Section 2. Then we introduce the online version of the EC algorithm in Section 3. Section 4 is devoted to the OS algorithm, where the theoretical derivation and some numerical investigation are presented. The performance of the EC algorithm is studied on several data sets; and the results are presented and discussed in Section 5. Finally, we make the conclusion in Section 6.

2 An Introduction to OOMs

The way that OOMs describe stochastic processes gives it its name. Let $(Y_n)_{n\in\mathbb{N}}$ be a discrete-time stochastic process on some probability space $(\Omega, \mathcal{F}, P)$ with values from a finite alphabet $O = \{1, 2, \cdots, \ell\}$. [2] According to OOMs theory, with each observable $a \in O$ one can assign a linear operator τ_a such that the probability distribution of $(Y_n)_{n\in\mathbb{N}}$ is completely determined by the operators $\{\tau_a\}_{a\in O}$ and an initial state vector $\boldsymbol{w}_0$. These operators τ_a are hence called *observable operators*,

[2] To simplify the notation, observables are represented by their indices in the alphabet.

which, together with the initial state $\boldsymbol{w}_0$, form an *observable operator model* of the process $(Y_n)_{n\in\mathbb{N}}$. Although OOMs can also be used to model continuous-time, arbitrary-valued processes [5], in this report we will only consider the discrete-time, finite-valued case.

2.1 The Basics of OOMs Theory

In this subsection we briefly introduce the basic results of OOMs theory as well as the notation conventions used by the report, see [6] and the references therein for the detail.

The definition of OOMs and the equivalence theorem

Throughout the report the alphabet is always denoted by $O = \{1, 2, \cdots, \ell\}$. For a stochastic process $(Y_n)_{n\in\mathbb{N}}$ with $Y_n \in O$, we write

$$P(a_1 a_2 \cdots a_n) := \Pr(Y_1 = a_1, Y_2 = a_2, \cdots, Y_n = a_n)\,,$$

where $n \in \mathbb{N}$ and $a_i \in O$ $(i = 1, 2, \cdots, n)$, for the probability that an initial sequence of outcomes $a_1 a_2 \cdots a_n$ is observed. Then an m-dimensional OOM of the process (Y_n) can be defined as a triple $\mathfrak{A} = (\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ with $\tau_a \in \mathbb{R}^{m\times m}$ and $\boldsymbol{w}_0 \in \mathbb{R}^m$ such that, for any initial sequence $a_1 a_2 \cdots a_n$,

$$P(a_1 a_2 \cdots a_n) = \mathbf{1}_m^{\mathsf{T}} \tau_{a_n} \cdots \tau_{a_2} \tau_{a_1} \boldsymbol{w}_0\,, \tag{2.1}$$

where $\mathbf{1}_m$ denotes the m-dimensional column vector of units.

To simplify the notation we use small letters with a bar ($\bar{a}$, $\bar{b}$, ...) to denote a finite sequence of observables in O; and O^* to denote the set of all such finite sequences, including the *empty sequence* ε. For any sequence $\bar{a} = a_1 a_2 \cdots a_n$ in O^*, we denote by $\tau_{\bar{a}}$ the *reverse-ordered* product $\tau_{a_n} \cdots \tau_{a_2} \tau_{a_1}$. With these shorthand notations, eqn (2.1) can be rewritten as $P(\bar{a}) = \mathbf{1}_m^{\mathsf{T}} \tau_{\bar{a}} \boldsymbol{w}_0$.

A stochastic process can be modelled by different OOMs through eqn (2.1), these OOMs are *equivalent* to each other. An OOM is said to be *minimal* if there is no other OOMs with lower dimension m in its equivalence class. By this definition of minimal OOMs and eqn (2.1), we get

Proposition 1 *The structure $\mathfrak{A} = (\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ is a minimal OOM of some stochastic process iff* (1) $\mathbf{1}_m^{\mathsf{T}} \boldsymbol{w}_0 = 1$; (2) $\mathbf{1}_m^{\mathsf{T}} \tau = \mathbf{1}_m^{\mathsf{T}}$, *where $\tau := \sum_{a\in O} \tau_a$ is the sum of τ_a's;* (3) $\mathbf{1}_m^{\mathsf{T}} \tau_{\bar{a}} \boldsymbol{w}_0 \geqslant 0$ *for any $\bar{a} \in O^*$; and* (4) *the two families of vectors $\{\tau_{\bar{a}} \boldsymbol{w}_0 : \bar{a} \in O^*\}$ and $\{\tau_{\bar{a}}^{\mathsf{T}} \mathbf{1}_m : \bar{a} \in O^*\}$ both span the whole space $\mathbb{R}^m$. Furthermore, $\mathfrak{A}$ describes a stationary process iff* (5) $\tau \boldsymbol{w}_0 = \boldsymbol{w}_0$.

The reader is referred to the discussion in Section 14.5 (pp.426–428) of [7] for an indirect proof of the above proposition and, more important, a general procedure for converting a given OOM to its equivalent minimal OOM. So in the sequel we may, and do, only consider minimal OOMs. Furthermore, this proposition gives us an equivalent algebraic definition of minimal OOMs.

Definition 1 *An m-dimensional minimal OOM is a triple* $\mathfrak{A} = (\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ *with* $\tau_a \in \mathbb{R}^{m\times m}$ *and* $\boldsymbol{w}_0 \in \mathbb{R}^m$*, for which the first four conditions from Proposition 1 hold.*

Now we are ready to state the *equivalence theorem*, which is of fundamental importance when developing learning algorithms for OOMs. It is clear that two minimal OOMs of different dimensions are not equivalent, so we can assume that the two minimal OOMs whose equivalence we wish to ascertain have the same dimension.

Proposition 2 (the equivalence theorem, see Proposition 5 of [6])
Two minimal OOMs $\mathfrak{A} = (\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ *and* $\mathfrak{B} = (\mathbb{R}^m, \{\varphi_a\}_{a\in O}, \boldsymbol{v}_0)$ *are equivalent if and only if there exists a nonsingular matrix* $\varrho \in \mathbb{R}^{m\times m}$ *such that* (1) $\mathbf{1}_m^\mathsf{T}\varrho = \mathbf{1}_m^\mathsf{T}$, (2) $\boldsymbol{v}_0 = \varrho\boldsymbol{w}_0$ *and* (3) $\varphi_a = \varrho\tau_a\varrho^{-1}$ *for all* $a \in O$.

It is based on the above equivalence theorem that one can establish the learning equation of OOMs, so far the starting point of all learning algorithms of OOMs (the robust, ES and our EC algorithm).

The learning equation of OOMs

Given an OOM $\mathfrak{A} = (\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$ of a process (Y_n), one can compute the distribution of (Y_n) by eqn (2.1). Now we consider the reverse problem, i.e., given the distribution of (Y_n), we want to identify all the operators τ_a [3] so that $\mathfrak{A}$ is an OOM for (Y_n). This can be done as follows.

Let O^d be the set of all sequences over O of length d and $\{\bar{a}_j\}_{j=1}^N$, $\{\bar{b}_i\}_{i=1}^N$ two enumerations of O^d (hence $N = \ell^d$). Note here the enumerations $\{\bar{a}_j\}_{j=1}^N$ and $\{\bar{b}_i\}_{i=1}^N$ may be different (in order). We define the *probability matrices*

$$\underline{V} = \big[P(\bar{a}_j\bar{b}_i)\big]_{i,j=1,2,\cdots,N}\ , \quad \underline{W}_a = \big[P(\bar{a}_j a\bar{b}_i)\big]_{i,j=1,2,\cdots,N}\ , \tag{2.2}$$

where the observable a runs over O; $\bar{a}_j\bar{b}_i$ denotes the concatenation of $\bar{a}_j$ and $\bar{b}_i$; and the notation $\bar{a}_j a\bar{b}_i$ has the similar meaning.

It follows from eqn (2.1) that $\underline{V} = US$ and $\underline{W}_a = U\tau_a S$, where U denotes the $N\times m$ matrix with $\mathbf{1}_m^\mathsf{T}\tau_{\bar{b}_i}$ as its i-th row and S the $m\times N$ matrix with j-th column $\tau_{\bar{a}_j}\boldsymbol{w}_0$. Since $\mathbf{1}_m^\mathsf{T}(\sum_{a\in O}\tau_a) = \mathbf{1}_m^\mathsf{T}\tau = \mathbf{1}_m^\mathsf{T}$ and since $\{\bar{b}_i\}_{i=1}^N$ is an enumeration of O^d, we have $\mathbf{1}_N^\mathsf{T}U = \sum_{i=1}^N \mathbf{1}_m^\mathsf{T}\tau_{\bar{b}_i} = \mathbf{1}_m^\mathsf{T}(\sum_{a\in O}\tau_a)^k = \mathbf{1}_m^\mathsf{T}$. Furthermore, by the definition of minimal OOMs, the matrices S and U both have rank m for sufficiently large d (e.g., $d \geqslant m$). So we can select matrices $P, Q \in \mathbb{R}^{m\times N}$ such that $\mathbf{1}_m^\mathsf{T}P = \mathbf{1}_N^\mathsf{T}$ and that PU, SQ^T both are invertible. Let $\varrho = PU$. Then $\mathbf{1}_m^\mathsf{T}\varrho = \mathbf{1}_N^\mathsf{T}U = \mathbf{1}_m^\mathsf{T}$ and

$$P\underline{W}_a Q^\mathsf{T} = PU\tau_a SQ^\mathsf{T} = \varrho\tau_a\varrho^{-1}PUSQ^\mathsf{T} = (\varrho\tau_a\varrho^{-1})(P\underline{V}Q^\mathsf{T})\,. \tag{2.3}$$

[3] The initial state vector $\boldsymbol{w}_0$ is usually evaluated from the OOMs conditions (1) and (5) from Proposition 1.

It follows from the equivalence theorem that, by selecting different matrices P and Q, we are able to construct the whole equivalence class of OOMs

$$\{\mathfrak{A}(P,Q)\} = \{(\mathbb{R}^m, \{\varrho\tau_a\varrho^{-1}\}_{a\in O}, \varrho\boldsymbol{w}_0)\} =: \{(\mathbb{R}^m, \{\tilde{\tau}_a\}_{a\in O}, \tilde{\boldsymbol{w}}_0)\}$$

of the given stochastic process (Y_n) via the formula

$$\tilde{\tau}_a(P,Q) = (P\underline{W}_aQ^{\mathsf{T}})(P\underline{V}Q^{\mathsf{T}})^{-1}\,. \tag{2.4}$$

Both eqn (2.3) and eqn (2.4) are therefore called the *learning equations*.

Estimating OOMs from empirical data

The learning equation (2.4) reveals how to reconstruct an OOM from the *known* distribution of a stochastic process. In practical modelling tasks, however, the distribution of the underlying process $(Y_n)_{n\in\mathbb{N}}$ is typically unknown. Instead, a finite length instantiation of the process is given and we are required to estimate an OOM that approximately models the process. In this report we will consider the following learning task: estimate an m-dimensional OOM from a finite sequence $\bar{s} = s_1s_2\cdots s_l$ which is assumed to be produced by a stationary and ergodic process. Here the dimension m is also assumed to be known.

For ergodic processes, the probability matrices $\underline{V}$ and $\underline{W}_a$ in eqn (2.3) can be asymptotically approximated by the following *counting matrices*

$$\underline{V}^{\#} = \left[\{\bar{a}_j\bar{b}_i\}^{\#}\right]_{i,j=1,2,\cdots,N}\,, \quad \underline{W}_a^{\#} = \left[\{\bar{a}_ja\bar{b}_i\}^{\#}\right]_{i,j=1,2,\cdots,N}\,, \tag{2.5}$$

respectively, where $\{\bar{a}_ja\bar{b}_i\}^{\#}$ denotes the occurrence number of the sequence $\bar{a}_ja\bar{b}_i$ in the training sequence $\bar{s}$, while $\{\bar{a}_j\bar{b}_i\}^{\#}$ is obtained from $s_1s_2\cdots s_{l-1}$, omitting the last observation s_l. This is because one should keep the same "counting factors" ($l-2d$ in our case) on both sides of the learning equation (2.3).

A remark on the indices l and t. — As we will also discuss the online version of the EC algorithm, it would be convenient to relate the length l of the training sequence (in offline EC) to the time step t (in online EC). The above discussion about how to get the counting matrices from the whole training sequence $\bar{s}$ is, of course, for offline learning task. For online learning, the length of $\bar{s}$ is potentially infinite; and at each time t only the initial part of $\bar{s}$ is known to us. To unify the two cases, we (1) for the offline case define the "effective length" $t := l-2d$ of the training sequence; (2) for the online case assume that at time t the initial part of $\bar{s}$ up to the $(t+2d)$-th symbol s_{t+2d} is observed. Then for the two (different) cases one may only consider the same sequence $s_1s_2\cdots s_{t+2d}$, with the index t meaning (1) the effective length of $\bar{s}$ in the offline case; or (2) the time, also the effective length of $\bar{s}$ up to the current time t in the online case.

To get the counting matrices $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$ online, we use an inspection window of width $2d+1$. The detailed scheme is as follows. The matrices $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$ are initially set to be zero. Assume at time t, we have gotten $\underline{V}^{\#} = \underline{V}^{\#}(t)$ and

$\underline{W}_a^{\#} = \underline{W}_a^{\#}(t)$. At time t, as supposed above, the subsequence $s_{t+1} \cdots s_{t+1+2d}$ is observed in the inspection window. Assume that

$$\begin{aligned} s_{t+1} \cdots s_{t+d} &= \bar{a}_j \,, & s_{t+1+d} &= c \,, \\ s_{t+1+d} \cdots s_{t+2d} &= \bar{b}_k \,, & s_{t+2+d} \cdots s_{t+1+2d} &= \bar{b}_i \,, \end{aligned}$$

for which we say *the event* (j, c, k, i) *is observed at time* $t+1$ and write $(j, c, k, i)_{t+1}$. Then we update the matrices $\underline{V}^{\#}$ and $\underline{W}_c^{\#}$ by adding one at their (k, j)-th and (i, j)-th position, respectively. This can be written in matrix notation as

$$(\underline{V}^{\#})^{+} = \underline{V}^{\#} + \boldsymbol{e}_k \boldsymbol{e}_j^{\mathsf{T}} \,, \quad (\underline{W}_a^{\#})^{+} = \underline{W}_a^{\#} + \delta_{ac} \boldsymbol{e}_i \boldsymbol{e}_j^{\mathsf{T}} \,, \tag{2.6}$$

where the superscript $^{+}$ denotes updated values (at the next time step) due to new observations; $\boldsymbol{e}_i$ is the i-th unit vector with dimension determined by the context; and δ_{ac} is the Kronecker symbol: $\delta_{ac} = 1$ if $a = c$ and $\delta_{ac} = 0$ otherwise. Notice that one can use eqn (2.6) also for offline learning task (to get the final counting matrices), bus as an iterative approach rather than update rules.

The basic (offline) procedure for estimating OOMs of dimension m from a given sequence $\bar{s}$ can now be summarized as follows. First we count the occurrences of $\bar{a}_j \bar{b}_i$ and $\bar{a}_j a \bar{b}_i$ via eqn (2.6) to get the matrices $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$, respectively. We then select two matrices $P, Q \in \mathbb{R}^{m \times N}$ by some way. Finally, the observable operators are estimated by the formula

$$\hat{\tau}_a = (P \underline{W}_a^{\#} Q^{\mathsf{T}})(P \underline{V}^{\#} Q^{\mathsf{T}})^{-1} \,, \tag{2.7}$$

which will also be referred to as the learning equation in this report.

Notice that the estimation (2.7) is asymptotically correct provide that the training sequence $\bar{s}$ were indeed generated by an m-dimensional OOM in the first place. In other words, the underlying OOM will be perfectly recovered (up to its equivalence class) almost surely as the effective length (or time) t goes to infinity. This is because the normalized counting matrices $t^{-1}\underline{V}^{\#}$ and $t^{-1}\underline{W}_a^{\#}$ converge almost surely to $\underline{V}$ and $\underline{W}_a$ when t tends to infinity.

About the choice of P and Q

The estimation $\hat{\tau}_a$ obtained by eqn (2.7), and hence the performance of the learning algorithm, depends crucially on the choice of P and Q. For instance, if the matrices P and Q are poorly selected that $P\underline{V}^{\#}Q^{\mathsf{T}}$ is nearly singular, then the algorithm will be unstable. Two methods have been developed for this problem: the robust algorithm [8] and the ES algorithm [7].

- The idea of the robust algorithm is quite intuitive: one just designs the matrices P and Q so that the condition number of $P\underline{V}^{\#}Q^{\mathsf{T}}$ is minimized. In [8], the matrices P, Q are restricted to be of a special form derived from the *singular value decomposition* (SVD) of $\underline{V}^{\#}$; and a systematic but somehow complicated method is proposed to minimize the condition number

of $P\underline{V}^{\#}Q^{\mathsf{T}}$. Although several positive experiments are reported, the criterion for designing P and Q used by the robust algorithm lacks a theoretical foundation, as will be discussed in the next subsection.

- In the ES algorithm, the matrix P is given the name a *characterizer* and is designed by minimizing the quantity $J_{\mathrm{ES}}(P) = \mathrm{tr}(PDP^{\mathsf{T}})$ (eqn (14.56) of [7]) under the constraint $PU = I_m$ (Proposition 14.14 of [7]), where D is the diagonal matrix of order N with $P(\bar{b}_i)$ $(i = 1, 2, \cdots, N)$ as its diagonal entries and U, as before, is the $N \times m$ matrix with i-th row $\mathbf{1}_m^{\mathsf{T}}\tau_{\bar{b}_i}$. According to the discussion in pp.443–444 of [7], the quantity $J_{\mathrm{ES}}(P)$ represents the "mean stochastic approximation error" caused by finite training data when the matrix P is used as the characterizer. So the principle here is to minimize the mean stochastic approximation error by selecting an appropriate P (the matrix Q is then set to be $(P\underline{V}^{\#})^{\dagger}$, the pseudo inverse of $P\underline{V}^{\#}$).

  As the matrix U is unknown, the ES algorithm is an iterative method: One first "guesses" (or trains using other methods) an OOM, from which the matrix U is computed; then he computes P, Q so that J_{ES} is minimized and updates $\hat{\tau}_a$'s via eqn (2.7). This procedure is repeated on the new updated OOM until some stop criterion is satisfied.

In this report we consider the problem of choosing P and Q from a purely algebraic viewpoint. More concretely, we want to minimize or control the error $\|\tilde{\tau}_a - \hat{\tau}_a\|$ between the "real" operator $\tilde{\tau}_a$ and the estimated $\hat{\tau}_a$. In the next subsection, an upper bound of this error is derived so that one can select the matrices P and Q by minimizing this upper bound. This is why we call our new algorithm the *error controlling* (EC) algorithm.

To simplify the notation in the sequel we will write τ_a for $\tilde{\tau}_a$.

2.2 Error Analysis and Controlling

If the distribution of a stochastic process is known, one can construct its OOM from the probability matrices $\underline{V}$ and $\underline{W}_a$ via eqn (2.4). When learning OOMs from empirical data, the probability matrices $\underline{V}$, $\underline{W}_a$ are approximated by the counting matrices $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$, respectively; and estimation errors in τ_a are then introduced. In this subsection we first investigate the influence of errors in counting matrices on the estimated operators $\hat{\tau}_a$, then briefly discuss the robust algorithm [8].

To fix the notations we let $\hat{V} = t^{-1}P\underline{V}^{\#}Q^{\mathsf{T}}$, $\hat{W}_a = t^{-1}P\underline{W}_a^{\#}Q^{\mathsf{T}}$ (recall that t is the effective length of the training sequence) and $V = P\underline{V}Q^{\mathsf{T}}$, $W_a = P\underline{W}_aQ^{\mathsf{T}}$. Then eqns (2.4), (2.7) can be rewritten as

$$\tau_a = W_aV^{-1} \quad \text{and} \quad \hat{\tau}_a = \hat{W}_a\hat{V}^{-1}\,, \tag{2.8}$$

respectively. Let $\underline{E} = t^{-1}\underline{V}^{\#} - \underline{V}$, $\underline{E}_a = t^{-1}\underline{W}_a^{\#} - \underline{W}_a$ be the error matrices of $\underline{V}$ and $\underline{W}_a$ respectively. For ergodic processes, $\underline{E}$, $\underline{E}_a$ asymptotically converge to

zero almost surely. So in the following analysis we (1) assume that the entries in $\underline{E}$ and $\underline{E}_a$ are sufficiently small and of the same order; (2) only consider the first order terms and omit those terms of higher orders.

It is well known for a sufficiently small matrix $A \in \mathbb{R}^{m\times m}$ (i.e., its *spectral radius* $\rho(A) < 1$), $(I_m - A)^{-1}$ exists and has the following Taylor expansion:

$$(I_m - A)^{-1} = I_m + A + A^2 + \cdots + A^n + \cdots = I_m + A + O(\|A\|^2)\,,$$

where $O(\|A\|^2)$ denotes the sum of all second and higher order terms. Since the error matrices $\underline{E}$ and $\underline{E}_a$ are small enough, by eqn (2.8) and the definition of $\underline{E}$ and $\underline{E}_a$ one can evaluate

$$\begin{aligned}
\hat{\tau}_a &= (W_a + P\underline{E}_a Q^{\mathsf{T}})(V + P\underline{E}Q^{\mathsf{T}})^{-1} \\
&= (W_a + P\underline{E}_a Q^{\mathsf{T}})(I_m + V^{-1}P\underline{E}Q^{\mathsf{T}})^{-1}V^{-1} \\
&= (W_a + P\underline{E}_a Q^{\mathsf{T}})\{I_m - V^{-1}P\underline{E}Q^{\mathsf{T}} + O(\|\underline{E}\|^2)\}V^{-1} \\
&= (W_a + P\underline{E}_a Q^{\mathsf{T}})\{V^{-1} - V^{-1}P\underline{E}Q^{\mathsf{T}}V^{-1} + O(\|\underline{E}\|^2)\} \\
&= \tau_a - \tau_a P\underline{E}Q^{\mathsf{T}}V^{-1} + P\underline{E}_a Q^{\mathsf{T}}V^{-1} + O(\|\underline{E}\|^2)\,. \qquad (2.9)
\end{aligned}$$

Similarly, $V^{-1} = (\hat{V} - P\underline{E}Q^{\mathsf{T}})^{-1} = \hat{V}^{-1} + \hat{V}^{-1}P\underline{E}Q^{\mathsf{T}}\hat{V}^{-1} + O(\|\underline{E}\|^2)$. Substituting this identity into eqn (2.9) and omitting all $O(\|E\|^2)$ terms, one obtains

$$\hat{\tau}_a \approx \tau_a - \tau_a P\underline{E}Q^{\mathsf{T}}\hat{V}^{-1} + P\underline{E}_a Q^{\mathsf{T}}\hat{V}^{-1}\,,$$

and so

$$\|\hat{\tau}_a - \tau_a\| \leqslant (\|\tau_a\| \cdot \|\underline{E}\| + \|\underline{E}_a\|) \cdot \|P\| \cdot \|Q^{\mathsf{T}}\hat{V}^{-1}\|\,. \qquad (2.10)$$

In the above discussion, $\|\cdot\|$ can be any *consistent* matrix norm, i.e., the norm with the property $\|AB\| \leqslant \|A\| \cdot \|B\|$ for any matrices A, B. In this report we will use the Frobenius norm $\|A\| := \sqrt{\mathrm{tr}(A^{\mathsf{T}}A)}$ for computational reasons.

According to eqn (2.10), to control the estimation error $\|\hat{\tau}_a - \tau_a\|$, one should control the value of $\kappa_1 := \|P\| \cdot \|Q^{\mathsf{T}}\hat{V}^{-1}\|$. The robust algorithm uses a similar quantity $\kappa_2 := \|\hat{V}\|_2 \cdot \|\hat{V}^{-1}\|_2$ [8], the condition number of the matrix $\hat{V}$ measured by the spectrum norm $\|\cdot\|_2$. By the definition of $\hat{V} = t^{-1}P\underline{V}^{\#}Q^{\mathsf{T}}$, the above two quantities κ_1 and κ_2 can be expanded into

$$\begin{aligned}
\kappa_1 &= t\|P\| \cdot \|Q^{\mathsf{T}}(P\underline{V}^{\#}Q^{\mathsf{T}})^{-1}\|\,, \qquad (2.11) \\
\kappa_2 &= \|P\underline{V}^{\#}Q^{\mathsf{T}}\|_2 \cdot \|(P\underline{V}^{\#}Q^{\mathsf{T}})^{-1}\|_2\,, \qquad (2.12)
\end{aligned}$$

They all characterize the condition of the learning equation (2.7); and we will call them *condition indicators* of the learning equation in the sequel.

We now briefly introduce Kretzschmar's work on how to minimize the condition indicator κ_2 under the constraint $\mathbf{1}_m^{\mathsf{T}}P = \mathbf{1}_N^{\mathsf{T}}$, given the counting matrix $\underline{V}^{\#}$. The reader is referred to [8] for the detailed procedure.

Let $\underline{V}^{\#} = L\Sigma R^{\mathsf{T}}$ be the SVD of $\underline{V}^{\#}$, where Σ is a diagonal matrix with diagonal entries $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_N (\geqslant 0)$ and $L, R \in \mathbb{R}^{N\times N}$ are *orthonormal matrices*, i.e.,

$L^\mathsf{T}L = R^\mathsf{T}R = I_N$ — the identity matrix of order N. In the robust algorithm, the SVD of $\underline{V}^{\#}$ is rewritten as

$$\underline{V}^{\#} = \begin{bmatrix} L_1 & L_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} R_1 & R_2 \end{bmatrix}^\mathsf{T} = L_1\Sigma_1 R_1^\mathsf{T} + L_2\Sigma_2 R_2^\mathsf{T}\,,$$

where $\Sigma_1 = \mathrm{diag}\{\sigma_1, \cdots, \sigma_m\}$ and 0, which denotes zero matrices here, and other matrices have the corresponding shape. The matrix Q is then fixed to be R_1^T and P is parameterized (by Φ and θ_i) as

$$P = \Phi \cdot \mathrm{diag}\{\theta_1, \cdots, \theta_m\} \cdot L_1^\mathsf{T} + \boldsymbol{e}_m \mathbf{1}_N^\mathsf{T} L_2 L_2^\mathsf{T}$$

with Φ being an $m \times m$ orthonormal matrix and $\theta_i \geqslant 0$ $(i = 1, 2, \cdots, m)$.

With the above setting of P and Q, direct computation shows that

$$\begin{aligned} P\underline{V}^{\#}Q^\mathsf{T} &= \Phi \cdot \mathrm{diag}\{\theta_1\sigma_1, \cdots, \theta_m\sigma_m\}\,, \\ \mathbf{1}_m^\mathsf{T} PL &= \begin{bmatrix} \mathbf{1}_m^\mathsf{T}\Phi \cdot \mathrm{diag}\{\theta_1, \cdots, \theta_m\}\,, & \mathbf{1}_N^\mathsf{T} L_2 \end{bmatrix}\,. \end{aligned}$$

So the constraint $\mathbf{1}_m^\mathsf{T} P = \mathbf{1}_N^\mathsf{T}$ holds iff $\mathbf{1}_m^\mathsf{T}\Phi \cdot \mathrm{diag}\{\theta_1, \cdots, \theta_m\} = \mathbf{1}_N^\mathsf{T} L_1$ and the matrix $P\underline{V}^{\#}Q^\mathsf{T}$ has the singular values $\{\theta_1\sigma_1, \cdots, \theta_m\sigma_m\}$. It follows that

$$\kappa_2 = \frac{\max\{\theta_i\sigma_i : i = 1, 2, \cdots, m\}}{\min\{\theta_i\sigma_i : i = 1, 2, \cdots, m\}}\,.$$

It is now obvious that $\kappa_2 \geqslant 1$ and the equality holds if and only if $\theta_i\sigma_i = \mu$ for all $i = 1, 2, \cdots, m$. Thus, the problem of minimizing κ_2 amounts to minimizing an error function, e.g., $J = \sum_{i=1}^m (\theta_i\sigma_i - \mu)^2$, under the constraints $\Phi^\mathsf{T}\Phi = I_m$ and $\mathbf{1}_m^\mathsf{T}\Phi \cdot \mathrm{diag}\{\theta_1, \cdots, \theta_m\} = \mathbf{1}_N^\mathsf{T} L_1$.

Kretzschmar then parameterized the matrix Φ by *anti-symmetric matrices*, i.e., by matrices ω with the property $\omega^\mathsf{T} = -\omega$. In fact, the matrix exponential $\exp(\omega) := \sum_{n=0}^\infty \omega^n/n!$ of any anti-symmetric matrix $\omega \in \mathbb{R}^{m\times m}$ is orthonormal. So one can set $\Phi = \exp(\omega)$ and evaluate θ_i by $\theta_i = (\mathbf{1}_N^\mathsf{T} L_1 \boldsymbol{e}_i)/(\mathbf{1}_m^\mathsf{T} \exp(\omega)\boldsymbol{e}_i)$, which is derived from the constraint $\mathbf{1}_m^\mathsf{T}\Phi \cdot \mathrm{diag}\{\theta_1, \cdots, \theta_m\} = \mathbf{1}_N^\mathsf{T} L_1$. In this way, we have converted the original constrained problem to an unconstrained optimization problem so that standard numerical methods can be utilized.

The robust learning algorithm of OOMs can now be summarized as follows: **1.** get the counting matrices $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$ from data; **2.** calculate the SVD of $\underline{V}^{\#}$; **3.** minimize $J(\omega, \mu) = \sum_{i=1}^m [\theta_i(\omega)\sigma_i - \mu]^2$ to get the minimal point ω^*; **4.** calculate the matrices P, Q from ω^*; **5.** estimate $\hat{\tau}_a$ by eqn (2.7). This procedure, however, is quite expensive and therefore unsuited to online learning.

Moreover, as has been pointed out earlier, although some positive results are reported for the robust algorithm, its starting criterion for selecting the matrices P, Q is not theoretically verified. Because one can arbitrarily select a matrix P such that $\mathbf{1}_m^\mathsf{T} P = \mathbf{1}_N^\mathsf{T}$ and $P\underline{V}^{\#}$ has rank m (almost all P with column sums 1 will do) and compute $Q^\mathsf{T} = (P\underline{V}^{\#})^\dagger$, then $P\underline{V}^{\#}Q^\mathsf{T} = I_m$ and κ_2 reaches its minimum

1. In other words, if we use minimization of κ_2 as the only criterion for designing P and Q, we are almost free to choose P (and then compute $Q^{\mathsf{T}} = (P\underline{V}^{\#})^{\dagger}$). So the complicated Kretzschmar computation is unnecessary; and the theoretical foundation of the robust algorithm is incomplete. In conclusion, one should use κ_1, not κ_2, as the target function for selecting the matrices P and Q.

3 The Error Controlling Algorithm

Based upon the above discussion, we sketch out the offline version of our error controlling algorithm in three steps:

1. Get the counting matrices $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$'s from the training sequence.
2. Minimize the condition indicator κ_1 to get the matrices P, Q.
3. Estimate the observable operators $\hat{\tau}_a$ by the learning equation (2.7).

To make this learning procedure online, one should consider the online form of the above three steps one by one. For the first step, the counting matrices $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$ are updated by eqn (2.6), which is already in online form. We next consider the other two steps.

3.1 Making the Second Step Online

Assume at time t we have gotten $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$. Then by eqn (2.11) in the second step we need to calculate the matrices P, Q via the optimization problem:

$$\begin{array}{rl} \text{minimize} & \kappa_1 = t\|P\| \cdot \|Q^{\mathsf{T}}(P\underline{V}^{\#}Q^{\mathsf{T}})^{-1}\|\,, \\ \text{subject to} & \mathbf{1}_m^{\mathsf{T}} P = \mathbf{1}_N^{\mathsf{T}}\,. \end{array} \tag{3.1}$$

Although numerical algorithms for general optimization problems such as the well known *gradient descent* (GD) method can be employed for problem (3.1), they are computationally too expensive to be used online because for online learning it requires the computation of the derivative of κ_1 w.r.t. P and Q for each time step. On the other hand, if one can keep the value of κ_1 under some predefined threshold κ during the learning procedure, then by (2.10) the estimation error $\|\hat{\tau}_a - \tau_a\|$ will converge to zero as t goes to infinity. This is because for ergodic processes the errors $\underline{E}, \underline{E}_a$ in probability matrices converges to zero when observations accumulate. So minimizing (3.1) at each time step is actually not necessary, what needed is a cheap updating scheme of P, Q so that the condition of the learning equation will not exceed some predefined threshold.

One such updating scheme will now be outlined. It is an iterative method: at each time step, if the target function κ_1 exceeds some predefined threshold κ, the matrices P, Q are modified column by column, unless some stop criterion is satisfied; otherwise, P and Q remain unchanged for this step. We call it the *column-by-column* (CbC) scheme. The detailed procedure of CbC is as follows.

We arrange the columns of P and Q in an ordered list $\{\boldsymbol{p}_1, \boldsymbol{q}_1, \cdots, \boldsymbol{p}_N, \boldsymbol{q}_N\}$, where $\boldsymbol{p}_n$ (resp. $\boldsymbol{q}_n$) denotes the n-th column of P (resp. Q); and make a cycle pointer running over the list. Whenever the threshold κ is exceeded, (only) the column under the pointer will be modified, with all other columns remaining unchanged. The pointer is then moved right to the next column and the same modification scheme is repeated until the condition indicator κ_1 drops below some level κ' or reaches the plateau of decreasing. Here κ' is usually set to be $0.1\kappa \sim 0.5\kappa$ so that the CbC scheme will not be triggered too frequently.

The above CbC scheme leads to two optimization problems: minimizing κ_1 w.r.t. $\boldsymbol{p}_n$ or $\boldsymbol{q}_n$, denoted by *min-*$\kappa_1(\boldsymbol{p}_n)$ and *min-*$\kappa_1(\boldsymbol{q}_n)$, respectively. The problem *min-*$\kappa_1(\boldsymbol{q}_n)$ has an analytical solution, as shown below.

Assume $\boldsymbol{q}_n$ is the currently pointed column and is modified to $\boldsymbol{q}_n^* = \boldsymbol{q}_n + \boldsymbol{h}$. In matrix notation this can be written as $Q^* = Q + \boldsymbol{h}\boldsymbol{e}_n^\mathsf{T}$. By eqn (2.11) we know $\kappa_1^* = t\|P\| \cdot \|Q^{*\mathsf{T}}(P\underline{V}^{\#}Q^{*\mathsf{T}})^{-1}\|$. Here and in the sequel, the superscript * always denotes the updated values due to the modification of P or Q. To simplify the notation we define three auxiliary matrices $X = P\underline{V}^{\#}Q^\mathsf{T}$, $Z = X^{-1}$ and $B = Q^\mathsf{T}Z$, then $X^* = P\underline{V}^{\#}Q^{*\mathsf{T}} = X + P\underline{V}^{\#}\boldsymbol{e}_n\boldsymbol{h}^\mathsf{T} =: X + \boldsymbol{u}_n\boldsymbol{h}^\mathsf{T}$. By the Sherman-Morrison theorem (rank-one modification of matrix inverse) [10] we know

$$Z^* = (X + \boldsymbol{u}_n\boldsymbol{h}^\mathsf{T})^{-1} = Z - \frac{Z\boldsymbol{u}_n\boldsymbol{h}^\mathsf{T}Z}{1 + \boldsymbol{h}^\mathsf{T}Z\boldsymbol{u}_n}\,. \tag{3.2}$$

Since P is fixed, minimizing κ_1^* is equivalent to minimizing the norm of

$$Q^{*\mathsf{T}}Z^* = (Q^\mathsf{T} + \boldsymbol{e}_n\boldsymbol{h}^\mathsf{T})Z^* = Q^\mathsf{T}Z - \frac{(Q^\mathsf{T}Z\boldsymbol{u}_n - \boldsymbol{e}_n)\boldsymbol{h}^\mathsf{T}Z}{1 + \boldsymbol{h}^\mathsf{T}Z\boldsymbol{u}_n}\,. \tag{3.3}$$

For the Frobenius norm of matrices $\|A\| = \sqrt{\mathrm{tr}(A^\mathsf{T}A)}$, direct computation from its definition reveals that

$$\|A + \boldsymbol{b}\boldsymbol{c}^\mathsf{T}\|^2 = \|A\|^2 + 2\boldsymbol{b}^\mathsf{T}A\boldsymbol{c} + \|\boldsymbol{b}\|^2 \cdot \|\boldsymbol{c}\|^2\,. \tag{3.4}$$

So by setting $\boldsymbol{g}_n = Q^\mathsf{T}Z\boldsymbol{u}_n - \boldsymbol{e}_n$ and $\boldsymbol{x}^\mathsf{T} = \boldsymbol{h}^\mathsf{T}Z/(1 + \boldsymbol{h}^\mathsf{T}Z\boldsymbol{u})$, one obtains

$$\|Q^{*\mathsf{T}}Z^*\|^2 = \|Q^\mathsf{T}Z\|^2 - 2\boldsymbol{g}_n^\mathsf{T}Q^\mathsf{T}Z\boldsymbol{x} + \|\boldsymbol{g}_n\|^2 \cdot \|\boldsymbol{x}\|^2\,.$$

This is a quadratic function of $\boldsymbol{x}$, with $\boldsymbol{x}_{\min} = \|\boldsymbol{g}_n\|^{-2} \cdot (Z^\mathsf{T}Q\boldsymbol{g}_n)$ as the unique minimum. By the definition of $\boldsymbol{x}$ we conclude $\boldsymbol{h}_{\min} = -Q\boldsymbol{g}_n/(\boldsymbol{e}_n^\mathsf{T}\boldsymbol{g}_n)$. [4]

Summing up, one modifies the matrix Q (column-wisely) by (i) computing $\boldsymbol{u}_n = P\underline{V}^{\#}\boldsymbol{e}_n$ and $\boldsymbol{g}_n = Q^\mathsf{T}Z\boldsymbol{u}_n - \boldsymbol{e}_n$; (ii) modifying $\boldsymbol{q}_n$ by $\boldsymbol{q}_n^* = \boldsymbol{q}_n + \boldsymbol{h}$ with

$$\boldsymbol{h} = \begin{cases} -Q\boldsymbol{g}_n/(\boldsymbol{e}_n^\mathsf{T}\boldsymbol{g}_n) & \text{if } \boldsymbol{e}_n^\mathsf{T}\boldsymbol{g}_n \neq 0 \\ Q\boldsymbol{g}_n & \text{otherwise} \end{cases}\,. \tag{3.5}$$

[4] If $\boldsymbol{e}_n^\mathsf{T}\boldsymbol{g}_n = 0$, then $\|Q^{*\mathsf{T}}Z^*\|^2$, as a function of $\boldsymbol{h}$, has no minimal point. For this case, one can set $\boldsymbol{h} = Q\boldsymbol{g}_n$, for which it can be easily proven that $\|Q^{*\mathsf{T}}Z^*\| < \|Q^\mathsf{T}Z\|$.

Now we consider the problem *min*-$\kappa_1(\boldsymbol{p}_n)$. Assume that $\boldsymbol{p}_n^* = \boldsymbol{p}_n + \boldsymbol{h}$, or equivalently, $P^* = P + \boldsymbol{h}\boldsymbol{e}_n^\mathsf{T}$, then $\kappa_1^* = t\|P^*\| \cdot \|Q^\mathsf{T}(P^*\underline{V}^{\#}Q^\mathsf{T})^{-1}\|$. From the constraint $\mathbf{1}_m^\mathsf{T}P = \mathbf{1}_m^\mathsf{T}$ we know $\mathbf{1}_m^\mathsf{T}\boldsymbol{h} = 0$. Thus, the vector $\boldsymbol{h}$ is determined by the optimization problem:

$$\begin{array}{rl} \text{minimize} & \|P + \boldsymbol{h}\boldsymbol{e}_n^\mathsf{T}\| \cdot \|Q^\mathsf{T}\{(P + \boldsymbol{h}\boldsymbol{e}_n^\mathsf{T})\underline{V}^{\#}Q^\mathsf{T}\}^{-1}\|\,, \\ \text{subject to} & \mathbf{1}_m^\mathsf{T}\boldsymbol{h} = 0\,. \end{array} \tag{3.6}$$

For this special problem we proposed an efficient method that iteratively searches for the *global* optimum on two orthogonal directions in $\mathbb{R}^m$, which is hence called the *orthogonal searching* (OS) algorithm. We postpone the introduction of the OS algorithm to the next section and turn now to the third step of the offline EC algorithm.

3.2 Making the Third Step Online

Assume that at time t we have gotten the matrices $\underline{V}^{\#}$, $\underline{W}_a^{\#}$; P, Q and $\hat{\tau}_a$; and that at time $t+1$ the event $(j, c, k, i)_{t+1}$ is observed. Then by eqns (2.6) and (2.7),

$$\begin{aligned} \hat{\tau}_a^+ &= \{P(\underline{W}_a^{\#} + \delta_{ac}\boldsymbol{e}_i\boldsymbol{e}_j^\mathsf{T})Q^\mathsf{T}\}\{P(\underline{V}^{\#} + \boldsymbol{e}_k\boldsymbol{e}_j^\mathsf{T})Q^\mathsf{T}\}^{-1} \\ &= (P\underline{W}_a^{\#}Q^\mathsf{T} + \delta_{ac}\boldsymbol{p}_i\boldsymbol{q}_j^\mathsf{T})(P\underline{V}^{\#}Q^\mathsf{T} + \boldsymbol{p}_k\boldsymbol{q}_j^\mathsf{T})^{-1}\,. \end{aligned} \tag{3.7}$$

It follows from the definition of $Z = X^{-1} = (P\underline{V}^{\#}Q^\mathsf{T})^{-1}$ and the rank-one modification of matrix inverse that $(P\underline{V}^{\#}Q^\mathsf{T} + \boldsymbol{p}_k\boldsymbol{q}_j^\mathsf{T})^{-1} = Z - (1 + \boldsymbol{q}_j^\mathsf{T}Z\boldsymbol{p}_k)^{-1}(Z\boldsymbol{p}_k\boldsymbol{q}_j^\mathsf{T}Z)$; and from eqn (2.7) that $\hat{\tau}_a = P\underline{W}_a^{\#}Q^\mathsf{T}Z$. Substituting these two equalities into eqn (3.7), we get the update rules for observable operators $\hat{\tau}_a$:

$$\hat{\tau}_a^+ = \hat{\tau}_a - [\hat{\tau}_a\boldsymbol{p}_k - \delta_{ac}\boldsymbol{p}_i]\boldsymbol{s}_{jk}^\mathsf{T} \quad \text{with} \quad \boldsymbol{s}_{jk}^\mathsf{T} := \frac{\boldsymbol{q}_j^\mathsf{T}Z}{1 + \boldsymbol{q}_j^\mathsf{T}Z\boldsymbol{p}_k}\,. \tag{3.8}$$

Thus far we have derived the online EC algorithm for estimating OOMs from empirical data, for which a block diagram is presented in Figure 1. From Figure 1 we see that the matrix $Z = (P\underline{V}^{\#}Q^\mathsf{T})^{-1}$ plays a central role in the online learning algorithm. Other quantities can be represented as functions of Z, for example, $\kappa_1 = t\|P\| \cdot \|Q^\mathsf{T}Z\|$. The update formula for Z is as follows: upon the observation of the event $(j, c, k, i)_{t+1}$,

$$Z^+ = Z - \frac{Z\boldsymbol{p}_k\boldsymbol{q}_j^\mathsf{T}Z}{1 + \boldsymbol{q}_j^\mathsf{T}Z\boldsymbol{p}_k} = Z - Z\boldsymbol{p}_k\boldsymbol{s}_{jk}^\mathsf{T}\,, \tag{3.9}$$

with the vector $\boldsymbol{s}_{jk}$ defined as in eqn (3.8).

Note that the condition indicator κ_1 of the learning equation can also be updated online. For instance, when $\kappa_1 \leqslant \kappa$, i.e., the matrices P, Q are fixed, we have $\kappa_1^+ = (t+1)\sqrt{F(P)F(Q^\mathsf{T}Z^+)} = (t+1)\sqrt{F(P)F(B^+)}$ with

$$F(B^+) = F(B) - 2\boldsymbol{p}_k^\mathsf{T}B^\mathsf{T}B\boldsymbol{s}_{jk} + \|B\boldsymbol{p}_k\|^2 \cdot \|\boldsymbol{s}_{jk}\|^2\,, \tag{3.10}$$

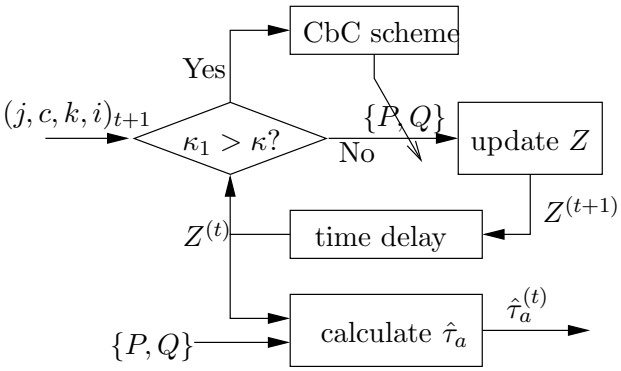


Figure 1: The block diagram of the online EC algorithm.

in which $F(\cdot) := \|\cdot\|^2$ means the square of Frobenius norm and the identity (3.4) is utilized. Other update rules, e.g., the one for κ_1 due to the modification of P or Q, can be derived in the similar way (see Table 2).

At this point, we find it convenient to list all involved variables and formulas that will be consistently used in this report in tables (see Table 1 and Table 2). Basically, all the formulas in Table 2 are derived from eqns (3.2) and (3.4). Note that symbols not listed in Table 1 may have different meanings in different places; and we believe the readers will not be confused by such abuse of symbols.

Table 1: Variables in the online algorithm

Symbols	**Description**
m, ℓ	resp. dimension of estimated OOMs, alphabet size;
d, N	resp. length and number of basic strings, $N = \ell^d$;
$\bar{s}$, t	resp. training data, current time or effective length of $\bar{s}$;
$\underline{V}^{\#}$, $\underline{W}_a^{\#}$	counting matrices; $\underline{V}^{\#}, \underline{W}_a^{\#} \in \mathbb{R}^{N\times N}$
P, Q	adjustable parameter matrices; $P, Q \in \mathbb{R}^{m\times N}$
X, Z, B	auxiliary matrices, $X = P\underline{V}^{\#}Q^{\mathsf{T}}$, $Z = X^{-1}$, $B = Q^{\mathsf{T}}Z$;
$F(\cdot)$	square of Frobenius norm, $F(A) = \|A\|^2$;
$\hat{\tau}_a$	estimated observable operators, $\hat{\tau}_a = P\underline{W}_a^{\#}B$;
κ_1	condition indicator, $\kappa_1 = t\sqrt{F(P)F(B)}$;

3.3 Determining the Threshold κ Theoretically

Figure 1 shows the overall framework of the online EC algorithm, in which two problems remain unsolved: the implementation of the CbC scheme and the value of the threshold κ. In this subsection we discuss the latter, leaving the former to the next section. More concretely, in the following we will derive (in theory), for

Table 2: Formulas in the online algorithm

Events	**(Possibly) involved formulas**
$(j,c,k,i)_{t+1}$	$(\underline{V}^{\#})^{+} = \underline{V}^{\#} + \boldsymbol{e}_k\boldsymbol{e}_j^{\mathsf{T}}$, $\quad (\underline{W}_a^{\#})^{+} = \underline{W}_a^{\#} + \delta_{ac}\boldsymbol{e}_i\boldsymbol{e}_j^{\mathsf{T}}$; [set $\boldsymbol{s}_{jk}^{\mathsf{T}} = (\boldsymbol{q}_j^{\mathsf{T}}Z)/(1+\boldsymbol{q}_j^{\mathsf{T}}Z\boldsymbol{p}_k)$], $Z^{+} = Z - Z\boldsymbol{p}_k\boldsymbol{s}_{jk}^{\mathsf{T}}$, $\quad B^{+} = B - B\boldsymbol{p}_k\boldsymbol{s}_{jk}^{\mathsf{T}}$; $\hat{\tau}_a^{+} = \hat{\tau}_a - [\hat{\tau}_a\boldsymbol{p}_k - \delta_{ac}\boldsymbol{p}_i]\boldsymbol{s}_{jk}^{\mathsf{T}}$; $F(B^{+}) = F(B) - 2\boldsymbol{p}_k^{\mathsf{T}}B^{\mathsf{T}}B\boldsymbol{s}_{jk} + \|B\boldsymbol{p}_k\|^2\|\boldsymbol{s}_{jk}\|^2$, $\kappa_1^{+} = (t+1)\sqrt{F(P)F(B^{+})}$.
$\boldsymbol{p}_n^{*} = \boldsymbol{p}_n + \boldsymbol{h}$	[set $\boldsymbol{v}_n^{\mathsf{T}} = \boldsymbol{e}_n^{\mathsf{T}}\underline{V}^{\#}Q^{\mathsf{T}}$, $\quad \boldsymbol{s}_n^{\mathsf{T}} = (\boldsymbol{v}_n^{\mathsf{T}}Z)/(1+\boldsymbol{v}_n^{\mathsf{T}}Z\boldsymbol{h})$], $Z^{*} = Z - Z\boldsymbol{h}\boldsymbol{s}_n^{\mathsf{T}}$, $\quad B^{*} = B - B\boldsymbol{h}\boldsymbol{s}_n^{\mathsf{T}}$; $\hat{\tau}_a^{*} = \hat{\tau}_a - \hat{\tau}_a\boldsymbol{h}\boldsymbol{s}_n^{\mathsf{T}} + \boldsymbol{h}\boldsymbol{e}_n^{\mathsf{T}}\underline{W}_a^{\#}B - (\boldsymbol{e}_n^{\mathsf{T}}\underline{W}_a^{\#}B\boldsymbol{h})\boldsymbol{h}\boldsymbol{s}_n^{\mathsf{T}}$; $F(P^{*}) = F(P) + 2\boldsymbol{h}^{\mathsf{T}}P\boldsymbol{e}_n + \|\boldsymbol{h}\|^2$, $F(B^{*}) = F(B) - 2\boldsymbol{h}^{\mathsf{T}}B^{\mathsf{T}}B\boldsymbol{s}_n + \|B\boldsymbol{h}\|^2\|\boldsymbol{s}_n\|^2$, $\kappa_1^{*} = t\sqrt{F(P^{*})F(B^{*})}$.
$\boldsymbol{q}_n^{*} = \boldsymbol{q}_n + \boldsymbol{h}$	[set $\boldsymbol{u}_n = P\underline{V}^{\#}\boldsymbol{e}_n$, $\gamma = (1+\boldsymbol{h}^{\mathsf{T}}Z\boldsymbol{u}_n)^{-1}$, $\boldsymbol{r}_n = \gamma Z\boldsymbol{u}_n$], [set $\boldsymbol{g}_n = B\boldsymbol{u}_n - \boldsymbol{e}_n$ and $\boldsymbol{x} = \gamma Z^{\mathsf{T}}\boldsymbol{h}$]; $Z^{*} = Z - \boldsymbol{r}_n\boldsymbol{h}^{\mathsf{T}}Z$, $\quad B^{*} = B - Q^{\mathsf{T}}\boldsymbol{r}_n\boldsymbol{h}^{\mathsf{T}}Z$; $\hat{\tau}_a^{*} = \hat{\tau}_a - \gamma\hat{\tau}_a\boldsymbol{u}_n\boldsymbol{h}^{\mathsf{T}}Z + \gamma P\underline{W}_a^{\#}\boldsymbol{e}_n\boldsymbol{h}^{\mathsf{T}}Z$; $F(B^{*}) = F(B) - 2\boldsymbol{g}_n^{\mathsf{T}}B\boldsymbol{x} + \|\boldsymbol{g}_n\|^2\|\boldsymbol{x}\|^2$, $\kappa_1^{*} = t\sqrt{F(P)F(B^{*})}$.

a given counting matrix $\underline{V}^{\#}$, an upper bound of the minimal value of κ_1, which can hence be taken as the threshold κ.

Let $\underline{V}_0^{\#} = t^{-1}\underline{V}^{\#}$ be the normalized counting matrix at time t, then eqn (2.11) can be rewritten as $\kappa_1 = \|P\| \cdot \|Q^{\mathsf{T}}(P\underline{V}_0^{\#}Q^{\mathsf{T}})^{-1}\|$. Assume κ_1 obtains its minimal $\kappa_1^{\min}$ at $P = P_1$ and $Q = Q_1$. Define $P_2 = P_1$ and $Q_2 = Q_1^{\mathsf{T}}(P_1\underline{V}_0^{\#}Q_1^{\mathsf{T}})^{-1}$. Then it holds that $\kappa_1(P_2,Q_2) = \kappa_1(P_1,Q_1)$ and that $P_2\underline{V}_0^{\#}Q_2^{\mathsf{T}} = I_m$. Thus, adding the constraint $P\underline{V}_0^{\#}Q^{\mathsf{T}} = I_m$ will not change the minimal value of κ_1; and the problem is reduced to minimizing the quantity $\|P\| \cdot \|Q^{\mathsf{T}}\|$ under the constraints $P\underline{V}_0^{\#}Q^{\mathsf{T}} = I_m$ and $\mathbf{1}_m^{\mathsf{T}}P = \mathbf{1}_N^{\mathsf{T}}$. That is, we need only to consider the problem

$$\min_{P,Q}\left\{\|P\| \cdot \|Q^{\mathsf{T}}\| : P\underline{V}_0^{\#}Q^{\mathsf{T}} = I_m, \mathbf{1}_m^{\mathsf{T}}P = \mathbf{1}_N^{\mathsf{T}}\right\}. \tag{3.11}$$

Fix the matrix P, then (3.11) is equivalent to a quadratic form of Q with the linear equality constraint $P\underline{V}_0^{\#}Q^{\mathsf{T}} = I_m$. By using the Lagrange function $\mathcal{L}(Q,\Lambda) = \frac{1}{2}\operatorname{tr}(Q^{\mathsf{T}}Q) - \operatorname{tr}[\Lambda^{\mathsf{T}}(P\underline{V}_0^{\#}Q^{\mathsf{T}} - I_m)]$, where $\Lambda \in \mathbb{R}^{m\times m}$ is the Lagrange multiplier, we get the analytical solution of (3.11): $Q^{\mathsf{T}} = (P\underline{V}_0^{\#})^{\dagger}$. Here the superscript † is the pseudo-inverse operation of matrices, which, for our case, is computed by $A^{\dagger} = A^{\mathsf{T}}(AA^{\mathsf{T}})^{-1}$. So the problem is further reduced to

$$\min_{P}\{\|P\| \cdot \|(P\underline{V}_0^{\#})^{\dagger}\| : \mathbf{1}_m^{\mathsf{T}}P = \mathbf{1}_N^{\mathsf{T}},\ \operatorname{rank}(P\underline{V}_0^{\#}) = m\}. \tag{3.12}$$

Next we will take a special form of P and consider the problem (3.12) for such special matrices P. Let $\underline{V}_0^{\#} = L \cdot \mathrm{diag}\{\sigma_1, \cdots, \sigma_N\} \cdot R^{\mathsf{T}}$ be the SVD of $\underline{V}_0^{\#}$ with $\sigma_1 \geqslant \cdots \geqslant \sigma_N \geqslant 0$; and take

$$P = \Phi \cdot \left[\theta_1 \boldsymbol{e}_{k_1}, \ \cdots, \ \theta_N \boldsymbol{e}_{k_N}\right] \cdot L^{\mathsf{T}}, \tag{3.13}$$

where $\Phi \in \mathbb{R}^{m\times m}$ is an orthonormal matrix, θ_i's are real numbers and $\boldsymbol{e}_{k_i}$, as mentioned before, denotes the k_i-th unit column vector of dimension, for this situation, m. So here all k_i's take values from $\{1, 2, \cdots, m\}$. It follows that

$$\begin{aligned} P\underline{V}_0^{\#} &= \Phi \cdot \left[\sigma_1\theta_1 \boldsymbol{e}_{k_1}, \ \cdots, \ \sigma_N\theta_N \boldsymbol{e}_{k_N}\right] \cdot R^{\mathsf{T}}, \\ (P\underline{V}_0^{\#})(P\underline{V}_0^{\#})^{\mathsf{T}} &= \Phi \cdot \mathrm{diag}\{\varphi_1, \cdots, \varphi_m\} \cdot \Phi^{\mathsf{T}}, \\ Q^{\mathsf{T}} = (P\underline{V}_0^{\#})^{\dagger} &= R \cdot \left[\sigma_1\theta_1\varphi_{k_1}^{-1} \boldsymbol{e}_{k_1}, \ \cdots, \ \sigma_N\theta_N\varphi_{k_N}^{-1} \boldsymbol{e}_{k_N}\right]^{\mathsf{T}} \cdot \Phi^{\mathsf{T}}, \end{aligned}$$

where $\varphi_j = \sum_{i:k_i=j} \sigma_i^2\theta_i^2$ for all $j = 1, 2, \cdots, m$. Thus,

$$\|P\|^2 \cdot \|Q^{\mathsf{T}}\|^2 = \left(\sum_{i=1}^{N} \theta_i^2\right) \cdot \left(\sum_{j=1}^{m} \varphi_j^{-2} \sum_{i:k_i=j} \sigma_i^2\theta_i^2\right) = \left(\sum_{i=1}^{N} \theta_i^2\right) \cdot \left(\sum_{j=1}^{m} \varphi_j^{-1}\right).$$

Put $\rho_j = \sum_{i:k_i=j} \theta_i^2$ for each $j = 1, 2, \cdots, m$ and $\alpha_i = (\sqrt{\rho_{k_i}})^{-1}|\theta_i|$ for each $i = 1, 2, \cdots, N$, by the above equality and Cauchy's inequality,

$$\begin{aligned} \|P\|^2 \cdot \|Q^{\mathsf{T}}\|^2 &= \left[\sum_{j=1}^m \rho_j\right] \cdot \left[\sum_{j=1}^m \rho_j^{-1}\Big(\sum_{i:k_i=j} \sigma_i^2\alpha_i^2\Big)^{-1}\right] \\ &\geqslant \left[\sum_{j=1}^m \Big(\sum_{i:k_i=j} \sigma_i^2\alpha_i^2\Big)^{-\frac{1}{2}}\right]^2, \end{aligned}$$

with the equality holds if and only if $\rho_j^2 \sum_{i:k_i=j} \sigma_i^2\alpha_i^2$ is a constant (not depending on j). Therefore, $\kappa_1' := \sum_{j=1}^m \big(\sum_{i:k_i=j} \sigma_i^2\alpha_i^2\big)^{-\frac{1}{2}}$ represents an attainable (lower) level of κ_1, an upper bound of $\kappa_1^{\min}$; and one should set the indices k_i as well as the weights α_i such that κ_1' is as small as possible. By the definition of α_i's we have $\sum_{i:k_i=j} \alpha_i^2 = 1$ for each j. So one can easily see that κ_1' reaches its minimal $\kappa_1' = \sum_{j=1}^m \sigma_j^{-1}$ under the settings

$$k_i = \begin{cases} i & \text{if } i \leqslant m \\ \text{irrelevant} & \text{if } i > m \end{cases}; \quad \alpha_i = \begin{cases} 1 & \text{if } i \leqslant m \\ 0 & \text{if } i > m \end{cases}.$$

Note that, the quantity $\kappa_1' = \sum_{j=1}^m \sigma_j^{-1}$ is of order $O(\sigma_m^{-1})$, which represents the best upper bound of $\kappa_1^{\min}$ (in the sense of order). In fact, by the definition of κ_1 we have $\kappa_1 \cdot \|\underline{V}_0^{\#}\| = \|P\| \cdot \|\underline{V}_0^{\#}\| \cdot \|Q^{\mathsf{T}}(P\underline{V}_0^{\#}Q^{\mathsf{T}})^{-1}\| \geqslant \|I_m\| = \sqrt{m}$; whereas $\|\underline{V}_0^{\#}\|^2 = \sum_{i=1}^N \sigma_i^2 \leqslant Nm^{-1}\sum_{i=1}^m \sigma_i^2$ for $\sigma_1, \cdots, \sigma_N$ are singular values of $\underline{V}_0^{\#}$ in decreasing order. Thus $\kappa_1 \geqslant mN^{-1/2}(\sum_{i=1}^m \sigma_i^2)^{-1/2} = O(\sigma_m^{-1})$. The value of κ_1', unfortunately, depends on the singular values of $\underline{V}_0^{\#}$. This prevents one directly using κ_1' as the threshold, for calculating the SVD of $\underline{V}_0^{\#}$ at each time step is too expensive.

To get an estimation of $\kappa_1' = \sum_{j=1}^m \sigma_j^{-1}$ here we would remind the reader of the meaning of model dimension m. Real-world systems usually have infinite dimension; and it is impossible and unnecessary to model a real system with 100% accuracy. For the case of OOMs, when we say a process can be modelled by an m-dimensional OOM, we actually mean that its distribution can be numerically approximated by the OOM. More concretely, this means the normalized counting matrix $\underline{V}_0^\#$, as an approximation of the probability matrix $\underline{V}$ (see eqn (2.2)), has numerical rank m. Therefore, if the model dimension m is somehow appropriately selected, the following two assertions must hold.

- The singular values σ_i $(i > m)$ of $\underline{V}_0^\#$ are small; for otherwise $\underline{V}_0^\#$ will have numerical rank $> m$.
- The quantity σ_m^2, the smallest one of σ_j^2's $(j = 1, 2, \cdots, m)$, is large enough; for otherwise the numerical rank of $\underline{V}_0^\#$ will be less than m.

If we define two parameters ϵ_1, ϵ_2 (both depend on m) to be such that

$$\sum_{j=1}^m \sigma_j^2 = (1-\epsilon_1) \sum_{i=1}^N \sigma_i^2 \quad \text{and} \quad \sigma_m^2 = \epsilon_2 m^{-1} \sum_{i=1}^m \sigma_i^2 \,, \tag{3.14}$$

the above two assertions imply the product $(1-\epsilon_1)\epsilon_2$ will not be too small. By the definition of $\underline{V}_0^\#$, its elements are all nonnegative and sum to 1; therefore $\sum_{i=1}^N \sigma_i^2 = \|\underline{V}_0^\#\|^2 \geqslant N^{-2}$, with the equality holds if and only if $[\underline{V}_0^\#]_{ij} = N^{-2}$ for all i, j. This inequality, together with (3.14), implies that

$$\kappa_1' = \sum_{j=1}^m \sigma_j^{-1} \leqslant m\sigma_m^{-1} \leqslant m \left[(1-\epsilon_1)\epsilon_2 m^{-1} N^{-2}\right]^{-\frac{1}{2}} = \gamma m N \sqrt{m} \,. \tag{3.15}$$

The above discussion did not consider the constraint $\mathbf{1}_m^\mathsf{T} P = \mathbf{1}_N^\mathsf{T}$, which would possibly increase the minimal value of κ_1. But as this is not a strong constraint and the upper bound (3.15) is already very loose, so in the online algorithm we use the following thresholds for turning on/off the CbC scheme.

$$\kappa = \gamma m N \sqrt{m} \,, \qquad \kappa' = 0.5 \gamma m N \sqrt{m} \,. \tag{3.16}$$

Notice here that the parameter γ still depends on the model dimension m and the matrix $\underline{V}_0^\#$. In Section 4 we will study its range by means of numerical simulation.

A remark on the parameters ϵ_1 and ϵ_2. — Intuitively, the singular values σ_i of $\underline{V}_0^\#$ can be seen as a measure of information about the underlying process contained in the i-th dimension of the estimated OOM. Thus the parameter ϵ_1 actually represents the loss of information when we use an m-dimensional OOM to describe the process; and ϵ_2 measures the relative information mass contained in the m-th dimension of the OOM. With this explanation (of ϵ_1 and ϵ_2), one can also view ϵ_1 or ϵ_2 as design parameters from which the model dimension m is determined by

$$\sum_{j=1}^m \sigma_j^2 \geqslant (1-\epsilon_1) \sum_{i=1}^N \sigma_i^2 \quad \text{or} \quad \sigma_m^2 \geqslant \epsilon_2 m^{-1} \sum_{i=1}^m \sigma_i^2 \,. \tag{3.17}$$

That is, we require that the estimated OOM describes the process with loss of information no more than ϵ_1; or that each dimension contains relatively large information of the underlying process.

3.4 The Overall Online EC Algorithm

Based on the above discussion, we can now outline the online EC algorithm for training OOMs.

- **Inputs:**
  - the model dimension m and the alphabet size ℓ;
  - the training data $\bar{s} = s_1 s_2 \cdots$ and the length of basic strings d;

- **Auxiliary variables:**
  - the counting matrices $\underline{V}^{\#}$, $\underline{W}_a^{\#}$'s; and the adjustable matrices P, Q;
  - the auxiliary matrices $X := P\underline{V}^{\#}Q^{\mathsf{T}}$, $Z := X^{-1}$ and $B := Q^{\mathsf{T}}Z$;
  - the condition indicator κ_1 and its turning on/off thresholds κ, κ'.

- **Initialization:**
  a. Randomly create an m-dimensional OOM $\mathfrak{A} = (\mathbb{R}^m, \{\tau_a\}_{a\in O}, \boldsymbol{w}_0)$.
  b. Initialize $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$ to be the probability matrices of $\mathfrak{A}$.
  c. Randomly initialize the matrices $P, Q \in \mathbb{R}^{m\times N}$ such that $\mathbf{1}_m^{\mathsf{T}} P = \mathbf{1}_N^{\mathsf{T}}$.
  d. Compute $Z = (P\underline{V}^{\#}Q^{\mathsf{T}})^{-1}$, $B = Q^{\mathsf{T}}Z$ and $\hat{\tau}_a = P\underline{W}_a^{\#}B$.
  e. Calculate the condition indicator $\kappa_1 = \|P\| \cdot \|B\|$.

- **Iteration:** for each new observation (j, c, k, i) do

  1. If $\kappa_1 > \kappa$ then (iteratively) modify the matrices P and Q according to the CbC scheme described in Subsection 3.1, until $\kappa_1 \leqslant \kappa'$ or the CbC procedure reaches its saturation point, i.e., κ_1 does not decrease significantly.

  2. Update the auxiliary matrices Z, B and the operators $\hat{\tau}_a$ to $Z^+, B^+, \hat{\tau}_a^+$, by the formulae listed in Table 2.

  3. Recompute the condition indicator κ_1 from the updated matrix B^+.

We end this section with some remarks on the above online algorithm.

- In the first step, the matrices P and Q are modified column by column, by adding a vector $\boldsymbol{h}$ to that column. When Q is modified, the vector $\boldsymbol{h}$ is computed via eqn (3.5); otherwise it is calculated by the OS algorithm, which will be discussed in Section 4. Whenever a column of P or Q is updated, one should immediately modify other relative quantities via the corresponding formula listed in Table 2.
- As indicated by eqns (2.2) and (2.5), we have $N = \ell^d$. This means, due to the limitation of memory-space, only small d's are allowed. So only the short-term information of the underlying process can be exploited by the online algorithm. For offline learning task, it is clear that the sum of all elements of $\underline{V}^{\#}$ ($\underline{W}_a^{\#}$'s) equals to t, the effective length of the training sequence $\bar{s}$. So for large d the matrices $\underline{V}^{\#}$, $\underline{W}_a^{\#}$'s must be sparse, which can be represented and manipulated using sparse matrices in Matlab.

- Like the RLS method for linear systems identification, one can introduce a *forgetting factor* $\beta \in (0,1]$ into the update rules listed in Table 2, getting an adaptive variation of the normal online learing algorithm. More precisely, the counting matrices $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$'s are now updated by

$$(\underline{V}^{\#})^{+} = \beta \underline{V}^{\#} + \boldsymbol{e}_k \boldsymbol{e}_j^{\mathsf{T}} \quad \text{and} \quad (\underline{W}_a^{\#})^{+} = \beta \underline{W}_a^{\#} + \delta_{ac} \boldsymbol{e}_i \boldsymbol{e}_j^{\mathsf{T}} \,.$$

  Correspondingly, in Table 2 the second, third and fifth lines now become

$$\begin{array}{l} [\text{set } \boldsymbol{s}_{jk}^{\mathsf{T}} = (\boldsymbol{q}_j^{\mathsf{T}} Z)/(\beta + \eta \boldsymbol{q}_j^{\mathsf{T}} Z \boldsymbol{p}_k)] \,, \\ Z^{+} = \beta^{-1}(Z - Z\boldsymbol{p}_k \boldsymbol{s}_{jk}^{\mathsf{T}}) \,, \qquad B^{+} = \beta^{-1}(B - B\boldsymbol{p}_k \boldsymbol{s}_{jk}^{\mathsf{T}}) \,; \\ F(B^{+}) = \beta^{-2}\left[F(B) - 2\boldsymbol{p}_k^{\mathsf{T}} B^{\mathsf{T}} B \boldsymbol{s}_{jk} + \|B\boldsymbol{p}_k\|^2 \|\boldsymbol{s}_{jk}\|^2\right] \,; \end{array}$$

  respectively, whereas all other lines ramain unchanged.

4 The Orthogonal Searching Algorithm

We have introduced the overall framework of the online EC algorithm: one observes the event (j,c,k,i) in the inspection window and updates the observable operators by eqns (3.9) and (3.8); when the condition indicator κ_1 becomes larger than the threshold κ, one should modify the matrices P and Q to decrease κ_1. While the modification rule of Q has been derived in Subsection 3.1; the modification of P amounts to the optimization problem (3.6), which we will discuss in this section.

Let $J_1(\boldsymbol{h})$ denote the target function of (3.6), then $\kappa_1^* = tJ_1(\boldsymbol{h})$. By the algebraic-geometric mean inequality, we have

$$J_1(\boldsymbol{h}) \leqslant \frac{1}{2}\left\{\alpha_0 F(P + \boldsymbol{h}\boldsymbol{e}_n^{\mathsf{T}}) + \beta_0 F(Q^{\mathsf{T}} Z^*)\right\} =: J_1^{\text{ub}}(\boldsymbol{h}) \,, \tag{4.1}$$

where α_0, β_0 are positive real numbers with product $\alpha_0\beta_0 = 1$; the superscript $^{\text{ub}}$ means *upper bound*; and the matrix Z^*, similar to eqn (3.2), is calculated by

$$Z^* = (P^* \underline{V}^{\#} Q^{\mathsf{T}})^{-1} = Z - \frac{Z\boldsymbol{h}\boldsymbol{e}_n^{\mathsf{T}} \underline{V}^{\#} Q^{\mathsf{T}} Z}{1 + \boldsymbol{e}_n^{\mathsf{T}} \underline{V}^{\#} Q^{\mathsf{T}} Z \boldsymbol{h}} \,. \tag{4.2}$$

In the following we will first present the basic OS algorithm for minimizing the upper bound $J_1^{\text{ub}}(\boldsymbol{h})$; then introduce an iterative procedure, which uses the basic OS algorithm as the inner optimizer, for minimizing κ_1; finally study the performance of OS through some numerical examples. It turns out that the OS method outperforms the standard numerical algorithms (see eqns (4.16) and (4.17)) on the (special) problem (3.1).

4.1 The Basic OS Algorithm

Instead of directly minimizing $J_1(\boldsymbol{h})$, we consider its upper bound $J_1^{\rm ub}(\boldsymbol{h})$. Let $\underline{\boldsymbol{r}}^{\mathsf{T}} := \boldsymbol{e}_n^{\mathsf{T}}\underline{V}^{\#}Q^{\mathsf{T}}Z = \boldsymbol{e}_n^{\mathsf{T}}\underline{V}^{\#}B$, then it follows from formula (3.4) and eqns (4.1), (4.2) that

$$\begin{aligned} J_1^{\rm ub}(\boldsymbol{h}) &= \frac{1}{2}\beta_0 F(B) - \beta_0\frac{\boldsymbol{h}^{\mathsf{T}}B^{\mathsf{T}}B\underline{\boldsymbol{r}}}{1+\underline{\boldsymbol{r}}^{\mathsf{T}}\boldsymbol{h}} + \beta_0\frac{\|\underline{\boldsymbol{r}}\|^2\cdot\|B\boldsymbol{h}\|^2}{2(1+\underline{\boldsymbol{r}}^{\mathsf{T}}\boldsymbol{h})^2} \\ &\quad + \frac{1}{2}\alpha_0 F(P) + \alpha_0\boldsymbol{h}^{\mathsf{T}}P\boldsymbol{e}_n + \frac{1}{2}\alpha_0\|\boldsymbol{h}\|^2\,. \end{aligned} \tag{4.3}$$

Setting $\underline{G} := B^{\mathsf{T}}B$, $\underline{\boldsymbol{u}} := P\boldsymbol{e}_n$ and omitting the constant $\frac{1}{2}[\alpha_0 F(P) + \beta_0 F(B)]$ in eqn (4.3), we get the following optimization problem:

$$\begin{aligned} &\text{minimize} \quad J(\boldsymbol{h}) = \frac{\beta_0\|\underline{\boldsymbol{r}}\|^2\boldsymbol{h}^{\mathsf{T}}\underline{G}\boldsymbol{h}}{2(1+\underline{\boldsymbol{r}}^{\mathsf{T}}\boldsymbol{h})^2} - \frac{\beta_0\boldsymbol{h}^{\mathsf{T}}\underline{G}\cdot\underline{\boldsymbol{r}}}{1+\underline{\boldsymbol{r}}^{\mathsf{T}}\boldsymbol{h}} + \frac{1}{2}\alpha_0\|\boldsymbol{h}\|^2 + \alpha_0\boldsymbol{h}^{\mathsf{T}}\underline{\boldsymbol{u}} \\ &\text{subject to} \quad \mathbf{1}_m^{\mathsf{T}}\boldsymbol{h} = 0\,. \end{aligned} \tag{4.4}$$

In this problem the constraint $\mathbf{1}_m^{\mathsf{T}}\boldsymbol{h} = 0$ can be eliminated by putting $\boldsymbol{h} = E\boldsymbol{d}$, with $\boldsymbol{d} \in \mathbb{R}^{m-1}$ and E being an $m \times (m-1)$ matrix whose columns form an orthonormal basis of the *null-space* of $\mathbf{1}_m$. In other words, here the matrix E has the properties $\mathbf{1}_m^{\mathsf{T}}E = 0$ and $E^{\mathsf{T}}E = I_{m-1}$. So if we define

$$\boldsymbol{r} := E^{\mathsf{T}}\underline{\boldsymbol{r}}\,, \quad G := E^{\mathsf{T}}\underline{G}E\,, \quad \boldsymbol{v} := \beta_0 E^{\mathsf{T}}\underline{G}\cdot\underline{\boldsymbol{r}}\,, \quad \boldsymbol{u} := \alpha_0 E^{\mathsf{T}}\underline{\boldsymbol{u}}\,, \tag{4.5}$$

the target function $J(\boldsymbol{h})$ of (4.4) can be rewritten as

$$J(\boldsymbol{h}) = J(\boldsymbol{d}) = \frac{\beta\boldsymbol{d}^{\mathsf{T}}G\boldsymbol{d}}{2(1+\boldsymbol{r}^{\mathsf{T}}\boldsymbol{d})^2} - \frac{\boldsymbol{d}^{\mathsf{T}}\boldsymbol{v}}{1+\boldsymbol{r}^{\mathsf{T}}\boldsymbol{d}} + \frac{1}{2}\alpha\|\boldsymbol{d}\|^2 + \boldsymbol{d}^{\mathsf{T}}\boldsymbol{u}\,, \tag{4.6}$$

with $\alpha := \alpha_0$ and $\beta := \beta_0\|\underline{\boldsymbol{r}}\|^2$.

The constrained problem (4.4) is now converted to the unconstrained one: $\min_{\boldsymbol{d}\in\mathbb{R}^{m-1}} J(\boldsymbol{d})$. An interesting observation is that, when $\boldsymbol{d}$ is orthogonal to $\boldsymbol{r}$, i.e., $\boldsymbol{r}^{\mathsf{T}}\boldsymbol{d} = 0$, $J(\boldsymbol{d})$ is a quadratic form of $\boldsymbol{d}$. This motivates us to search for the minimum of (4.6) iteratively along the direction $\boldsymbol{r}$ and a direction orthogonal to $\boldsymbol{r}$. On each direction a *global* minimal point can be obtained, as will be discussed in the sequel.

Let $\eta = \|\boldsymbol{r}\|$ be the norm of $\boldsymbol{r}$ and $\boldsymbol{r}_0 = \eta^{-1}\boldsymbol{r}$ the unitary vector on the direction $\boldsymbol{r}$. [5] Let $y = 1 + \boldsymbol{d}^{\mathsf{T}}\boldsymbol{r}$, then $\boldsymbol{d}^{\mathsf{T}}\boldsymbol{r}_0 = \eta^{-1}(\boldsymbol{d}^{\mathsf{T}}\boldsymbol{r}) = \eta^{-1}(y-1)$. Thus the quantity $\eta^{-1}(y-1)$ represents the length of the projection of $\boldsymbol{d}$ on $\boldsymbol{r}_0$ and $\boldsymbol{x} := \boldsymbol{d} - \eta^{-1}(y-1)\boldsymbol{r}_0$ is the residual vector after the projection. It follows that $\boldsymbol{r}_0^{\mathsf{T}}\boldsymbol{x} = 0$ and the target function $J(\boldsymbol{d})$ can be rewritten as

$$\begin{aligned} J(\boldsymbol{d}) &= \tfrac{1}{2}\beta y^{-2}\{\boldsymbol{x}+y_0\boldsymbol{r}_0\}^{\mathsf{T}}G\{\boldsymbol{x}+y_0\boldsymbol{r}_0\} - y^{-1}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{v} - y^{-1}y_0\boldsymbol{r}_0^{\mathsf{T}}\boldsymbol{v} \\ &\quad + \tfrac{1}{2}\alpha(\|\boldsymbol{x}\|^2 + y_0^2) + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{u} + y_0\boldsymbol{r}_0^{\mathsf{T}}\boldsymbol{u} \quad [y_0 := \eta^{-1}(y-1)] \\ &=: c_1y^2 + c_2y + c_3(\boldsymbol{x}) + c_4(\boldsymbol{x})y^{-1} + c_5(\boldsymbol{x})y^{-2} \qquad (4.7) \\ &=: \tfrac{1}{2}\boldsymbol{x}^{\mathsf{T}}H(y)\boldsymbol{x} + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{g}(y) + c(y)\,, \qquad (4.8) \end{aligned}$$

[5] If $\eta = 0$, then $\boldsymbol{r}^{\mathsf{T}}\boldsymbol{h} = 0$ and $J(\boldsymbol{d})$ becomes a quadratic form. So the problem $\min J(\boldsymbol{d})$ has a unique analytical solution.

where, just by brute-force calculation,

$$\begin{array}{lcl} c_1 & = & \frac{1}{2}\alpha\eta^{-2}\,, \\ c_2 & = & \eta^{-1}\boldsymbol{r}_0^{\mathsf{T}}\boldsymbol{u} - \alpha\eta^{-2}\,, \\ c_3(\boldsymbol{x}) & = & \frac{1}{2}\alpha\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x} + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{u} + \frac{1}{2}\eta^{-2}(\alpha + \beta\boldsymbol{r}_0^{\mathsf{T}}G\boldsymbol{r}_0) - \eta^{-1}\boldsymbol{r}_0^{\mathsf{T}}(\boldsymbol{u}+\boldsymbol{v})\,, \\ c_4(\boldsymbol{x}) & = & \boldsymbol{x}^{\mathsf{T}}(\beta\eta^{-1}G\boldsymbol{r}_0 - \boldsymbol{v}) + \eta^{-1}\boldsymbol{r}_0^{\mathsf{T}}(\boldsymbol{v} - \beta\eta^{-1}G\boldsymbol{r}_0)\,, \\ c_5(\boldsymbol{x}) & = & \frac{1}{2}\beta\boldsymbol{x}^{\mathsf{T}}G\boldsymbol{x} - \beta\eta^{-1}\boldsymbol{x}^{\mathsf{T}}G\boldsymbol{r}_0 + \frac{1}{2}\beta\eta^{-2}\boldsymbol{r}_0^{\mathsf{T}}G\boldsymbol{r}_0\,; \end{array} \tag{4.9}$$

and

$$\begin{array}{lcl} H(y) & = & \alpha I_m + y^{-2}\beta G\,, \\ \boldsymbol{g}(y) & = & \boldsymbol{u} + y^{-1}(\beta\eta^{-1}G\boldsymbol{r}_0 - \boldsymbol{v}) - y^{-2}\beta\eta^{-1}G\boldsymbol{r}_0\,, \end{array} \tag{4.10}$$

and $c(y)$ is a scalar irrelevant to the algorithm.

Now we are ready to formulate the OS algorithm for minimizing (4.6). As mentioned before, the OS algorithm searches for the optimum of $J(\boldsymbol{d})$ along two orthogonal directions. More precisely, we first fix y to some initial value $y^{(0)}$ and allow $\boldsymbol{x}$ to vary. This leads to the quadratic programming problem:

$$\begin{array}{ll} \text{minimize} & J(\boldsymbol{d}) = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}H(y)\boldsymbol{x} + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{g}(y) + c(y)\big|_{y=y^{(0)}} \\ \text{subject to} & \boldsymbol{r}_0^{\mathsf{T}}\boldsymbol{x} = 0\,. \end{array} \tag{4.11}$$

It is clear that $H(y)$ is positive definite, so (4.11) has the unique global optimum

$$\boldsymbol{x}^{(1)} = H^{-1}(y)\{\lambda\boldsymbol{r}_0 - \boldsymbol{g}(y)\}\big|_{y=y^{(0)}}\,, \tag{4.12}$$

where $\lambda \in \mathbb{R}$ is the Lagrange multiplier determined by the constraint $\boldsymbol{r}_0^{\mathsf{T}}\boldsymbol{x} = 0$. We then let $\boldsymbol{x} = \boldsymbol{x}^{(1)}$ be fixed and minimize the single-variable function

$$J(\boldsymbol{d}) = c_1 y^2 + c_2 y + c_3(\boldsymbol{x}) + c_4(\boldsymbol{x})y^{-1} + c_5(\boldsymbol{x})y^{-2}\,. \tag{4.13}$$

Taking the derivative of $J(\boldsymbol{d})$ with respect to y and letting it equal to 0, we get the quartic equation

$$2c_1y^4 + c_2y^3 - c_4(\boldsymbol{x})y - 2c_5(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{x}^{(1)}} = 0\,, \tag{4.14}$$

whose roots have an algebraic expression. [6] So one can get the global minimum $y = y^{(1)}$ of (4.13) by comparing the values of $J(\boldsymbol{d})$ on all real roots of (4.14). Next, we fix $y = y^{(1)}$ and evaluate the optimum $\boldsymbol{x}^{(2)}$ of (4.11), and so forth.

Iteratively applying the above method we get a sequence: $y^{(0)} \to \boldsymbol{x}^{(1)} \to y^{(1)} \to \boldsymbol{x}^{(2)} \to y^{(2)} \to \cdots$, for which it holds that

$$J(y^{(0)}, \boldsymbol{x}^{(1)}) \geqslant J(y^{(1)}, \boldsymbol{x}^{(1)}) \geqslant J(y^{(1)}, \boldsymbol{x}^{(2)}) \geqslant J(y^{(2)}, \boldsymbol{x}^{(2)}) \geqslant \cdots\,.$$

If we define $\boldsymbol{d}^{(k)} = \boldsymbol{x}^{(k)} + \eta^{-1}[y^{(k)} - 1]\boldsymbol{r}_0$, then $J(\boldsymbol{d}^{(1)}) \geqslant J(\boldsymbol{d}^{(2)}) \geqslant \cdots$; moreover, the function $J(\boldsymbol{d})$ is clearly bounded from below, so *the OS algorithm converges to some local minimal of* $J(\boldsymbol{d})$.

The following are some more observations concerning the OS algorithm.

[6] See, e.g., `http://mathworld.wolfram.com/QuarticEquation.html`.

1. In each OS iteration one gets the global minima of (4.11) and (4.13), so OS is more efficient than general numerical methods such as GD on the special problem (4.4) (see Subsection 4.3).
2. We find in most cases the OS algorithm is more efficient and more unlikely being trapped by local optima if we let $y^{(0)} \to 0$ in (4.11). For this case the target function $J(\boldsymbol{d})$ is dominated by $c_5(\boldsymbol{x})$, from which we get

$$\boldsymbol{x}^{(1)} = \eta^{-1}\boldsymbol{r}_0 + \lambda G^{-1}\boldsymbol{r}_0 = \eta^{-1}\boldsymbol{r}_0 - (\eta \boldsymbol{r}_0^{\mathsf{T}} G^{-1}\boldsymbol{r}_0)^{-1} G^{-1}\boldsymbol{r}_0\,. \tag{4.15}$$

   In our numerical experiments, this $\boldsymbol{x}^{(1)}$ will be used as the starting point of the OS algorithm, unless the contrary is explicitly claimed.
3. Each OS-iteration involves the computation of (4.12), (4.14): while the latter has constant complexity, the former requires about $\frac{1}{3}m^3$ flops due to the Cholesky factorization of the symmetric, positive definite matrix $H(y)$. So the complexity of the OS algorithm is about $\frac{1}{3}Km^3$, where K is the number of iterations. When K is large, the OS procedure can be simplified by diagonalizing the matrix G via a rotation transform, as follows.
4. Since G is symmetric and positive definite, it has the *Schur decomposition* $G = RDR^{\mathsf{T}}$, where D is diagonal and R orthonomal. Applying the rotation transform $[\boldsymbol{d}, \boldsymbol{r}, \boldsymbol{u}, \boldsymbol{v}] \leftarrow R^{\mathsf{T}}[\boldsymbol{d}, \boldsymbol{r}, \boldsymbol{u}, \boldsymbol{v}]$, one can replace G by D in eqn (4.6), with all other symbols remaining unchanged. The matrix $H(y)$ in (4.10) now becomes a diagonal matrix $I_m + y^{-2}\beta D$ and eqn (4.12) can be efficiently evaluated. Of course, one should rotate the solution $\boldsymbol{d}$ back to $\boldsymbol{d} \leftarrow R\boldsymbol{d}$ after the OS procedure. The complexity of this rotated OS is about $9m^3$ flops because of the Schur decomposition of G (see pp.420–421 of [4]). So theoretically the rotated version of OS is faster than the normal OS only when $K > 27$.

4.2 Minimizing κ_1 by the OS Algorithm

The above basic OS algorithm is developed for minimizing the function $J(\boldsymbol{h})$, which represents the upper bound (4.1) of κ_1, the original target function of the online algorithm (cf. eqns (4.1), (4.3) and (4.4)). One can also (iteratively) use the same OS algorithm to minimize κ_1, by selecting appropriate parameters α_0 and β_0 in eqn (4.1) such that the equality holds for each iteration. This gives us the offline version of the EC algorithm.

It is well known that the equality in eqn (4.1) holds when the two terms of $J_1^{\text{ub}}(\boldsymbol{h})$ are equal. So if we set $\alpha_0 = \|Q^{\mathsf{T}}Z\|/\|P\|$ and $\beta_0 = \|P\|/\|Q^{\mathsf{T}}Z\|$, then $J_1(\mathbf{0}) = \|P\| \cdot \|Q^{\mathsf{T}}Z\| = J_1^{\text{ub}}(\mathbf{0})$. With this setting of (α_0, β_0), we apply the OS algorithm to problem (4.4) and get a local minimum $\boldsymbol{h}$. [7] It follows that $J_1^{\text{ub}}(\boldsymbol{h}) \leqslant J_1^{\text{ub}}(\mathbf{0}) = J_1(\mathbf{0})$ and thus $J_1(\boldsymbol{h}) \leqslant J_1^{\text{ub}}(\boldsymbol{h}) \leqslant J_1(\mathbf{0})$. We then modify the matrix P (or Q) by adding $\boldsymbol{h}$ to its corresponding column to get a new P^*

[7] For the modification of columns of Q, eqn (3.5) would be utilized.

(or Q^*). It is clear that $\kappa_1 \geqslant \kappa_1^*$, where κ_1^* denotes the value of κ_1 on (P^*, Q) or (P, Q^*). Repeating this procedure: compute the parameters α_0, β_0 and update the matrix P or Q by the OS algorithm or eqn (3.5), we get a monotonically decreasing sequence of κ_1: $\kappa_1 \geqslant \kappa_1^* \geqslant \kappa_1^{**} \geqslant \cdots$.

It is now quite clear how the basic OS algorithm can be used to solve the original problem (3.1), as the following procedure shows.

a. compute the parameters $\alpha_0 = \|Q^\mathsf{T} Z\|/\|P\|$ and $\beta_0 = \|P\|/\|Q^\mathsf{T} Z\|$;
b. get the vector $\boldsymbol{h}$ via eqn (3.5) or by solving the problem (4.4) with OS;
c. update the matrix P or Q by adding $\boldsymbol{h}$ to the corresponding column;
d. move to the next column and repeat steps **a**-**c**, until $\kappa_1 < \kappa'$ or κ_1 shows no significant decrease.

Note that the above iterative method for minimizing κ_1 can be used in the offline EC algorithm: one first collects the counting matrices $\underline{V}^\#$ and $\underline{W}_a^\#$'s, then he calculates a pair of appropriate matrices P, Q via the above steps **a.-d.**, and finally gets the estimated $\hat{\tau}_a$'s by eqn (2.7).

4.3 Numerical Examples Concerning the OS Algorithm

In this miscellaneous subsection we discuss the performance and the offline use of the OS algorithm, through several numerical examples. For comparison here we consider two other numerical methods, which are easier to be thought out than OS, for minimizing (4.6) and (3.6) respectively.

- *The GD method for the problem* $\min J(\boldsymbol{d})$ *with* $J(\boldsymbol{d})$ *defined by* (4.6):

$$\begin{aligned} \boldsymbol{d}^{(t)} &= \boldsymbol{d}^{(t-1)} - \eta \frac{\partial J}{\partial \boldsymbol{d}}\left(\boldsymbol{d}^{(t-1)}\right), \quad \text{with} \\ \frac{\partial J}{\partial \boldsymbol{d}} &= \alpha \boldsymbol{d} + \boldsymbol{u} - \frac{\boldsymbol{v}}{1+\boldsymbol{r}^\mathsf{T}\boldsymbol{d}} + \frac{\beta G\boldsymbol{d} + (\boldsymbol{d}^\mathsf{T}\boldsymbol{v})\boldsymbol{r}}{(1+\boldsymbol{r}^\mathsf{T}\boldsymbol{d})^2} - \frac{(\beta \boldsymbol{d}^\mathsf{T} G\boldsymbol{d})\boldsymbol{r}}{(1+\boldsymbol{r}^\mathsf{T}\boldsymbol{d})^3}, \end{aligned} \tag{4.16}$$

  where the *learning rate* $\eta > 0$ is determined by the following scheme.

  At the t-th iteration, the value of $J(\boldsymbol{d}^{(t)})$ is determined by η and so can be written as $J_t(\eta)$. We put $\eta_0 = 0$, $\eta_1 = 0.04$, $\eta_2 = 0.2$ and calculate the interpolating polynomial $\hat{J}_t(\eta) = a\eta^2 + b\eta + c$ of the points $(\eta_i, J_t(\eta_i))$ $(i = 0, 1, 2)$. [8] If $a \leqslant 0$, then we take $\eta = \eta_2$; otherwise assume η^* is the minimal of $\hat{J}_t(\eta)$ and take the one of $\{\eta_1, \eta_2, \eta^*\}$ that makes $J_t(\eta)$ minimal as η, i.e., $\eta = \arg\min_{\eta' \in \{\eta_1, \eta_2, \eta^*\}} J_t(\eta')$.

- *A direct iterative method for* (3.11) *— the "pseudo-inverse" method*:

  In Subsection 3.3, we have shown that if P is fixed such that $\mathbf{1}_m^\mathsf{T} P = \mathbf{1}_N^\mathsf{T}$, then $Q^\mathsf{T} = (P\underline{V}_0^\#)^\dagger$ is the minimum of (3.11). Similarly, if Q is fixed such

[8] If $J_t(\eta_1) > J_t(\eta_0)$, one should reduce η_i by some factor, e.g., $\eta_i \leftarrow 0.8\eta_i$ $(i = 1, 2)$; and do the interpolation again.

that $\mathbf{1}_N^{\mathsf{T}}\underline{V}_0^{\#}Q^{\mathsf{T}} = \mathbf{1}_m^{\mathsf{T}}$, which is derived from the two constraints in (3.11), then $P = (I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^{\mathsf{T}})(\underline{V}_0^{\#}Q^{\mathsf{T}})^{\dagger} + \frac{1}{m}\mathbf{1}_m\mathbf{1}_N^{\mathsf{T}}$ is the minimal of (3.11). This verifies the following updating formulas of P and Q:

$$\begin{array}{rcl} Q^{(t)\mathsf{T}} & = & \left[P^{(t-1)}\underline{V}_0^{\#}\right]^{\dagger}, \\ P^{(t)} & = & \left(I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^{\mathsf{T}}\right)\left[\underline{V}_0^{\#}Q^{(t)\mathsf{T}}\right]^{\dagger} + \frac{1}{m}\mathbf{1}_m\mathbf{1}_N^{\mathsf{T}}\,. \end{array} \tag{4.17}$$

A toy example — to get more insights into OS

OS is a special method for minimizing the function $J(\boldsymbol{d})$ defined by (4.6). To get an intuitive insight into this algorithm we consider a simple example of $J(\boldsymbol{d})$ with $\alpha = \beta = 1$, $G = [4, 2; 2, 2]$ (in Matlab's notation), $\boldsymbol{r} = [1, 1]^{\mathsf{T}}$, $\boldsymbol{u} = [3.5, -2]^{\mathsf{T}}$ and $\boldsymbol{v} = [3, 2]^{\mathsf{T}}$. Writing $\boldsymbol{d} =: [x, y]^{\mathsf{T}}$, one obtains

$$J = \frac{2x^2 + 2xy + y^2}{(1 + x + y)^2} - \frac{3x + 2y}{1 + x + y} + \frac{1}{2}(x^2 + y^2) + 3.5x - 2y\,, \tag{4.18}$$

to which the OS algorithm and, for comparison, the GD method described in (4.16) will be applied.

The function $J(x, y)$ goes to infinity when $x + y \to -1$ or $\|\boldsymbol{d}\| \to \infty$, so there are local minima on both sides of the line $x + y = -1$. Figure 2 shows the surface (**a.**) and the contour map (**b.**) of J: along the line $x + y = -1$ is an infinitly high "ridge", on each side of the ridge is a relatively shallow "basin" (compare to the height of the ridge). To make the two basins visible, here the z-axis is scaled by a base-10 logarithmic transform $z = \lg(J - J_{\min} + 0.1)$.

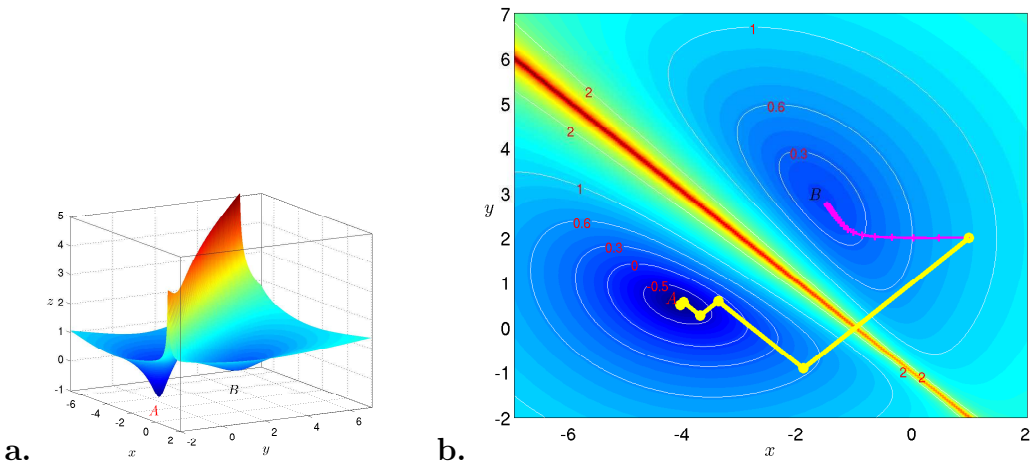


Figure 2: Searching procedure of GD/OS for the minimal of $J(\boldsymbol{d})$

Figure 2**b.** also shows the searching trace of OS (yellow line marked by ∘) and GD (magenta line marked by +), both starting from the initial point $[1, 2]^{\mathsf{T}}$. One observes that the OS algorithm reaches the global minimal point $A(-4.12, 0.53)$

in three iterations (searching on every two orthogonal directions is seen as an OS iteration), better than GD, which converges to a local minimum $B(-1.58, 2.88)$ afte about twenty iterations. [$J_A = -6.76$, $J_B = -5.55$]

Although this toy example is quite simple and of little interest in itself, it offers us several insights into the general problem (4.6) and the OS algorithm.

- The target function $J(\boldsymbol{d})$ goes to infinity when $1 + \boldsymbol{r}^\mathsf{T}\boldsymbol{d} \to 0$. This forms an infinitely high ridge along the hyper plane $\boldsymbol{r}^\mathsf{T}\boldsymbol{d} = -1$, making an uncrossable region (the red belt in Figure 2**b.** for our example) in the searching space $\mathbb{R}^m$ for all gradient-based methods. So any GD-like method has at least half chance being trapped by some local minimum.
- On the other hand, the second OS step searches for the global minimum of $J(\boldsymbol{d})$ along the direction $\boldsymbol{r}$, enabling it to pass through the high ridge freely. Moreover, when $|1 + \boldsymbol{r}^\mathsf{T}\boldsymbol{d}| \gg 0$ (the region far from the ridge), by (4.6) $J(\boldsymbol{d}) \approx \frac{1}{2}\alpha\|\boldsymbol{d}\|^2 + \boldsymbol{d}^\mathsf{T}\boldsymbol{u}$ becomes a quadratic function of $\boldsymbol{d}$, of which the global minimum is easily to obtain. This partly explains why OS usually converges to the global minimum of $J(\boldsymbol{d})$, although we cannot theoretically prove it yet.

The performance of OS on the auxiliary optimization problem

In this experiment, $10 \times 10 = 100$ functions of form (4.6) with dimension m ranging from 10 to 500 are randomly created as the target, to which the basic OS algorithm and the GD method described by (4.16) are applied. The initial value of OS is set as eqn (4.15); whereas for the GD method, we tried 10 different initial vectors $\boldsymbol{d}^{(0)}$: one is $\boldsymbol{d}^{(0)} = \boldsymbol{0}$ and the other 9 are randomly procured.

The average number of iterations K of both methods for problems with the same dimension are shown in Figure 3**a.** Here the stopping criterion is that both the decreasing value and rate of the target function between two subsequent iterations are less than 10^{-6}, i.e., $\max\{J_{K-1} - J_K,\ (J_{K-1} - J_K)/|J_K|\} < 10^{-6}$; or K reaches its maximal value 10000. We see that the number of OS iterations is much less than that of GD, especially for high dimensional problems. The CPU-time per OS iteration, however, is larger than GD. So the overall time of OS is only a little bit less that that of GD, as depicted in the same figure.

Although the OS algorithm has no significant advantage in speed comparing to GD, it obtains better result than GD on the problem (4.6). It turns out that OS *always* gets smaller target values on the above 100 examples than GD (despite the fact that GD is initialized by different $\boldsymbol{d}_0$'s). We plotted the optimal values obtained by the OS and GD (with the 8-th initial vector $\boldsymbol{d}_0$) methods in Figure 3**b**: each point for one problem, with its x-coordinate being the minimal value obtained by OS and y-coordinate the difference between GD and OS on the same problem. To make the picture clearer a logarithmic transform $y = \log(J_{\mathrm{GD}} - J_{\mathrm{OS}})$ is used for y-axis. Therefore, for instance, the point $A(-175.5, 1.28)$ means that, for some

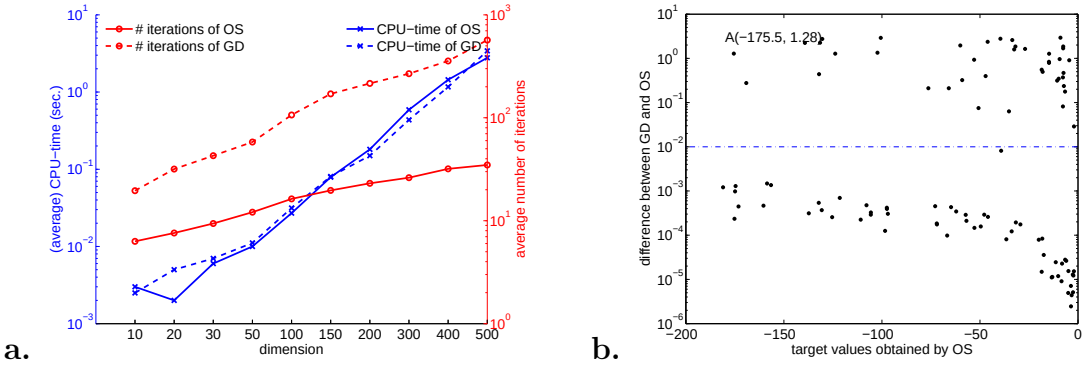


Figure 3: Comparison of the OS and GD algorithms

example the OS method gets the minimal value -175.5, while GD converges to a local optimal with the target value $J = -175.5 + 1.28 = -174.2$.

Now we set a threshold $\delta = 0.01$ for checking whether the GD method gets a local minimal: if $J_{\text{GD}} - J_{\text{OS}} > \delta$ for some problem, then we thought that GD is trapped by a local minimum on this problem. Figure 3**b.** shows that on 41 problems (the points above the deshed line) OS obtains smaller optimum than GD. The similar phenomena are observed when GD is initialized with different $\boldsymbol{d}_0$. This is because, as has been mentioned earlier, the surface of $J(d)$ has an infinitely high ridge that separates the whole space into two parts, of which only one (depends on the initial vector $\boldsymbol{d}_0$) is reachable by GD. Therefore, the GD method can only get a local minimal when $\boldsymbol{d}_0$ and the global minimum of $J(\boldsymbol{d})$ lie in different parts of the searching space.

The performance of OS on the original optimization problem

We now investigate the performance of the OS algorithm on the original optimization problem (3.1). Let $\underline{V_0^{\#}}$ denote the normalized counting matrix with elements summing up to unity, then the target function κ_1 can be written as $\kappa_1 = \|P\| \cdot \|Q^{\mathsf{T}}(P\underline{V_0^{\#}}Q^{\mathsf{T}})^{-1}\|$. To make the simulation more "real-life", here the matrix $\underline{V_0^{\#}}$ is created by the following procedure:

1. randomly create three matrices U, S and E by, e.g., the Matlab codes $U = \underline{\text{rand}}(m, N)$; $S = \underline{\text{rand}}(m, N)$; $E = \underline{\text{rand}}(N, N)$;
2. normalize the above three matrices so that U and S have row sums 1 and E has elements sum m, i.e., $U\mathbf{1}_N = S\mathbf{1}_N = \mathbf{1}_m$ and $\mathbf{1}_N^{\mathsf{T}}E\mathbf{1}_N = m$ after the normalization;
3. set $V = U^{\mathsf{T}}S + \epsilon E$, where ϵ is a small positive real number representing the amplitude of perturbations on the matrix V;
4. calculate the normalized counting matrix by $\underline{V_0^{\#}} = [m(1+\epsilon)]^{-1} \cdot V$.

Thus, the created matrix $\underline{V_0^{\#}}$ is characterized by three parameters: m, N and ϵ. In this example, the dimension m is in $\{3, 6, 10, 15, 20, 30, 50, 80\}$; the size N is determined by the ratio N/m, which takes values from $\{2, 4, 7, 10, 15\}$; and the parameter ϵ ranges from 0.1 to 0.5, with step length 0.1. All together there are $8 \times 5 \times 5 = 200$ different configurations of (m, N, ϵ). For each such configuration we create 10 matrices $\underline{V_0^{\#}}$ using the above method. [9]

So far we have constructed $200 \times 10 = 2000$ testing problems of form (3.1), to which the CbC scheme with OS as the inner minimizer of (4.4) [CbC+OS] and the pseudo-inverse method described in (4.17) [PsInv] are to be applied. First we investigate the sensitivity of the two methods to the initial values P_0 and Q_0. To lighten the computation load here we randomly selected 40 testing matrices $\underline{V_0^{\#}}$'s and for each $\underline{V_0^{\#}}$ randomly procured 25 pairs of (P_0, Q_0) with $\mathbf{1}_m^{\mathsf{T}} P_0 = \mathbf{1}_m^{\mathsf{T}}$ as the initial values, by, e.g., the Matlab codes:

```
1  P0 = 2 * rand(m,N) - 1.0;
2  Q0 = 2 * rand(m,N) - 1.0;
3  P0 = P0 - repmat(mean(P0), m, 1) + 1/m;
```

This leads to 1000 ($= 40 \times 25$) instances divided naturally into 40 groups. If a method is insensitive to initial values, then it should obtain essentially the same minimum for examples in the same group. So the *deviation-mean ratio* (DMR) of the minima in one group obtained by a given method can be seen as a measure of its sensitivity to initial values. More precisely, here we use the quantity

$$\text{DMR} = \bar{J}^{-1} \sqrt{\frac{1}{24} \textstyle\sum_{k=1}^{25} (J_k - \bar{J})^2} \quad \text{with } \bar{J} := \frac{1}{25} \textstyle\sum_{k=1}^{25} J_k \,,$$

where J_k denotes the minimal value of the k-th example obtained by the method under consideration, as the measure of the sensitivity to initial values.

Figure 4 shows the average minimal values obtained by the CbC+OS and PsInv methods (**a.**); and their corresponding DMR-values (**b.**). The examples are sorted so that the minimal values obtained by CbC+OS is increasing. For these examples the minimal values obtained by PsInv are about 60.6% (2.1dB) larger than that obtained by CbC+OS in average (note the logarithmic scaling of the y-axis in Figure 4**a.**). Moreover, CbC+OS is much less sensitive to initial values than PsInv: the average DMR-value of CbC+OS on the examples is 5.0×10^{-5}, versus 0.067 of PsInv. Practically, when used to minimize (3.1), the CbC+OS method always gets the same minimal value regardless of how the matrices P, Q are initialized.

Next, we minimize the whole family of the above constructed 2000 problems by the CbC+OS method, with P, Q initialized by $P_0 = Q_0 = [I_m, \cdots, I_m]$, the block-matrix consisting of N/m copies of I_m. According to its construction (cf. p.25), the normalized counting matrix $\underline{V_0^{\#}}$, and thereby the minimal value $J_{\min}$ of

[9] Just for one can reproduce the results, the random numbers generator rand is initialized by the k-th initial state ($k = 9, 19, \cdots, 99$): rand('state', k) when creating the matrices $\underline{V_0^{\mathsf{T}}}$.

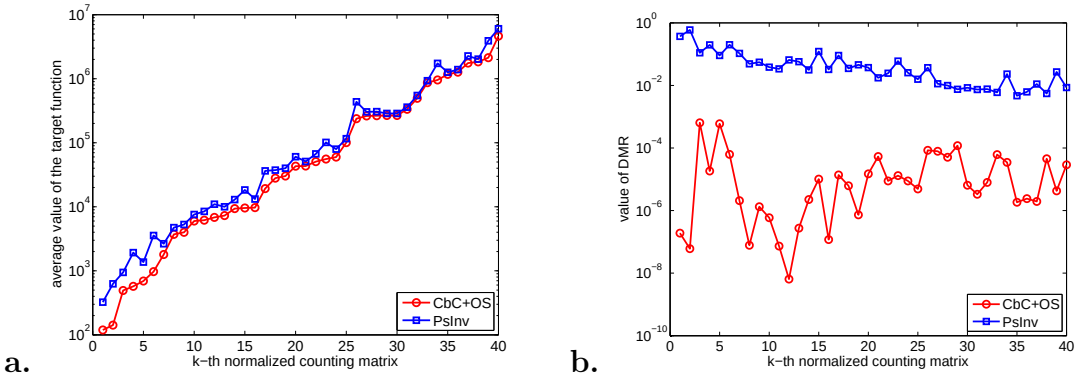


Figure 4: Comparison of the CbC+OS and pseudo-inverse methods

the corresponding problem obtained by CbC+OS, are (uniquely) determined by four parameters: the dimension m, the ratio N/m, the perturbation amplitude ϵ and the initial state k of the random numbers generator. So below we will write $J_{\min} = J_{\min}(m, N/m, \epsilon, k)$, a 4-D array in Matlab.

Figure 5 shows, on the plane of the dimension m and the ratio N/m, two exemplary contour maps of the maximal one of $J_{\min}$'s with m, N/m and ϵ being fixed, i.e., of $J^*_{\min}(m, N/m, \epsilon) = \max_k J_{\min}(m, N/m, \epsilon, k)$. Note that in Figure 5 all the quantities are logarithmically scaled. Also note that the contour curves are almost equidistant parallel straight lines. This finding motivates us to do the following multiple linear regression:

$$\log(J_{\min}) = a \cdot \log(m) + b \cdot \log(N/m) + c \cdot \log(\gamma) + E_{\mathrm{r}}\,, \tag{4.19}$$

where γ is computed from (the singular values of) $\underline{V}_0^{\#}$ as in (3.15); and E_{r} is the *regression error*. Using our 2000 testing matrices $\underline{V}_0^{\#}$, we computed the coefficients a, b, c in (4.19) as: $a \approx 2.6$, $b \approx 1.1$ and $c \approx 0.7$, with the regression error E_{r} ranging from -0.121 to 0.077. Substituting these values into (4.19) we get $J_{\min} \approx (75.7\text{–}119.4)\% \cdot \gamma^{0.7} m^{1.5} N^{1.1}$, which represents a reachable (lower) level for the condition indicator κ_1.

A comparison of this numerical result with eqn (3.15) might be interesting. For a given matrix $\underline{V}_0^{\#}$ assume $\kappa_1^{\min}$ is the minimal value of the problem (3.1). Then (3.15) shows that $\kappa_1^{\min} \leqslant \gamma m^{3/2} N$, whereas the numerical simulation here reveals that $\kappa_1^{\min} \leqslant 1.2\gamma^{0.7} m^{3/2} N^{1.1}$. Both bounds are of approximately the same order $\mathcal{O}(m^{3/2}N)$ on m and N, although in (3.15) the constraint $\mathbf{1}_m^{\mathsf{T}} P = \mathbf{1}_N$ was not considered at all. For the order of γ, the numerical simulation gets a lower value than (3.15), due to the reason that in (3.15) the matrix P is restricted to be of a special form (3.13). On our 2000 examples the parameter γ ranges from 4.35 to 35.04, see Figure 6 for a histogram of γ. As Figure 6 approximately represents the distribution of γ, according to eqn (4.19) it is safe to set the threshold of κ_1 to be $\kappa = 35.04^{0.7} m^{3/2} N \approx 12 m^{3/2} N$ for the online EC algorithm of OOMs.

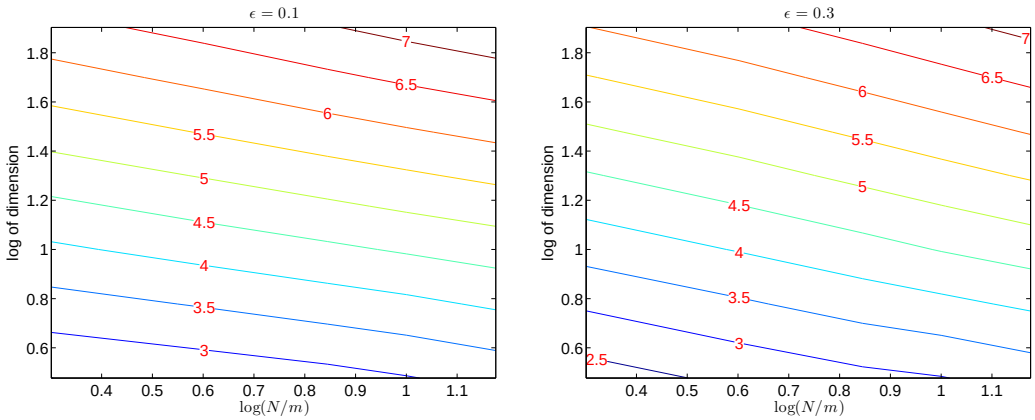


Figure 5: Contour maps of $\log(J^*_{\min})$ on the $\log(N/m)$-$\log(m)$ plane

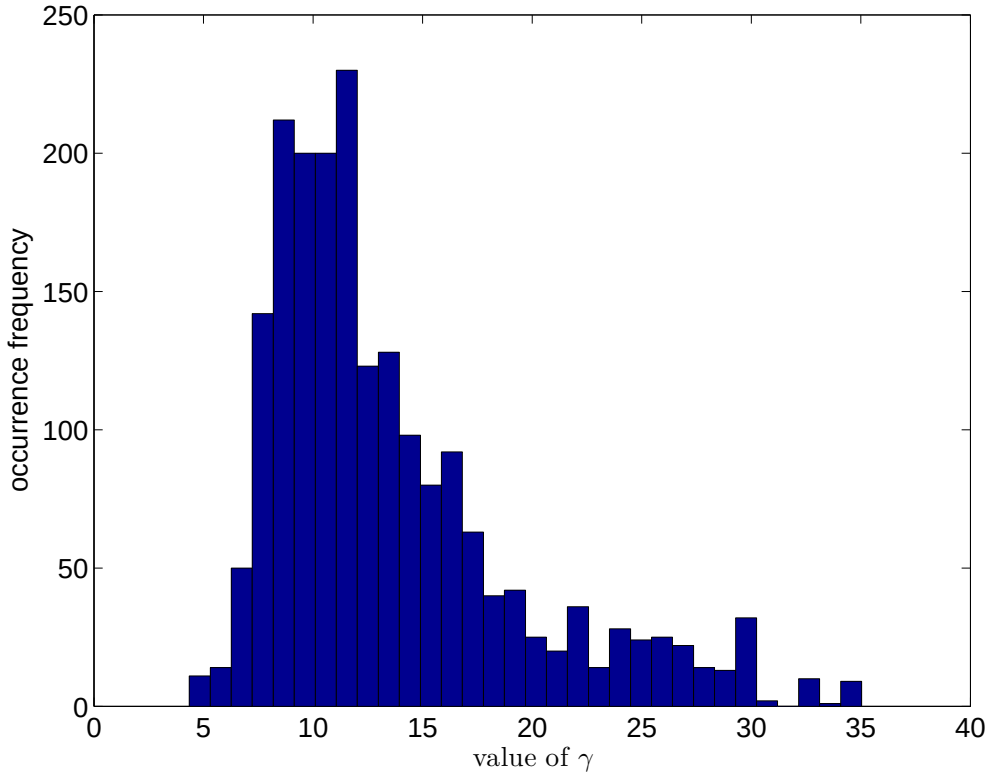


Figure 6: The distribution of the value of γ's.

In conclusion, from the above numerical experiments we see that CbC+OS is an efficient and stable numerical method for the optimization problem (3.1), and can be used in the EC algorithm for estimating OOMs from data online (cf. Subsection 3.4) or offline (using it to minimize the condition indicator of learning equation).

5 Numerical Experiments

In this section we check the performance of the EC algorithm via some symbolic sequence modelling experiments. More precisely, the EC algorithm will be utilized for estimating OOMs from the data sets generated by the "probability clock" [6]; by a symbolic logistic system; and by a system switching between two different HMMs (hence the created sequences are nonstationary and the online EC algorithm is required). To measure the quality of learnt models, we define a quantity called *description accuracy* (DA) by

$$\mathrm{DA}(\mathfrak{A}, S) := f[1 + \mathrm{NLL}(\mathfrak{A}, S)] := f\left[1 + (S^{\#})^{-1} \log_{\ell} \Pr(S|\mathfrak{A})\right] ,$$

where $\mathfrak{A}$ is the model whose quality we want to measure; S is a data set generated by the underlying process and $S^{\#}$ denotes the total number of symbols in S, i.e., the sum of length of all sequences in S; ℓ is the alphabet size; and f is a nonlinear function which maps the infinite interval $(-\infty, 1]$ to the finite one $(-1, 1]$ via

$$f(x) = \begin{cases} x & \text{if} \quad x \geqslant 0\,, \\ (1 - e^{-0.25x})/(1 + e^{-0.25x}) & \text{if} \quad x < 0\,. \end{cases}$$

Intuitively, $\mathrm{NLL}(\mathfrak{A}, S)$ is the *normalized log-likelihood* of $\mathfrak{A}$ on the data set S per symbol and assumes values from $-\infty$ to 0. Therefore the range of $\mathrm{DA}(\mathfrak{A}, S)$ is the interval $(-1, 1]$: $\mathrm{DA} = 1$ means the model describes the data S perfectly well (it can predict the data with probability one); $\mathrm{DA} = 0$ means the model is irrelevant to the data for it provides no more information about the process than just randomly "guessing" such a data set; $\mathrm{DA} < 0$ means the model is even worse than a randomly created one, which, as one can imagine, rarely happens in practice. In one word, the larger the DA-values are, the better the learnt model is. In the following numerical experiments the quality of all learnt models will be measured by its DA-values on the training sequence and testing sequences from the same source as that of the training data.

Before the simulation we need to clarify some "universal" settings and remarks.

- The EC algorithm does not solve the *negative probability problem* (the main shortcoming of the theory of OOMs): the condition (3) from Proposition 1 might be violated by the estimated model, which may assign "negative probabilities" $\hat{P}(\bar{a}) = \mathbf{1}_m^{\mathsf{T}} \hat{\tau}_{\bar{a}} \hat{\boldsymbol{w}}_0 < 0$ to some (rare) sequences $\bar{a} \in O^*$. However, heuristic methods to master this problem exist (for the detailed procedure see Appendix J of [7]), and are used in our experiments.

- We also train HMMs of the same size as OOMs from each data set by the EM algorithm [11], and compare their performance to OOMs estimated by EC. There are two kinds of HMMs: the (usually used) *state-emission* HMMs (SE-HMMs), in which the outcomes are "emitted" by the hidden states; and the *transition-emission* HMMs (TE-HMMs), in which the symbol emitted at time t depends on the hidden states at times t and $t+1$. Note that, a TE-HMM can be seen as an OOM whose parameters are all nonnegative; whereas a SE-HMM has less parameters than an OOM of the same dimension.

- For the estimation of HMMs, the EM algorithm is terminated when the increasing of the DA-value on the training sequence between two subsequent iterations is less than $\delta = 10^{-6}$ [10] . For the learning of OOMs, we choose the length d of basic strings (cf. Subsection 3.4) such that $\ell^d \sim (5\text{–}100)m$, where ℓ is the alphabet size and m is the model dimension.

5.1 Modelling the Probability Clock

The probability clock is a 3-dimensional OOM $(\mathbb{R}^3, \{\tau_i\}_{i\in\{1,2\}}, \boldsymbol{w}_0)$ with

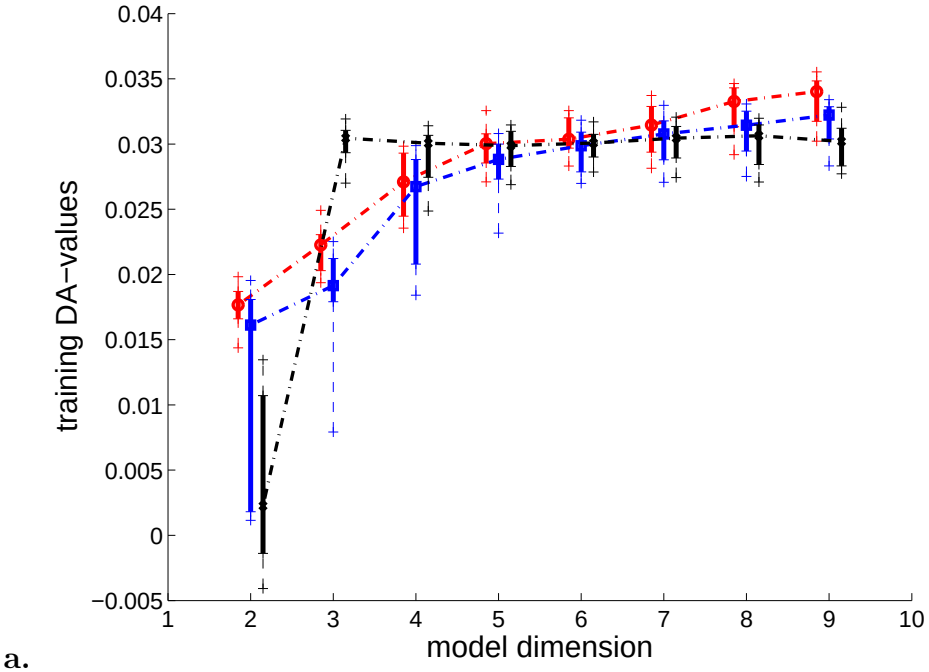
$$\tau_1 = 0.5 \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & s \\ 0 & -s & c \end{bmatrix}, \quad \tau_2 = 0.5 \begin{bmatrix} .75 & .75(2-c+s) & .75(2-c-s) \\ 0 & 0 & 0 \\ .25 & .25(2-c+s) & .25(2-c-s) \end{bmatrix}$$

and $\boldsymbol{w}_0 = \begin{bmatrix} .75 & 0 & .25 \end{bmatrix}^{\mathsf{T}}$, where $c = \cos(1)$ and $s = \sin(1)$. It should be noticed that the process described by the probability clock is not a HMM process, i.e., it cannot be modelled by any finite-dimensional HMMs (see Section 6 of [6]).

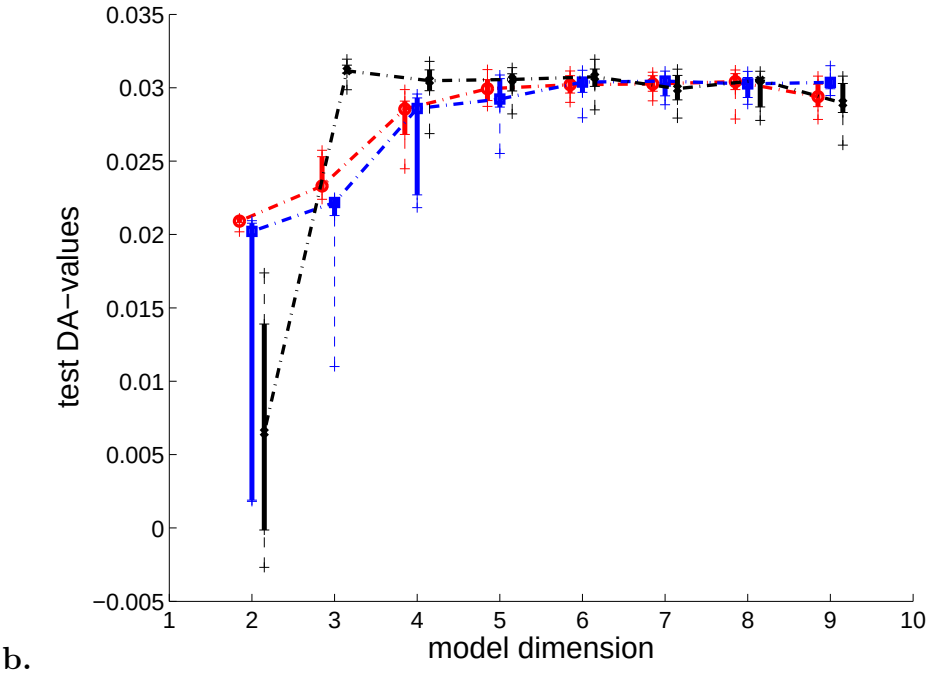
Running the probability clock we create 60 sequences: the first 20 are of length 5000 used as training sequences, the other 40 are all of length 300 reserved as the testing data. From each training sequence several OOMs and HMMs of different dimensions $m \in \{2, 3, \cdots, 9\}$ are estimated, using the offline EC algorithm (with κ_1 minimized by the CbC+OS scheme) and the EM algorithm, respectively. When learning OOMs, the length of basic strings is chosen to be $d = 4$ for $m \leqslant 3$ and $d = 5$ for $m \geqslant 4$. This is because for higher dimensions m we need larger counting matrices $\underline{V}^{\#}$ and $\underline{W}_a^{\#}$.

Thus, for each dimension m and each model type (SE-HMM, TE-HMM and OOM), we obtain from the 20 training sequences 20 models; and compute their DA-values on the corresponding training sequence and on the 40 test sequences. These training and test DA-values are boxplotted in Figure 7, in which we marked the 10%-percentile, lower quantile, median, upper quantile and 90%-percentile positions of the DA-distribution of the learnt 20 models (of the same dimension and the same model-type, but learned from different training sequences).

[10] We have also tried $\delta = 10^{-5}$, which just stops the EM algorithm too early (and only "immature" HMMs are learned).



a.



b.

Figure 7: Training and test DA-distribution of OOMs (black line marked by $\times$), SE-HMMs (blue line marked by $\square$) and TE-HMMs (red line marked by $\circ$) on the data set generated by the probability clock.

From Figure 7 we see that for $m = 3$ (the "true" dimension of the underlying process) the DA-values of OOMs are significantly higher than that of SE-HMMs and TE-HMMs. When the model dimension m increases, the performance of HMMs trained by EM becomes better, but still OOMs trained with EC outperform HMMs a little bit. For even larger dimensions $m = 7, 8, 9$, the train DA-values of HMMs increase further but the test DA-values do not, indicating the EM algorithm begins to overfit the training data; whereas EC still works well: the overfitting phenomenon is not observed for OOMs.

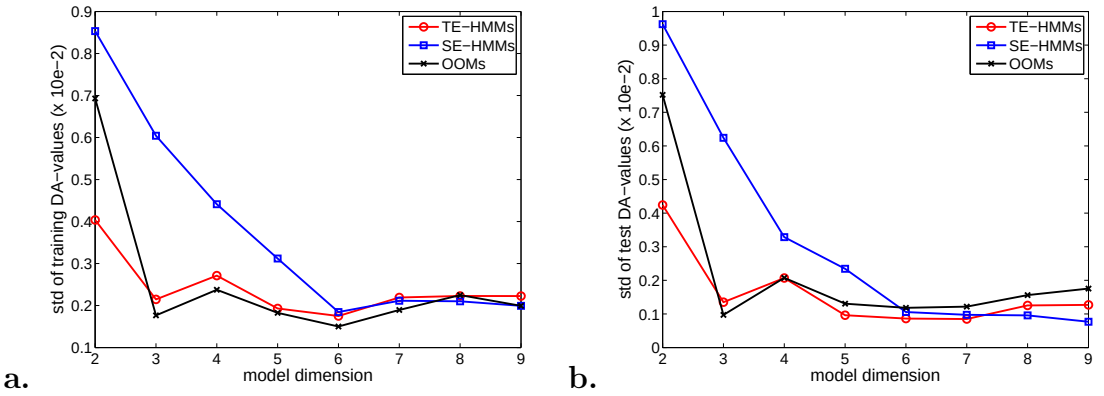


Figure 8: Standard deviation of DA-values of TE-HMMs, SE-HMMs and OOMs on the data set generated by probability clock (the y-axis is scaled by 10^{-2}).

In Figure 8 the standard deviation (std) of DA-values for each model dimension and each model type on the training (**a.**) and test (**b.**) sequences are plotted. It is shown that HMMs and OOMs have approximately the same deviation of DA-values for $m \geqslant 6$; whereas for $m \leqslant 5$, SE-HMMs have higher DA-deviation than that of TE-HMMs and OOMs. One possible reason for this is that SE-HMMs have fewer parameters than TE-HMMs and OOMs of the same dimension and so for low model dimensions SE-HMMs are more sensitive to the initial model (EM is an iterative method and one should first "guess" an initial model) than TE-HMMs.

5.2 Modelling the Quantized Logistic System

In the second experiment we consider the logistic system $y(t+1) = \mu y(t)[1-y(t)]$ with $\mu = 4$. The attractor of this dynamical system, the closed interval $[0, 1]$, is devided into 10 equidistant sub-intervals, yielding a quantized logistic system with alphabet size $\ell = 10$. Running this system with different initial values $y(1)$ we get 50 sequences: 30 are of length 5000 as the training data; and the other 20 are of length 500 used as the test data.

As before, several OOMs, SE-HMMs and TE-HMMs with dimensions $m \in \{3, 5, 8, 12, 16, 20, 25, 30, 40, 50, 60\}$ are learned from each training sequence; and the DA-values of these estimated models are computed on the training and test

data. When learning OOMs using the EC algorithm, the length of basic strings is set to be $d = 4$. Figure 9 shows the distribution of DA-values on the training (**a.**) and test (**b.**) data set for the three model classes. From the figure we see that

1. for low dimensions TE-HMMs obtain higher DA-values than OOMs and SE-HMMs, on both training data and test data;
2. when model dimension m increases, the performance of OOMs becomes better than that of SE-HMMs, and is comparable to TE-HMMs;
3. when m becomes even larger ($\geqslant 40$), TE-HMMs begin to overfit the training data, whereas OOMs still work well.

These observations can be explained as follows: the "real dimension" of the logistic system may be high; and one just can not describe such a high-dimensional system using a low-dimensional model. So for low model dimensions m, the EC algorithm fails to grasp the underlying process, which is reflected by the low DA-values; and it is not surprising that TE-HMMs get higher DA-values than OOMs for this situation, since the target of EM is to maximize the model's log-likelihoods, while EC aims to minimize the estimation error. With the increase of the dimension m, the model class becomes richer and richer, so there are steady and big jumps in the DA-values of OOMs when $m \leqslant 12$, making OOMs comparable to TE-HMMs for $m \geqslant 12$.

As the EC algorithm is designed to minimize the estimation error, in principle it should be more insensitive to concrete training data than the EM algorithm. This is reflected by the curves of the standard deviation of DA-values, see Figure 10**a.** and **b.**. It turns out that when $m \geqslant 12$, OOMs trained by EC have lower DA-deviation than that of HMMs trained by EM.

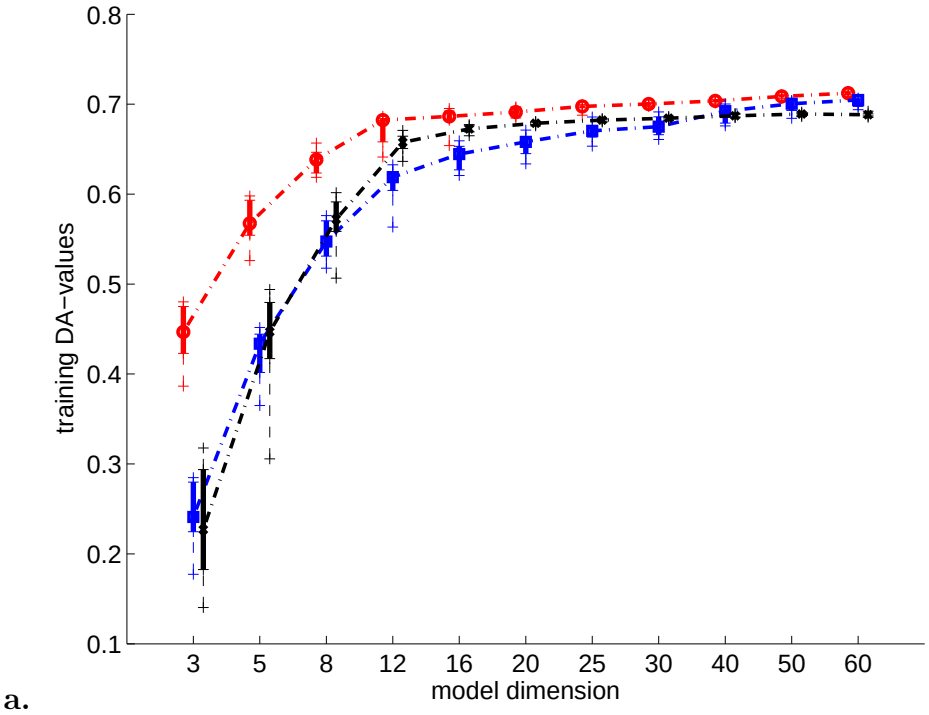
5.3 Using EC with CbC+OS for Online Learning

In this experiment, we use a 4-state-3-output time-varying "cyclic" HMM to generate training and test sequences. Figure 11 shows its structure, in which the parameters a and c are dependent on the time t:

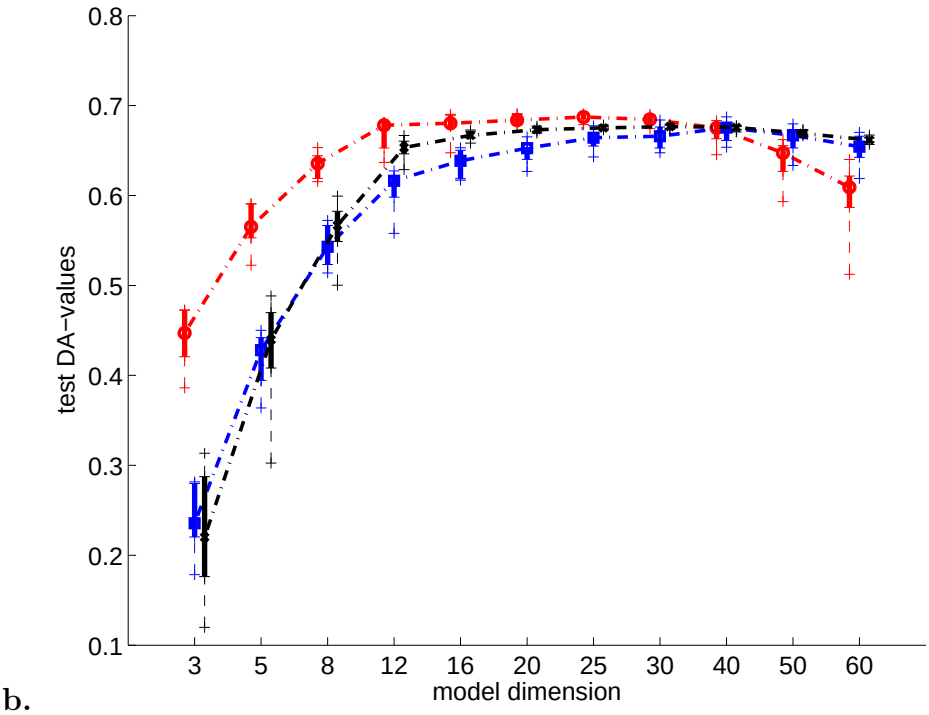
$$(a(t), c(t)) = \begin{cases} (0.97, 0.95) & \text{if} \quad t \leqslant 1000\,, \\ (0.03, 0.05) & \text{if} \quad t > 1000\,. \end{cases}$$

So the process produced by this HMM is nonstationary and all its sample paths consist of two segments with different statistical properties.

The goal here is to check whether the online EC algorithm can adaptively modify its estimation when the underlying process varies with time. To this end we use 51 sequences of length 2000 generated by the above HMM: one for training and the other 50 for testing. From the training sequence a 4-dimensional OOM is learnt using the online EC algorithm with forgetting factor $\beta = 0.99$. During the learning procedure, the "predictive" DA-values of the current estimated OOM are computed over the next 20 symbols for each time step (we update the operators



a.



b.

Figure 9: DA-distributions of OOMs (black line marked by $\times$), SE-HMMs (blue line marked by $\square$) and TE-HMMs (red line marked by $\circ$) on the data set generated by the quantized logistic system.

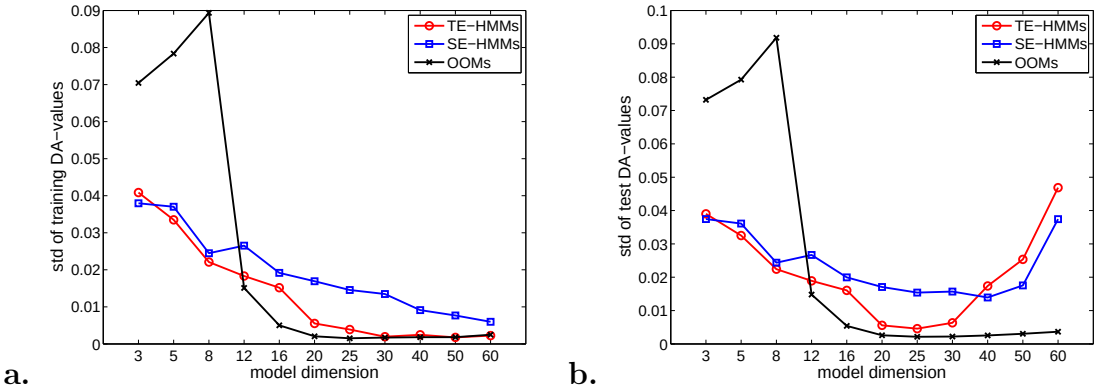


Figure 10: Standard deviation of DA-values of TE-HMMs, SE-HMMs and OOMs on the data set generated by the quantized logistic system.

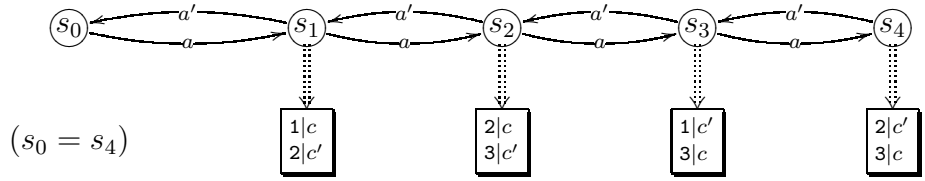


Figure 11: The time-varying HMM: the transition probabilities a, a' and the emission probabilities c, c' both have sums 1 and are dependent on the time t.

$\hat{\tau}_a$ and at the same time drive the estimated model by the training and test data). Figure 12 shows the curve of the average predictive DA-values on the test data (blue solid line); and the variation in the condition indicator κ_1 of the learning equation (red dashed line) for each time step.

In Figure 12 the curve of the test predictive DA-value shows a big drop at time $t = 1000$, revealing the underlying process is somehow changed at this point. Later it remains at a row level for long time, meanwhile the condition indicator κ_1 raises steadily, indicating the learning procedure becomes more and more instable. When κ_1 exceeds some threshold κ, the OS algorithm is triggered (indicated by the vertical black lines in the figure) and the matrices P, Q are adjusted to decrease the value of κ_1. At the same time, there is a jump in the DA-value of the estimated OOM. Shortly after that, the OS algorithm is triggered and the DA-value increases again. All these observations illustrate the importance of an appropriate selection of the adjustable matrices P and Q in learning algrithms of OOMs.

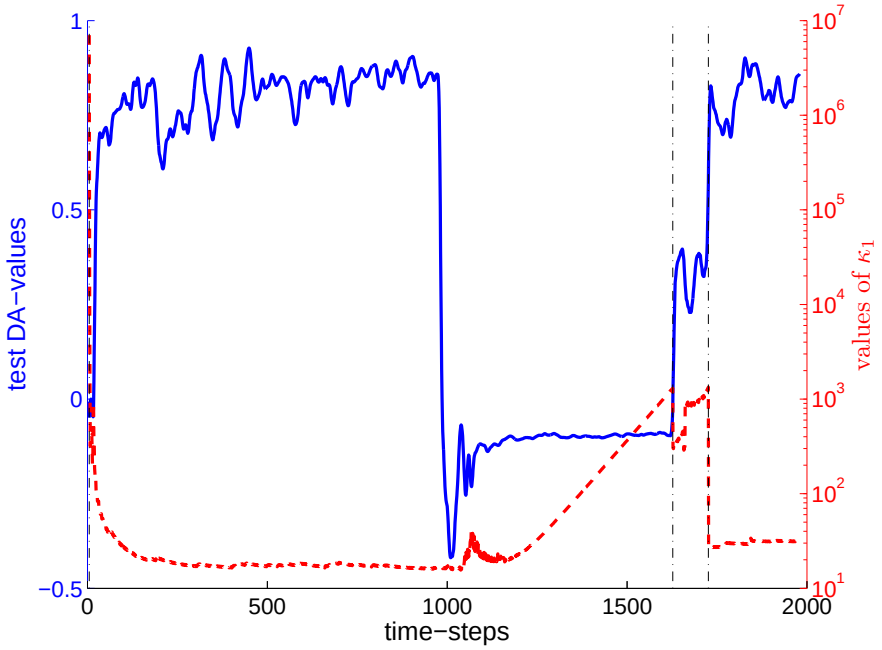


Figure 12: The (predictive) DA-values of OOMs on the test data and the variance of the condition indicator κ_1 in an online learning task.

6 Conclusion

The equivalence theorem is of fundamental importance in OOMs' theory. It is from this theorem that the learning equation involving two adjustable matrices P, Q was derived and a general procedure for learning OOMs from empirical data was developed. Two principles for designing P, Q have been discovered, leading to two different learning algorithms: the ES algorithm introduced in [7] and the EC algorithm in this report.

Unlike the ES algorithm, EC can be used in an online form, which modifies the OOM estimation on each new observed symbol via a set of RLS-like formulae. In this sense, the online EC algorithm can be seen as a statistical version of RLS estimation for system identification. Moreover, by introducing a forgetting factor, the online EC method, like the RLS estimator, is also adaptive to the change of the underlying process, as illustrated by the numerical experiments.

The main numerical problem of the EC algorithm is to minimize (for offline learning) or control (for online learning) the condition of the learning equation. For this we developed the OS algorithm, which, together with the CbC scheme, is demonstrated to be an efficient and appropriate method in our simulation. Furthermore, the numerical experiments for offline learning illustrate the EC algorithm is a robust method for learning OOMs.

Finally, we point out some future works. In our experiments, when training

OOMs by EC, the length d of basic strings is selected by hand. Intuitively, larger d would enable us to discover more information about the underlying process. This is true in principle if the training data are infinite. But for finite training sequence, large d may introduce noise into the counting matrices. In fact, one can easily imagine that, if d is too large comparing to the length l of the training sequence in that $\ell^d \gg l$, the counting matrices will be very sparse and only consist of 0 and 1; and it is difficult to estimate a useful OOM from such matrices. So methods for choosing d automatically are expected and should be studied in future.

Another possible way to overcome this problem is to use the suffix tree to get the counting matrices, as has been done by the ES algorithm [7]. The detailed procedure is also waiting for further investigation.

References

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

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

[3] R. Elliott, L. Aggoun, and J. Moore. *Hidden Markov models: estimation and control*, volume 29 of *Application of mathematics*. Springer Verlag, New York, 1995.

[4] G. H. Golub and C. F. V. Loan. *Matrix computations*. The Johns Hopkins University Press, 3 edition, 1996.

[5] H. Jaeger. Characterizing distributions of stochastic processes by linear operators. GMD Report 62, GMD, Sankt Augustin, `http://www.faculty.iu-bremen.de/hjaeger/pubs/oom_distributionsTechRep.pdf`, 1999.

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

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

[8] K. Kretzschmar. Learning symbol sequences with observable operator models. GMD Report 161, Fraunhofer Institute AIS, `http://omk.sourceforge.net/files/OomLearn.pdf`, 2003.

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

[10] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Annals of Mathematical Statistics*, 20(4):621, 1949.

[11] H. Xue and V. Govindaraju. *Stochastic models combining discrete symbols and continuous attributes in handwriting recognition.* `http://www.cedar.buffalo.edu/~govind/sto.pdf`, 2004.